Android Application Development for GPS Based Location Tracker & NITR Attendance Management System

Asit Kumar Parida

National Institute of Technology, Rourkela
Odisha, India - 769008

Android Application Development for GPS Based Location Tracker & NITR Attendance Management System

----

Thesis submitted in May, 2013 in partial fulfilment of the requirements for the degree of

Bachelor of Technology

in

Electronics & Communication Engineering

by

Asit Kumar Parida

(109EC0228)

Under the guidance of

Prof. Sarat Kumar Patra

Department of Electronics & Communication Engineering
National Institute of Technology, Rourkela
Odisha, India - 769008

Department of Electronics & Communication Engineering
National Institute of Technology, Rourkela
Odisha, India – 769008

May 11, 2013

Certificate

This is to certify that the thesis titled ***Android Application Development for GPS Based Location Tracker & NITR Attendance Management System*** submitted by *Asit Kumar Parida,* bearing Roll No. 109EC0228, is a record of authentic work carried out under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering during the session 2012-2013.

Prof. Sarat Kumar Patra
Department of Electronics & Communication Engineering,
National Institute of Technology, Rourkela
Odisha, India – 769008

Place : Rourkela
Date :

Acknowledgement

I owe a great many thanks to my friends, peers & the many people who helped and supported my idea and work on this very project.

My sincere gratitude to Prof. Sarat Kumar Patra, Department of Electronics & Communication Engineering, NIT Rourkela for having faith in me and thus allowing me to carry out a project on a technology completely new to me. He helped immensely by guiding me throughout the course of the project, inspiring me to take up new challenges along the road, and at the same time providing valuable suggestions & constructive criticisms.

I would like to thank Mr. Manas Ranjan Pattanayak, Scientific Officer, Computer Centre, NIT Rourkela for giving valuable insight into the working of the automated attendance system and the corresponding database handling. A great many thanks to Mr. Harshwardhan Kumar for training me well in Android Basics.

I would also thank our Institution & other faculty members of Department of Electronics & Communication Engineering without whom this project would have been a distant reality. I also extend my heartfelt thanks to all well wishers.

Asit Kumar Parida

109EC0228
Electronics & Communication Engineering
National Institute of Technology, Rourkela

Abstract

Accounting for more than half of the presently used hand-held devices, Android, as an operating system, has provided users with great opportunity to innovate and get things done in a mobile device. Starting as a phone OS, the array of devices compatible with Android is even driving the market in the direction of PC experience with rumours that Intel and some of the partners are working on laptop prototypes with Atom processors. And so, the need for portability has risen by leaps and bounds. People have started developing apps for every other need.

The first part of the project involves Android Application Development of a GPS based Location Tracker in which with the help of any mobile device (app installed); any other GPS enabled handset (app installed) could be located. Though target user may be located anywhere in the world, he must have network connectivity and be GPS enabled. Initially, the app is developed for Android platform only, but can be expanded to cross-platform use with device specific support in terms of Google Maps, Nokia Maps & iOS Maps Service. The app is free and currently online.

The second part of the project involves porting the Attendance System of NIT Rourkela onto Android enabled devices. The present system in intranet only, but with the new application, the extranet system would serve as an effective management tool resulting in the reduction of the no. of man hours spent in uploading the attendance for each of the subjects for the semester. The professors won’t have to double up the work as to take pen & paper attendance and then upload onto the online management. Instead a single swipe would take care of everything. Even the intuitive interface is easy to understand so that professors can easily adapt themselves to use it. Also, the application is secure allowing only one user per phone.

List of Figures

Table of Contents

CHAPTER 1

Introduction

Android as an Operating System

GPS Based Location Tracker

NITR Attendance Management System in Android

Thesis Organisation

Summary

Chapter 1. **Introduction**

1.1. **Android as an Operating System**

Being a mobile operating system, android OS is a modified version of Linux, originally developed by a start-up, Android, Inc. As Google entered mobile market, it purchased Android and in a bid to encourage independent development works, it released the developer tools under the open source Apache License. The permissive licensing allows the OS and related software to be modified and distributed by enthusiastic developers, network operators & device manufacturers.

1.1.1. **Technology Features**

Dalvik VM

A modified version of JAVA programming language is used for app development with Dalvik VM used to run the apps on Android devices. Dalvik VM can be viewed as modified version of JVM constrained in terms of memory and processor speed and converts the java bytecode (in form of JVM compatible *.class* files) to Dalvik compatible *.dex* executables before installation.

Application Interface & H/W Support

Based on Direct Manipulation, the on screen objects have been programmed to respond to real world actions like swiping, touching etc. Boasting of a fast & responsive fluidic touch screen, the OS supports various dedicated hardware like proximity sensors, gyroscopes, magnetometer and accelerometer etc. The Home Screen is analogue to the Desktop in a Windows OS.

Powered by Google Play Store, millions of apps can be readily downloaded and used. Apps are available in the *.apk* format. Google provide the SDK free of cost and it supports a comprehensive set of developing tools which primarily includes an IDE (Eclipse), a debugger, and support for emulator and sample codes etc. It even supports C/C++ extensions or bytecodes through JNI and the support is available through Native Development Kit (NDK).

Architecture

Based on Linux kernel, most of the middleware, APIs & libraries are written in C. The hardware platform is generally of ARM architecture (hence parallel processing) with later support being available for x86 & MIPS also. All GNU libraries are not supported, hence restricting porting of Windows applications onto Android. Device owners are not given ROOT access and hence they have access only to /data partition on flash storage and not to the /system which holds OS and boot files and other sensitive read-only partitions.

Memory Management

The OS supports multi-threading but depending on the instant memory availability, it can kill application so as to reduce overloading. The RAM management is such so that power consumption is at minimum. As far as third party applications are considered, the SDK provides with ample library entities such as Services, Background Tasks & Foreground Tasks for working with application lifetime.

Security & Privacy

Though the OS is immune to normal user usage, but the security flaws can be exploited, as done by the open source community, to get ROOT access (can be used for malicious purposes by crackers) and modify device capabilities. Except that, the device owners' applications are mostly run in an isolated area of OS called sandbox which restricts access to the system resources and hardware unless the user explicitly gives the access permissions during installation itself. Hence, the app gains access to /data partition through this method and the /data partition only. The newest Android OS versions have enhanced security features such as malware scanners built into system to keep a tab on malicious software downloaded through Google Play or any other third party application. Newer applications now rely on OAUTH 2.0 for secure access to internet.

Network Connectivity

The OS supports a full range of connectivity solutions ranging from Bluetooth to ZigBee (through accessory support) and from 2G to LTE support. It supports data packet transmissions through GPRS/EDGE support. Internet can also be accessed through Wi-Fi, WiMAX and shared among other devices through tethering (both over Wi-Fi & USB) support. PC communication is established through device management software using USB & Bluetooth. HTTP service is supported and through use of Google APIs, the phone is an effective GPS enabled device.

1.1.2. Android Versions till Date

Version	Code Named
1.1	Petit Four
1.5	Cupcake
1.6	Donut
2.0/2.1	Éclair
2.2	Froyo`
2.3	Gingerbread
3.0 /3.1/ 3.2	Honeycomb
4.0	Ice Cream Sandwich
4.1/4.2	Jelly Bean

1.2. GPS Based Location Tracker

1.2.1. Problem Formulation

The App "GPS based Location Tracker" is a GPS service based application which would help us in locating the exact geo-position of people (any single entity of a large set) depending upon their current location/whereabouts. Geo-position would be displayed on the map-view on our android set and display functioning can analogue to the current usage of Google Maps / Nokia Maps / iOS Map Service. Some Key points about the App:

- All users' locations would be retrieved from an online database so as to centrally control the permissions for viewing.
- For restricting user access, user authentication would be supported.
- Periodic refreshing has to be present so that each time the geo–location changes or after a fixed interval of time the values in database should be updated.
- All devices would be having a unique ID (UID) and this would be used for searching for the user.
- The app would have additional support in terms of
    - Street View & Satellite View
    - Pin Points on the Map

- Getting Address from the Map
- Locating Multiple users (support for Multiple Pin Points)
- Zooming In / Zooming Out
- Application User Data Manipulation (password)
- Change of UID support

1.2.2. **Application Overview**

In our application, we have used MapViews as supported by Google APIs 10 or higher which would allow the use of app in devices starting from Gingerbread itself. We have used an Apache Server with PHP & MySQL support for remote database use. The data transaction from or to the database occurs with the help of PHP scripts and in the form of JSON objects. The android end of the app handles this JSON objects through HTTP clients.

Onboard compass & map controllers are enabled. Locations are extracted from the device with the help of the GPS module available. A form of passive GPS use, the device decides on the best content with the information available from different providers. On touching the overlay on the map, options are asked ranging from extracting address to locating any other user on the same view.

1.2.3. **General Constraints**

Hardware

- Any Android Enabled Handheld
- Android OS Version : Gingerbread & Above
- Google Maps API 1.0 Version Compatible

Software

- Server Side : Web Server – Apache 1.2, Apache 2.x
- Server Side : Database Server – MySQL 4.1 or higher
- Server Side : PHP – PHP 4.4.0 or higher (5.2 recommended)
- Client End : Network Enabled system with Eclipse IDE and ADT Plug-in (for emulator use & debugger)

1.3. NITR Attendance Management System in Android

1.3.1. Problem Formulation

The App “Attendance Management System” is a prototype design for use by professors of NIT Rourkela which would help them in managing & uploading attendance for M Tech. / PhD. / B Tech. students. The present system requires the professors to do the following:

- For B Tech. students, to take pen & paper attendance first
- Then, upload the same to the management system through *eims.nitrkl.ac.in* before month end.
- For M Tech. & PhD. Students, the daily attendance is uploaded to the management system through *eims.nitrkl.ac.in*
- Then, approve the same at the end of week by the supervisor.

The above process needs to be simplified through:

- Porting of the attendance system to the hand-held devices.
- At any time of day, from anywhere in the world, and with zero network connectivity, the offline app should store each day’s attendance.
- Support for uploading the local database to the remote server hosted at NIT Rourkela any time of day but before the month end.
- Swiping Support with appropriate colour coding

1.3.2. Application Overview

The app thus developed would go a long way in reducing the man hours spent in managing the attendance of around 5000 odd students. Being offline, the app would be able to work in zero connectivity areas and once done, the professor can upload the same to the online database with just a single click.

As soon as the student list pops for the adding day/class, a simple swipe takes care of everything. Left to Right marks the students as present and Right to Left marks him/her as absent. Again, Left to Right gives the view entity a GREEN colour whereas the same is RED colour for Right to Left. Exceptions like opening the attendance twice in a single have been taken care thereof.

It eases the system of Biometric attendance supervision wherein the professor has to go to the web portal only when a change in previous month or week is needed. As far as security is concerned, the app is so designed as to whenever another user successfully signs in somehow, the app would crash.

1.3.3. **General Constraints**

Hardware

- Any Android Enabled Handheld
- Android OS Version : Gingerbread & Above

Software

- Server Side : Web Server – Apache 1.2, Apache 2.x
- Server Side : Database Server – MySQL 4.1 or higher
- Server Side : PHP – PHP 4.4.0 or higher (5.2 recommended)
- Client End : Network Enabled system with Eclipse IDE and ADT Plug-in (for emulator use & debugger)

1.4. **Thesis Organization**

The following thesis has been organized so as to give a clear view of what and how the apps behave. Chapter 1 gives a clear introduction to why android was chosen as the target platform. It also tells about why the apps were built and to what ends the apps are required plus the scope of the applications. Chapter 2 speaks in details about GPS Based Location Tracker application. It gives a clear view as to every aspect of the application starting from the design phase to the testing and simulation on an emulator. Chapter 3 gives a detailed description about the NITR Attendance Management system ported to an Android device. We can find a description of the app's functions and the software/hardware constraints if any. All Emulator and live phone use were shown with screenshots of all possible inputs & outcomes. Chapter 4 goes into details of the technologies used, software used & backend configurations. It also gives view on how to implement and connect to those technologies from Android Application and then back to the device end.

1.5. **Summary**

This chapter dealt with questions like why the application was created & what does it stand for. Overview or general working principles have been provided. The problem statement for each has been detailed and analyzed well. The nature of these prototypes has been explained. An introduction into why android was selected as target OS has also been provided.

CHAPTER 2

GPS Based Location Tracker

GPS

Software Modules

Workflow Sequences

Data Transaction Module Analysis

Foreground Service Module Analysis

Testing: UI Screenshots

Testing: DB Screenshots

Summary

Chapter 2. **GPS Based Location Tracker**

2.1. **GPS**

Turning ON the GPS module on the phone would not cost us anything but getting a location usually involves transaction with cell phone service provider so as to extract the location fast and with as little network connectivity as possible plus non visibility of satellites. In short: no cell phone service implies any GPS location, as far as handheld devices are considered.

Normal GPS

The method is called trilateration. The receiver listens to a particular frequency and gets data packets in the form of time coded messages from satellites. The receiver figures which satellites it can hear from. It starts gathering those messages containing time information from atomic clocks, current satellite positions etc. Nominal time to get a location is around 30-60 seconds. The same information needs to be confirmed by at least two other satellites.

Only Using Mobile Services

The user location in an area is calculated with the help of signal measurements with the information received from cell towers. Information analyzed are angle to approach towers, multipath fading characteristics with signal strength comparisons. No GPS module used.

Assisted GPS

This is what a cell phone normally uses for mapping and GPS use purposes. User location information is retrieved within 5-10 seconds. The GPS components are shared with other mobile components and hence simultaneous use of GPS and normal voice/video usage is done. First, gross positioning information from service provider based on what cell tower is being accessed and the same is fed to the GPS receiver. Next, the phone switches from phone to GPS mode for around 0.1 seconds and collects raw GPS data from satellites. It then switches back to phone mode and sends the data to the service provider to be analyzed.

The service provider uses its servers to process the data and send the most accurate location back to the phone to be displayed on a map overlay.

Full Chip GPS Receiver

The module still gets data from the service provider such as tower positioning and satellites to hear from. Switching is for 1 second but after that the receiver keeps track of information with very low power drawn from circuit.

2.2. Software Modules Used

Software modules being used in Android application can be better described in form of different activities used. An Activity is a library entity which describes the response of a layout to the user behaviour i.e. serves as an interface. The behaviour or activity is defined through a class file and an associated layout. Multiple activities have been use here so as to handle different functional requirements:

a. AboutApp.java (*R . layout . aboutapp*)
   - Show welcome information to the user. Next is authentication screen.

b. AuthenticateUser.java (*R . layout . authen*)
   - For first time user, it first registers the user onto the device database.
   - Once registered, for normal user, it checks login ID and password and authenticates the same against those stored internally.
   - On proper authentication, the user is taken to the application menu.

c. CustomPin.java
   - Describes the behaviour of the pin point which would be used to point to any arbitrary location user touches on the screen
   - Initializes the pin point image as a Green pointer with 'G' inscribed.
   - Any pinpoint would be added as a overlay on the main MapView

d. FrendPin.java
   - Describes the behaviour of the pin point which would be used to point to the user whose location was retrieved from the server
   - Initializes the pin point image as a Brown pointer with 'U' inscribed.

- Any pinpoint would be added as a overlay on the main MapView

e. HomePin.java
   - Describes the behaviour of the pin point which would be used to point to the device location or own location on the MapView
   - Initializes the pin point image as a Blue pointer with 'A' inscribed.
   - Any pinpoint would be added as a overlay on the main MapView

f. JSONParser.java
   - Important class used to define the behaviour of the entity which would handle the transaction with remote server using network connection
   - Data is formatted as *Java Script Object Notation* (JSON) objects.
   - The method *getServerResponse(url , method , params)* returns a JSON object.
   - If the method is *"POST"*, then an Http Client is used to simply send instructions or data contained in *"params"* to the remote server designated by *"url"*. The response received contains no appreciable data apart from success or failure execution information when this connection method is used..
   - If the method is *"GET"*, then an Http Client is used to simply send information contained in *"params"* to the remote server designated by *"url"* and the response is received from the server containing required data when this connection method is used.
   - The response received is built to strings and encoded into JSON format to be parsed latter by other object so as to retrieve information.

g. MapsMain.java (*R . layout . maps_main , R . layout . createuser*)
   - Initializes the *MapView* which is the primary view being used in our application
   - The MapView implements the Google APIs for maps
   - Implements toggling between *Street View&Satellite View* for Maps
   - Implements addition of compass, controller to animate to location and zoom in/out& extraction of the best *Criteria* for selection of *Provider*
   - Initializes home location and recurring refreshing of the same
   - Implement extraction of the address from the particular coordinates and can place a pinpoint at that particular position &removal of all pin points
   - Implement *AsyncTasks* for (execution on a separate thread so as not to overload the main thread) getting the JSON encoded location of user though his/her UID, to create the user in the remote database if the app is being used for the first time &to delete the same user data in the remote database.

h. Menu.java
   - Gateway to different activities, each managing *MapView*, our preferences, broadcasting of our location

i. ServiceSetup.java (*R.layout.updatediag*)
   - Implements the layout for using the service interface
   - Wherein we can explicitly start or stop broadcast of our location to the remote location. It setups the objects of UpdateService Class

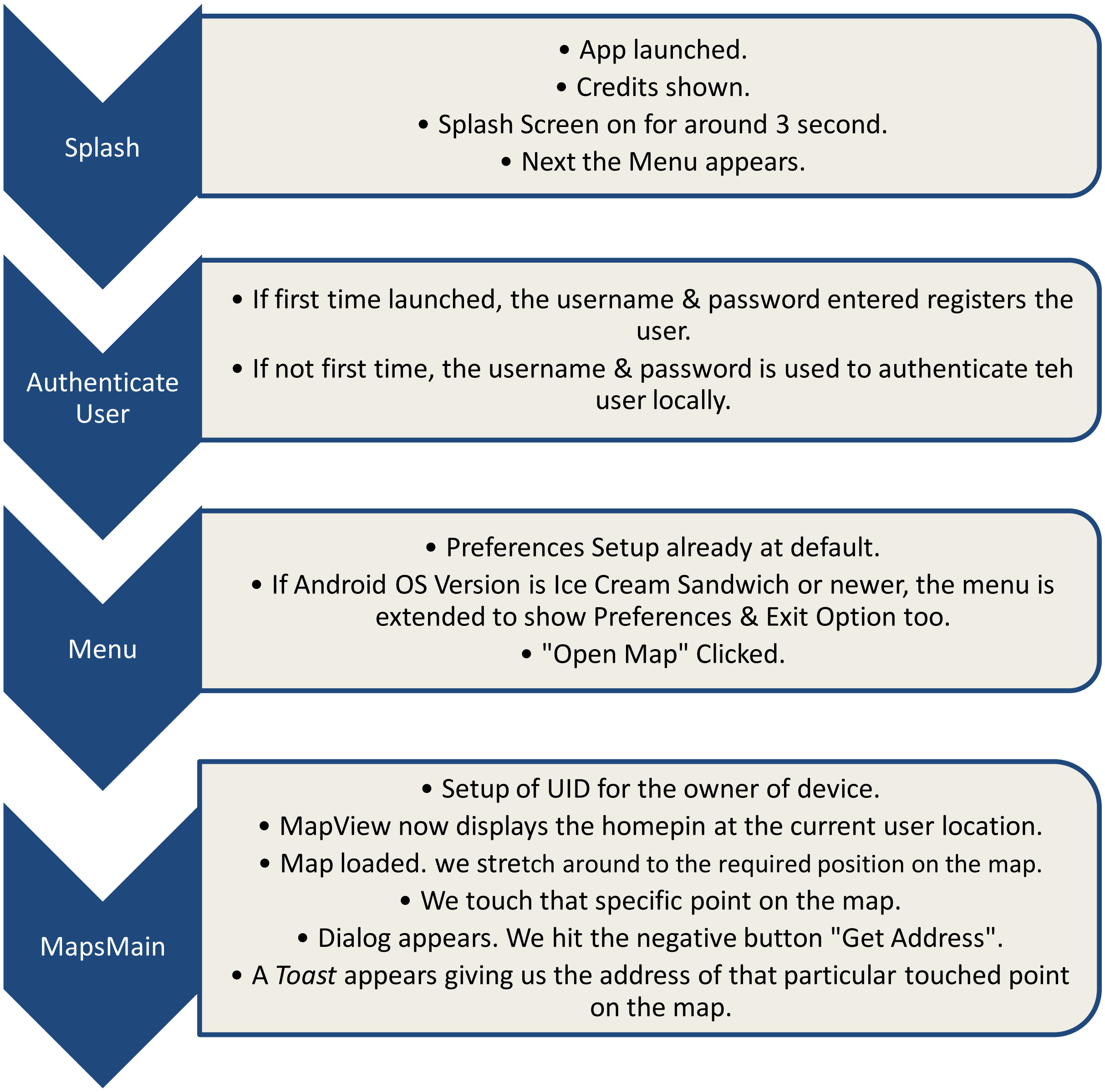
j. Splash.java (*R . layout . splash*)
   - Implements the first screen when the app is launched showing app name and credits

k. UpdateService.java
   - Implements a background service which periodically refreshes our location in the remote database against our UID

2.3. Work Flow Sequences

To Extract Address from a touched point on the MapView, we must touch the screen for more than 2.5 seconds leading to generation of the dialog. We hit the button “Get Address” & the view finds the x & y coordinates on the screen & correlates it with that of the map. The view connects to the server giving it the lat& long for that point & hence, displays address.

Figure 2-1 : Extraction of Address From Touched Point on MapView



Splash

- App launched.
- Credits shown.
- Splash Screen on for around 3 second.
- Next the Menu appears.

Authenticate User

- If first time launched, the username & password entered registers the user.
- If not first time, the username & password is used to authenticate teh user locally.

