ANDROID APPLICATION OF THROCK MOBILE

By

AKHILA VIDIYALA

B.E, Osmania University, India, 2009

A REPORT

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2013

Approved by:

Major Professor
Dr. Daniel Andresen

Abstract

The aim of this project is to develop an android application for managing and organizing activities of various departments in ThrockMorton building at K-State. Mobile application development is a growing trend in computer industry. With the advancements in mobile technologies and efficient 3G and 4G wireless communications, a number of desktop applications are now becoming available as mobile applications. Android has the leading market share in the entire smart phone OS available. It gives lot of space for creative development as it is open source. There are various discussion forums and official android development support websites that encourage mobile and tablet application development.

The ThrockMobile application provides many features for managing inventory at ThrockMorton in Kansas State University. The features include scanning a barcode of an asset and displaying its details and can even edit any of the fields if needed. The access to this application is restricted to only those devices whose device id is existing in the database.

One can request access to the application through an email client integrated with the application. Look up feature lets you look up for a user, room, ip-address and asset. Preferences module lets you enter the details of the server from which data is to be requested.

This application has been tested on android devices of varying screen sizes and android OS versions. The application serves requests at an average rate of 1.5sec/request, which is above the industry average time. I mentioned in detail the reason for the above performance average times and as future enhancements I have discussed the possible solutions.

Table of Contents

List Of Figures

List Of Tables

Acknowledgements

I would like to take this opportunity to thank my Major Professor, Dr. Daniel Andresen, for his valuable guidance and encouragement throughout the project.

I would also like to thank Professor, Dr. Mitchell Neilsen and Professor, Dr. Torben Amtoft, for graciously accepting to serve on my committee. I would also like to thank all my professors of the CIS department at Kansas State University for giving me an opportunity to take classes and work with them. The learning experiences and invaluable guidance will serve as a valuable asset in my career.

I would also like to thank my employer at ThrockMorton, Arthur Selman, for giving me opportunity to gain hands on experience in android development.

1. Introduction

ThrockMobile is an android application, used as an inventory management system at ThrockMorton. The application has the ability to lookup for a user, room, asset and its ip-address and to assign new devices to people/room. Only few devices will have access to the application whose device ids are registered with the database with permission from the admin. User authentication to access the application can be made through email client integrated within the application.

The application also has an integrated bar code scanner which scans the barcode on the go and tells for whom it is assigned and when it was last updated or replaced. One can also enter the numbers on the bar code manually and search for the details. The main aim is to make the application as simple as possible so every user can understand each and every feature that it offers.

2. Motivation

There are various departments in the ThrockMorton building at Kansas State University. The administrator has to manage a lot of users and rooms.

For example, the administrator assigns has to assign a new room, his devices, phone numbers etc. to a professor who resides in the building. The usual practice is for him/her to note down these details and then re enter them in the database manually which is tedious.

With almost each and every individual using a smart phone, the idea of ThrockMobile, being an android application for inventory management purpose at Throckmorton was intuitive. I think it is the best possible way the administrator can maintain assets in the building on the go.

Hence with the valuable guidance of Arthur Selman, Network Specialist/Instructor at the Agronomy Department at ThrockMorton, I started this project.

3. Requirement Analysis

3.1 Requirements Gathering

I have collected all the information required for the project from Mr. Arthur Selman. Regular meetings with him at every phase of the project helped in obtaining the requirements of the project. Requirements gathering played a crucial role as providing a user friendly Graphical User Interface (GUI) to use the application was important. This helped the users to navigate through the application with the minimum number of key stroke and hardly any use-cases to enter data.

One of the important requirement analysis involved choosing a suitable android platform version which is compatible with most of the android phones and android tablets in market today and also make it compatible for upcoming android versions. Market survey shows that many android devices use version of 2.3(Gingerbread), the latest version being 4.2 (Jelly Bean). This project is developed on version 2.3.3 Gingerbread platform and made it compatible to work on 4.0 (Ice cream sandwich) as well.

The features of the application are

1. Scan Barcode: A barcode scanner which scans the barcode of an asset and identifies the name, the room number it belongs to, the IP address assigned to it and the owner. Asset may be desktops, printers, phones etc.,
2. Lookup User/Room: Search tool to search for any user/room in the building. For a user, we can view the assets he/she owns, theirs phone numbers. For a room, we can look up the assets and ip-address that are located in that room.
3. Request Access: The usage is restricted to only some devices which are registered. Registration request can be sent through the email agent integrated with the application.
4. Registration: Assign a new user to a room, also a desktop or any other assets in the building and vice-versa.
5. Contact User: Search for a user and his contact number and make a call/message depending on weather it is a mobile or landline phone.

3.2 Requirement Specification

3.2.1 Software Requirements:

For developing the application the following are the Software Requirements:

Operating System: Windows 7

Platform: Android SDK Framework 10 or higher

Database: MYSQL, XAMPP Server

Tools: Eclipse SDK 3.5, ADT plug-in for eclipse, Aptana Plug-in(PHP Editor)

Technologies used: Java, Php, JSON, Android

Debugger: Android Dalvik Debug Monitor service

For running the application the following are the Software Requirements:

Operating System: Android 2.3 or higher versions

Network: Wi-Fi Internet or cellular Network

3.2.2 Hardware Requirements

For developing the application the following are the Hardware Requirements:

Processor: Intel Pentium IV or higher
RAM: 256 MB
Space on disk: minimum 250MB

For running the application following are the Hardware Requirements::
Device: Smart phone with Android version 2.3 and higher
Camera: 1.3MP rear camera
Minimum space to execute: 5.0MB

4. System Architecture and Design

4.1 Android Architecture:

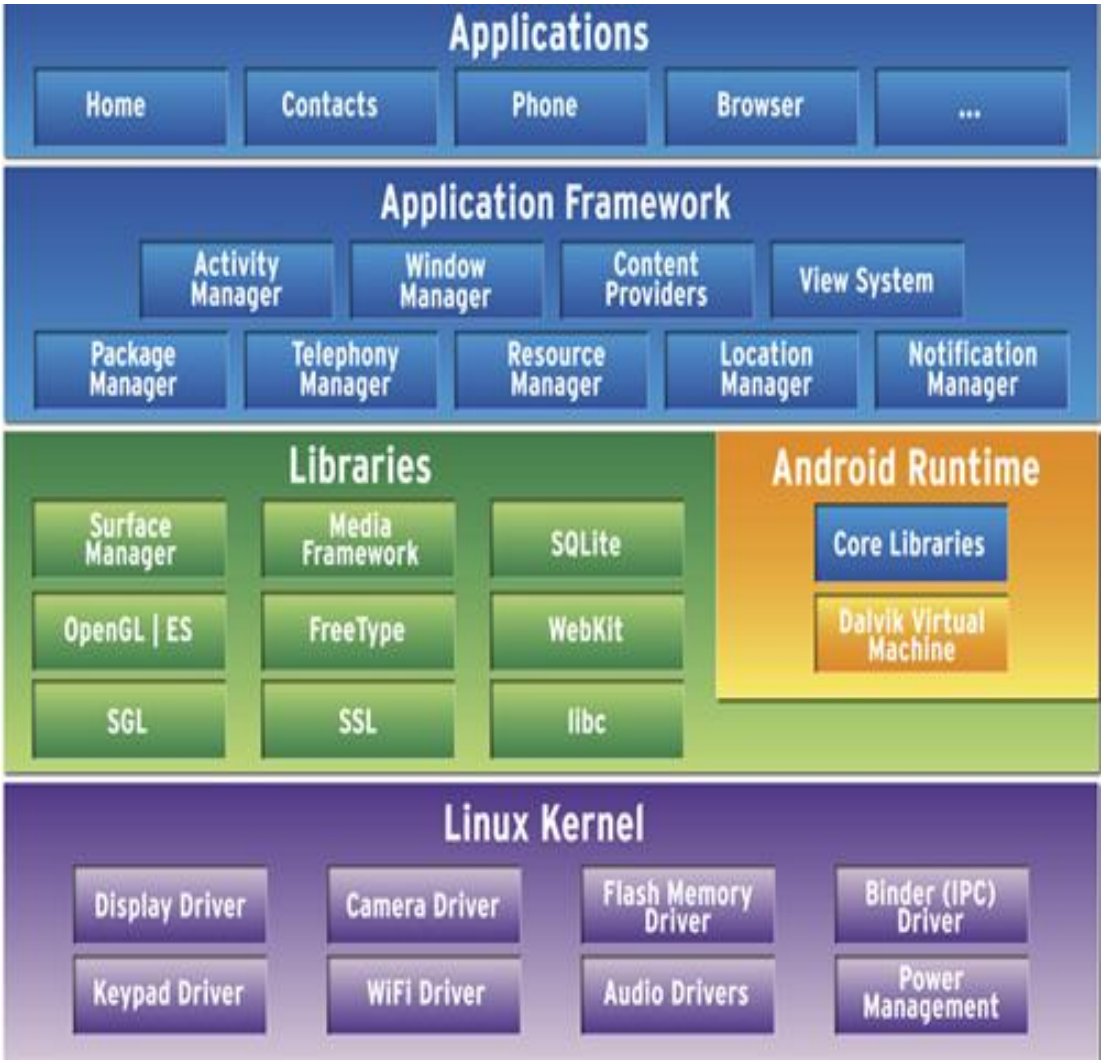

Figure 1: Android Architecture (ecnmag,2011**)**

An Android system acts like a stack of software components built upon the Linux 2.6 kernel. Linux provides basic system functions such as process and memory management, multithreading, and security. Also, the Linux kernel includes device drivers that take the pain out of interfacing to peripheral hardware.