Menu

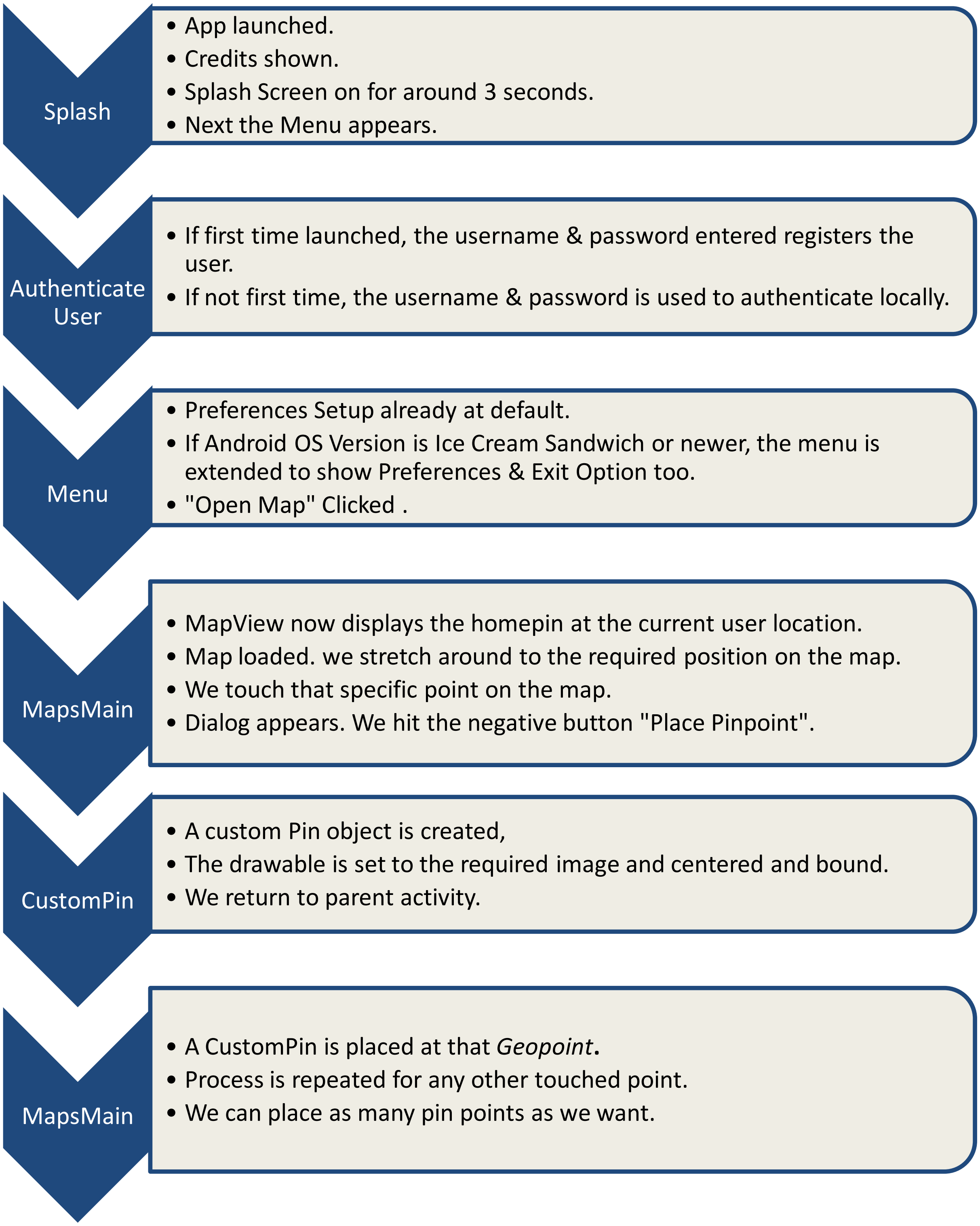
- Preferences Setup already at default.
- If Android OS Version is Ice Cream Sandwich or newer, the menu is extended to show Preferences & Exit Option too.
- "Open Map" Clicked.

MapsMain

- Setup of UID for the owner of device.
- MapView now displays the homepin at the current user location.
- Map loaded. we stretch around to the required position on the map.
- We touch that specific point on the map.
- Dialog appears. We hit the negative button "Get Address".
- A *Toast* appears giving us the address of that particular touched point on the map.

To place a pinpoint on the MapView, we follow the same procedures as to the point of displaying of dialog, and then we hit the button "Place PinPoint".

Figure 2-2 : Placing a PinPoint on a Desired Location



Splash

- App launched.
- Credits shown.
- Splash Screen on for around 3 seconds.
- Next the Menu appears.

Authenticate User

- If first time launched, the username & password entered registers the user.
- If not first time, the username & password is used to authenticate locally.

Menu

- Preferences Setup already at default.
- If Android OS Version is Ice Cream Sandwich or newer, the menu is extended to show Preferences & Exit Option too.
- "Open Map" Clicked .

MapsMain

- MapView now displays the homepin at the current user location.
- Map loaded. we stretch around to the required position on the map.
- We touch that specific point on the map.
- Dialog appears. We hit the negative button "Place Pinpoint".

CustomPin

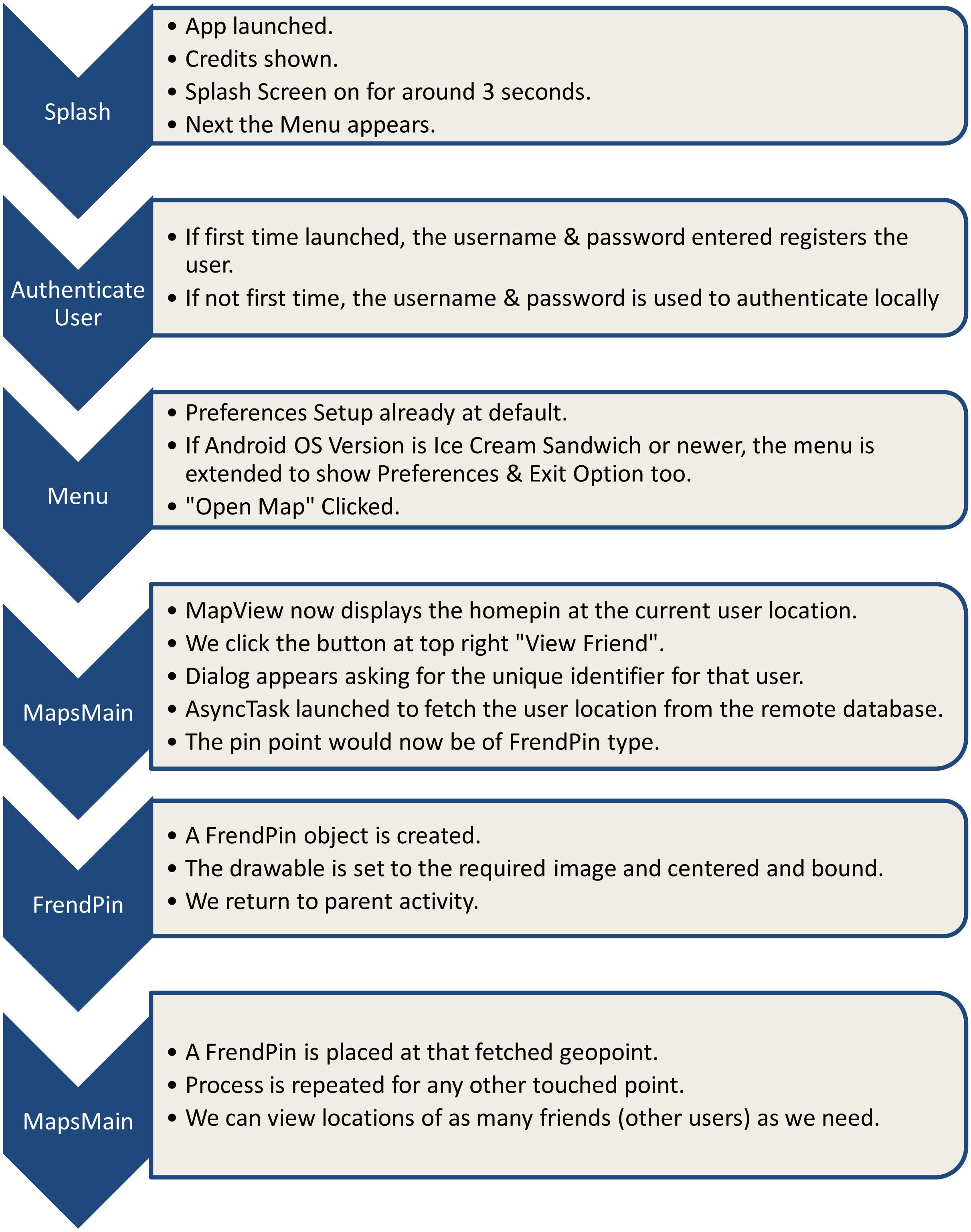
- A custom Pin object is created,
- The drawable is set to the required image and centered and bound.
- We return to parent activity.

MapsMain

- A CustomPin is placed at that *Geopoint*.
- Process is repeated for any other touched point.
- We can place as many pin points as we want.

To locate another user on the MapView, we hit the button “Find Friend” on top-right, the dialog asks for UID of the person, the location is to be fetched from the remote database.

Figure 2-3 : Locating Another User/Friend in MapView



Splash

- App launched.
- Credits shown.
- Splash Screen on for around 3 seconds.
- Next the Menu appears.

Authenticate User

- If first time launched, the username & password entered registers the user.
- If not first time, the username & password is used to authenticate locally

Menu

- Preferences Setup already at default.
- If Android OS Version is Ice Cream Sandwich or newer, the menu is extended to show Preferences & Exit Option too.
- "Open Map" Clicked.

MapsMain

- MapView now displays the homepin at the current user location.
- We click the button at top right "View Friend".
- Dialog appears asking for the unique identifier for that user.
- AsyncTask launched to fetch the user location from the remote database.
- The pin point would now be of FrendPin type.

FrendPin

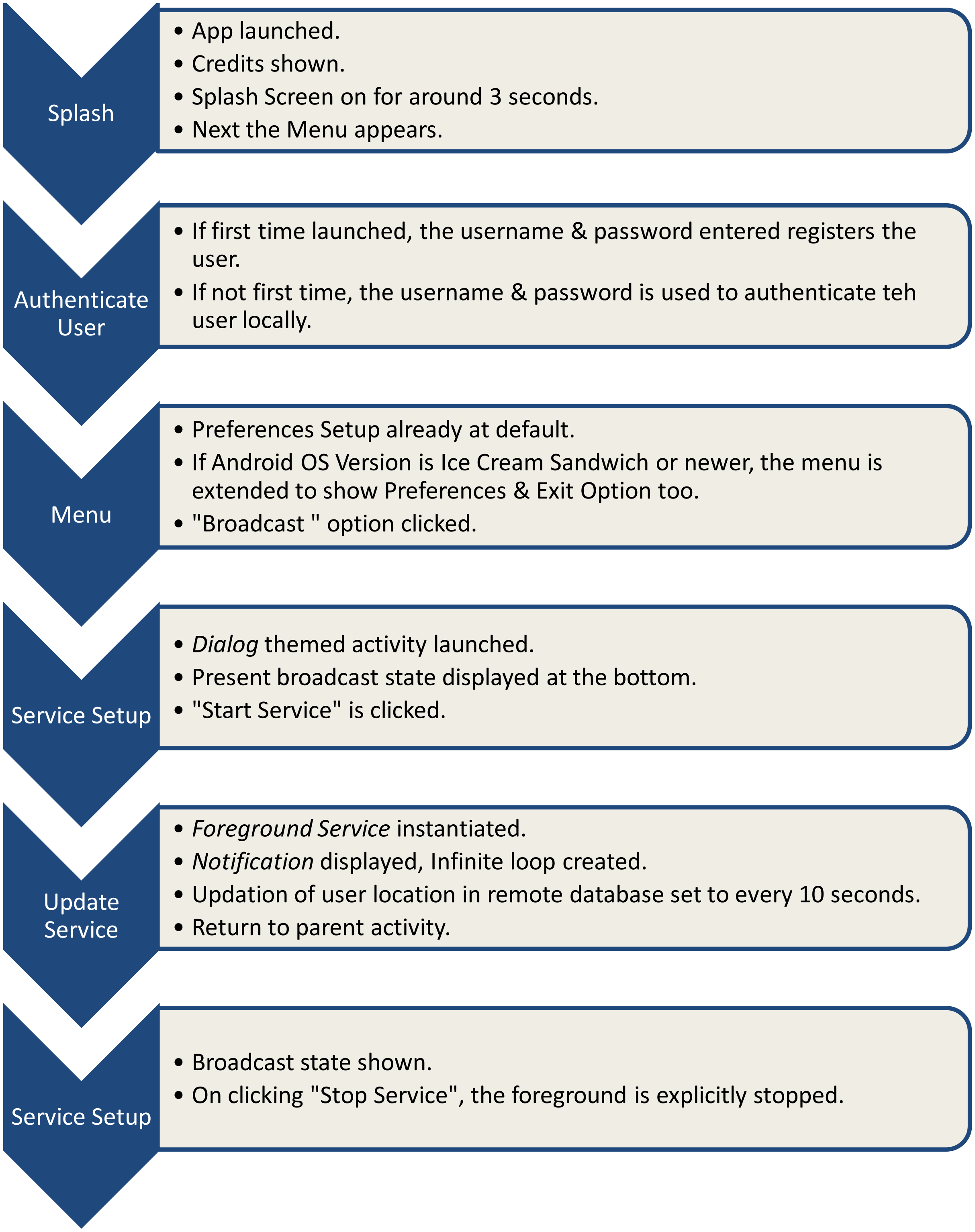
- A FrendPin object is created.
- The drawable is set to the required image and centered and bound.
- We return to parent activity.

MapsMain

- A FrendPin is placed at that fetched geopoint.
- Process is repeated for any other touched point.
- We can view locations of as many friends (other users) as we need.

To enable broadcast of our location, we launch the same from Menu, and then start the service to periodically send our location to be stored in the remote database.

Figure 2-4 : Setting Up Broadcast Action



Splash

- App launched.
- Credits shown.
- Splash Screen on for around 3 seconds.
- Next the Menu appears.

Authenticate User

- If first time launched, the username & password entered registers the user.
- If not first time, the username & password is used to authenticate teh user locally.

Menu

- Preferences Setup already at default.
- If Android OS Version is Ice Cream Sandwich or newer, the menu is extended to show Preferences & Exit Option too.
- "Broadcast " option clicked.

Service Setup

- *Dialog* themed activity launched.
- Present broadcast state displayed at the bottom.
- "Start Service" is clicked.

Update Service

- *Foreground Service* instantiated.
- *Notification* displayed, Infinite loop created.
- Updation of user location in remote database set to every 10 seconds.
- Return to parent activity.

Service Setup

- Broadcast state shown.
- On clicking "Stop Service", the foreground is explicitly stopped.

2.4. Data Transaction Module Analysis

For any data transaction over the network arising in both the applications we have developed, we have used *AsyncTask* Class type objects. The sole reason for this is not to overload the main thread. Whenever an AsyncTask object is created, the execution of it behavior is done in a separate UI thread. Then, we need to add the following unimplemented methods:

a. onPreExecute()
   - Throughout our applications, we have displayed a progress dialog here.
   - The dialog shows a message while the execution of *doInBackground*() is complete.

b. doInBackground()
   - A new thread is created, data to be sent to the server is added as a *NameValuePair* object encoded in a list.
   - Using an object of *JSONParser* Class, the *getServerResponse()* is called, with three arguments being supplied: the URL, method type & the parameters to be sent.
   - *getServerResponse()* method returns a *JSONObject*.
   - The object is then parsed to get information about:
     - ✓ Successful Execution of script on server,
     - ✓ Error message, if any,
     - ✓ Data such as student list, subject list etc.

c. onPostExecute()
   - The execution is about to end.
   - Progress dialog is closed.
   - Result implemented, if needed.

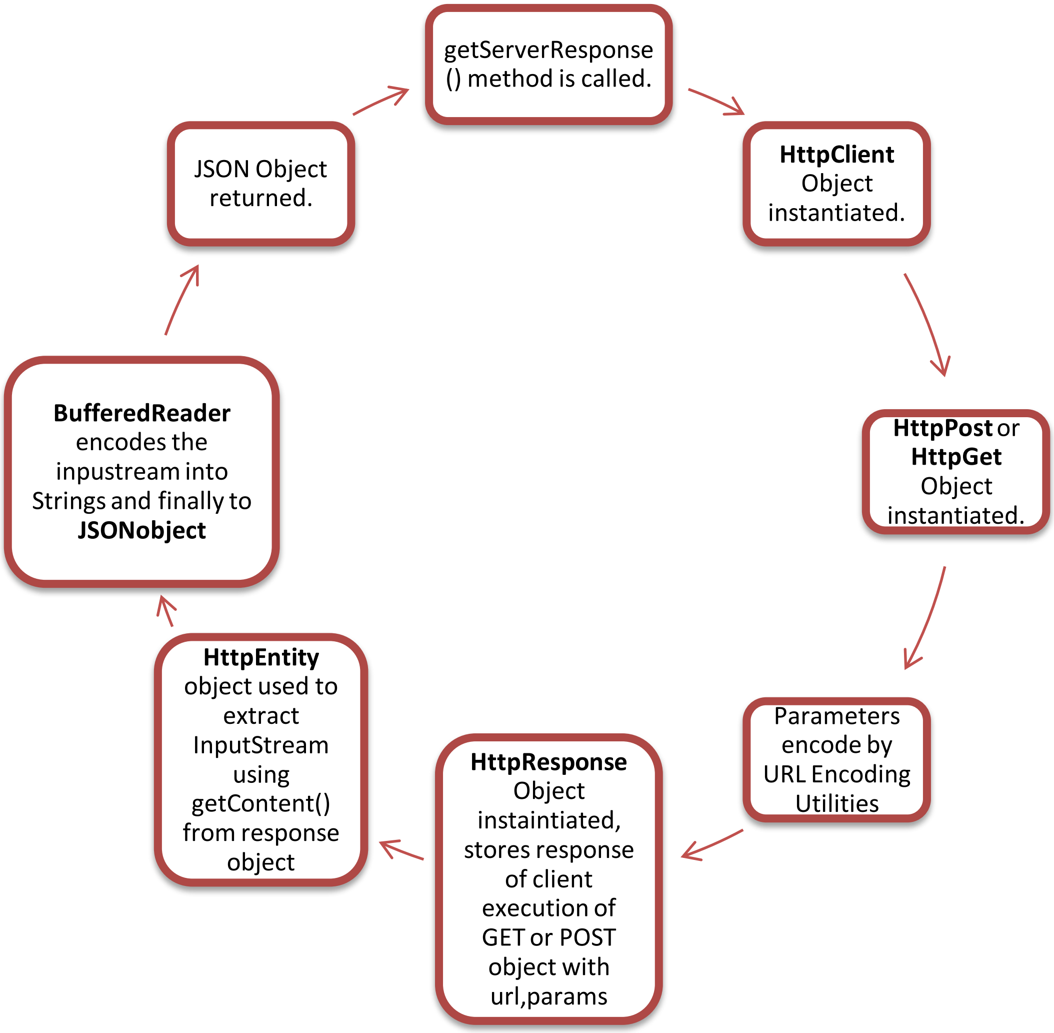
JSONParser Class

Very briefly, it defines the method *getServerResponse()*. There are two method types: “GET” or “POST”.

- *HttpClient* is instantiated.
- If “GET” method type is selected, the user is trying to get some viable data back from the remote server and as parameters, it is sending some data in params. The query is encoded in an *HttpGet* type object.

- If "POST" method type is selected, the user is just trying to send some viable data to the remote server and as parameters, it is sending some data in params. The query is encoded in an *HttpPost* type object.
- The response of the client's execution of this object is stored in an *HttpResponse* type object.
- Response thus received is parsed into entities.
- The content received is extracted from the response object through the *getContent()* method executed on the response by an *HttpEntity* object.
- The content is an *inputstream*. And using *BufferedReader*, the content received, thus, is translated into Strings.
- The strings are finally encoded into a JSON object.

Figure 2-5 : getServerResponse() Execution Cycle in Android


getServerResponse() method is called.
HttpClient Object instantiated.
HttpPost or **HttpGet** Object instantiated.
Parameters encode by URL Encoding Utilities
HttpResponse Object instaintiated, stores response of client execution of GET or POST object with url,params
HttpEntity object used to extract InputStream using getContent() from response object
BufferedReader encodes the inpustream into Strings and finally to **JSONobject**
JSON Object returned.

2.5. Foreground Service Analysis (Updating Service)

Service is an android library entity which is used to execute a chunk of code in the background, without direct interaction with the user. Android kernel, itself, runs many a system services responsible for managing the normal use of phone. System Services can be accesses through *getSystemService()* method. Services run with a priority higher than visible & invisible activities and therefore memory manager is less likely to terminate it in case of memory overloading. Like other classes, services also need to be defined in the manifest. Also, the service may be compelled to run in its separate process so that it can have its own memory space. But it does make the communication of main application with the service a bit tricky. These background services are also killed when memory overload exceeds the limits, but as soon as memory if freed, the service starts again until explicitly stopped.

Any Activity that extends the Service Library must add and define the following unimplemented methods:

a. onBind()
    - System calls this method to bind another component with the service and so calls *bindService()*.
    - Interface need for clients to communicate with the service through an *IBinder* object returned.
    - If we don't want to implement, then we return null instead of an *IBinder*.

b. onStartCommand()
    - Code to be executed when service starts.
    - If we want a regular/periodic updating service, we can create an infinite loop and force the thread to sleep for some time to make it periodic and not overload the thread.
    - Called when the service is started by calling the *startService()* Method
    - If we don't want the service to be killed under memory overload i.e. treat the service as a system component, we need to declare it as *Foreground Service*.
    - A foreground service provides a notification for the status bar. A foreground service is started by calling the function *startForeground()* within the *onStartCommand()* and stopped by calling the function *stopForeground().*
    - If the integer returned is *Service.START_STICKY*, the service is restarted after being killed by memory manager and not restarted if integer returned is *Service.START_NOT_STICKY*.

c. onDestroy()

- Called when explicitly *stopService()* has been called.
- *stopForeground()* is called here.