The Android runtime kernel operates above the Linux kernel, as shown in Figure 1, and it contains both runtime libraries and the Dalvik virtual machine (VM). The runtime libraries comprise open-source building blocks such as the WebKit browser, the SQLite database, and the FreeType font engine.

4.2 System Architecture:

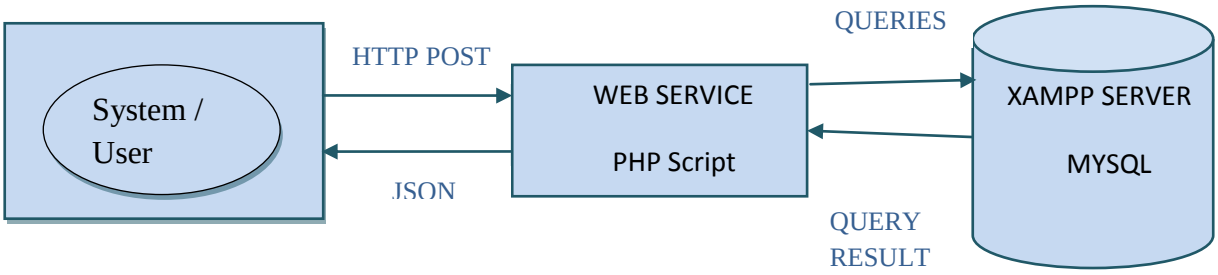

Figure 2: System Architecture

The above diagram explains the system architecture on which the application is based on.

Steps involved are:

- A php script is stored on the remote server (in our case, the XAMPP server).
- The system invokes the php script when it needs to connect to the database (again, on the XAMPP server).
- A command corresponding to the required action is passed to the php script along with the url, based on which the appropriate action is taken in the database
- Additional data is also encoded in the URL String and sent with an HTTP POST request.
- The result of the action is passed back to the front-end in form of JSON.

The application interacts via the touch input of the user. The main screen has list of actions which the application is capable of. When the user touches the screen, the respective action takes place depending on where the user touches and the user is propagated to the next screen. PHP is used here because of the interaction it can offer with the databases and it is easy to deploy on the XAMPP Server. JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange and it is used in this application to send data from Android device to PHP Script

4.3 System Design:

After the requirements gathering and analysis, the design of the system is created using UML, Unified Modelling Language. Various activities and entities have been identified and structured using Use case and Class Diagrams. The Unified Modelling Language includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems. A detailed explanation of those two diagrams is given below.

4.3.1 Use Case Diagram:

Use Case Diagram describes the functionality in terms of actors, use cases, and any dependencies among those use cases from an external point of view. It is type of behavioural diagram in UML. The goals of the user and the sequence of actions within application to achieve some functionality are depicted as a use cases. It describes various ways that users interact with the system.

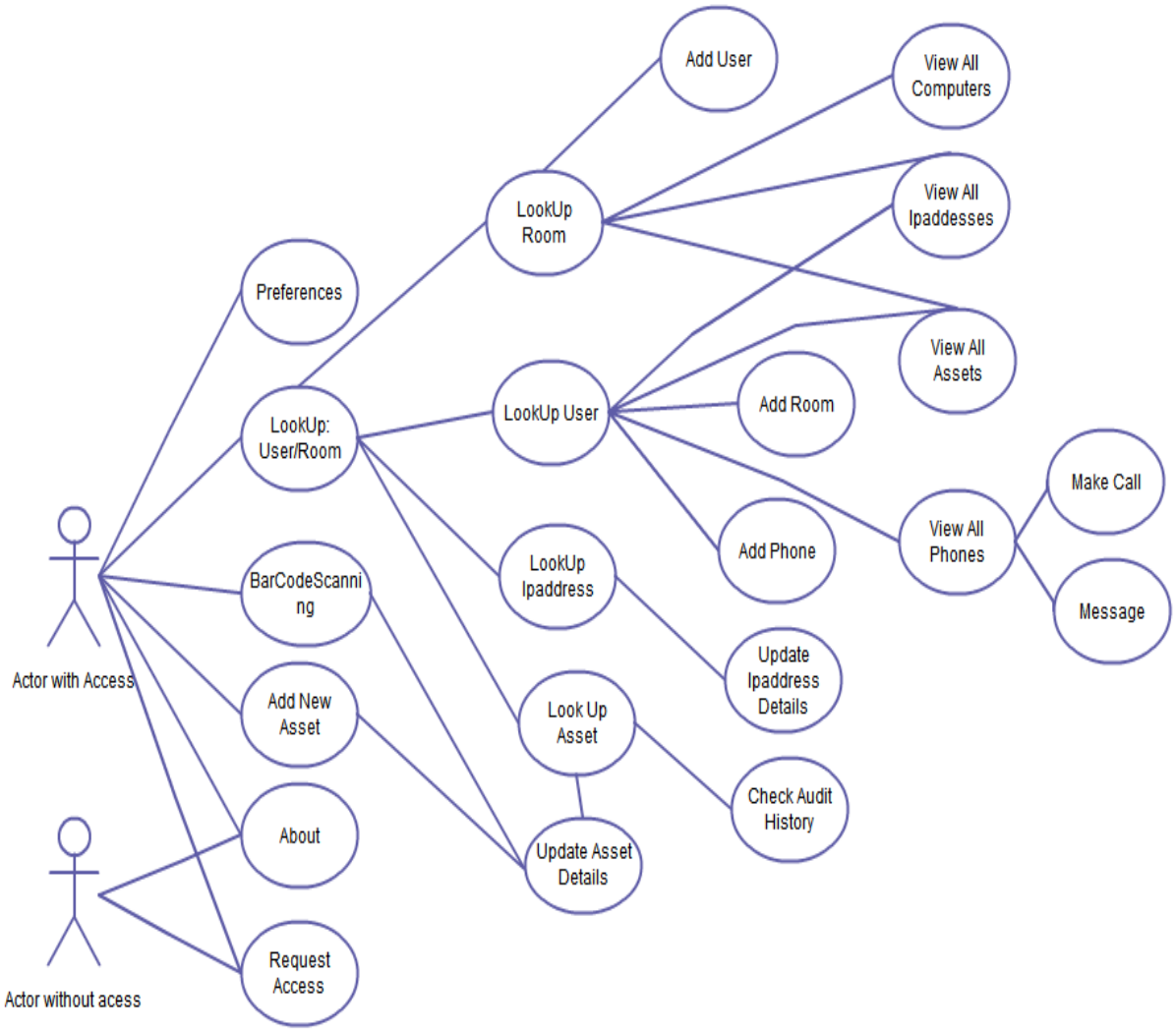

Figure 3: Use Case Diagram

4.3.1.1 Use case 1: Preferences

- **Description:** This use case allows the user to enter the user specific application preferences which is required for the application to run.
- **Precondition:** User's Md5 must exist in the database. One has to know the location of database and the script.
- **Main Success Scenario:**
    - User first opens the application from the android phone.
    - Enters the ip-address of the server which has the database.
    - Select from various databases like agronomy and horticulture.
    - Enters the Md5 key which is unique for a device.
    - One has to enter the web service script location as well.
    - End of Use case.
- **Post condition:** The user is good to access the applications additional features.

4.3.1.2 Use case 2a: Look up user

- **Actor:** Administrator
- **Description:** This use case allows the admin to look up a user with their EID.
- **Precondition:** User enters the Preferences. User enters a search value in the text box.
- **Main Success Scenario:**
    - Can search for another user of the department selected.
    - Can select a user from the auto populated list or can enter a new one.
    - Displays the detailed description of the user selected.
    - Displays a list view which has number of computers, ip-addresses, assets, rooms and phone numbers associated with the user selected.
    - Can add a new room for the user.
    - Can add a new phone number to the user.
- **Post condition:** The profile corresponding to the EID searched is shown by the application in case the profile exists.
- **Alternate Scenario:** A message is displayed saying that the profile does not exist, in case the application is unable to find the profile.

4.3.1.3 Use case 2a.1: Add room

- **Actor:** Admin
- **Description:** This use case allows the user to register a room for a particular user which was looked up.
- **Precondition:** User enters the Preferences. User is in the User Details page
- **Main Success Scenario:**
    - Can register a room.
    - Can make that room as the primary room.
    - Pop up will be displayed to confirm to make the room as primary if there is already a primary room for that user in the database.
- **Post condition:** Successful registration of a room to the user.

- **Alternate Scenario:** A message saying “duplicate” is displayed if the room already exists for that user.

4.3.1.4 Use case 2a.2: Add phone

- **Actor:** Admin
- **Description:** This use case allows the user to register a phone for a particular user which was looked up.
- **Precondition:** User enters the Preferences. User is in the User Details page
- **Main Success Scenario:**
    - Can register a phone whether a personal one or work cell.
    - Can make that phone as a primary one.
    - Pop up will be displayed to confirm to make the phone as primary if there is already a primary phone for that user in the database.
- **Post condition:** Successful registration of a phone to the user.
- **Alternate Scenario:** A message saying “duplicate” is displayed if the phone already exists for that user.

4.3.1.5 Use case 2a.3: View all phones

- **Actor:** Admin.
- **Description:** User can view the list of Phones assigned for a particular user which was looked up.
- **Precondition:** Admin looks up user information.
- **Main Success Scenario:**
    - User is in the user details page.
    - User can view the list of phones that were allocated for the room if he/she is in the room details page.
    - Can select one to view detailed information
- **Post condition:** Success in viewing the list of Phones.