Figure 2-6 : Normal Background Service Resumption by START_STICKY

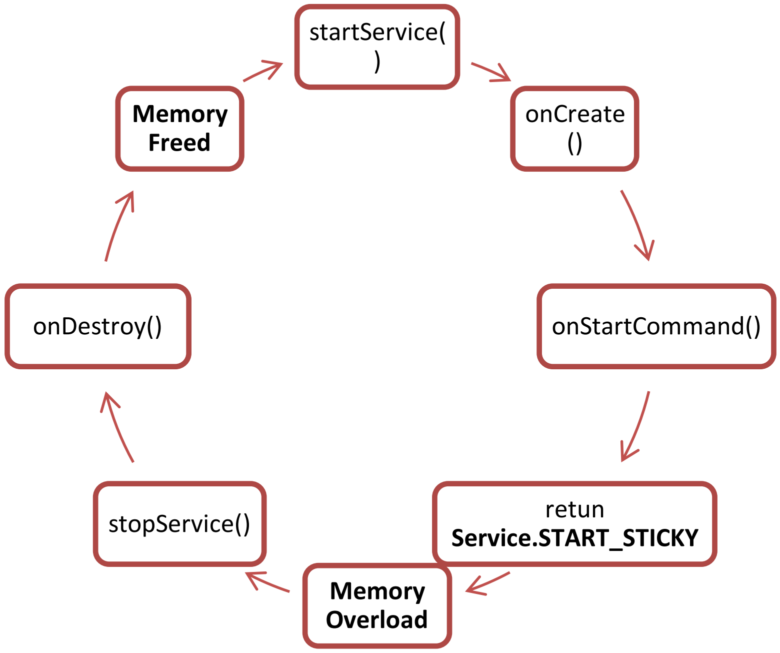
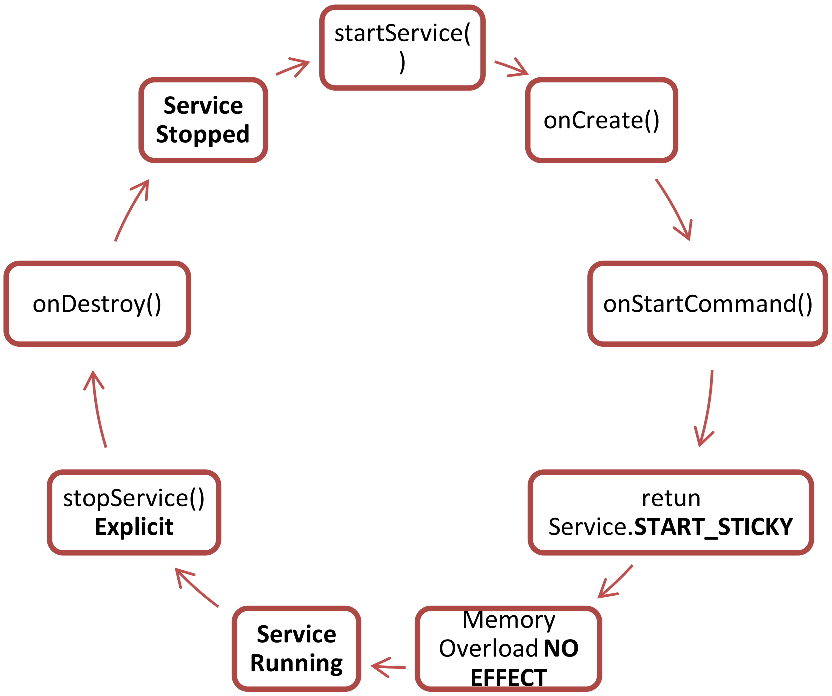
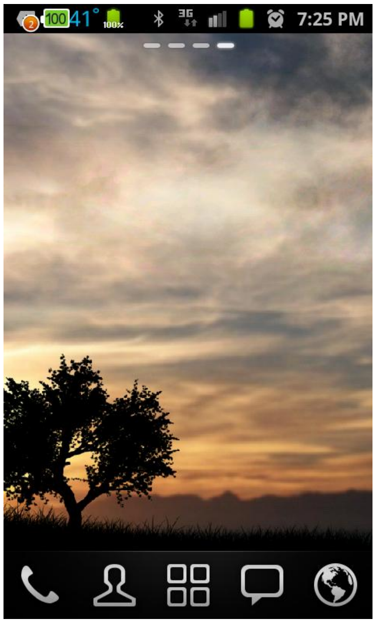


Figure 2-7 : Normal Foreground Service Cycle in Android

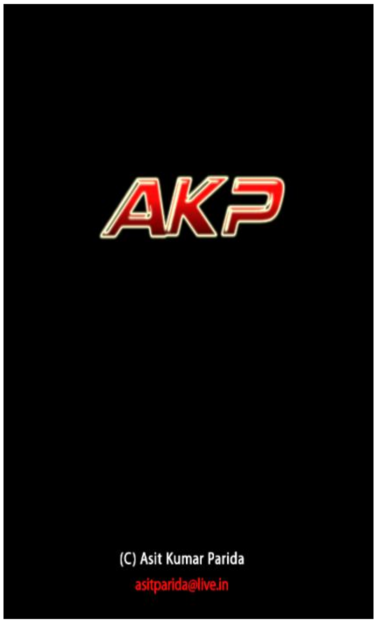


2.6. Testing: UI Screenshots

Home Screen Interface

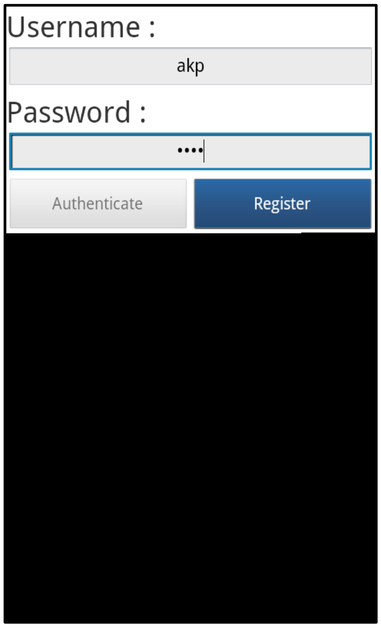


Application Launch



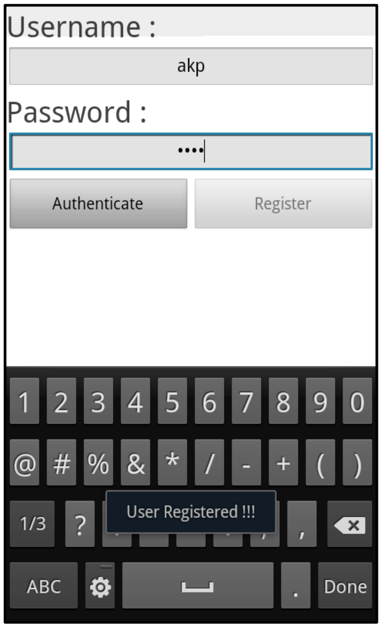
(C) Asit Kumar Parida
asitparida@live.in

Registration Screen



Username :
akp
Password :
••••

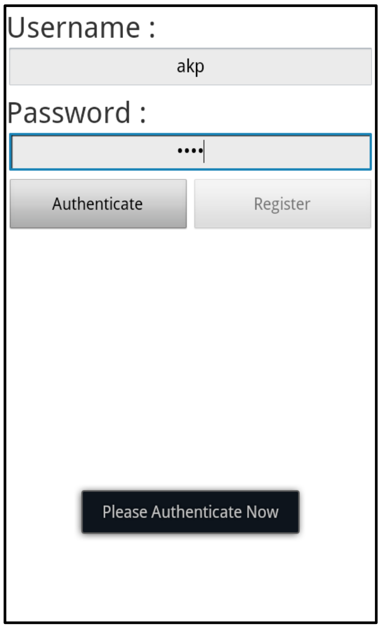
Registration Successful



Username :
akp
Password :
••••

User Registered !!!

Authentication Screen

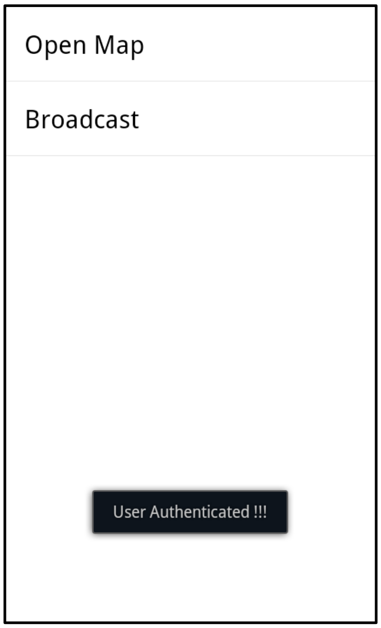


Username : akp

Password : ••••

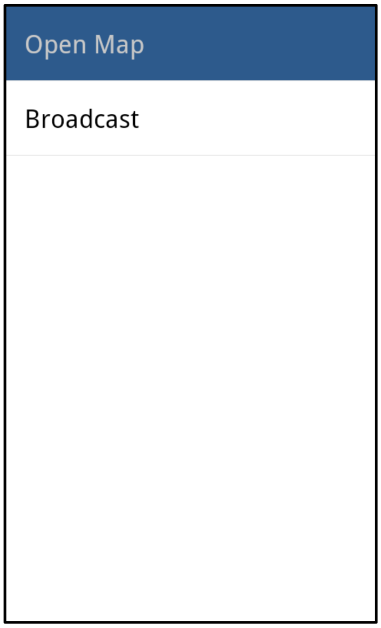
Please Authenticate Now

Authentication Successful

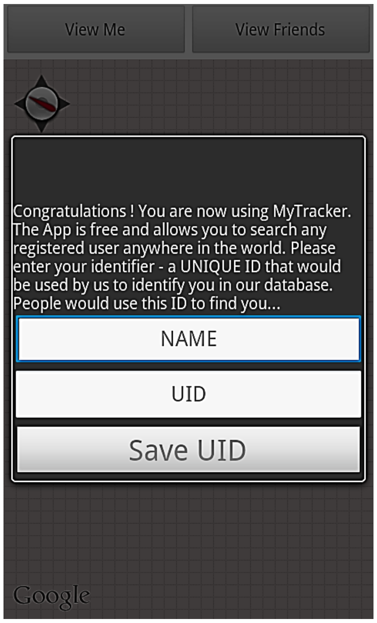


User Authenticated !!!

Menu Open & "Open Map" Clicked

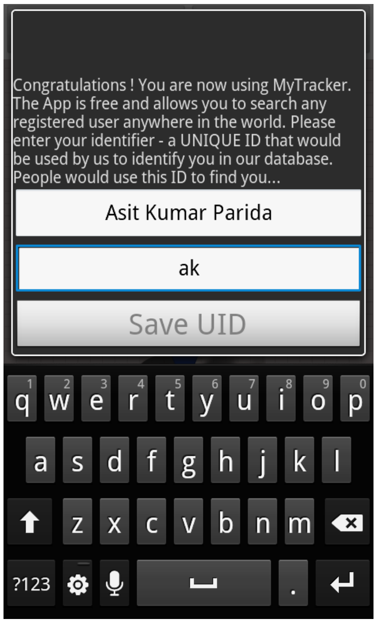


UID Registration Screen



Congratulations ! You are now using MyTracker. The App is free and allows you to search any registered user anywhere in the world. Please enter your identifier - a UNIQUE ID that would be used by us to identify you in our database. People would use this ID to find you...

Data Entry into UID Dialog

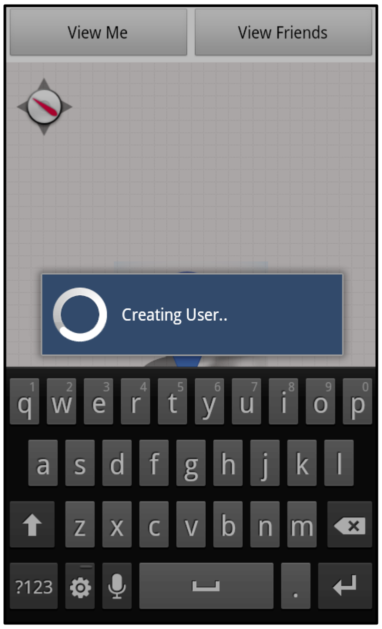


Congratulations ! You are now using MyTracker. The App is free and allows you to search any registered user anywhere in the world. Please enter your identifier - a UNIQUE ID that would be used by us to identify you in our database. People would use this ID to find you...

Asit Kumar Parida

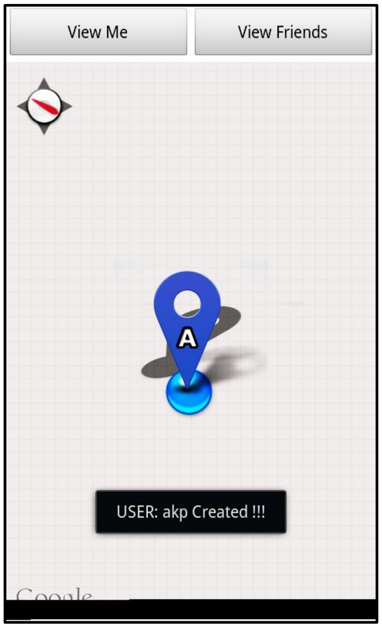
ak

Creating User in Remote Database



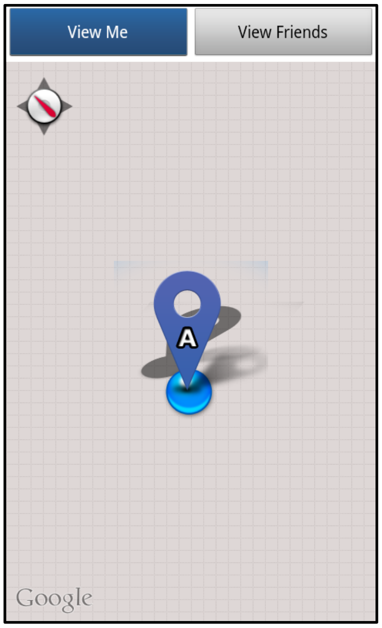
Creating User..

User Successfully Created

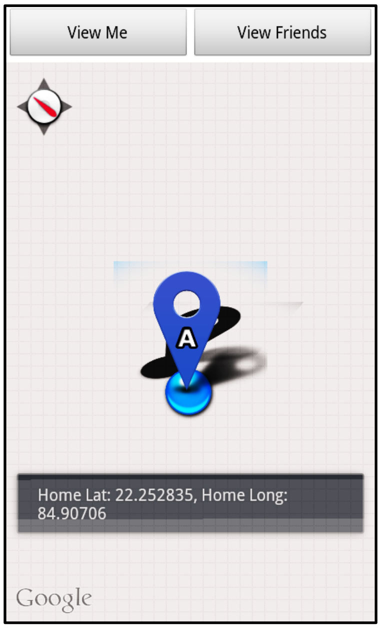


USER: akp Created !!!

Button To Show Own Location Clicked

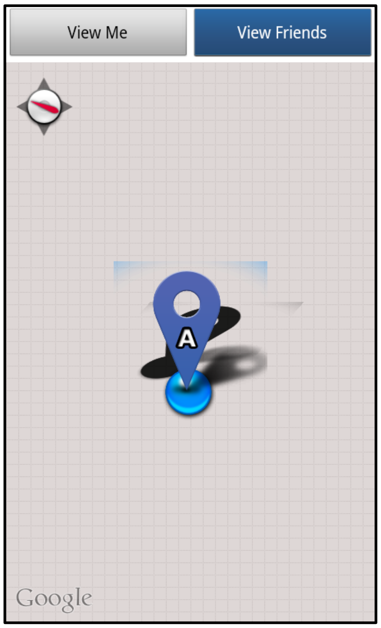


Home Location Displayed with "A"

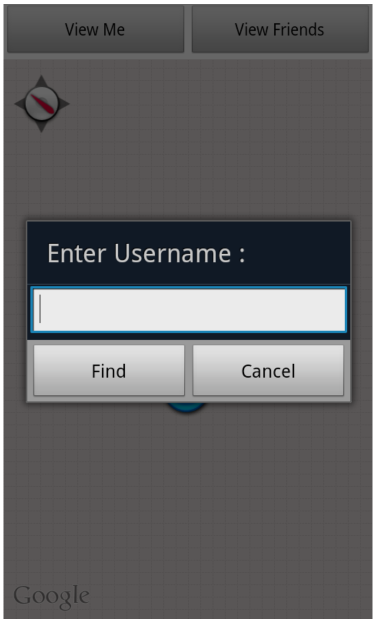


Home Lat: 22.252835, Home Long: 84.90706

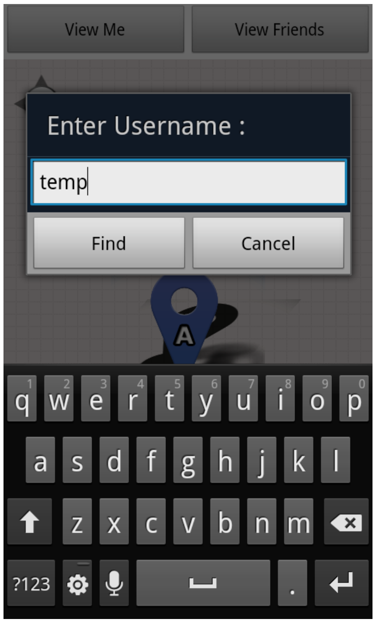
Friend Finder Button



Enter UID of Person to Find



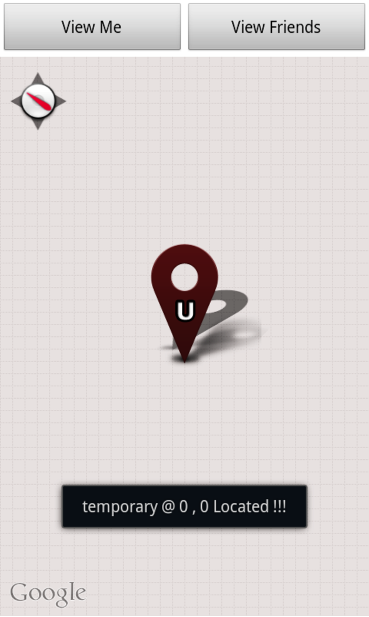
UID of Person to Find Entered



Enter Username :

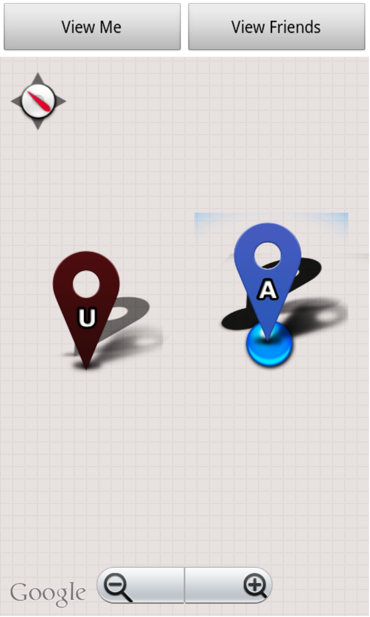
temp

User found and pinpointed with "U"

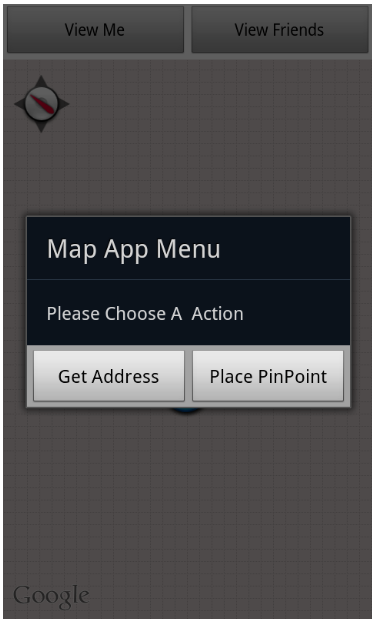


temporary @ 0 , 0 Located !!!

Viewing both Me & User



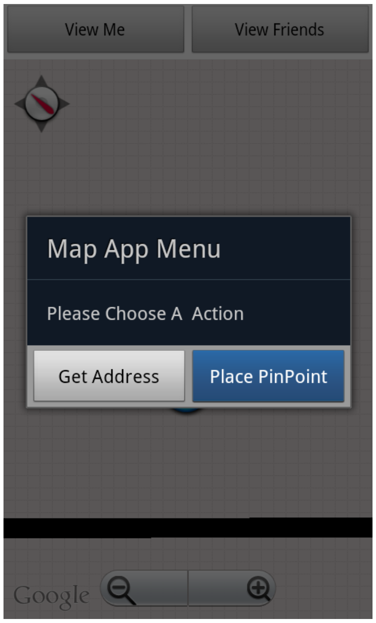
Map Clicked for More than 2.5 Seconds



Map App Menu

Please Choose A Action

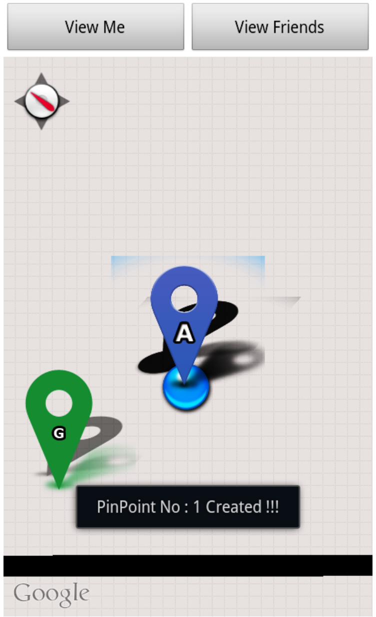
Placing Custom Pin Point



Map App Menu

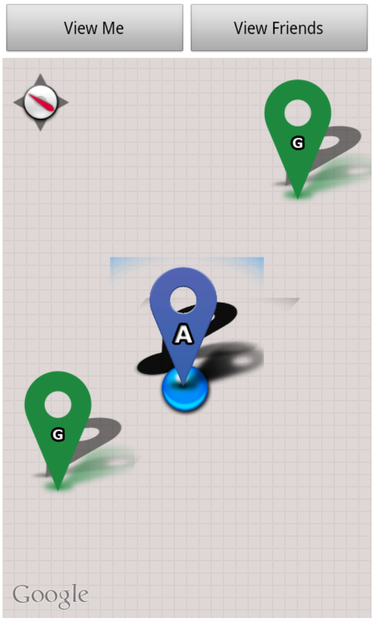
Please Choose A Action

Custom PinPoint 1 Created

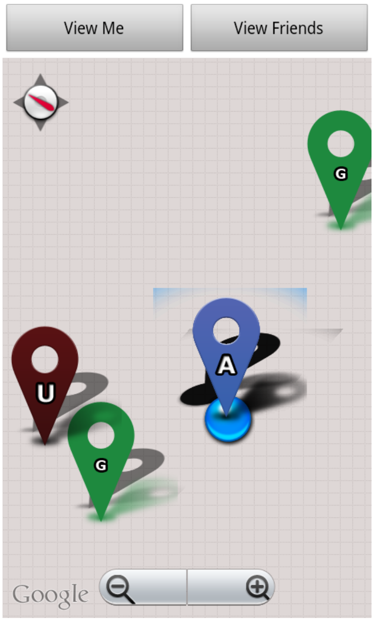


PinPoint No : 1 Created !!!