4.3.1.6 Use case 2a.3.1: Call phone

- **Actor:** Admin.
- **Description:** User can view the list of Phones assigned for a user and can make a call.
- **Precondition:** User looks up for list of phones.
- **Main Success Scenario:**
    - User is in the list view of phones.
    - User can make a call by clicking the button beside the phone number in the list view
- **Post condition:** Success in making a call to the phone number listed.

4.3.1.7 Use case 2a.3.2: Message

- **Actor:** Admin.

- **Description:** User can view the list of Phones assigned for a User and can send a text message to that number
- **Precondition:** Admin looks up for list of phones for a particular user.
- **Main Success Scenario:**
    - User is in the list view of phones.
    - User can message by clicking the button beside the phone number in the list view
- **Post condition:** Success in sending a text message to the phone number listed.

4.3.1.8 Use case 2b: Look up room

- **Actor:** Admin
- **Description:** This use case allows the user to look for a room with a room number.
- **Precondition:** User enters the Preferences. Enters a value in the text box
- **Main Success Scenario:**
    - Can search for a room.
    - Can select a room number from the auto populated list or can enter a new one.
    - Displays the detailed description of the room selected.
    - Displays a list view which has number of computers, ip-addresses, assets, users, computers and phone numbers associated with the room selected.
    - Can add a new user for the room.
- **Post condition:** The profile corresponding to the room searched is shown by the application in case the profile exists.
- **Alternate Scenario:** A message is displayed saying that the profile does not exist, in case the application is unable to find the profile.

4.3.1.9 Use case 2b.1: Add user

- **Actor:** Admin
- **Description:** This use case allows the user to register a user for a particular room which was looked up.
- **Precondition:** User enters the Preferences. User is in the Room Details page
- **Main Success Scenario:**
    - Can register a User.
    - Can make that room as a primary one.
    - Pop up will be displayed whether to make the user as primary if there is already a primary user for that room in the database.
- **Post condition:** Successful registration of a user to the room.
- **Alternate Scenario:** A message saying “duplicate” is displayed if the room already exists for that user.

4.3.1.10 Use case 2b.2: View all computers

- **Actor:** Admin.
- **Description:** User can view the list of computers under a room

- **Precondition:** User looks up for a room.
- **Main Success Scenario:**
    - User is in the room details page
    - User can view the list of computers that were in the room
    - Can select one to view detailed information
- **Post condition:** Success in viewing the list of computers.

4.3.1.11 Use case 2b.3 / 2a.4: View all ip-addresses

- **Actor:** Admin.
- **Description:** User can view the list of ip-addresses assigned for a room/user.
- **Precondition:** User looks up for a room or another user information.
- **Main Success Scenario:**
    - User is in the room details page or user details page
    - User can view the list of ip-addresses that were allocated for the room if he/she is in the room details page.
    - User can view the list of ip-addresses that were allocated for the user if he/she is in the user details page.
    - Can select one to view detailed information
- **Post condition:** Success in viewing the list of ip-addresses.

4.3.1.12 Use case 2b.4 / 2a.5: View all assets

- **Actor:** Admin.
- **Description:** User can view the list of assets assigned for a room/user.
- **Precondition:** User looks up for a room or another user info
- **Main Success Scenario:**
    - User is in the room details page or user details page
    - User can view the list of assets that were allocated for the room if he/she is in the room details page.
    - User can view the list of assets that were allocated for the user if he/she is in the user details page.
    - Can select one to view detailed information
- **Post condition:** Success in viewing the list of assets.

4.3.1.13 Use case 2c: Look up asset

- **Actor:** Admin.
- **Description:** This use case allows the user to look for an asset.
- **Precondition:** User enters the Preferences. Enters a value in the text box
- **Main Success Scenario:**
    - Can search for a asset.
    - Can select asset number from the auto populated list or can enter a new one.
    - Displays the detailed description of the asset selected.

    - Can edit the details if desired to do.
    - Can check the audit history the asset like when it was last updated or checked.
- **Post condition:** Details of the asset is displayed.
- **Alternate Scenario:** A message is displayed saying that the profile does not exist, in case the application is unable to find the asset.

4.3.1.14 Use case 2c.1: Update asset details

- **Actor:** Admin.
- **Description:** This use case allows the user to look for an asset
- **Precondition:** User looks up for an Asset
- **Main Success Scenario:**
    - Can edit certain details of an asset.
- **Post condition:** Details of the asset are successfully updated.

4.3.1.15 Use case 2c.2: Check audit history

- **Actor:** Admin.
- **Description:** This use case allows the user to look for audit details of asset.
- **Precondition:** User Looks up for an Asset.
- **Main Success Scenario:**
    - Can check audit details of an asset.
    - Audit details has the information of when was an asset last updated.
    - Physical check means when an asset was last viewed.
    - Update info has information of what fields of the asset were updated and when.
- **Post condition:** Details of the asset are successfully checked.

4.3.1.16 Use case 2d: Look up ip-address

- **Actor:** Admin.
- **Description:** User can view the existing ip-address details.
- **Precondition:** User enters the Preferences. Enters a value in the text box.
- **Main Success Scenario*:***
    - User searches for an ip-address record.
    - Detailed description of ip-address is displayed.
    - User can modify any of the details and submits to database.
- **Post condition:** A successful modification or display of ip-address details.
- **Alternate Scenario:** A message is displayed saying that the profile does not exist, in case the application is unable to find the ip-address.

4.3.1.17 Use case 2d.1: Update ip-address details

- **Actor:** Admin.

- **Description:** User can update the existing ip-address details.
- **Precondition:** User looks up for an ip-address.
- **Main Success Scenario:** Can edit certain details of an ip-address.
- **Post condition:** A successful modification of ip-address details.

4.3.1.18 Use case 3: Bar code scanning

- **Actor:** Admin.
- **Description:** User can scan any barcode for getting its description.
- **Precondition:** User enters the Preferences. Zxing application should be installed.
- **Main Success Scenario:**
    - User searches for anything by just scanning its barcode.
    - User can also enter the details printed on the barcode to lookup for its details.
- **Post condition:** The detailed description of the barcode is listed like the user, room and the description
- **Alternate Scenario:** A message is displayed saying that the barcode details don't exist, in case the application is unable to find the barcode.

4.3.1.19 Use case 4: Add new asset

- **Actor:** Admin.
- **Description:** User can add an asset to the database.
- **Precondition:** User enters the Preferences. Must enter the database of the department to which the asset must be assigned to.
- **Main Success Scenario:**
    - User can add a new asset to the database
    - User can select whether it is a departmental asset or university asset.
    - After creating an asset, user is directed to the page where he can edit further details of the asset like name, user to whom it is assigned and room where it is located etc.,
- **Post condition:** The Admin has successfully added the new asset.

4.3.1.20 Use case 5: About screen

- **Actor:** Admin.
- **Description:** User can view the details of the application
- **Precondition:** User has the application installed.
- **Main Success Scenario:**
    - Opens up the about screen.
    - The screen is the home page of the application which has the ip-address of the server and device id listed
- **Post condition:** Success in viewing the about screen.

4.3.1.21 Use case 6: Request access

- **Actor:** Admin.
- **Description:** User can request access to application through this use case and request is approved by the admin.
- **Precondition:** An email client must be setup in the device on which the application is running.
- **Main Success Scenario:**
    - Opens up the email client with repopulated email address of the admin, subject.
    - User must enter the device id and send it to the admin.
- **Post condition:** The admin grants permission to the user if its valid and enters the device id to the database and replies back the user with a MD5 key which the user has to enter onto the preferences screen to enter into the application.

4.3.2 Class Diagram:

It describes the structure of a system with help of classes, their attributes, operations (or methods), and the relationships among the classes. The class diagram is the main building block of object oriented modelling. Each activity in android application is depicted as a class and the interaction between them is explained as intent navigation.

The figure 4, class diagram 1, depicts the relationship of MenuActivity with other activities. The figure 5, class diagram 2, depicts the relationship between LoookupActivity and all the users, rooms, ipaddress and assets. The multiplicity of these relationships are one to one.