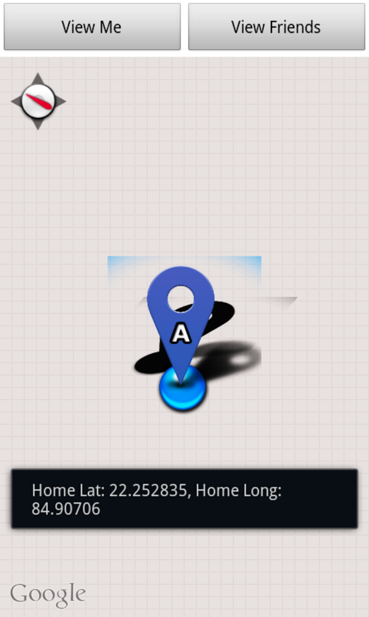
Custom PinPoint marked with "G"



View showing Myself, Friends & Cutsom Pins



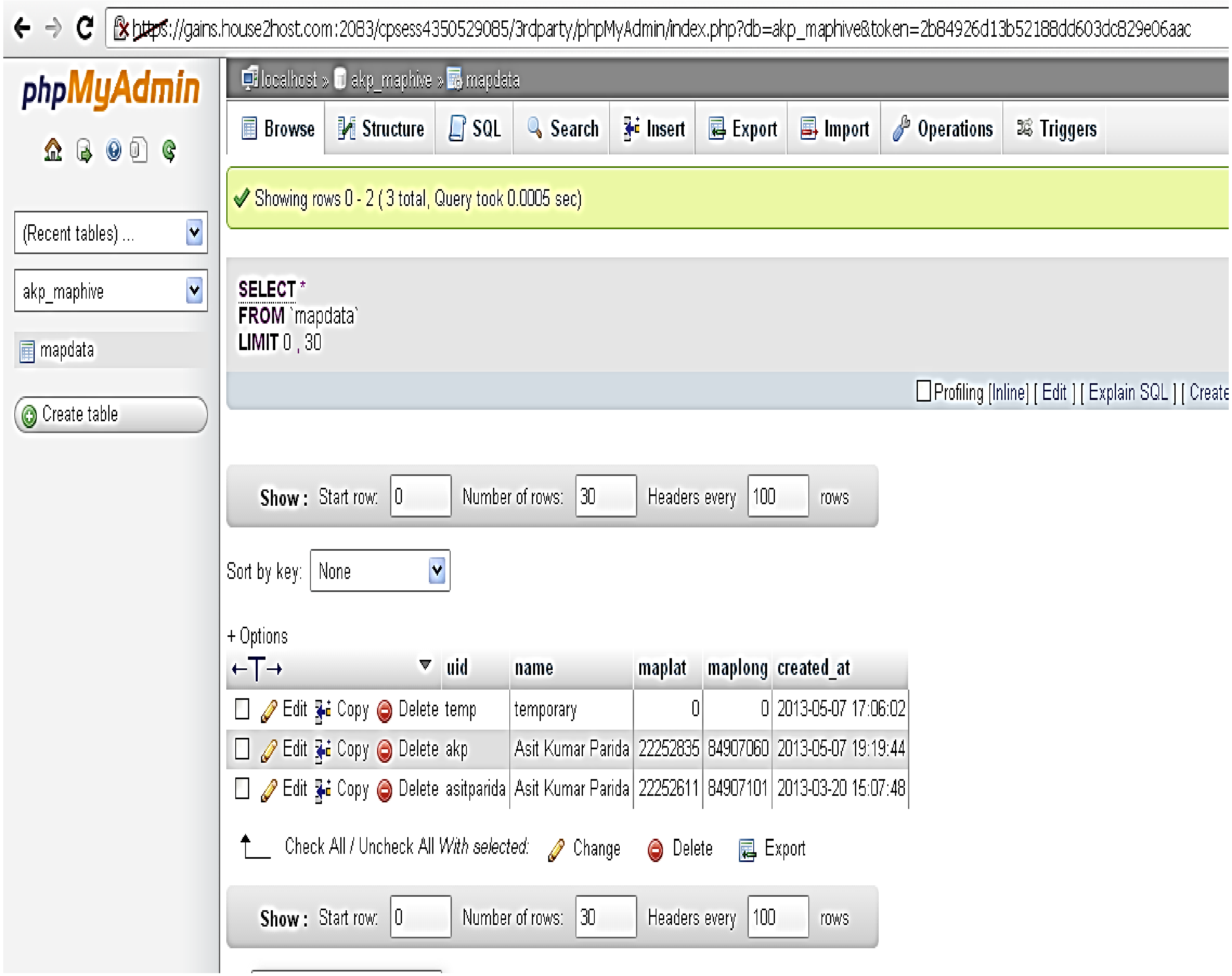
Switched back to Home Location



Home Lat: 22.252835, Home Long:
84.90706

2.7. Testing: DB Screenshots

Figure 2-8 : DB Screenshot For Tracker Application



~~https~~://gains.house2host.com:2083/cpsess4350529085/3rdparty/phpMyAdmin/index.php?db=akp_maphive&token=2b84926d13b52188dd603dc829e06aac
phpMyAdmin
(Recent tables) ...
akp_maphive
mapdata
Create table
localhost » akp_maphive » mapdata
Browse
Structure
SQL
Search
Insert
Export
Import
Operations
Triggers

✓ Showing rows 0 - 2 (3 total, Query took 0.0005 sec)

```
SELECT *
FROM `mapdata`
LIMIT 0 , 30
```

□ Profiling [Inline] [Edit] [Explain SQL] [Create

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

←T→			uid	name	maplat	maplong	created_at
□	Edit	Copy	Delete	temp	temporary	0	0	2013-05-07 17:06:02
□	Edit	Copy	Delete	akp	Asit Kumar Parida	22252835	84907060	2013-05-07 19:19:44
□	Edit	Copy	Delete	asitparida	Asit Kumar Parida	22252611	84907101	2013-03-20 15:07:48

Check All / Uncheck All *With selected:* Change Delete Export

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

2.8. **Summary**

The chapter spoke in detail about the various aspects of the application designed to emulate GPS based Location Tracker. First of all, GPS was introduced. Assisted GPS, the more common form is talked about and showed to be the predominant case of usage. Workflow sequences detailing how activities are launched and in what sequence in correspondence to the objective at that instant, have also been shown in figures. Two important modules have been explained: the data transaction module working with the help of HTTP client & Foreground Service Module handling the regular/periodic updating of user location in remote server. The screenshots regarding various testing scenarios have also been presented.

CHAPTER 3

NITR Attendance Management System

Present Attendance System

Software Modules

Workflow Sequences

Swipe Detector Module Analysis

SQLite DBMS Module Analysis

Testing: UI Screenshots

Testing: DB Screenshots

Summary

Chapter 3. NITR Attendance Management System

3.1. Present Attendance Management

The current attendance system is managed differently with relation to the category to which the student belongs. B. Tech, 5 year Integrated M Sc& Dual Degree students fall into one category and the M.Tech & PhD student in one category.

For the former category, each semester some papers are offered. Each student is assigned a unique ID for that particular paper for that particular semester. For each class, the professor takes pen & paper attendance. Before the end of the month, the professor uploads the same attendance onto the web portal. At the start of each month, the previous and cumulative attendance record can be accessed by the student. The process continues for each month till the semester ends. If any change is required, the professor has to go the web portal and carry out the change. After the final approval at the end of month, the database can't be modified with the exception of the super administrator. It should be noted that for the same professor, different lists of students would exist corresponding to the no. of papers he would be taking that semester

For the latter category of students, the performance is judged throughout the year through biometric attendance management. The month wise approval system is replaced by a week based system i.e. the approval needs to be done before the month ends. The professor needs to open the web portal to approve each day's attendance before final approval. The system marks absentees in RED and on approval/change by the supervisor, it turns to GREEN. Hence, we have used the same colour coding to reflect attendance status for each student. It should be noted that for the same professor, only one list of students would exist corresponding to all students under his supervision.

3.2. Software Modules used

Software modules being used in this Android application can be better described in form of different activities being used. The behaviour or activity is defined through a class file and an associated layout. Multiple activities have been use here so as to handle different functional requirements:

a. AddDay.java (*R . layout .addday*)
    - With respect to category 2, the module adds the current day in local database
    - With respect to category 2, the module also populates the list of student under the professor's supervision
    - With respect to category 1, the module also populates the list of student in professor's current subject. In category 1, module is called *AddClass.jav*a

b. AuthenticateUser.java (*R . layout . authen*)
    - For first time user, it first authenticates the user from the remote database and stores the login data onto the local storage.
    - If not the first time, the user authentication depends on whether offline authentication is enabled or disabled.
    - On proper authentication, the user is taken to the Subject List or Biometric List.

c. BioSpecs.java (*R . layout . biospec*)
    - Manages Biometric Attendance i.e. for category 2 students.
    - Student List can be viewed.
    - Month addition necessary before any day in that month can be added.
    - Once any day's attendance has been generated, the sync button commands the app to synchronize & upload the local database onto the remote database.

d. JSONParser.java
    - Important class used to define the behaviour of the entity which would handle the transaction with remote server using network connection.
    - Data is formatted as Java Script Object Notation (JSON) objects.
    - The method *getServerResponse(url , method , params)* handles the data transaction over network.

e. MainActivity.java (*R . layout . activity_main*)
    - Implements the first screen when the app is launched showing app name and credits.

f. ProfessorUI.java (*R . layout . profui*)
    - Just to smooth transitioning between various functions. Takes us to biometric management activity (for category 2) or takes us to the list of subjects or papers being taught by the professor.

g. * SelectClass.java (*R . layout . selclass*)
  - The activity helps in implementing the B. Tech. attendance scenario wherein attendance is added in form of a class rather than a day.
  - Moreover, not every day classes are taken, hence we implemented *DatePicker* module so as to set the date also.
  - Support for multiple classes on the same day is also implemented.

h. SQLDayListHandle.java
  - Handles the tables which store the dates or days on which attendance have been added to the database.
  - Instantiates a *SQLiteOpenHelper* class object which would help manage the tables in a *SQLiteDatabase* type database.

i. SQLMonthHandle.java
  - Handles the tables which store the attendance for each student.
  - Days/Dates are added to the tables of this type only.
  - These tables are uploaded or synced with the remote server.

j. SQLStulistHandle.java
  - Handles the tables which store the student list under supervision of professor or enrolled in different subjects being taught by the professor.

k. Stulist.java (*R . layout . stulist*)
  - The activity only displays the list of students.
  - On being opened the first time, the list is fetched from the remote server.
  - On subsequent launches, the list is populated from local database.

l. * SubjectSpec.java (*R . layout . subjspec*)
  - Manages category 1 students' attendance.
  - Student List can be viewed.
  - Month addition necessary before any day in that month can be added.
  - Once any day's attendance has been generated, the sync button commands the app to synchronize & upload the local database onto the remote database.

m. SwipeDetector.java
  - Handle and defines the behavior of swiping action on the screen.
  - Responsible for recognizing a swiping action.
  - Implements return of type of swipe generated.

n. * ViewSubjects.java ($R\,.\,layout\,.\,subjz$)

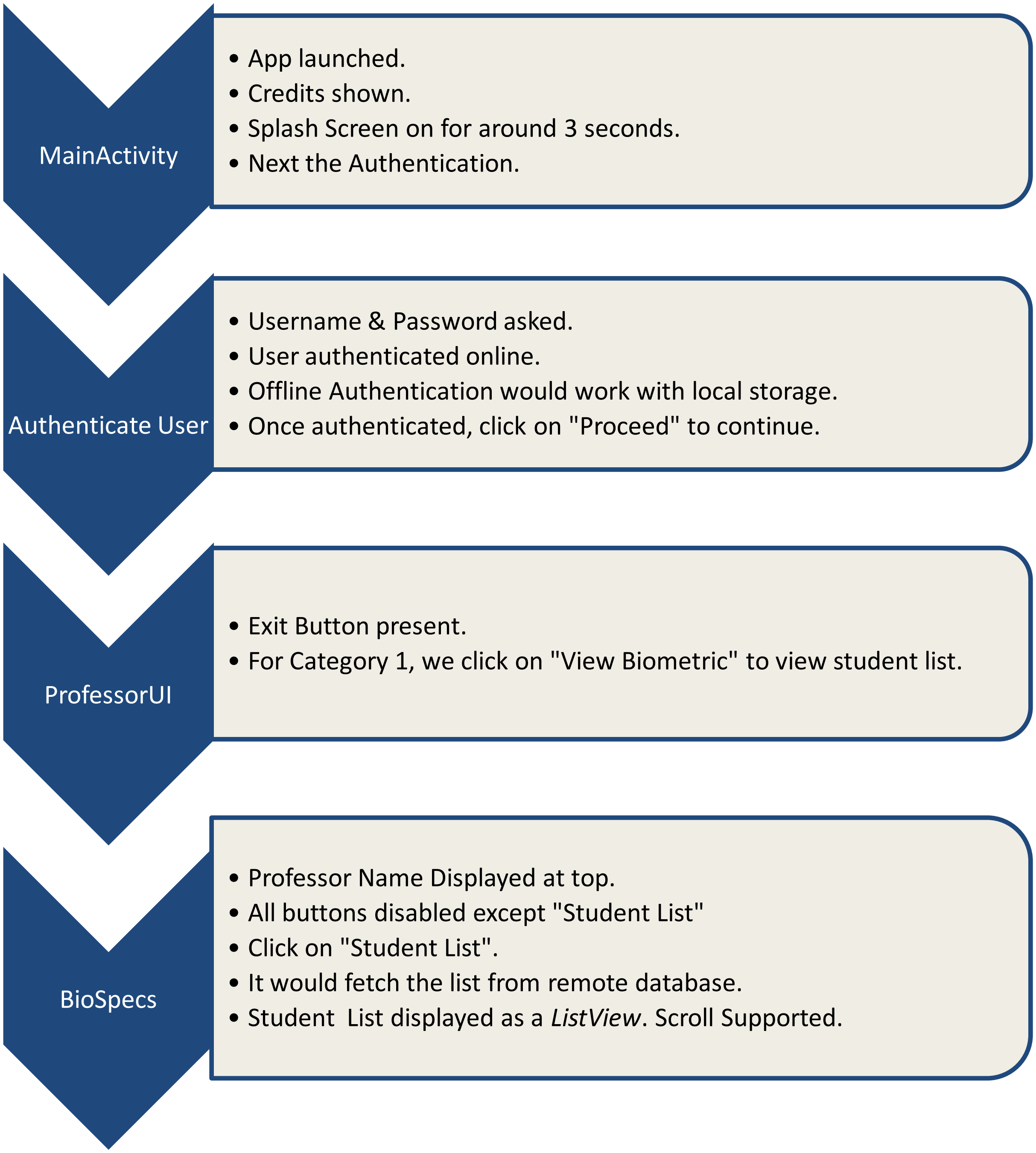
- The activity lists all the subjects being taken by the professor.
- On clicking any item, that subject specific menu appears.

NB. : Al the * listed activities/classes are exclusive to B. Tech attendance system. For now, the Category 1 & Category 2 systems have not been integrated into one.

3.3. Work Flow Sequence

To view the list of students under the professor's supervision, simply launch the application and follow the steps below to configure first launch.

Figure 3-1 : Viewing List of Student List Under Supervision



MainActivity

- App launched.
- Credits shown.
- Splash Screen on for around 3 seconds.
- Next the Authentication.

Authenticate User

- Username & Password asked.
- User authenticated online.
- Offline Authentication would work with local storage.
- Once authenticated, click on "Proceed" to continue.

ProfessorUI

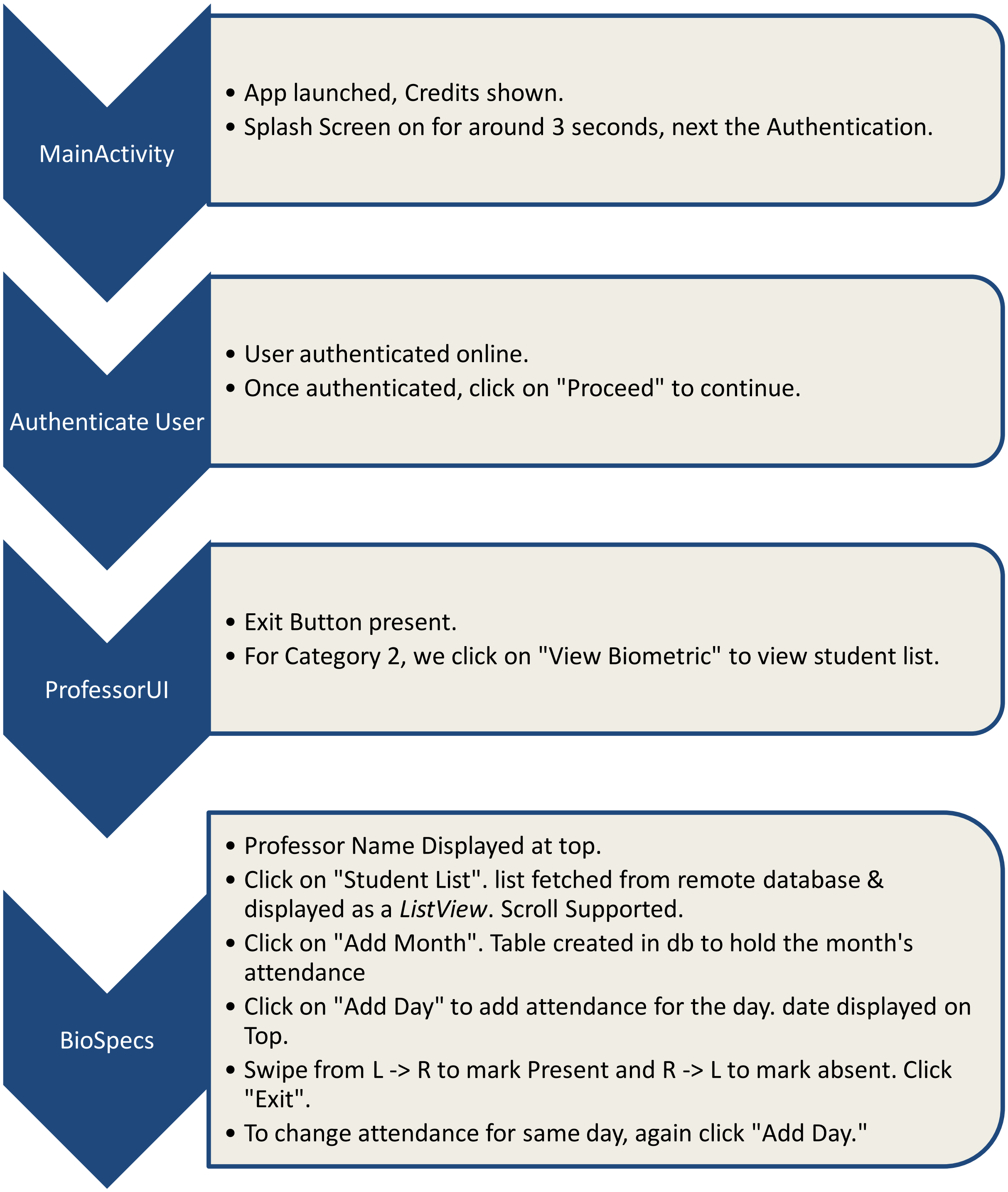
- Exit Button present.
- For Category 1, we click on "View Biometric" to view student list.

BioSpecs

- Professor Name Displayed at top.
- All buttons disabled except "Student List"
- Click on "Student List".
- It would fetch the list from remote database.
- Student List displayed as a *ListView*. Scroll Supported.

Here we are interested to implement the way attendance for a Category 2 student is taken i.e. how we are implementing storage of attendance day wise.

Figure 3-2 : Workflow for Attendance of Category 2 Students



MainActivity

- App launched, Credits shown.
- Splash Screen on for around 3 seconds, next the Authentication.

Authenticate User

- User authenticated online.
- Once authenticated, click on "Proceed" to continue.

ProfessorUI

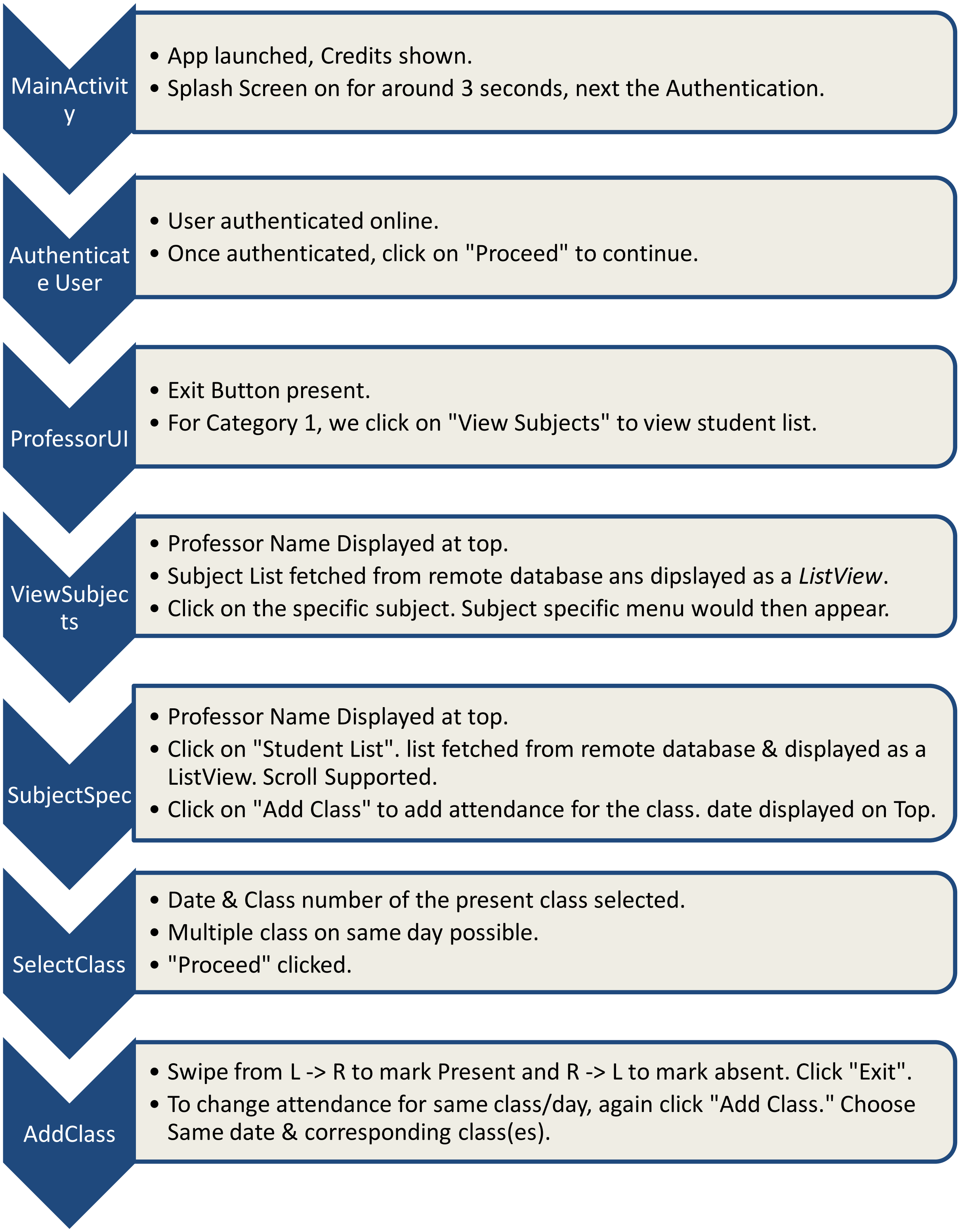
- Exit Button present.
- For Category 2, we click on "View Biometric" to view student list.