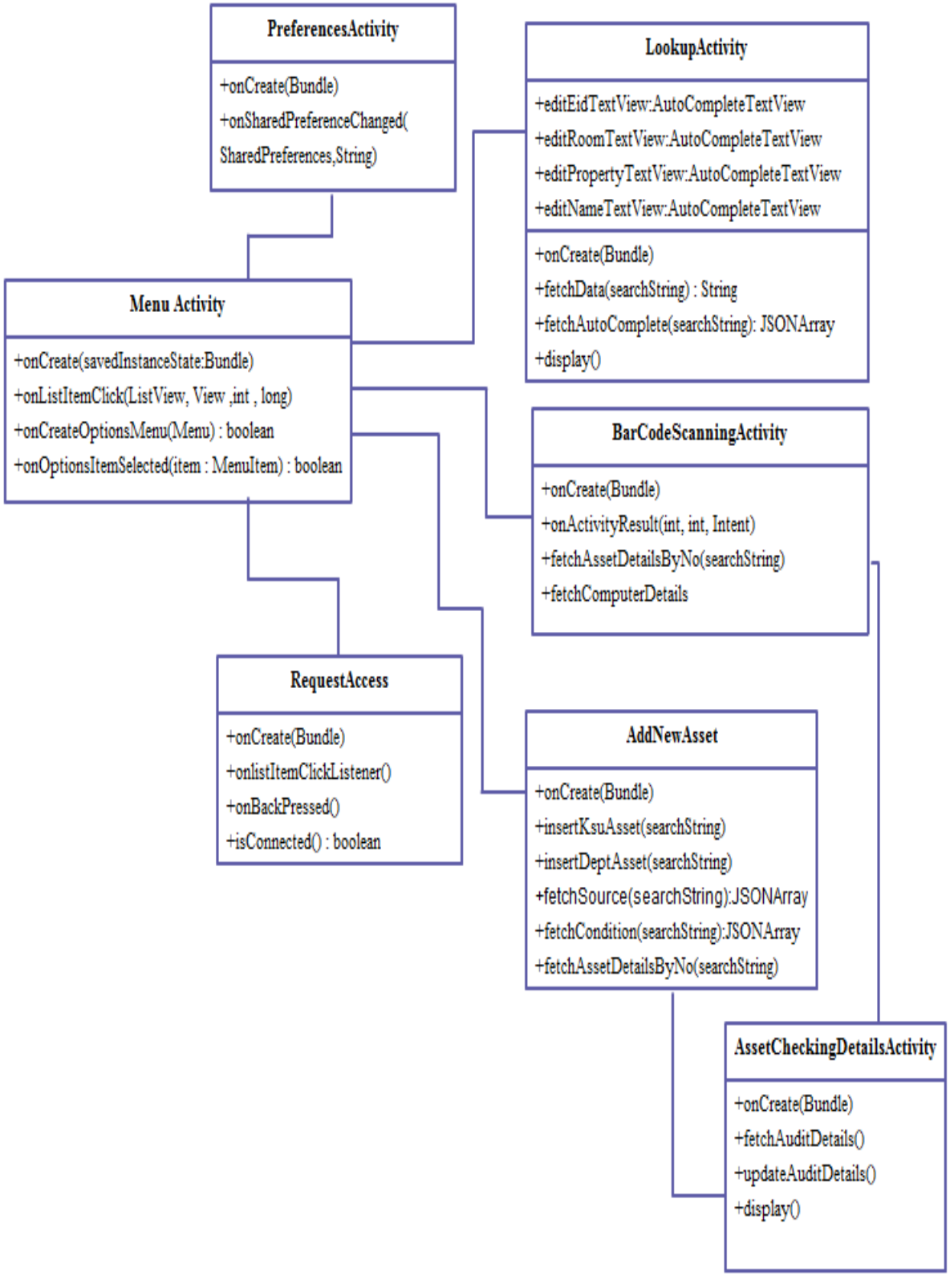

Figure 4: Class Diagram 1

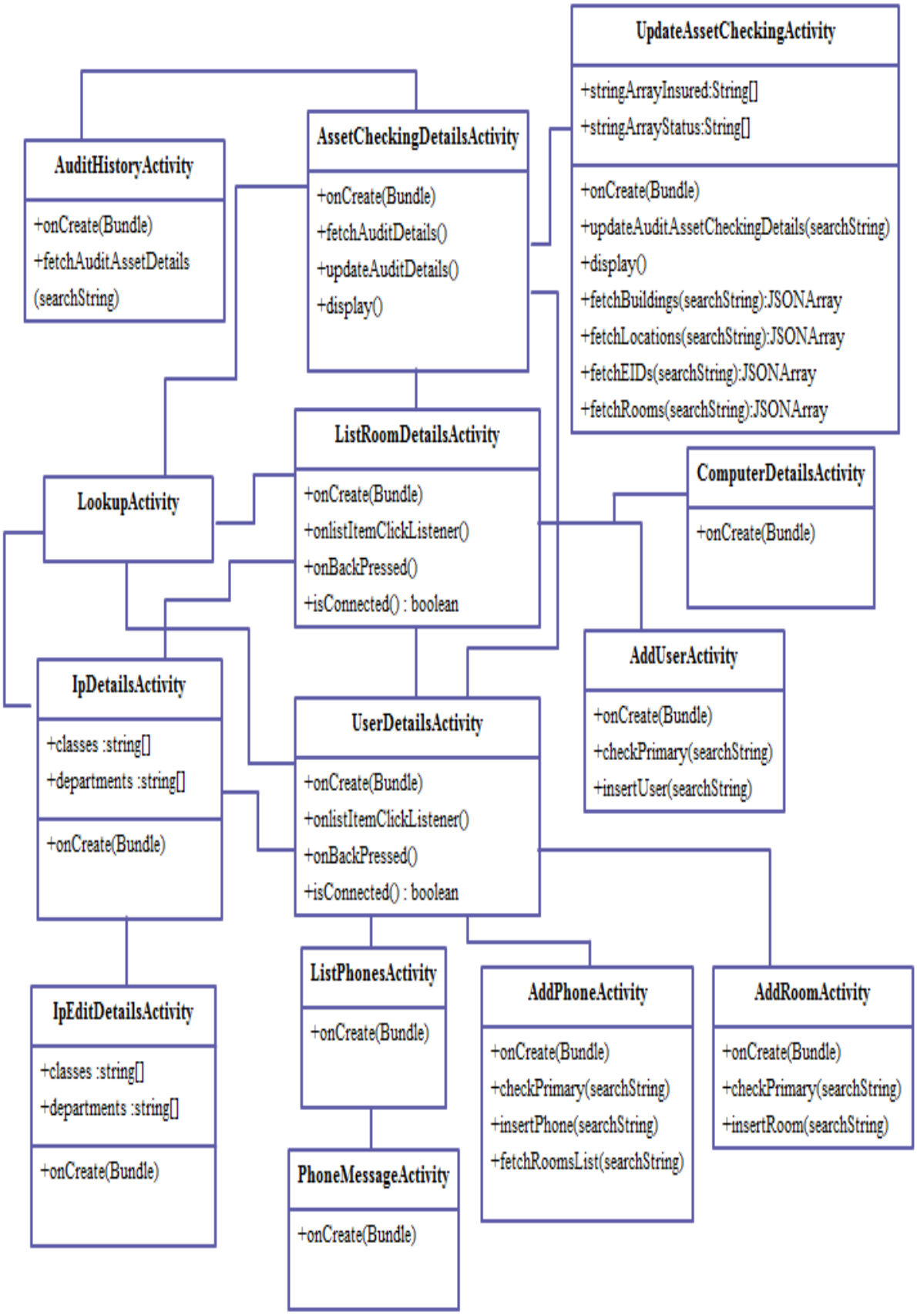

Figure 5: Class Diagram 2

5. Android Framework Components

Android applications are developed in JAVA and on eclipse IDE. The Android SDK tools compile the code along with any data and resource files into an Android package, an archive file with an '.apk' suffix. All the code in a single .apk jar is considered to be one application and is the jar that Android-powered devices use to install the application.

Activities are the main components of android application. Intent is used to activate the actions performed by an activity. All the activities present in the application must be declared in a manifest file which is present in its root directory. The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code. Detailed explanation of these components in this android application is explained in this section.

5.1 Android Manifest.xml:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0" package="edu.ksu.agron.thcs.ThrockMobile">
  <uses-sdk android:minSdkVersion="8" />
  <supports-screens android:resizeable="true"
      android:largeScreens="true" android:normalScreens="true"
      android:anyDensity="true"></supports-screens>
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.CAMERA"/>
  <uses-permission android:name="android.permission.CALL_PHONE"/>
  <uses-permission android:name="android.permission.SEND_SMS"/>
  <uses-permission android:name="android.permission.RECEIVE_SMS"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE">
  </uses-permission>
  <application android:label="@string/app_name"
          android:icon="@drawable/ic_launcher_throck"
          android:debuggable="true"
          android:description="@string/app_name">
     <activity android:name=".Splash" android:label="@string/app_name">
       <intent-filter>
         <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.LAUNCHER"/>
       </intent-filter>
     </activity>
     <activity android:name=".LoginActivity"
           android:label="@string/app_name">
       <intent-filter>
           <action android:name="edu.ksu.agron.thcs.ThrockMobile.
                                                   LOGINACTIVITY" />
```

```
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>

    <activity
        android:screenOrientation="landscape"
        android:name="com.google.zxing.client.android.CaptureActivity"
        android:configChanges="orientation|keyboardHidden"
        android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
        android:windowSoftInputMode="stateAlwaysHidden">
      <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.DEFAULT"/>
        </intent-filter>
        <intent-filter>
              <action android:name="com.google.zxing.client.android.SCAN"/>
              <category android:name="android.intent.category.DEFAULT"/>
        </intent-filter>
      </activity>
  </application>
</manifest>
```

Manifest file contains information about the characteristics of application. It has the list of permissions the application uses like the internet, network state of the device to check whether an internet connection exists or not and also camera to request for a barcode scanning. If an application needs access to a feature protected by permission, it must declare that it requires that permission with a <uses-permission> element in the manifest. Then, when the application is installed on the device, the installer determines whether or not to grant the requested permission by checking the authorities that signed the application's certificates and, in some cases, asking the user. If the permission is granted, the application is able to use the protected features.

All the activities must be listed out in this file or else android run time exception will occur. The minimum SDK that is required for the application is to be mentioned too. The main and launcher indicates the start up activity of the application. The application icon which is used on devices is mentioned in the application tag of manifest file.

Application requesting data from some other application must also be written in this manifest file. ThrockMobile application uses Zxing barcode scanner. Compatibility between screen sizes and among different android SDK versions should also be mentioned in this file.

An activity can restrict itself to be available only on portrait/landscape mode by locking itself by listing it like “`android:screenOrientation="landscape"`”, which locks

itself in the landscape mode. Activity specific permissions or constraints like no keyboard on loading; no title bar must be encapsulated in activity tag.

5.2 Activity:

Activities are the parts of an application that users can see and with which they may interact. An Activity has a view which is built from xml that you create, and it can contain widgets like buttons, text input fields, date pickers, and other common user-interface components. It also contains these views' click-listeners and any other supporting code.

There are several call-back methods that an activity might receive, due to a change in its state whether the system is creating it, stopping it, resuming it, or destroying it which are part of activity lifecycle process and each call-back provides you the opportunity to perform specific work that's appropriate to that state change .

Views are the GUI placed in "`setContentView`" method and Activities are initialized with "`onCreate(Bundle)`" method in which various components can be accessed like text fields, images etc by using `findViewById`

The method which loads the view of the xml is given below

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.splash);
    TextView textDevID = (TextView)findViewById(R.id.textDevID);
}
```

5.3 Intent:

Intent provides a facility for performing late runtime binding between different activities. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.

```
Intent openLoginActivity = new  Intent(this ,"edu.ksu.agron.thcs.ThrockMobile.
                                                MENUACTIVITY");
Bundle b = new Bundle();

b.putString("bufferDataStrings", resultObj.toString());
```

```
b.putString("bufferDataStringsFrom", "room");

b.putString("bufferDataStringsFromObj", room);
intentLookup.putExtras(b);
 startActivity(openLoginActivity);
```

Intent object holds the information of the activity that is to be launched from present context. The method `startActivity()` launches the activity implicitly. There is also another method `startActivityForResult()` which launches the activity based on the result obtained from a series of actions in the present activity. One can also pass data between activities through an intent by declaring a bundle. Bundle encapsulates the data and is passed to another activity from the present activity

Components advertise their capabilities — the kinds of intents they can respond to — through *intent filters*. Since the Android system must learn which intents a component can handle before it launches the component, intent filters are specified in the manifest as `<intent-filter>` elements.

6. Implementation

The main objective of ThrockMobile application is to make an inventory management system for ThrockMorton on an android platform. Only few devices will have access to the application. The device ids are being inserted in the database to check for authentication of devices.

This Android application is developed on Eclipse Helios IDE with an ADT plug-in which provides android development environment. Android SDK 2.3.3 and 4.0.1 is installed in the system along with JAVA 1.6. The business logic is written in Java. Front End GUI is developed using XML.

Middle layer is a web service that interacts with the database (MYSQL) and is written in PHP. XAMPP server is installed on the server side where the web service is located and maintained. Aptana plug-in is installed in eclipse for php compilation and

execution. The data object that is passed from Android to web service and back and forth is JSON. JSON is used as a data format as it is easier and faster to decode.

Total lines of code written for this application are:

Programming Language	LOC
JAVA	4920
PHP	2100
XML	1798

Table 1: Programming Languages Vs LOC

A total of 380 working hours spanning over 6 months have been spent on designing and developing this application and an approximate of additional 5 hours have been spent on testing the application for its correct functionality after the application has been fully developed.

Debugging of application is done using Dalvik Debug Monitor Server (DDMS) console in eclipse. LogCat is integrated into DDMS, and outputs the messages that you print out using the Log class along with other system messages such as stack traces when exceptions are thrown.

6.1 Graphical User Interface:

GUI of the application is very simple, understandable and easy to navigate. The user has very less data to enter and minor actions to perform. Almost every text field is auto populated with the elements that exist in database. The UI has been compatible for almost all the android screen sizes and for operating systems 2.3.3 or higher as well.

6.1.1 Splash Screen:

The Figure 6 below is the application start-up screen. This screen gives a glimpse of the application logo, server details and device ID on which the application is running.

Figure 6: Splash Screen

6.1.2 Menu:

Figure 7, Menu lists the features that the application has to offer. The Menu activity starts up after the splash screen. User can select any of the features listed in the menu.

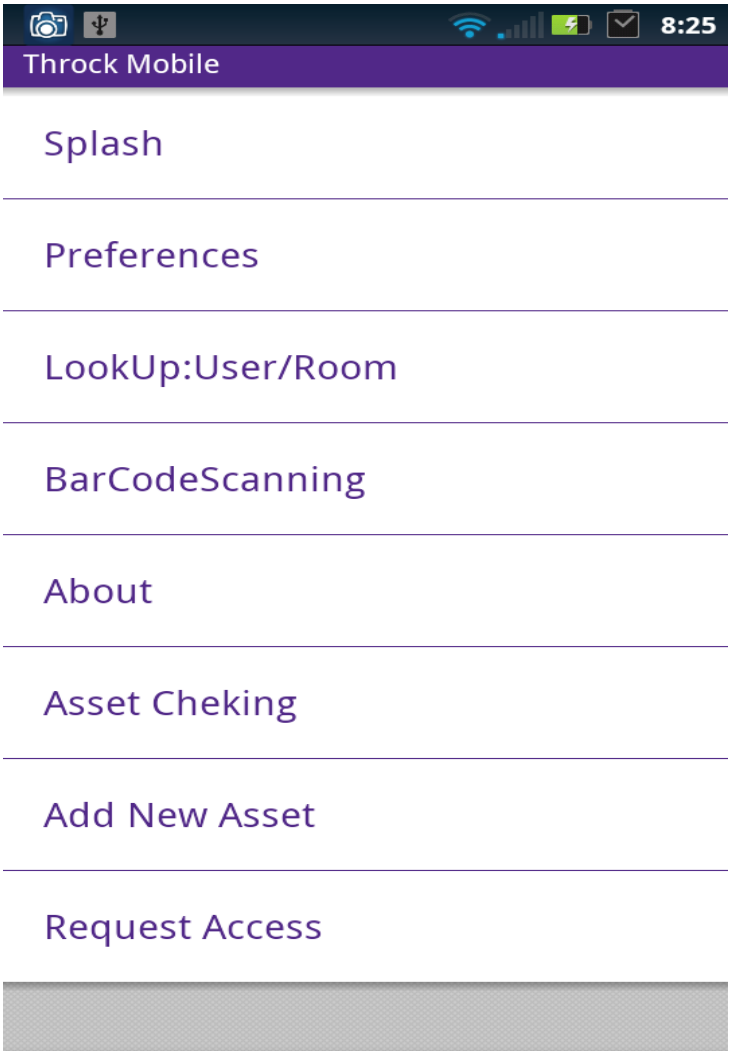

Figure 7: Menu Screen

6.1.3 Preferences:

Figure 8, Preferences screen is the backbone of the application where in you enter the details about the back end database. The application doesn't let you access any of the other features unless the fields have been entered in preferences screen. The fields include server ip address, path of the web service, device id and MD5 unique key assigned for a device. One can choose which department's database can be accessed for the application as there are many departments in the Throckmorton building. Various databases pertaining to the departments have been maintained on the server. Figure 9 and 10 depicts how to select a database and also enter or modify the server details respectively

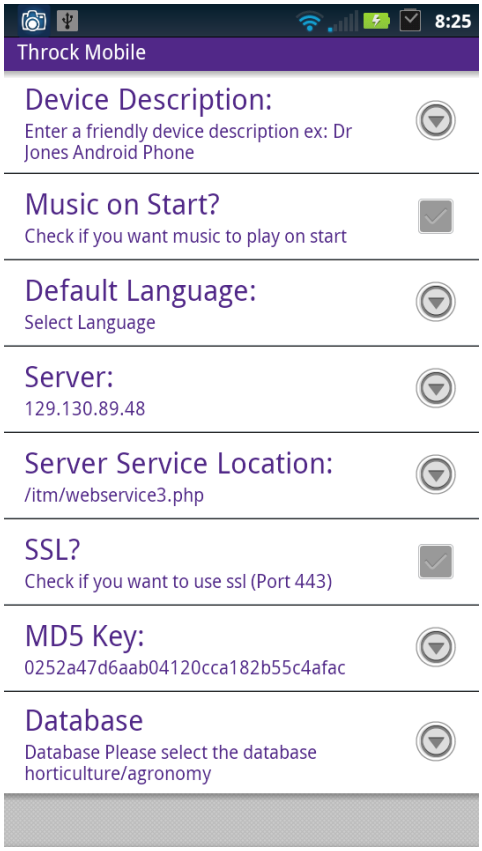

Figure 8: Preferences Screen

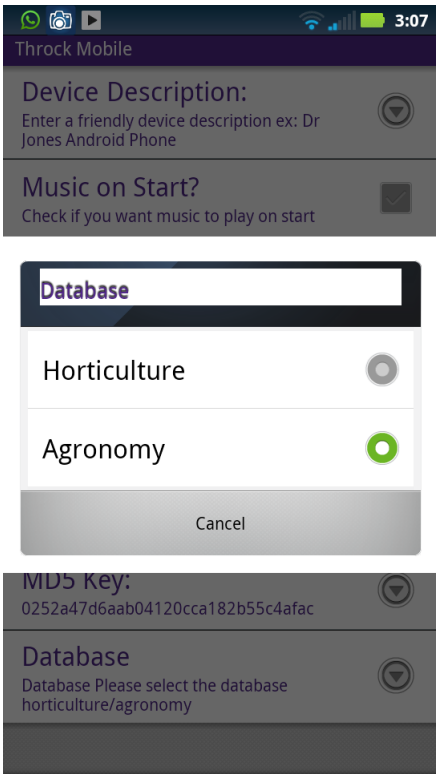

Figure 9: Database Spinner: Spinner Shows up onclick database on Figure 8

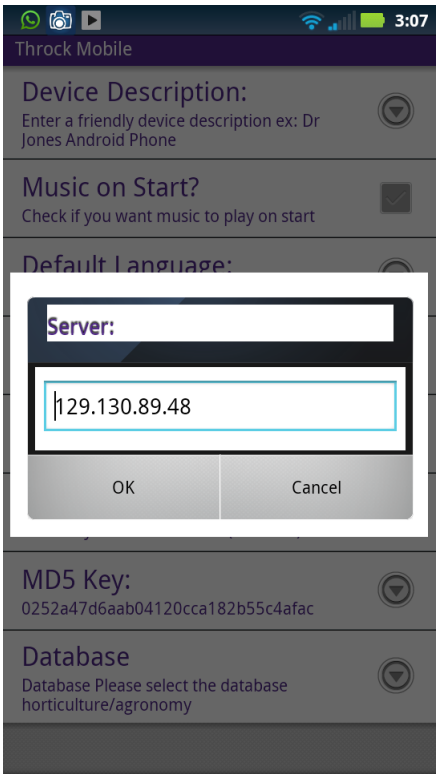

Figure 10: Server EditPreference: Window shows up onclick of Server in Figure8

6.1.4 Request access:

One can request access to the applciation through this screen which opens up an email client. An email client must already be setup on the device. After selecting an email client, compose mail opens up with some prepopulated fields, the to address of the admin, subject and a hint is given as what to send in the email.