BioSpecs

- Professor Name Displayed at top.
- Click on "Student List". list fetched from remote database & displayed as a *ListView*. Scroll Supported.
- Click on "Add Month". Table created in db to hold the month's attendance
- Click on "Add Day" to add attendance for the day. date displayed on Top.
- Swipe from L -> R to mark Present and R -> L to mark absent. Click "Exit".
- To change attendance for same day, again click "Add Day."

Here we are interested to implement the way attendance for a Category 1 student is taken i.e. how we are implementing storage of attendance class wise.

Figure 3-3 : Workflow for Attendance of Category 1 Students



- App launched, Credits shown.
- Splash Screen on for around 3 seconds, next the Authentication.

- User authenticated online.
- Once authenticated, click on "Proceed" to continue.

- Exit Button present.
- For Category 1, we click on "View Subjects" to view student list.

- Professor Name Displayed at top.
- Subject List fetched from remote database ans dipslayed as a *ListView*.
- Click on the specific subject. Subject specific menu would then appear.

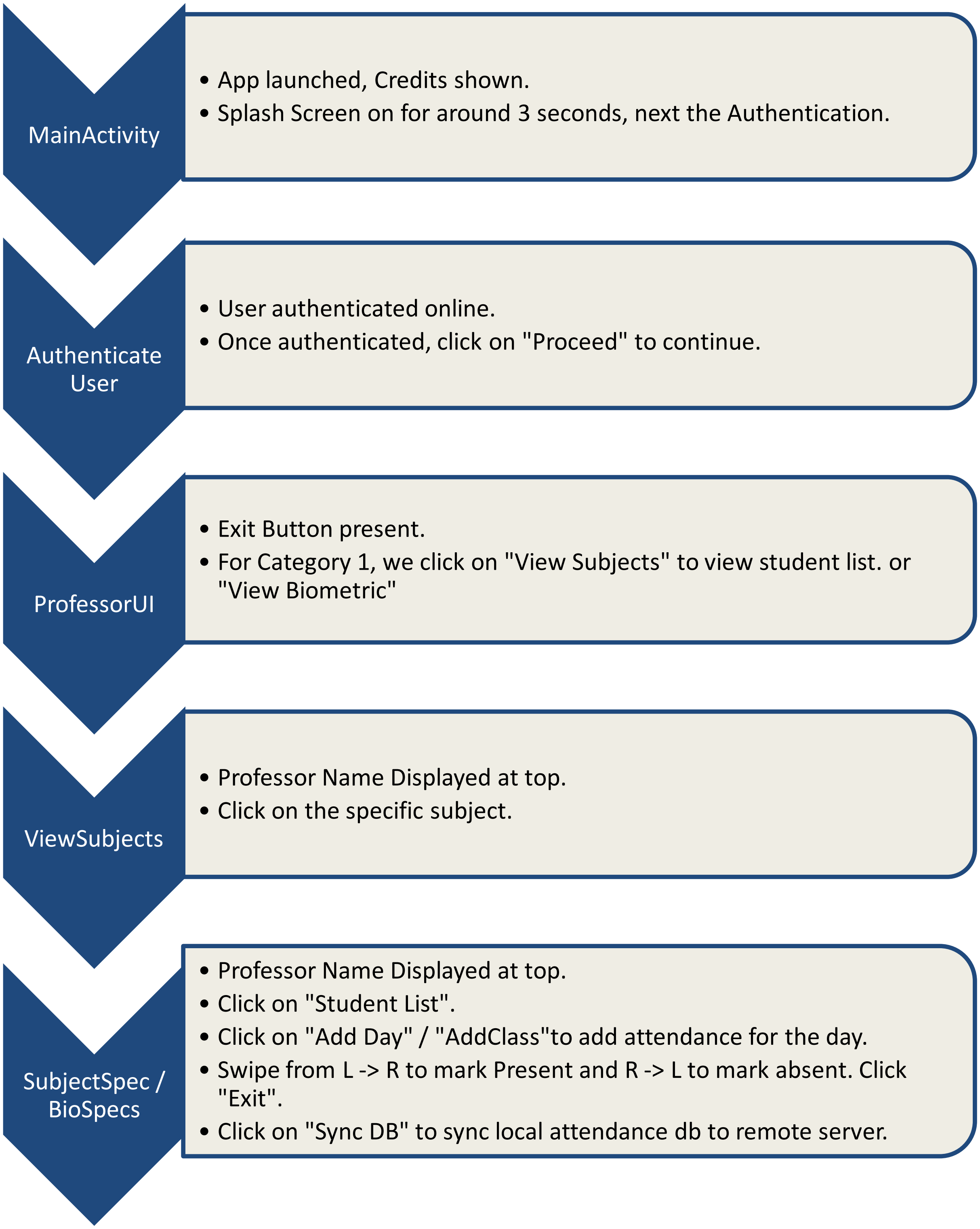
- Professor Name Displayed at top.
- Click on "Student List". list fetched from remote database & displayed as a ListView. Scroll Supported.
- Click on "Add Class" to add attendance for the class. date displayed on Top.

- Date & Class number of the present class selected.
- Multiple class on same day possible.
- "Proceed" clicked.

- Swipe from L -> R to mark Present and R -> L to mark absent. Click "Exit".
- To change attendance for same class/day, again click "Add Class." Choose Same date & corresponding class(es).

Now, we see how to sync local attendance database to remote server for all categories of students.

Figure 3-4 : Workflow for Syncing Local DB to Remote Server



MainActivity

- App launched, Credits shown.
- Splash Screen on for around 3 seconds, next the Authentication.

Authenticate User

- User authenticated online.
- Once authenticated, click on "Proceed" to continue.

ProfessorUI

- Exit Button present.
- For Category 1, we click on "View Subjects" to view student list. or "View Biometric"

ViewSubjects

- Professor Name Displayed at top.
- Click on the specific subject.

SubjectSpec / BioSpecs

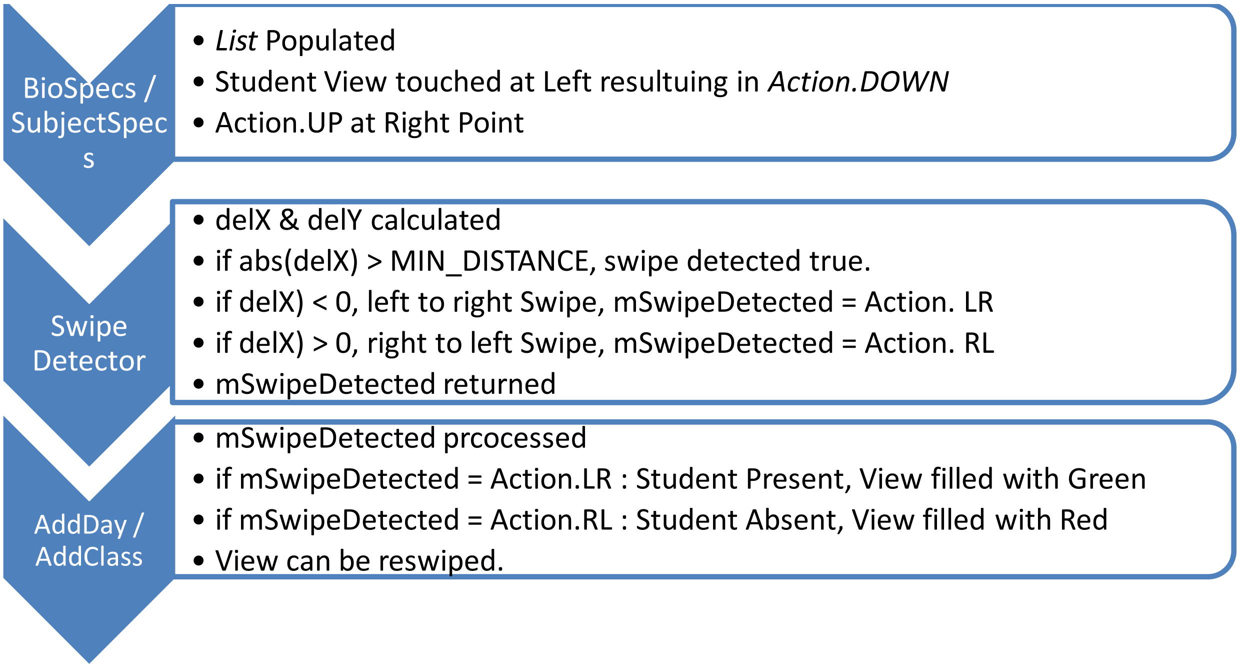
- Professor Name Displayed at top.
- Click on "Student List".
- Click on "Add Day" / "AddClass"to add attendance for the day.
- Swipe from L -> R to mark Present and R -> L to mark absent. Click "Exit".
- Click on "Sync DB" to sync local attendance db to remote server.

3.4. Swipe Detector Module Analysis

Swipe was used to make the attendance application more lucrative. Used on a student list, each student name is enclosed within mini views. So swiping action occurs on a single view. Actually our module extends the *View.onTouchListener* class and modifies the *onTouch()* method so as to recognize a elongated touch as a swipe.

- So, the class SwipeDetector.java implements the *onTouchListener* library.
- Since four swiping actions are detectable : left to right, right to left, top to bottom & bottom to top, we define a *enum* datatype "Action". Enum is a set of predefined constraints. "Action" has four constraints defined: LR, RL, TB & BT.
- The min. distance swiped is defined as MIN_DISTANCE, the *onTouch()* method extracts the distance between the *ACTION.DOWN&ACTION.UP* touches it. It processes the same & finds out which swipe is detected with information about distance covered in both X and Y coordinates.
- *swipeDetected()* returns true if any of the four swipe is detected.
- *leftSwipeDetected()* returns true if a left to right swipe or Action.LR is detected.
- *rightSwipeDetected()* returns true if a right to left swipe or Action.RL is detected.
- *getAction()* returns the type of swipe detected i.e. it returns any one of the following: Action.LR, Action.RL, Action.TB & Action.BT.
- *returnTouchTime()* returns the duration of touch felt by the screen.

Figure 3-5 : Workflow for Swipe Detection



BioSpecs / SubjectSpecs

- *List* Populated
- Student View touched at Left resultuing in *Action.DOWN*
- Action.UP at Right Point

Swipe Detector

- delX & delY calculated
- if abs(delX) > MIN_DISTANCE, swipe detected true.
- if delX) < 0, left to right Swipe, mSwipeDetected = Action. LR
- if delX) > 0, right to left Swipe, mSwipeDetected = Action. RL
- mSwipeDetected returned

AddDay / AddClass

- mSwipeDetected prcocessed
- if mSwipeDetected = Action.LR : Student Present, View filled with Green
- if mSwipeDetected = Action.RL : Student Absent, View filled with Red
- View can be reswiped.

3.5. SQLite DBMS Module Analysis

In our application dealing with attendance management, we did deal with the use of database. But actually the DBMS compatible with android is *SQLite* contained in a small programming library of around 350 KB. It supports most of the standard SQL using a dynamically but weakly typed SQL syntax. It is hence a popular storage medium for use as embedded database. In case of android, to access any SQLite database we need a handler, an object of a class extending *SQLiteOpenHelper*. The following methods are added & defined in the class:

a. Constructor
   - Just calls the super function.

b. onCreate()
   - The method is called when the handler is instantiated the first time ever for the application.
   - It's better to move the Table Creation instructions out of the *onCreate()*

c. onUpgrade()
   - drops the table if created and re-calls the *onCreate()*

d. createTable()
   - Not a library method, it's a custom user defined method wherein a database object executes a *CREATE* instruction.

Apart from the above, the following need to be noted:

- For any transaction of data to or from the database or any of its tables, the handler must be opened i.e., the context should be assigned and a writable database should be extracted using *getWritableDatabase()* method.
- The class should have a method which should call the *createTable()* method of the helper.
- After all transactions and querying are complete, the database handler must be dereferenced or closed.
- For creation of rows or entry of data, values are encoded in *ContentValue* type entities.
- Any query instruction returns bundled data embedded in a *Cursor* type entity.. First, the column indexes are extracted using *getColumnIndex()* method and then using *getString()* method, the corresponding values are extracted.
- Any SQL statement can be executed using *execSQL()* method.

Figure 3-6 : Creation of Table in SQLite Database in Android

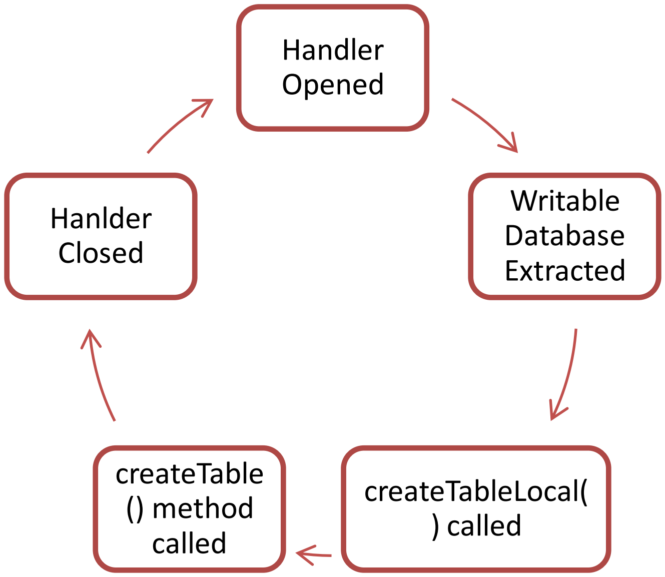

Handler Opened
Writable Database Extracted
createTableLocal( ) called
createTable () method called
Hanlder Closed

Figure 3-7 : Data Entry into Table in SQLite Database in Android

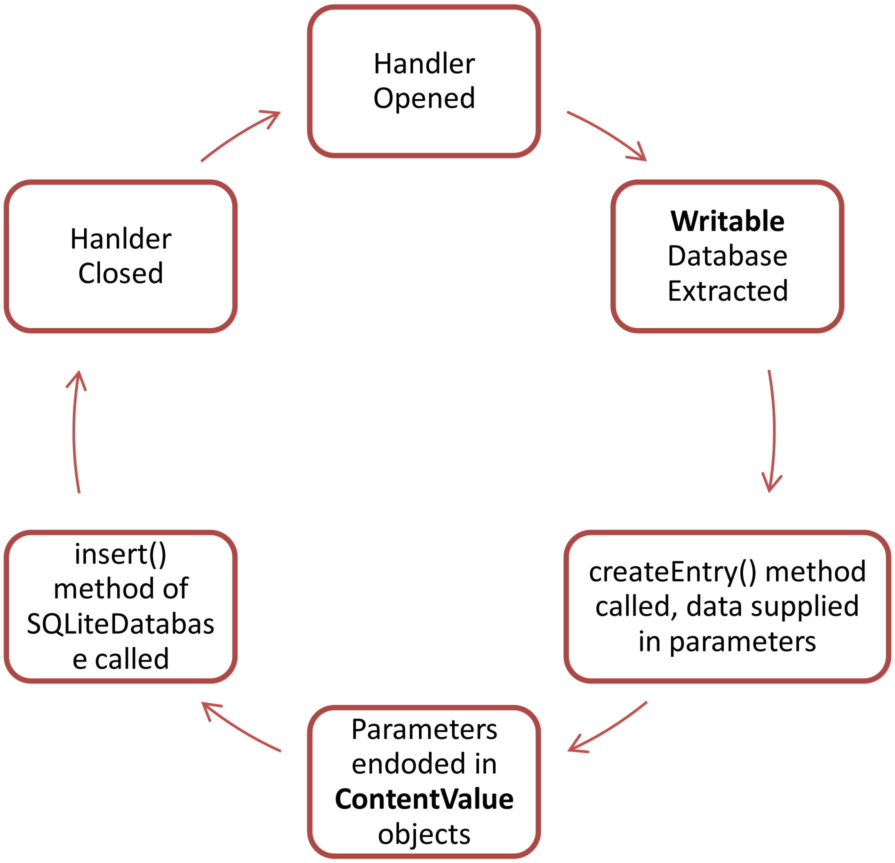
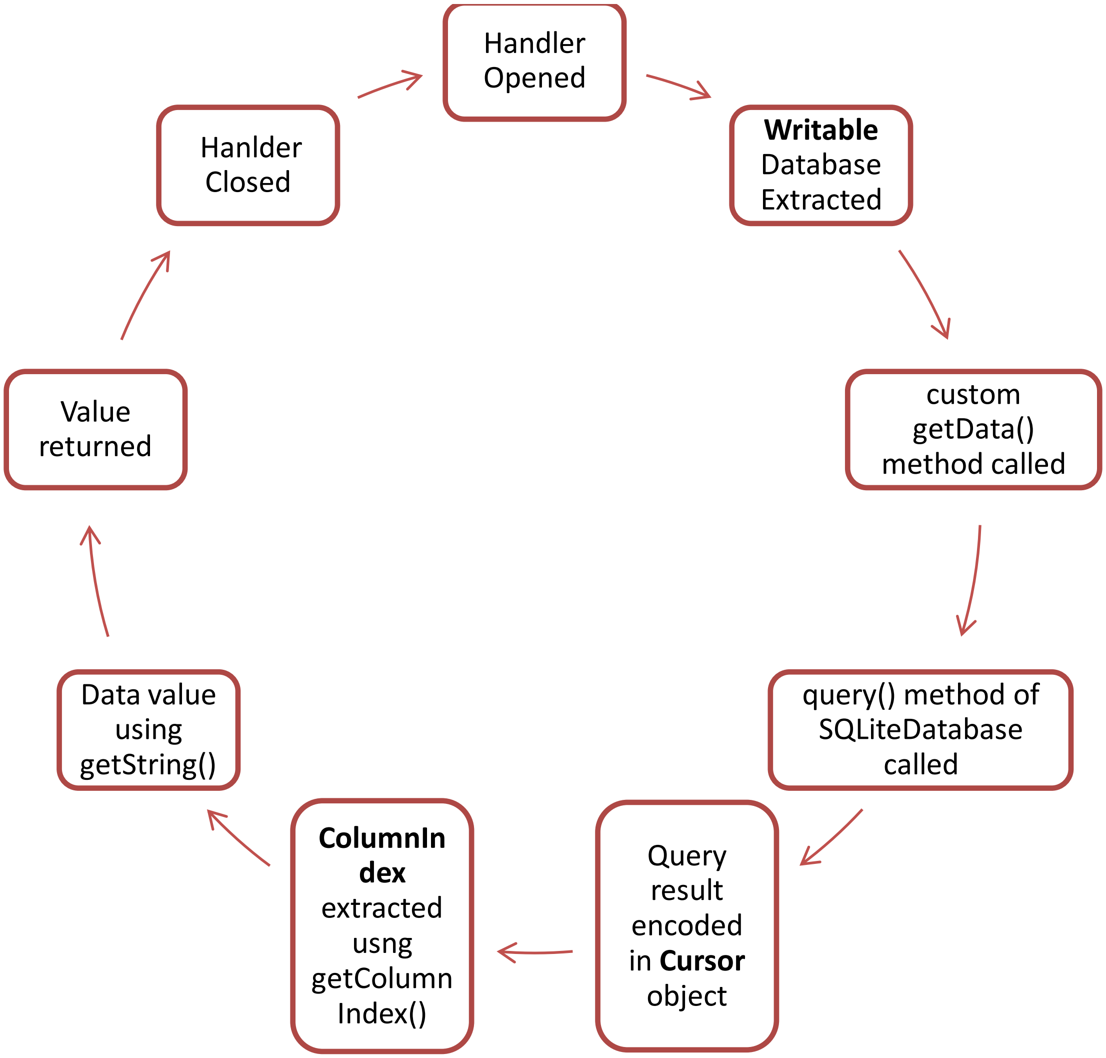

Handler Opened
Writable Database Extracted
createEntry() method called, data supplied in parameters
Parameters endoded in **ContentValue** objects
insert() method of SQLiteDatabas e called
Hanlder Closed

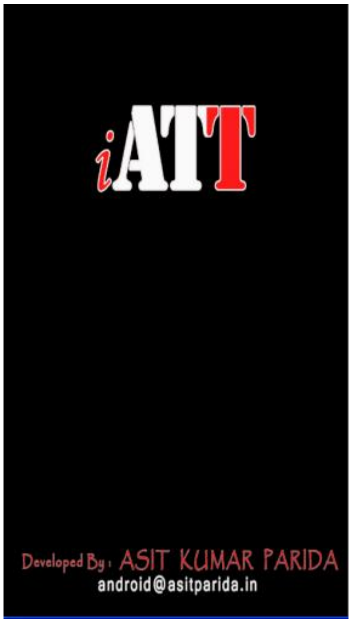
Figure 3-8 : Data Query from Table in SQLite Database in Android


Handler Opened
Writable Database Extracted
custom getData() method called
query() method of SQLiteDatabase called
Query result encoded in **Cursor** object
ColumnIn dex extracted usng getColumn Index()
Data value using getString()
Value returned
Hanlder Closed