Figure 11 is the result of a click on Request Access Field on Menu. Figure 12**,** Compose Mail opens after the user has selected a gmail email client.

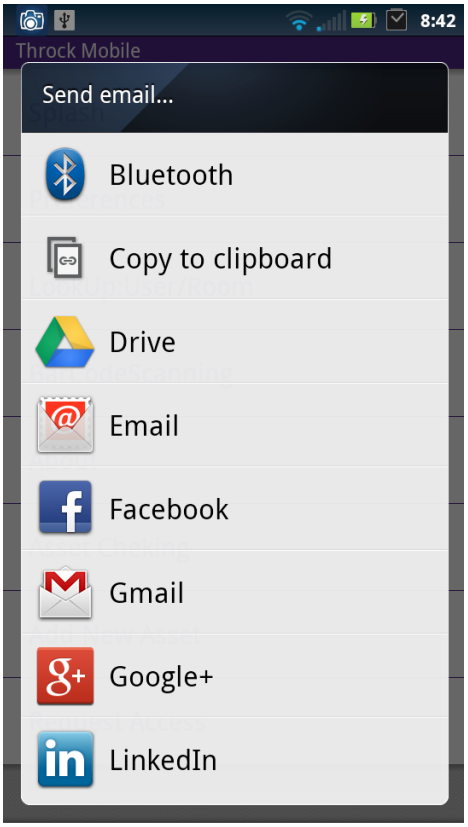

Figure 11: Email Client

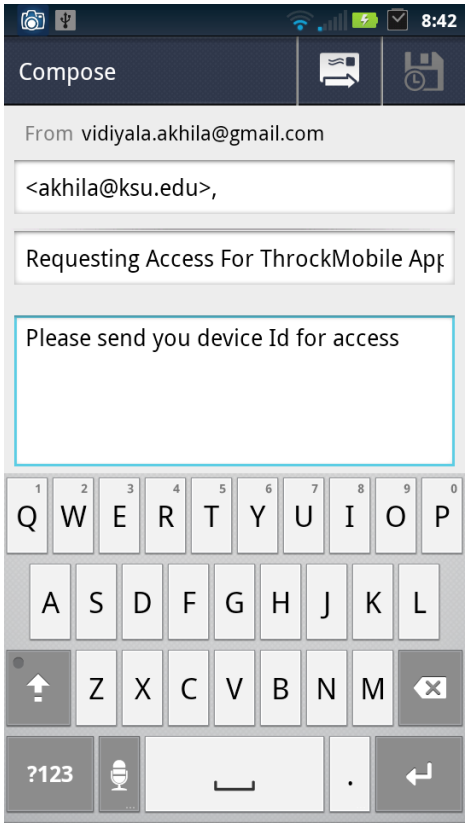

Figure 12: Compose Mail

6.1.5 Look up User/Room:

This screen lets you access the feature of looking up for something in the database. One can look up for a room, user, ip address or an asset. All the text fields in the screen are autopopulated with the items that were already existing in the database. One just has to enter two alphabets and a list is being populated with items starting with those alphabets and you just have to choose one of them for more detailed information.

The Lookup Screen can be scrolled. Figure 13, 14 below are the top and bottom views of the lookup screen respectively.

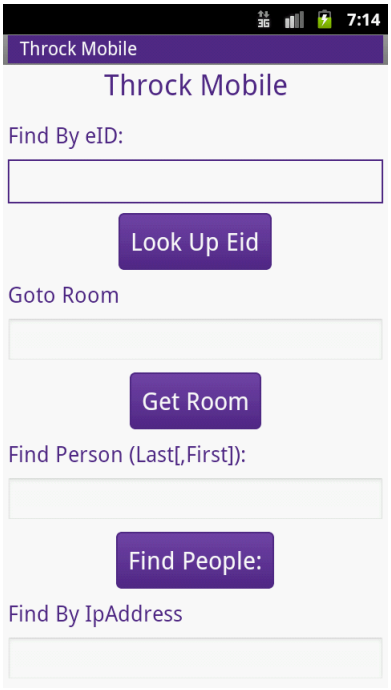

Figure 13: LookUp Screen 1

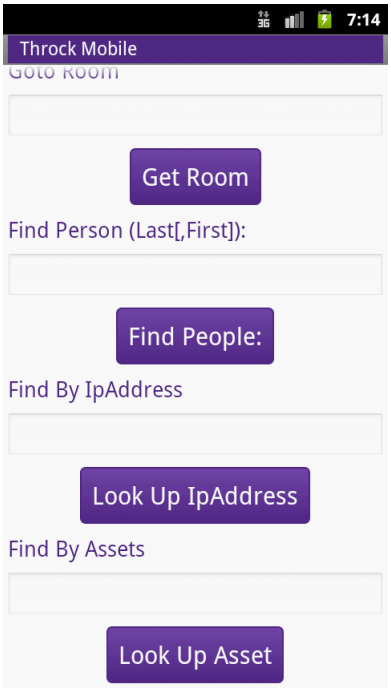

Figure 14: LookUp Screen 2

6.1.6 Look up Eid:

Figure 15, lets you choose an eid and look up for a detailed information about that person like which room he was assigned to and a count of resources that were allocated to him/her. Figure 16, displays information about acselman.

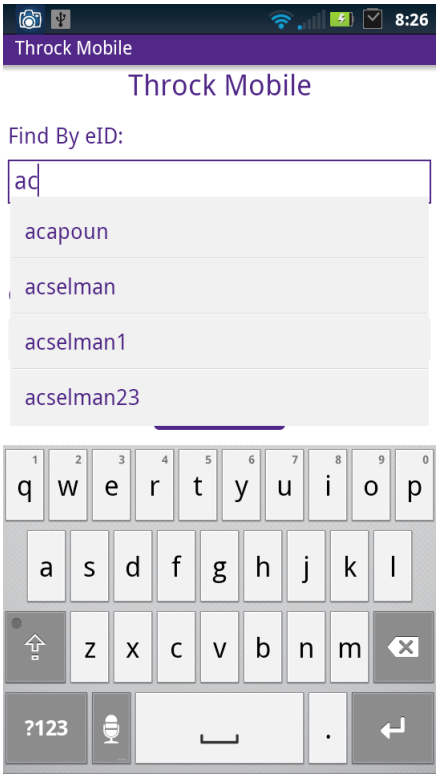

Figure 15: LookUp Eid

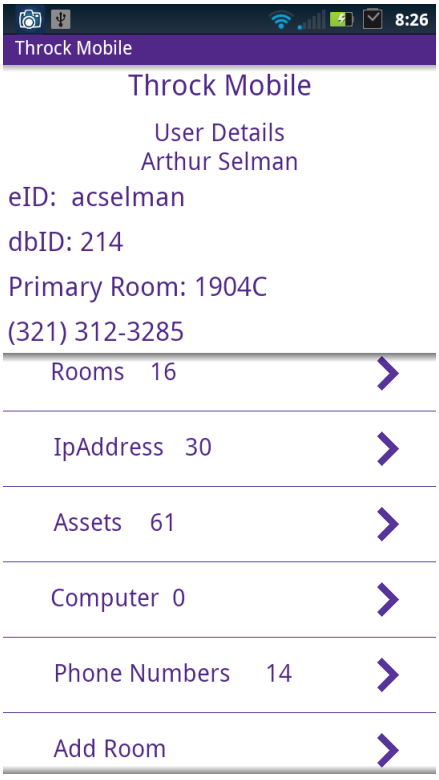

Figure 16: UserDetails

Below are lists of scenarios that can happen through various functions performed on user details info page.

Scenario 1:

From Figure 16, User details page one can view the list of rooms which was listed as “Rooms 16”, of which 16 is the count of Rooms which were under that user’s supervision or which were allocated to him. On Clicking the Rooms, the Room List in Figure 17 will be displayed as a pop-up screen, of which one can be chosen. The Figure 18 is the display of the Room details info when Room 1006 is selected.

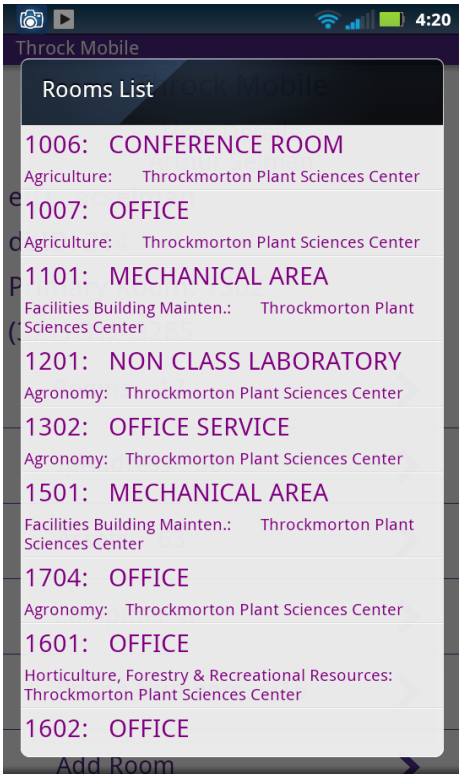

Figure 17: Rooms List

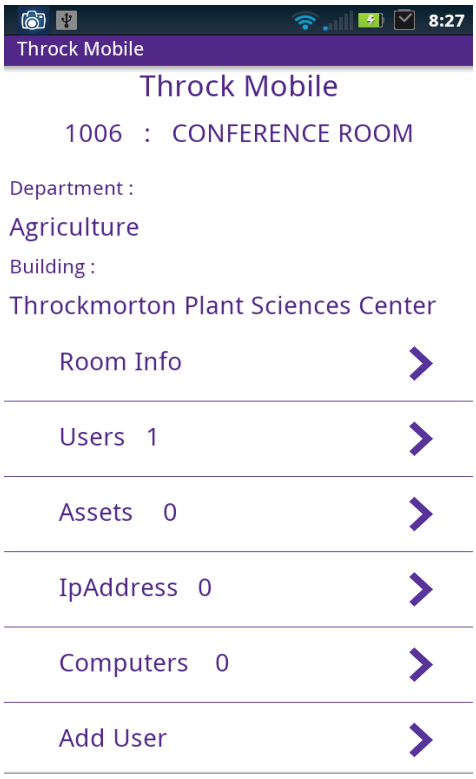

Figure 18: Room Details

Scenario 2:

We can view list of ip-addresses (Figure 19) that were allocated to the user by clicking on “Ip-address 30” as on Figure 16, which means 30 ip-addresses were allocated for this particular user. Clicking any one ip-address from the list view of ip-addresses navigates the user to IP address Info page (Figure 20) where you can view or edit the details of it as well. One can click on edit button of ip-address detail info, the functionality of edit ipaddress will be discussed in 6.1.8. Look up IpAddress section.

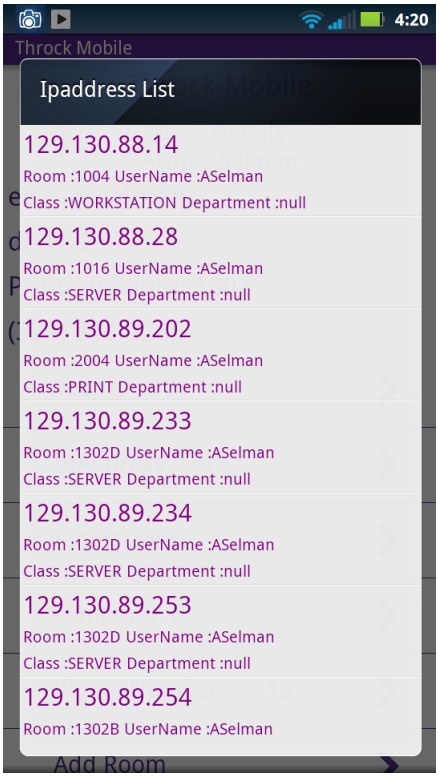

Figure 19: Ipaddress List

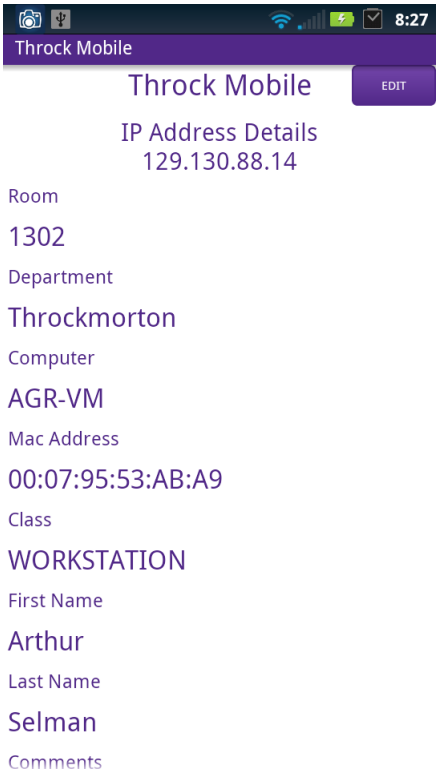

Figure 20: Ipaddress Details

Scenario 3:

Clicking on “Assets 61” of Figure 16, User Details page will popup the list of assets as in below Figure 21. Choosing one of the asset navigates to Figure 22, asset details page. Asset details page has detailed overview of the asset information. Go back to user info by clicking on ok button. The edit functionality will be discussed in detail in 6.1.9 Look Up Asset section.

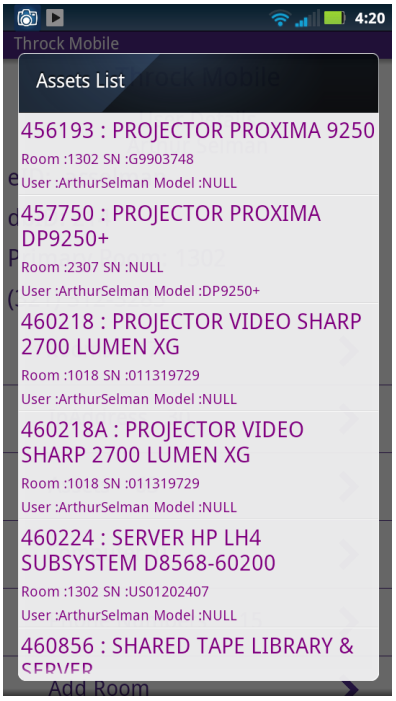

Figure 21: Assets List

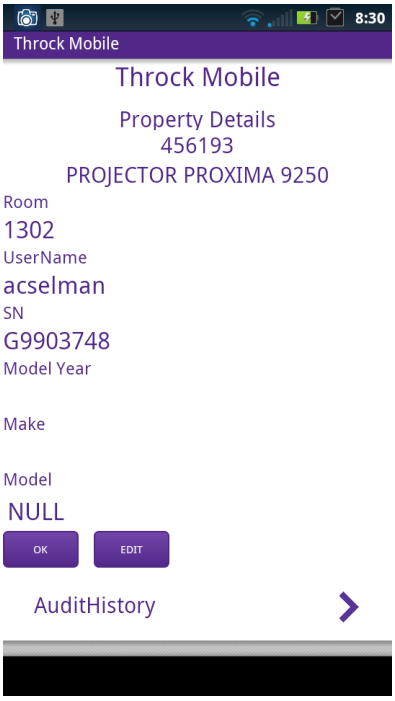

Figure 22: Asset Details

Scenario 4:

Figure 23 will give an overview of list view of phone numbers displayed on click of "Phone numbers 14" as on Figure 16. The icons besides the phone number has the functionality to call or text that number as per Figure 24 and 25 respectively.

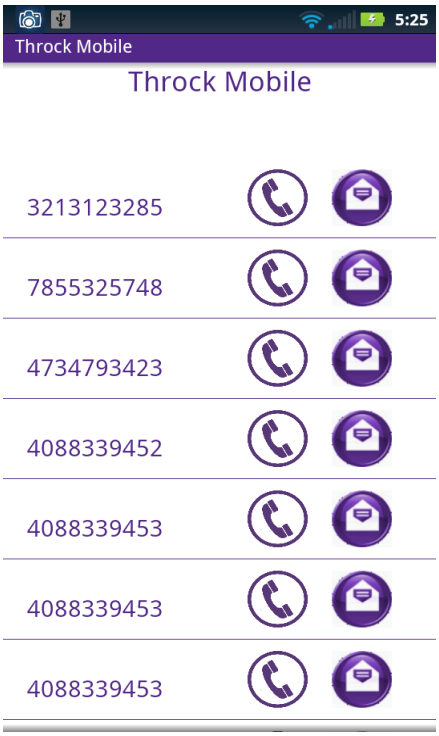

Figure 23: Phone Numbers List

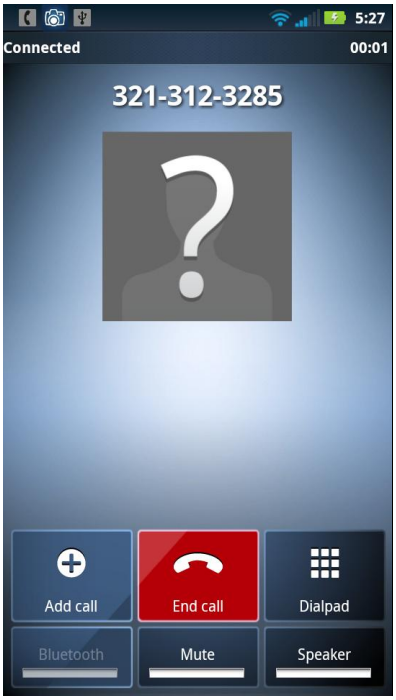

Figure 24: Calling phone

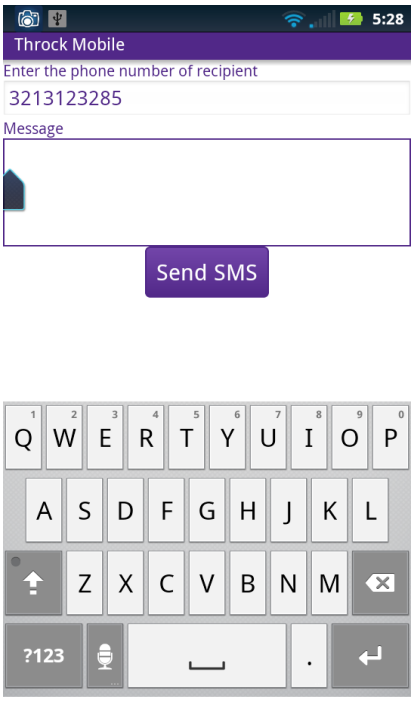

Figure 25: Send SMS

Scenario 5:

One can assign new room to the user of Figure 16 or can make a room primary. Primary room means the users main room where he actually resides or works. From Figure 16, primary room of user is 1904C. The user is able to assign a new room to be the primary room as per the figures 26 and 27 below. The Primary Room and also count of rooms have been changed in user detail info page Figure 28.