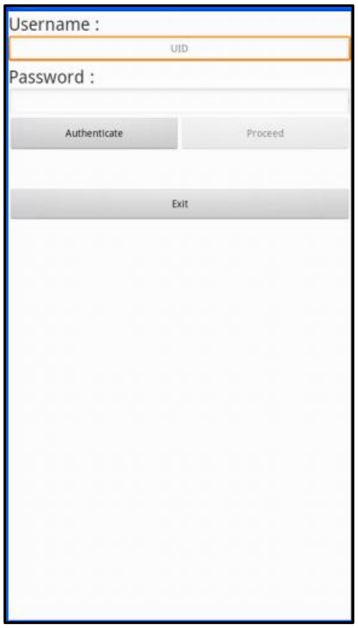
3.6. Testing: UI Screenshots

3.6.1. Attendance Management System (Category 1 Students)

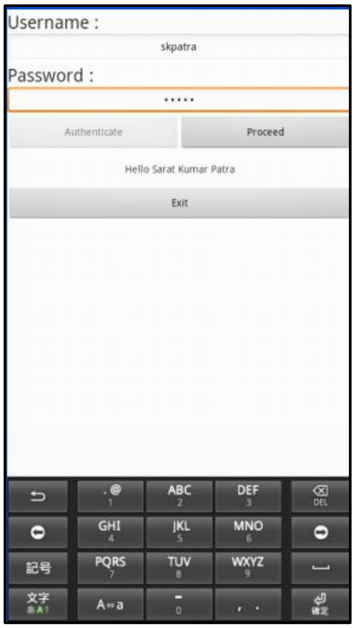
Splash Screen On StartUp



Authentication Screen



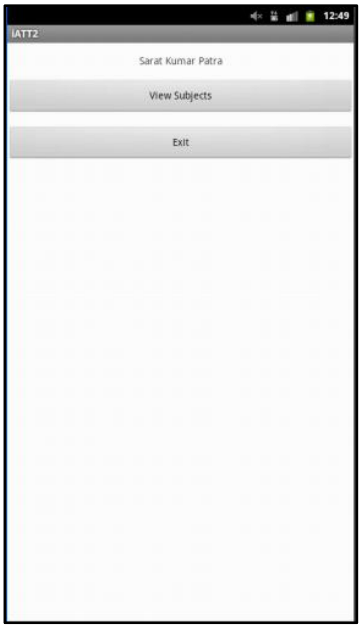
Authentication Successful


Username : skpatra

Password : •••••

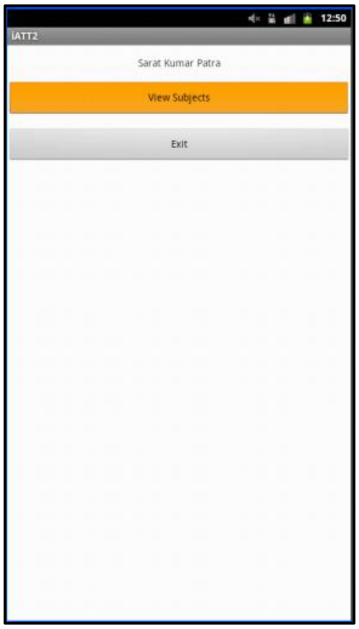
Hello Sarat Kumar Patra

Professor User Interface



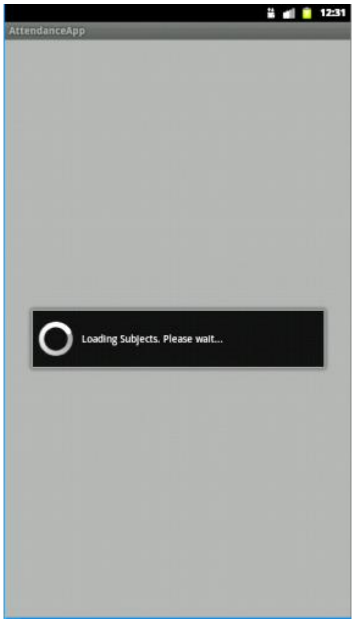
Sarat Kumar Patra

Button "View Subjects" Clicked



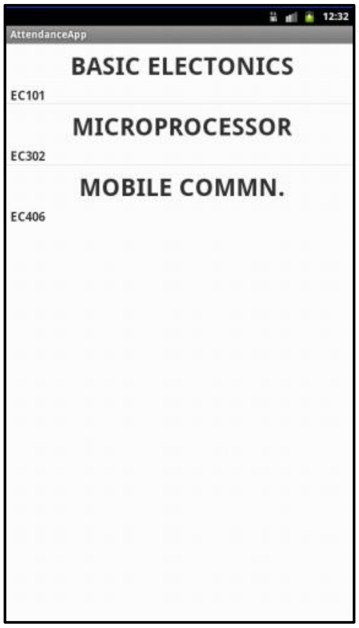
Sarat Kumar Patra

Subjects being Fetched From Remote Server



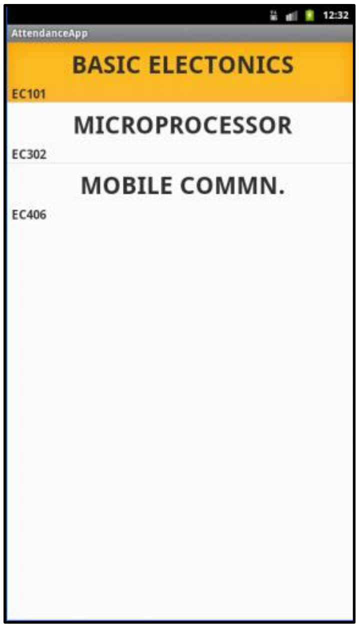
Loading Subjects. Please wait...

List Of Subjects Taken By Professor



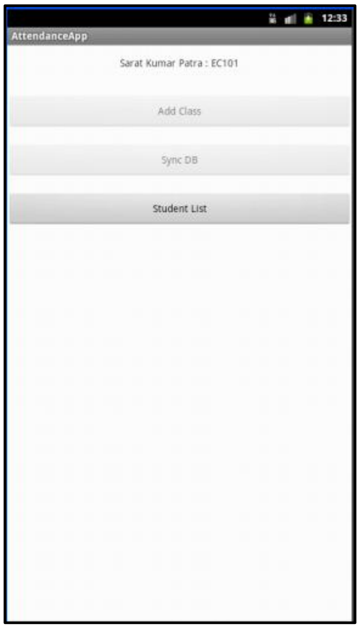
BASIC ELECTONICS
EC101
MICROPROCESSOR
EC302
MOBILE COMMN.
EC406

Selection of Particular Subject

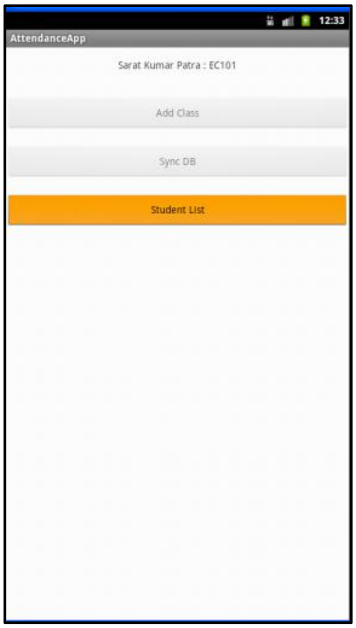


BASIC ELECTONICS
EC101
MICROPROCESSOR
EC302
MOBILE COMMN.
EC406

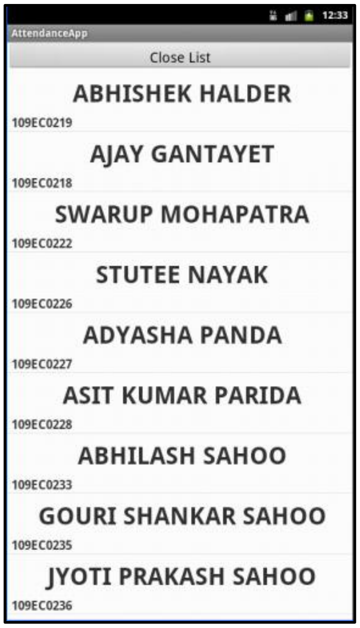
Subject Specific Screen Opens



Student List for Subject Being Fetched for First Time



Student List For Subject Displayed



ABHISHEK HALDER
109EC0219

AJAY GANTAYET
109EC0218

SWARUP MOHAPATRA
109EC0222

STUTEE NAYAK
109EC0226

ADYASHA PANDA
109EC0227

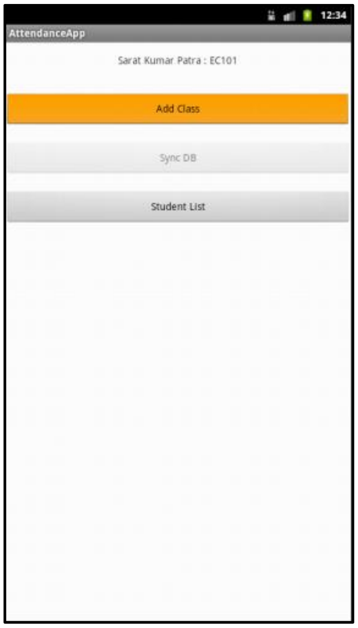
ASIT KUMAR PARIDA
109EC0228

ABHILASH SAHOO
109EC0233

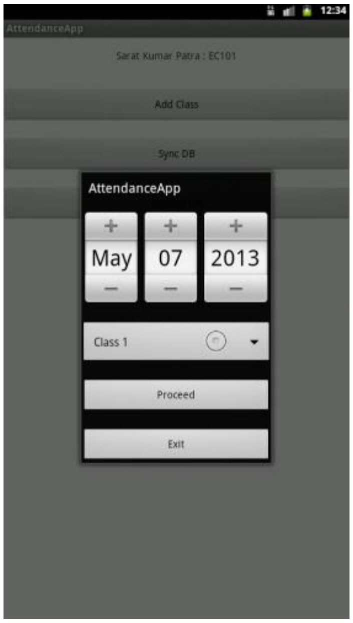
GOURI SHANKAR SAHOO
109EC0235

JYOTI PRAKASH SAHOO
109EC0236

Adding Class for Attendance of One Class



Date & Class Selection

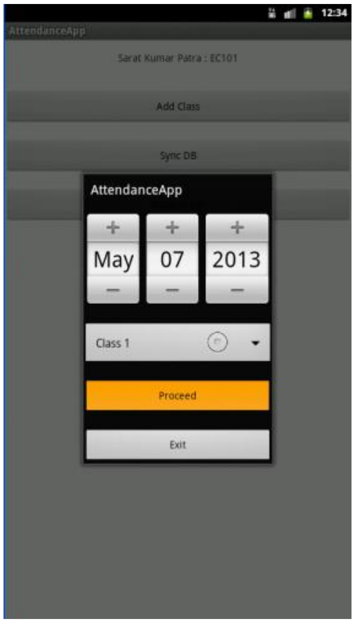


Sarat Kumar Patra : EC101

May 07 2013

Class 1

Date & Class Selected and We Proceed

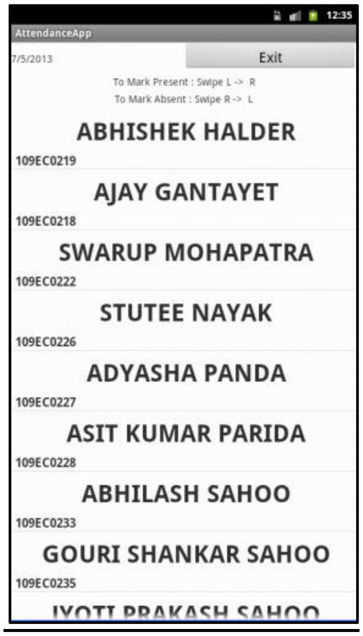


Sarat Kumar Patra : EC101

May 07 2013

Class 1

List Populated, Date Displayed. Ready for Swiping,



7/5/2013

To Mark Present : Swipe L -> R
To Mark Absent : Swipe R -> L

ABHISHEK HALDER
109EC0219

AJAY GANTAYET
109EC0218

SWARUP MOHAPATRA
109EC0222

STUTEE NAYAK
109EC0226

ADYASHA PANDA
109EC0227

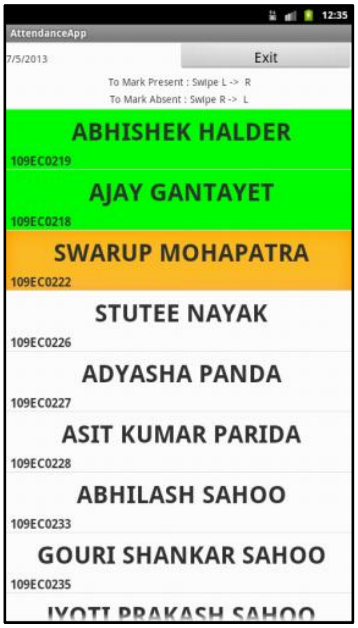
ASIT KUMAR PARIDA
109EC0228

ABHILASH SAHOO
109EC0233

GOURI SHANKAR SAHOO
109EC0235

JYOTI PRAKASH SAHOO

Swiping Started (Yellow)
Green : Present, Red: Absent



7/5/2013

To Mark Present : Swipe L -> R
To Mark Absent : Swipe R -> L

ABHISHEK HALDER
109EC0219

AJAY GANTAYET
109EC0218

SWARUP MOHAPATRA
109EC0222

STUTEE NAYAK
109EC0226

ADYASHA PANDA
109EC0227

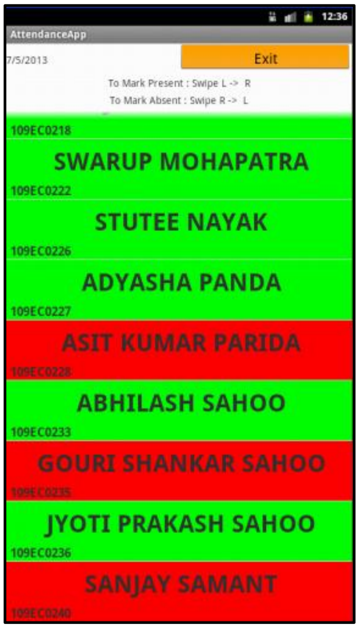
ASIT KUMAR PARIDA
109EC0228

ABHILASH SAHOO
109EC0233

GOURI SHANKAR SAHOO
109EC0235

JYOTI PRAKASH SAHOO

Swiping Completed, We Exit



12:36

AttendanceApp

7/5/2013 Exit

To Mark Present : Swipe L -> R

To Mark Absent : Swipe R -> L

109EC0218

SWARUP MOHAPATRA

109EC0222

STUTEE NAYAK

109EC0226

ADYASHA PANDA

109EC0227

ASIT KUMAR PARIDA

109EC0228

ABHILASH SAHOO

109EC0233

GOURI SHANKAR SAHOO

109EC0235

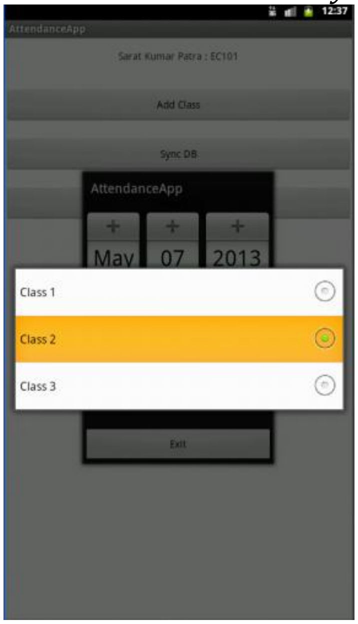
JYOTI PRAKASH SAHOO

109EC0236

SANJAY SAMANT

109EC0240

Changing Class, Taking Attendance for Second Class on Same day



12:37

AttendanceApp

Sarat Kumar Patra : EC101

Add Class

Sync DB

AttendanceApp

May 07 2013

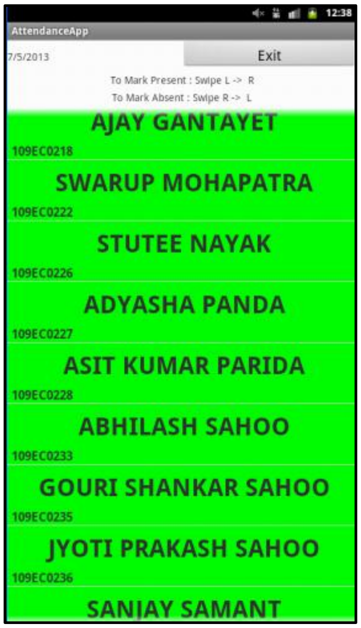
Class 1 ○

Class 2 ●

Class 3 ○

Exit

Attendance For Second Class Taken



12:38

AttendanceApp

7/5/2013 Exit

To Mark Present : Swipe L -> R

To Mark Absent : Swipe R -> L

AJAY GANTAYET

109EC0218

SWARUP MOHAPATRA

109EC0222

STUTEE NAYAK

109EC0226

ADYASHA PANDA

109EC0227

ASIT KUMAR PARIDA

109EC0228

ABHILASH SAHOO

109EC0233

GOURI SHANKAR SAHOO

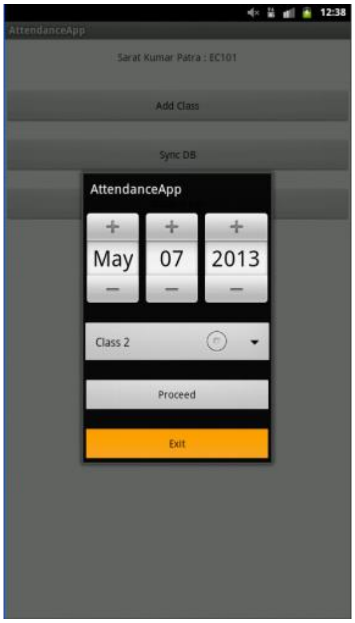
109EC0235

JYOTI PRAKASH SAHOO

109EC0236

SANJAY SAMANT

We Exit From Date & Class Selection



12:38

AttendanceApp

Sarat Kumar Patra : EC101

Add Class

Sync DB

AttendanceApp

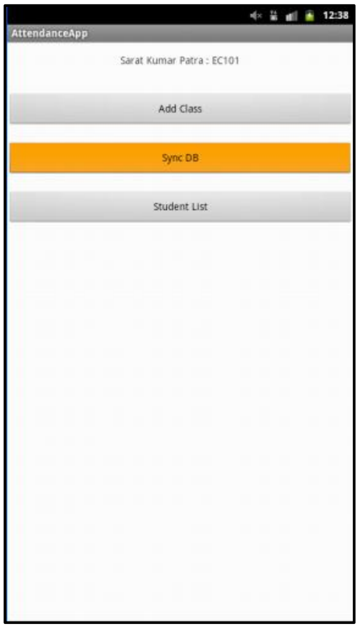
May 07 2013

Class 2 ○

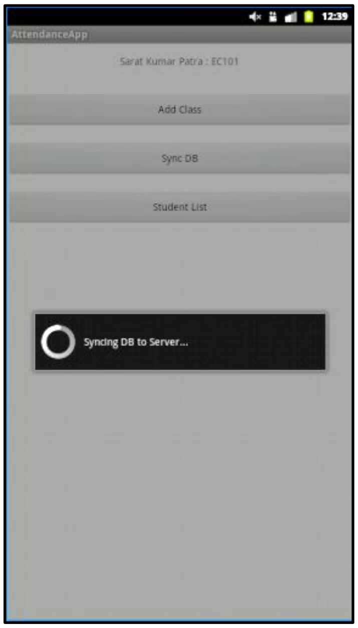
Proceed

Exit

“Sync” to sync local DB to Server



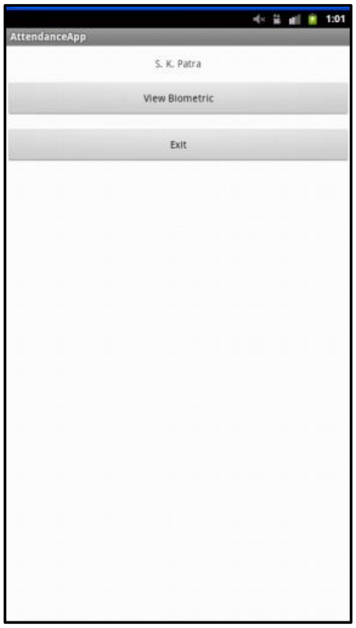
Syncing in Progress



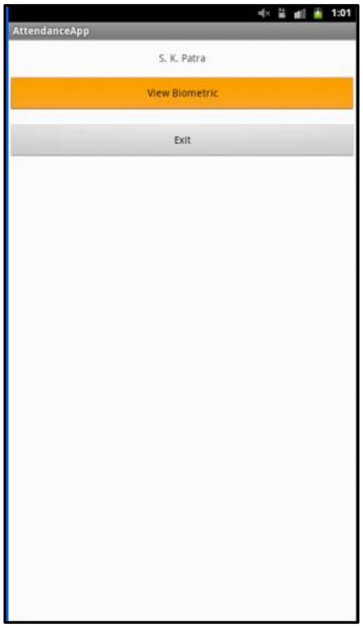
Syncing DB to Server...

3.6.2. **Attendance Management System (Category 2 Students)**

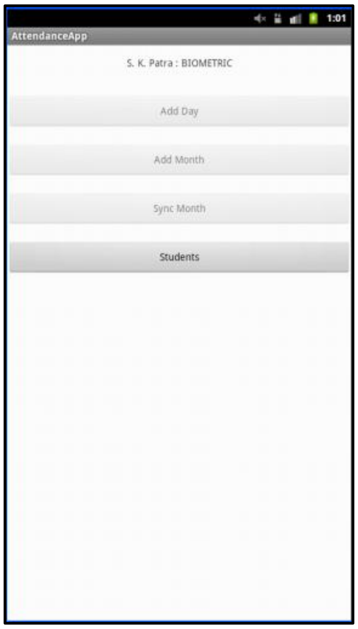
Biometric UI For Professors



“View Biometric” Clicked



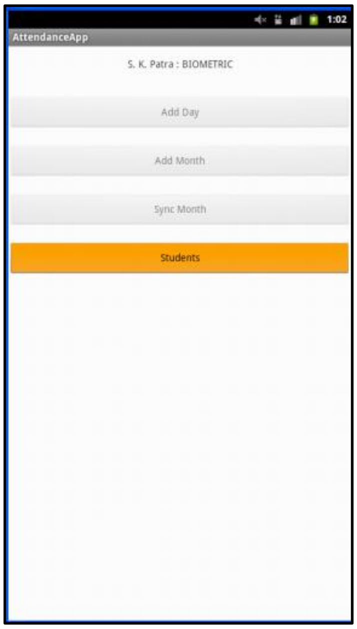
Biometric UI Opened



AttendanceApp

S. K. Patra : BIOMETRIC

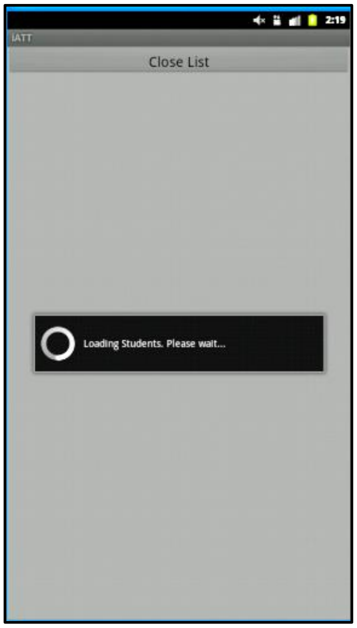
Student List for Subject for First Time



AttendanceApp

S. K. Patra : BIOMETRIC

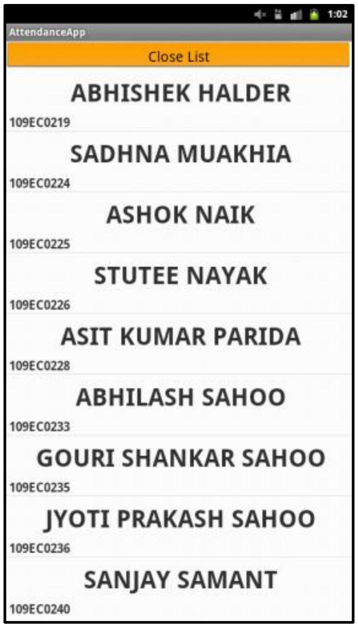
List of Student Under Supervision being Fetched



IATT

Loading Students. Please wait...

List Populated & Closed



AttendanceApp

ABHISHEK HALDER
109EC0219

SADHNA MUAKHIA
109EC0224

ASHOK NAIK
109EC0225

STUTEE NAYAK
109EC0226

ASIT KUMAR PARIDA
109EC0228

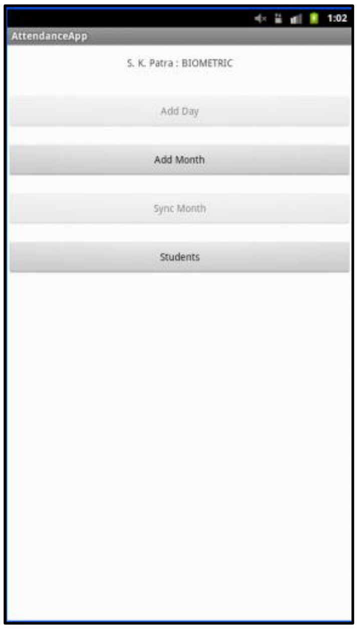
ABHILASH SAHOO
109EC0233

GOURI SHANKAR SAHOO
109EC0235

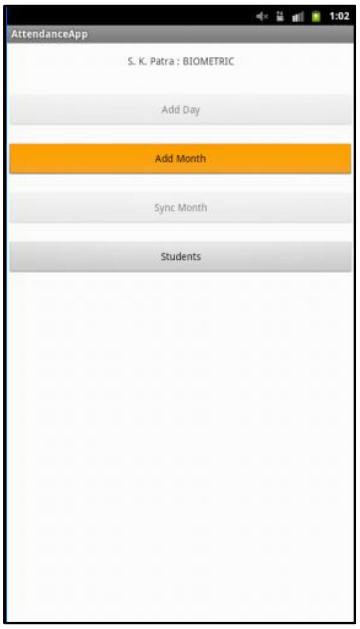
JYOTI PRAKASH SAHOO
109EC0236

SANJAY SAMANT
109EC0240

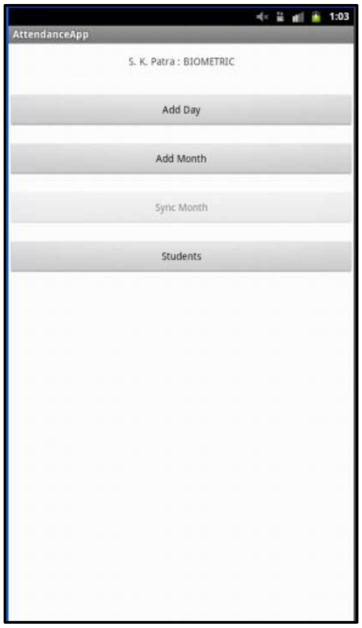
Add Month Button Activated



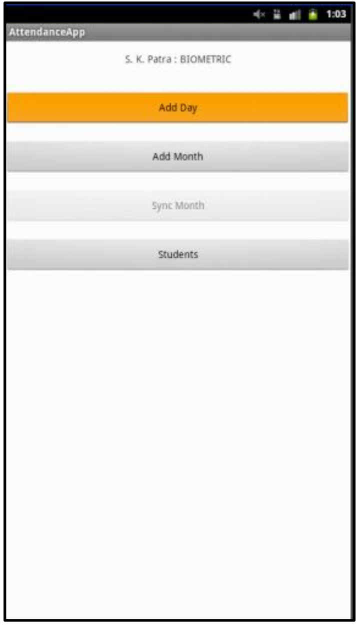
Add Month before taking Attendance



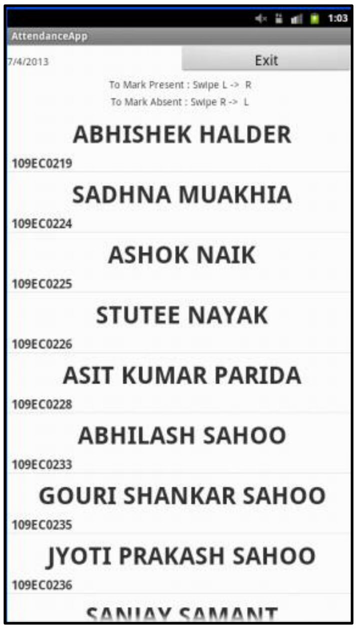
Add Day Button Activated



Add Day Clicked to Take Attendance For the day



Student List Populated



7/4/2013

To Mark Present : Swipe L -> R
To Mark Absent : Swipe R -> L

ABHISHEK HALDER
109EC0219

SADHNA MUAKHIA
109EC0224

ASHOK NAIK
109EC0225

STUTEE NAYAK
109EC0226

ASIT KUMAR PARIDA
109EC0228

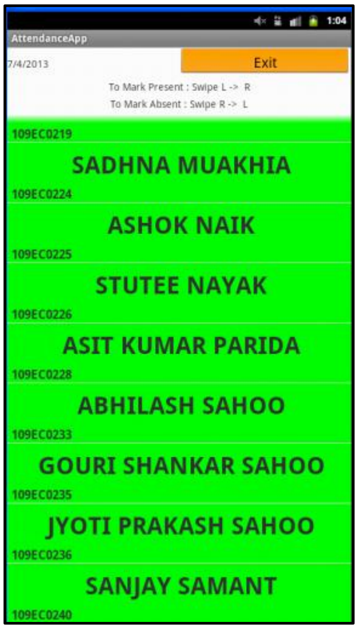
ABHILASH SAHOO
109EC0233

GOURI SHANKAR SAHOO
109EC0235

JYOTI PRAKASH SAHOO
109EC0236

SANJAY SAMANT

Normal Swiping Process for Attendance



7/4/2013

To Mark Present : Swipe L -> R
To Mark Absent : Swipe R -> L

109EC0219

SADHNA MUAKHIA
109EC0224

ASHOK NAIK
109EC0225

STUTEE NAYAK
109EC0226

ASIT KUMAR PARIDA
109EC0228

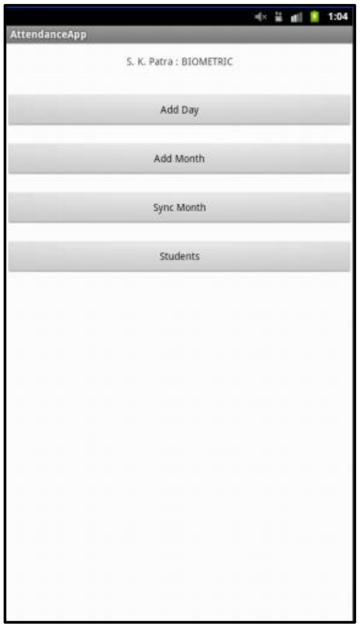
ABHILASH SAHOO
109EC0233

GOURI SHANKAR SAHOO
109EC0235

JYOTI PRAKASH SAHOO
109EC0236

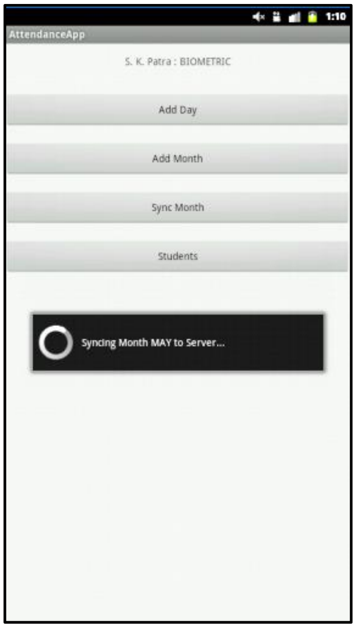
SANJAY SAMANT
109EC0240

Sync Button Activated



S. K. Patra : BIOMETRIC

Syncing in Progress



S. K. Patra : BIOMETRIC

Syncing Month MAY to Server...

3.7. Testing: DB Screenshots

Main Database For B Tech

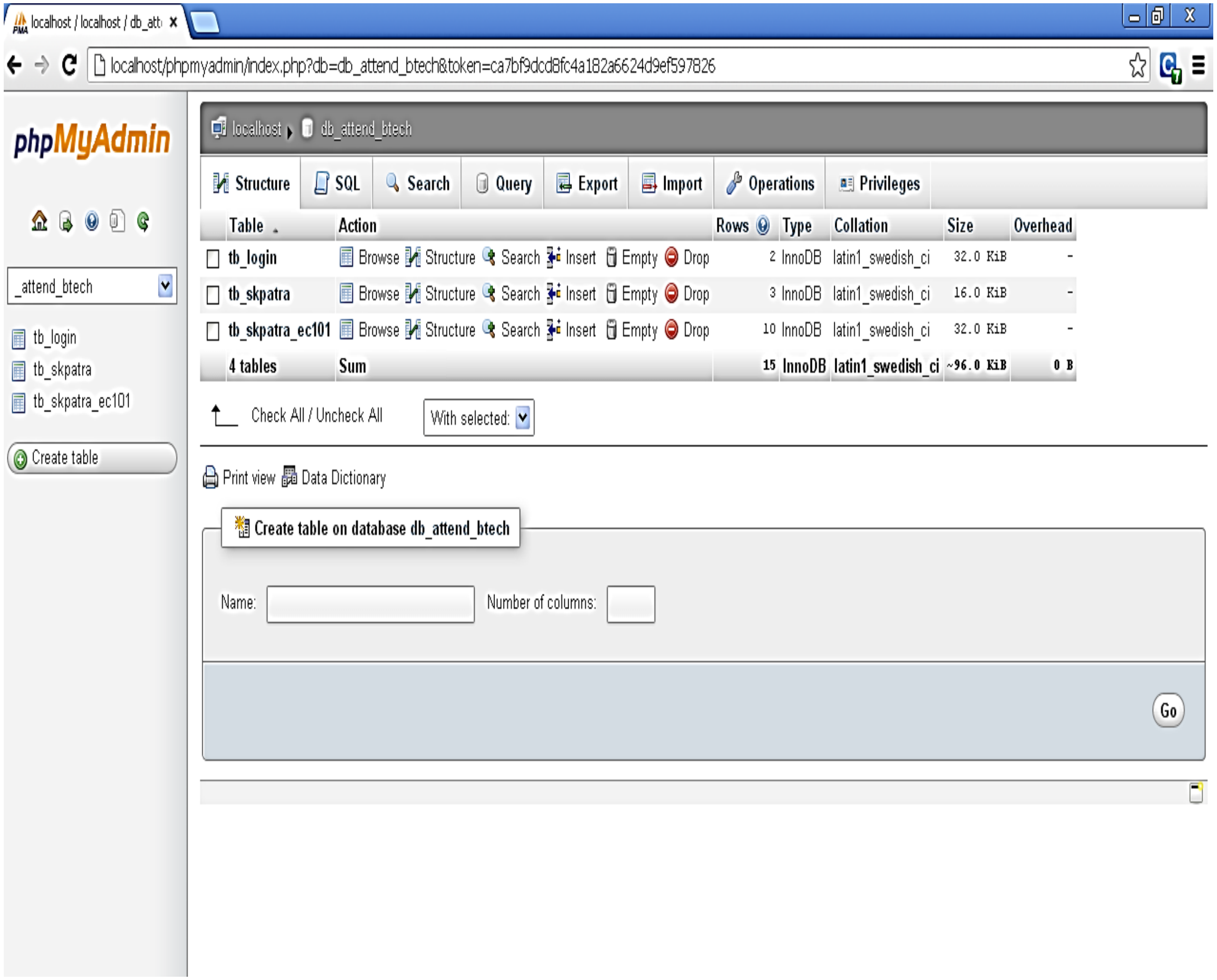
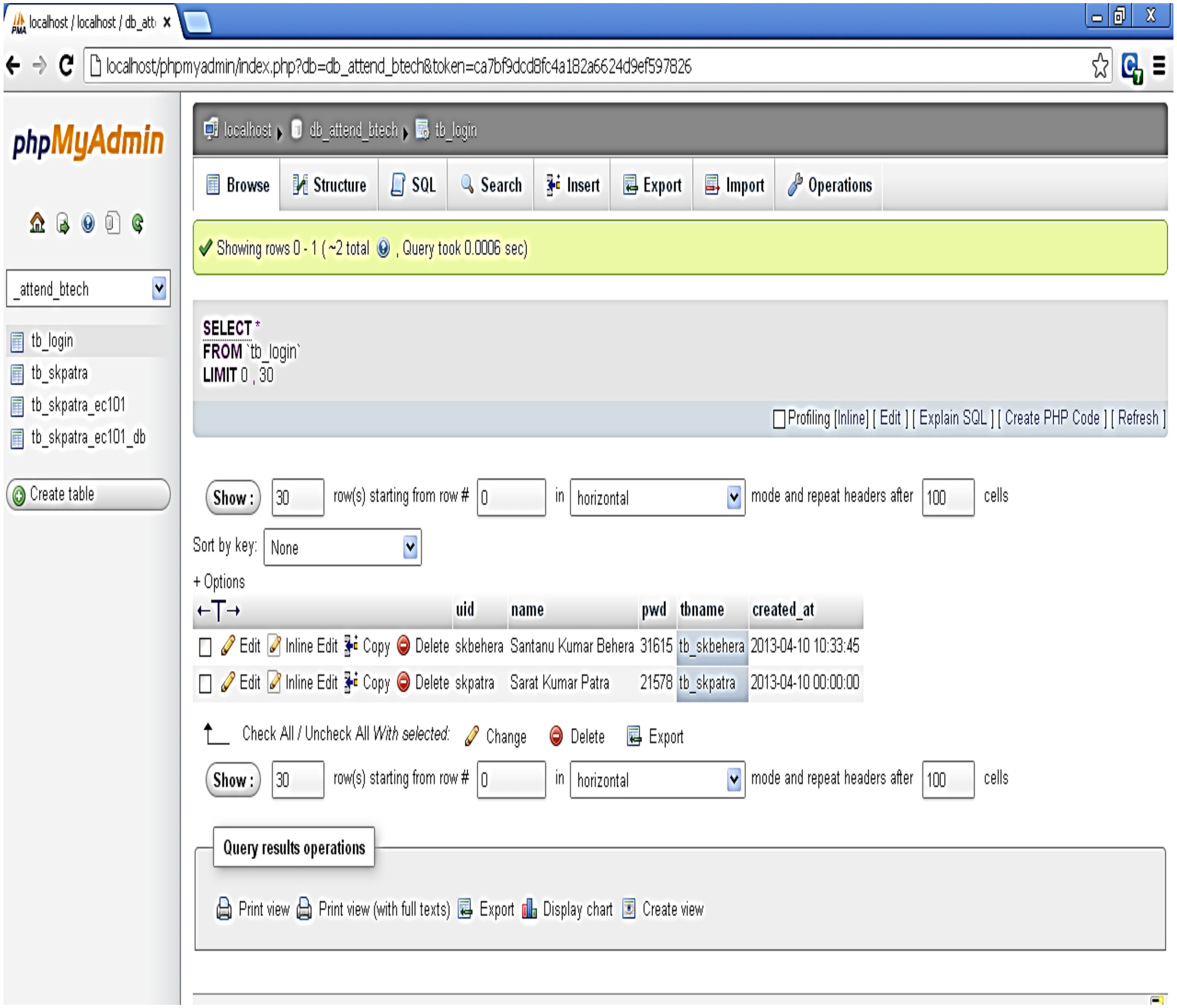


Table	Action	Rows	Type	Collation	Size	Overhead
☐ tb_login	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	32.0 KiB	-
☐ tb_skpatra	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16.0 KiB	-
☐ tb_skpatra_ec101	Browse Structure Search Insert Empty Drop	10	InnoDB	latin1_swedish_ci	32.0 KiB	-
4 tables	Sum	15	InnoDB	latin1_swedish_ci	~96.0 KiB	0 B

Table With Log In Information



localhost / localhost / db_att
localhost/phpmyadmin/index.php?db=db_attend_btech&token=ca7bf9dcd8fc4a182a6624d9ef597826
phpMyAdmin
_attend_btech
tb_login
tb_skpatra
tb_skpatra_ec101
tb_skpatra_ec101_db
Create table
localhost ▸ db_attend_btech ▸ tb_login
Browse
Structure
SQL
Search
Insert
Export
Import
Operations

✓ Showing rows 0 - 1 (~2 total , Query took 0.0006 sec)

```
SELECT *
FROM `tb_login`
LIMIT 0 , 30
```

□ Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

Sort by key: None

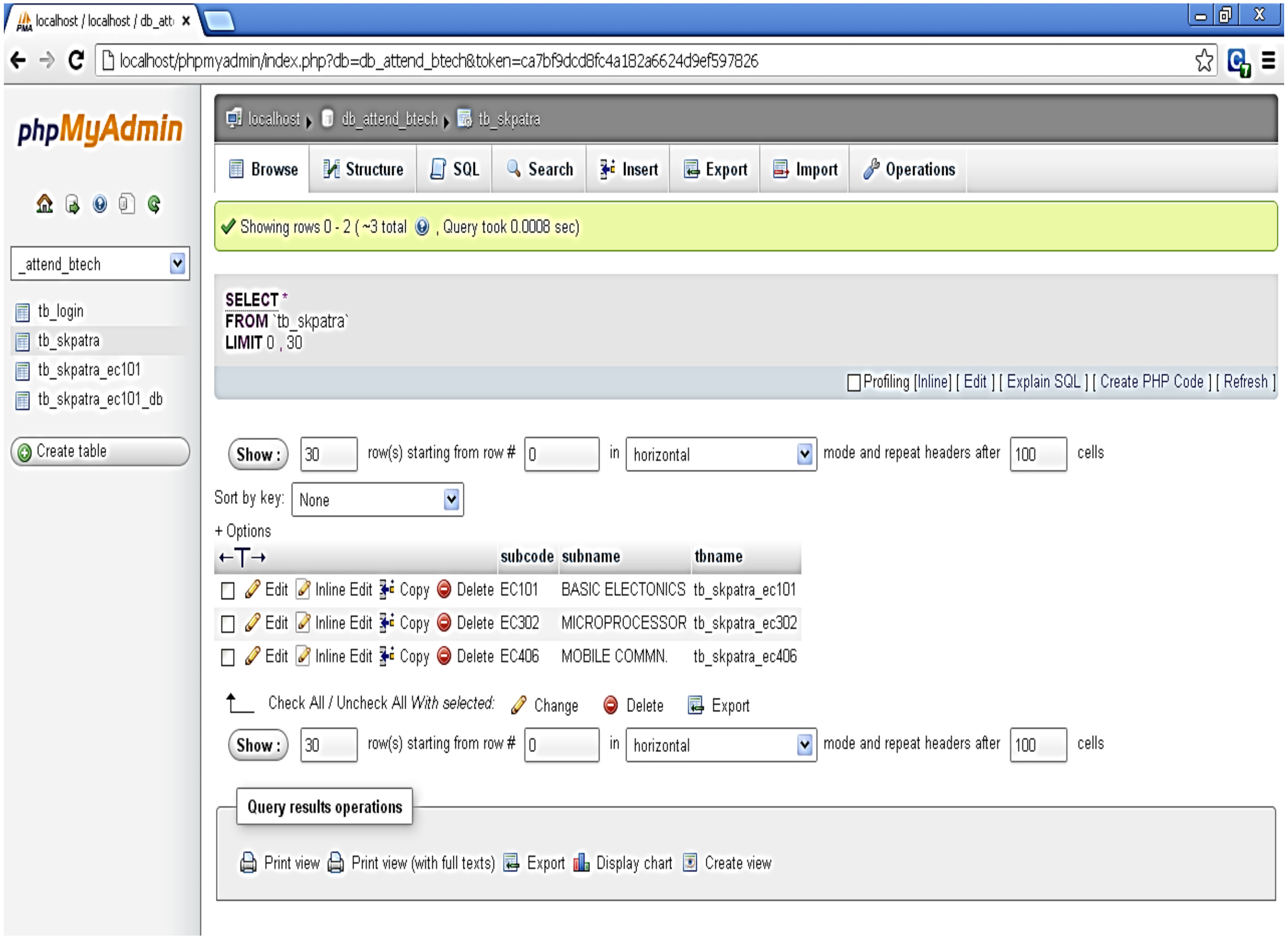
+ Options

←T→	uid	name	pwd	tbname	created_at
□ Edit Inline Edit Copy Delete	skbehera	Santanu Kumar Behera	31615	tb_skbehera	2013-04-10 10:33:45
□ Edit Inline Edit Copy Delete	skpatra	Sarat Kumar Patra	21578	tb_skpatra	2013-04-10 00:00:00

Check All / Uncheck All *With selected:* Change Delete Export

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

Table Showing List of Subjects


localhost / localhost / db_att
localhost/phpmyadmin/index.php?db=db_attend_btech&token=ca7bf9dcd8fc4a182a6624d9ef597826
phpMyAdmin
_attend_btech
tb_login
tb_skpatra
tb_skpatra_ec101
tb_skpatra_ec101_db
Create table
localhost ▸ db_attend_btech ▸ tb_skpatra
Browse
Structure
SQL
Search
Insert
Export
Import
Operations

✓ Showing rows 0 - 2 (~3 total , Query took 0.0008 sec)

```
SELECT *
FROM `tb_skpatra`
LIMIT 0 , 30
```

□ Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

Sort by key: None

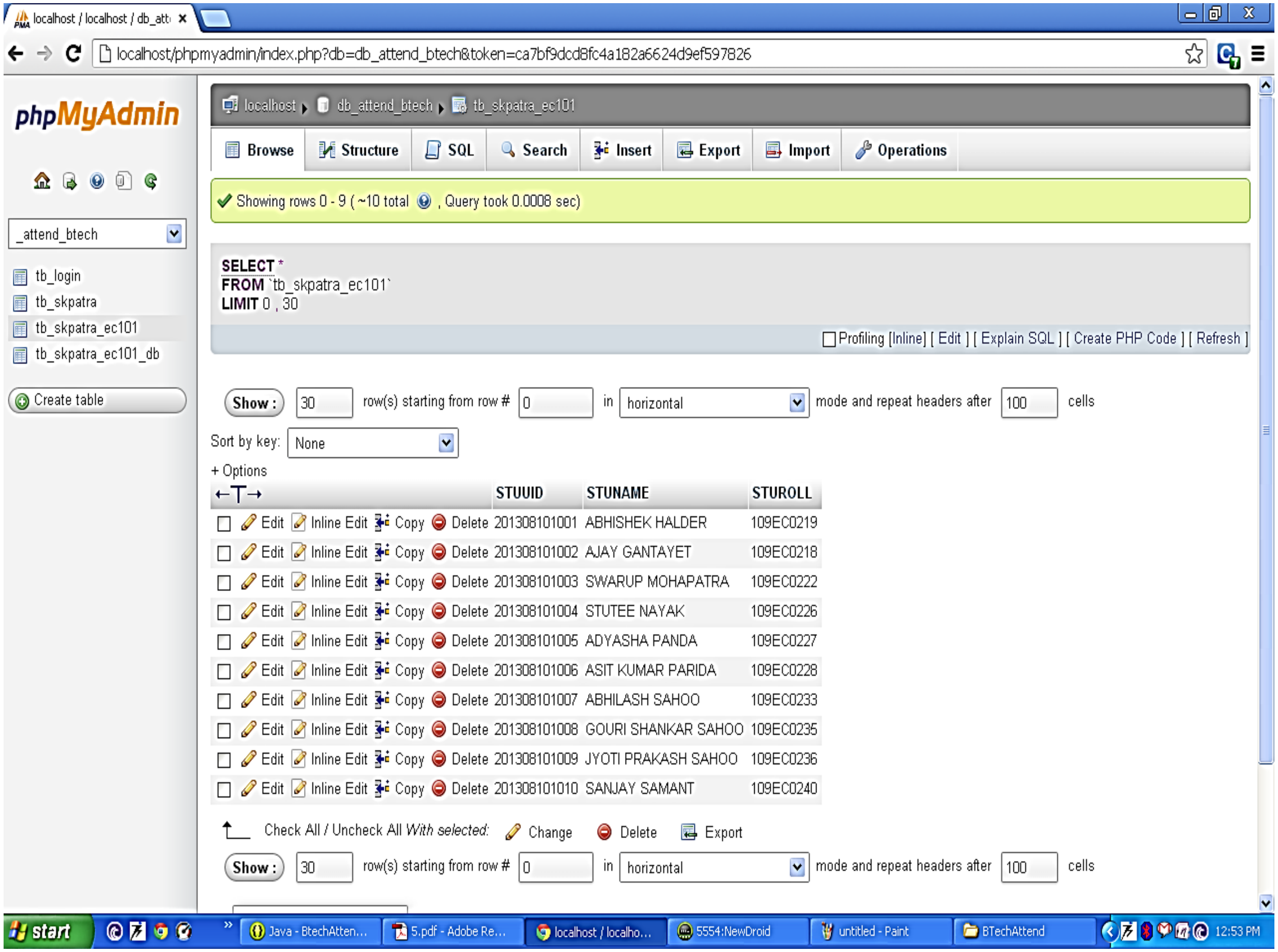
+ Options

←T→	subcode	subname	tbname
□ Edit Inline Edit Copy Delete	EC101	BASIC ELECTONICS	tb_skpatra_ec101
□ Edit Inline Edit Copy Delete	EC302	MICROPROCESSOR	tb_skpatra_ec302
□ Edit Inline Edit Copy Delete	EC406	MOBILE COMMN.	tb_skpatra_ec406

Check All / Uncheck All *With selected:* Change Delete Export

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

Table of List Of Student in a Subject



localhost/phpmyadmin/index.php?db=db_attend_btech&token=ca7bf9dcd8fc4a182a6624d9ef597826

localhost ▸ db_attend_btech ▸ tb_skpatra_ec101

✓ Showing rows 0 - 9 (~10 total , Query took 0.0008 sec)

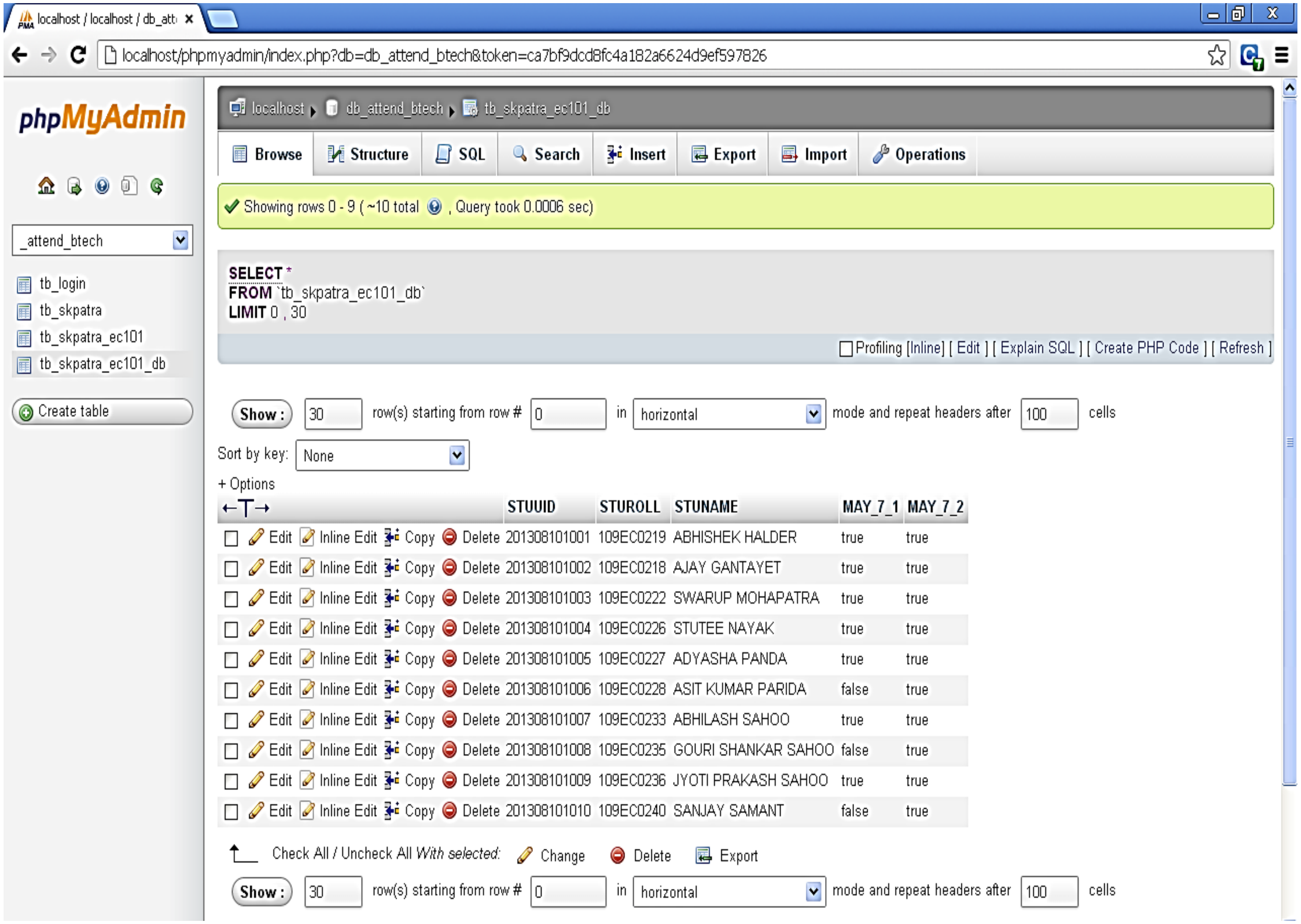
```
SELECT *
FROM `tb_skpatra_ec101`
LIMIT 0 , 30
```

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

Sort by key: None

STUID	STUNAME	STUROLL
201308101001	ABHISHEK HALDER	109EC0219
201308101002	AJAY GANTAYET	109EC0218
201308101003	SWARUP MOHAPATRA	109EC0222
201308101004	STUTEE NAYAK	109EC0226
201308101005	ADYASHA PANDA	109EC0227
201308101006	ASIT KUMAR PARIDA	109EC0228
201308101007	ABHILASH SAHOO	109EC0233
201308101008	GOURI SHANKAR SAHOO	109EC0235
201308101009	JYOTI PRAKASH SAHOO	109EC0236
201308101010	SANJAY SAMANT	109EC0240

Table Showing Attendance for 2 Classes For Category 1



localhost / localhost / db_att
localhost/phpmyadmin/index.php?db=db_attend_btech&token=ca7bf9dcd8fc4a182a6624d9ef597826
phpMyAdmin
_attend_btech
tb_login
tb_skpatra
tb_skpatra_ec101
tb_skpatra_ec101_db
Create table
localhost ▸ db_attend_btech ▸ tb_skpatra_ec101_db
Browse
Structure
SQL
Search
Insert
Export
Import
Operations

✓ Showing rows 0 - 9 (~10 total , Query took 0.0006 sec)

```
SELECT *
FROM `tb_skpatra_ec101_db`
LIMIT 0 , 30
```

□ Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

Sort by key: None

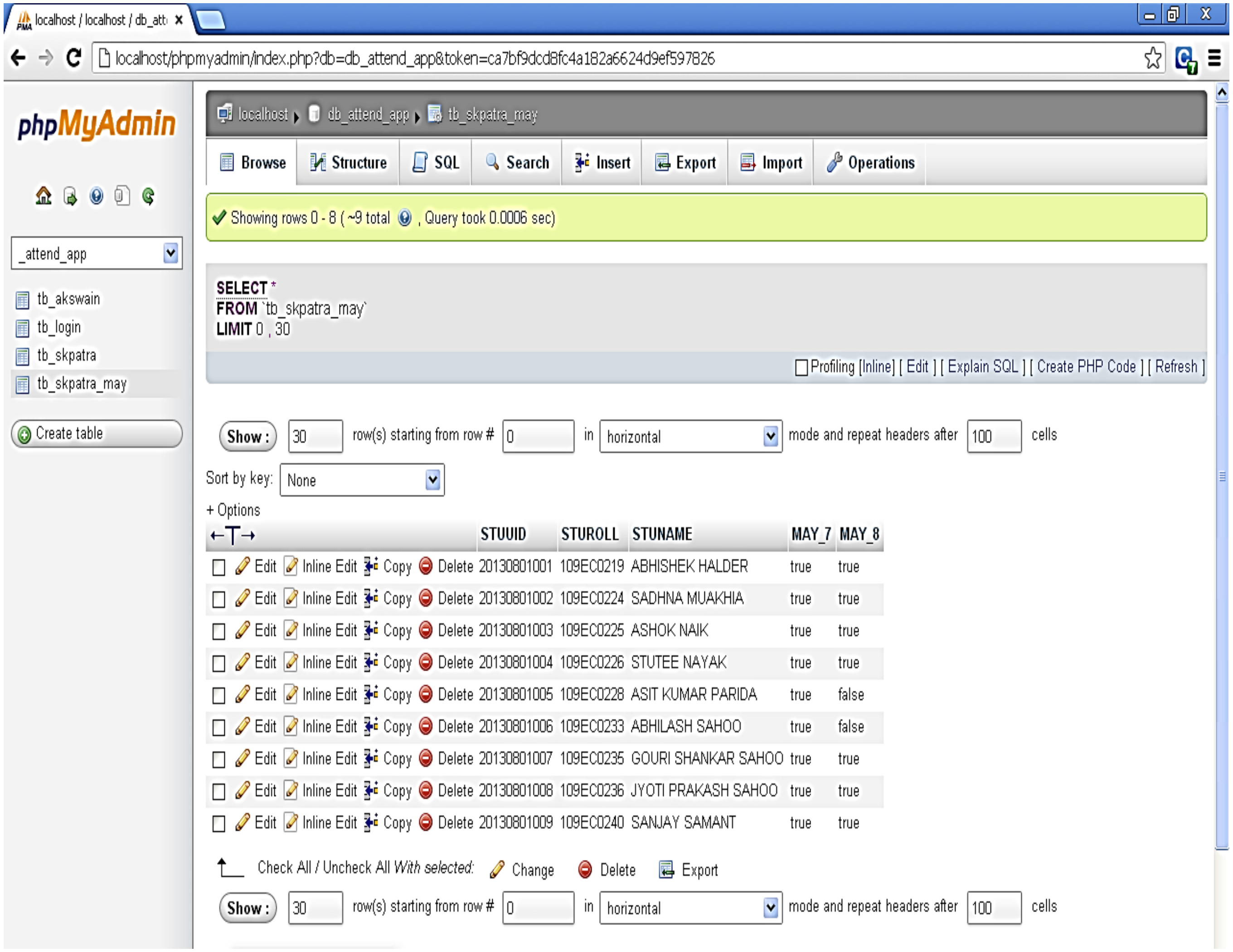
+ Options

←T→					STUUID	STUROLL	STUNAME	MAY_7_1	MAY_7_2
□	Edit	Inline Edit	Copy	Delete	201308101001	109EC0219	ABHISHEK HALDER	true	true
□	Edit	Inline Edit	Copy	Delete	201308101002	109EC0218	AJAY GANTAYET	true	true
□	Edit	Inline Edit	Copy	Delete	201308101003	109EC0222	SWARUP MOHAPATRA	true	true
□	Edit	Inline Edit	Copy	Delete	201308101004	109EC0226	STUTEE NAYAK	true	true
□	Edit	Inline Edit	Copy	Delete	201308101005	109EC0227	ADYASHA PANDA	true	true
□	Edit	Inline Edit	Copy	Delete	201308101006	109EC0228	ASIT KUMAR PARIDA	false	true
□	Edit	Inline Edit	Copy	Delete	201308101007	109EC0233	ABHILASH SAHOO	true	true
□	Edit	Inline Edit	Copy	Delete	201308101008	109EC0235	GOURI SHANKAR SAHOO	false	true
□	Edit	Inline Edit	Copy	Delete	201308101009	109EC0236	JYOTI PRAKASH SAHOO	true	true
□	Edit	Inline Edit	Copy	Delete	201308101010	109EC0240	SANJAY SAMANT	false	true

Check All / Uncheck All *With selected:* Change Delete Export

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

Table Showing Biometric Attendance for category 2



localhost/phpmyadmin/index.php?db=db_attend_app&token=ca7bf9dcd8fc4a182a6624d9ef597826

✓ Showing rows 0 - 8 (~9 total , Query took 0.0006 sec)

```
SELECT *
FROM `tb_skpatra_may`
LIMIT 0 , 30
```

□ Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

Sort by key: None

+ Options

←T→					STUUID	STUROLL	STUNAME	MAY_7	MAY_8
□	Edit	Inline Edit	Copy	Delete	20130801001	109EC0219	ABHISHEK HALDER	true	true
□	Edit	Inline Edit	Copy	Delete	20130801002	109EC0224	SADHNA MUAKHIA	true	true
□	Edit	Inline Edit	Copy	Delete	20130801003	109EC0225	ASHOK NAIK	true	true
□	Edit	Inline Edit	Copy	Delete	20130801004	109EC0226	STUTEE NAYAK	true	true
□	Edit	Inline Edit	Copy	Delete	20130801005	109EC0228	ASIT KUMAR PARIDA	true	false
□	Edit	Inline Edit	Copy	Delete	20130801006	109EC0233	ABHILASH SAHOO	true	false
□	Edit	Inline Edit	Copy	Delete	20130801007	109EC0235	GOURI SHANKAR SAHOO	true	true
□	Edit	Inline Edit	Copy	Delete	20130801008	109EC0236	JYOTI PRAKASH SAHOO	true	true
□	Edit	Inline Edit	Copy	Delete	20130801009	109EC0240	SANJAY SAMANT	true	true

Check All / Uncheck All *With selected:* Change Delete Export

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

3.8. **Summary**

The chapter deals with the second part of the project – NITR Attendance Management System. The android app for the above was designed keeping in mind the present structure of the system i.e. the Category 1 & Category 2 student have different type of attendance storage in main database at server. Also, as in previous cases the test results in the form of screenshots were shown & workflow sequences depicted in figures. The special modules detailed here are the Swipe Detector module which recognises finger swipes on the name of students & SQLite DBMS module which handles local storage of attendance in the android device.

CHAPTER 4

Technologies Used

Eclipse IDE & ADT Setup

Apache HTTP Server

PHP Scripting

MySQL Database Handling

Summary

Chapter 4. Technologies Used

4.1. Eclipse IDE & ADT setup

Eclipse is an integrated development environment for multi-language software & web application development. Its extensible plug-in system helps in customizing the IDE to work on a particular technology, like Android in this case. It supports Java as the principal coding language and through plug-ins; it supports C, C++, Python, Ruby, PHP, JavaScript, Perl etc. Under Eclipse Public License, the SDK is free and open source.

For android application development, we need to download the Android Development Tools (ADT) plug-in for eclipse and install it. After this, we need to run the SDK manager so as to download the Android SDKs for developing onto newer android OS versions and also the Google APIs and other support functions so as to use Google licensed android services like Maps, OAUTH 2.0 etc to full use.

The eclipse configuration where I have developed the applications is as follows:

Eclipse	:	Version	Juno (4.2.1)
ADT Package	:	Version	21.0.1
Android SDK Tools	:	Revision	21.1
Android SDK Platform-Tools	:	Revision	16.0.2
Android Support Library	:	Revision	12
Android SDK Platform	:	4.2.2 – API 17 – Revision 2	
		2.3.3 – API 10 – Revision 2	
Google APIs	:	4.2.2 – API 17 – Revision 2	
		2.3.3 – API 10 – Revision 2	

4.2. Apache HTTP Server

Apache is a web server software. Any web server is either hardware (a computer) or software (the computer application) and thus it helps deliver content which is accessed through the computer. It is the back end that provides the support for scripts to work, databases to seamlessly integrate with the web application. Apache played a vital role in initial phases of development of World Wide Web. It helps in implementing core modules for handling server side programming language, authentication schemes, socket transactions & layer securities. While supporting ftp, it has inbuilt HTML authorizing tool & inbuilt search engine. The large public library of add-ons helps one to customize the server end. Wamp Server is used to run a local server on PC or laptop.

4.3. PHP Scripting

It serves as a server side scripting language. The interpreter used at server end is a dedicate PHP processor module and interprets the chunk of code within the delimiters. Acronym for Hypertext Preprocessor, PHP helps in designing dynamic web content. PHP code is processed in command-line mode performing desired operating system operations and producing program output on its standard output channel. The three main reasons to work with PHP are: first, it fits in greatly with HTML, being interchangeable with it, and only adds new content. Secondly, its user interface provides a richer experience as compared to HTML only. Third and foremost, it’s easy to learn and we can easily get started by using only a few functions.

4.4. MySQL Database

MySQL is a world renowned open source Relational DBMS supporting standard SQL. At the server end, it can be configured to provide single user or multi user access to a number of databases and tables. Some of the best features are: cross-platform support, updatable views, cursors, information schema, query caching etc. It can easily be integrated into PHP scripts. It is developed, distributed & supported by Oracle foundation.

4.5. Summary

In this chapter, the various technologies which have been used to manage & develop the applications have been discussed. Support for both localhost and remote server has been discussed. They are instrumental in making the applications stand up to their quality while seamlessly implementing the work flow.

Conclusions and Future Enhancements

Conclusions

At the end, we find that, the applications were really useful. The tracker unlike others is free of cost. The network connection module developed would be helpful in N number of scenarios where synchronization or data exchange between devices is desired. The pin point module helps us in locating users and at the same time differentiating between custom locations, home location & friends' locations. The attendance app is unique for the management of the same in NIT Rourkela. Both the B. Tech & M. Tech systems are very well in sync with norm being followed in the institute i.e. attendance stored as days for M. Tech. & attendance stored in classes for B. Tech. A good number of man-hours would be reduced by the use of same. Attendance can be taken even when the professor is mobile. It does not depend on network very often. Except first time usage, the app is offline. And the only online usage occurs when we need to sync. Even if we forget to sync data for the previous month, by changing the system date, we can sync data for any previous months.

Future Enhancements

- Tracker can be modified so as to implement Google Maps V2 APIs. They are more advanced & support 3D projections.
- Deployment of Attendance Application throughout the institute requires scripting for connecting the database to main NIT Rourkela server.
- The UI can definitely be improved by using Action Bars & other new end graphics introduced in Ice Cream Sandwich & Jelly Bean versions of Android.

References

[1] Wei-Meng Lee, 2011, *Beginning Android Application Development*, Wiley India Pvt. Ltd.

[2] Reto Meier , *Professional Android 2 Application Development*,

[3] Marko Gargenta, *Learning Android,* O'Reily

[4] Travis Cornelius, *New Boston Series : Android Application Development Tutorials*

[5] Web Resource : http://www.vogella.com

[6] Web Resource : http://www.stackoverflow.com

[7] Web Resource : http://developer.android.com/index.html

[8] Web Resource : https://developers.google.com/maps/