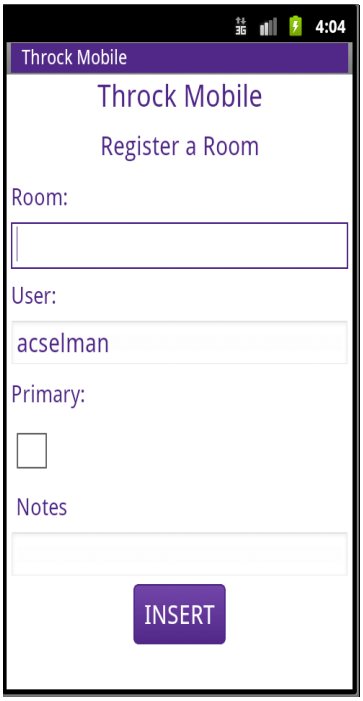

Figure 26: Add Room

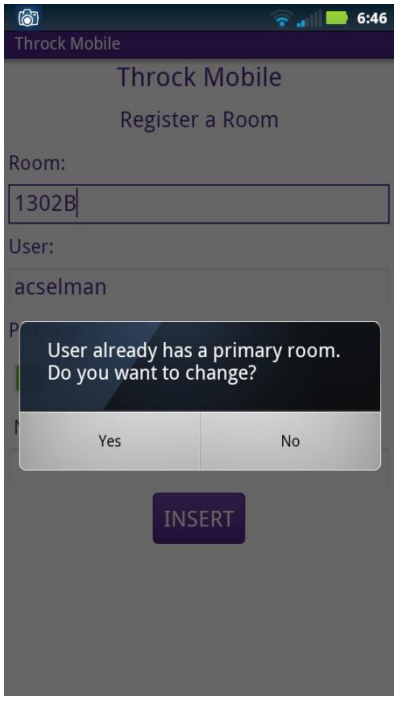

Figure 27: Primary Room

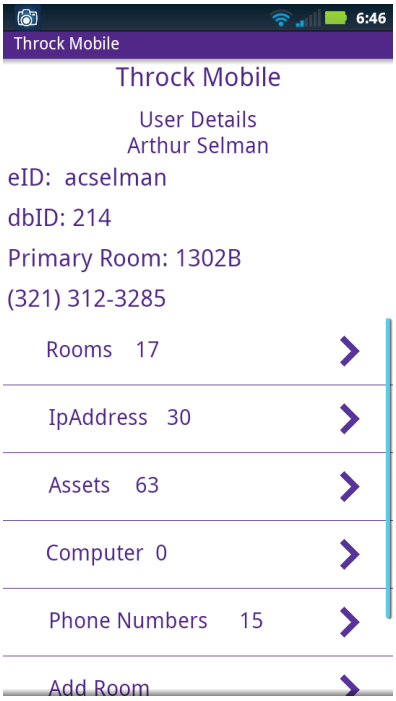

Figure28: UpdatedUserDetails

Scenario 6:

This gives an overview of adding a phone to the user.

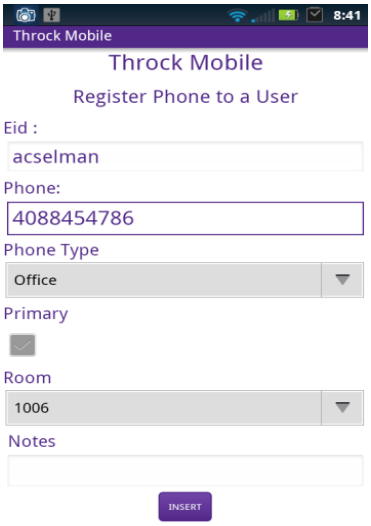

Figure29: Add Phone

6.1.7 Look up Room:

This feature lets you get detailed information about a room. The text field in Figure 30 is auto populated. The series of actions done on room details info is very similar to the section 6.1.6 described above of User Details page. Below in the scenario is the explanation of feature which was different from user page.

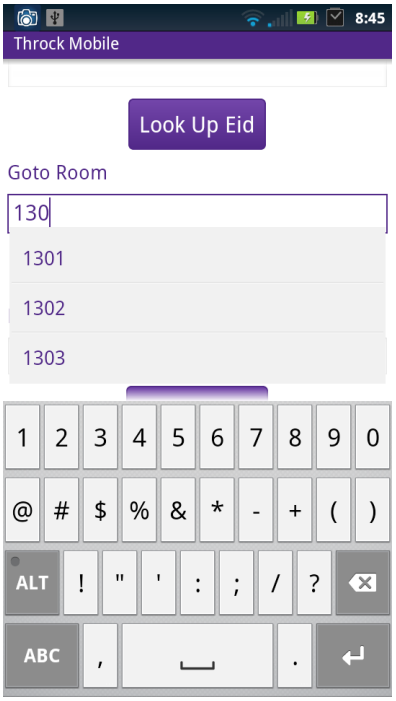

Figure30: Look Up Room

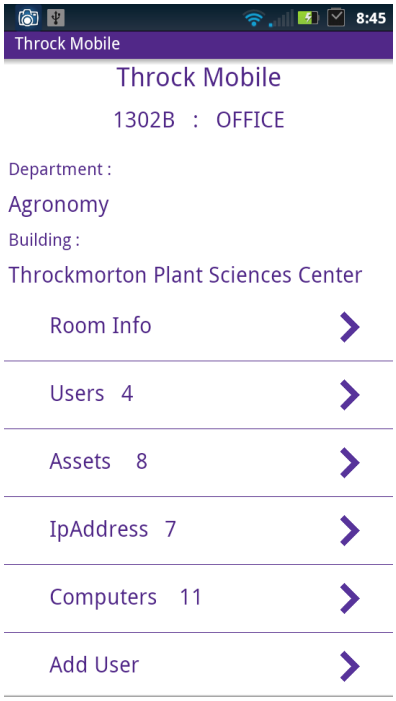

Figure31: Room Details

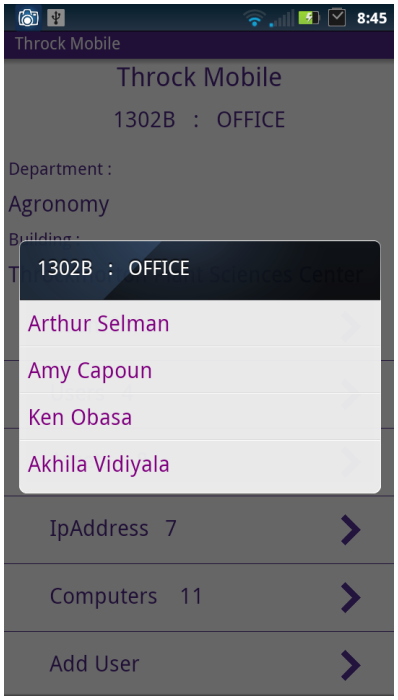

Figure32: Users List: UsersList is shown on click of "Users 4" in Figure30

Scenario 1:

Figure 33 gives the overview of the list of computers that are present in the room or allocated to the room in Figure31. Click on one of the computer for a detailed information as of Figure 34.

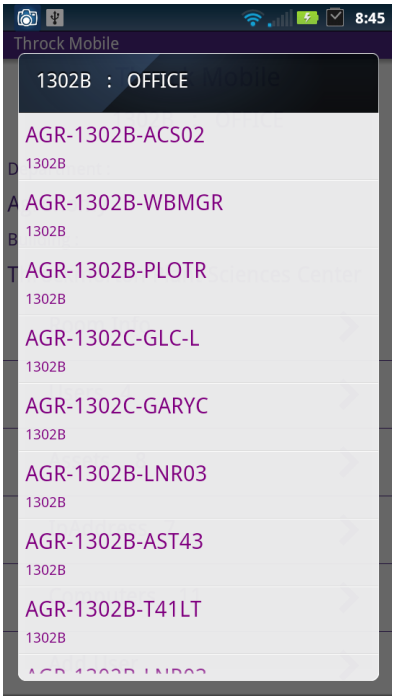

Figure33: Computers List

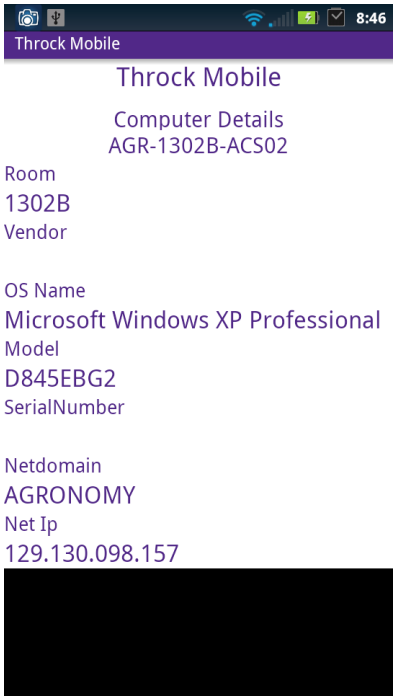

Figure34: Computer Details

6.1.8 Look up Ip-Address:

One can look up for an ip-address and can edit the details of it. Just enter last two numbers of ip-addresses. That will be prefixed with 129.130 and then searched.

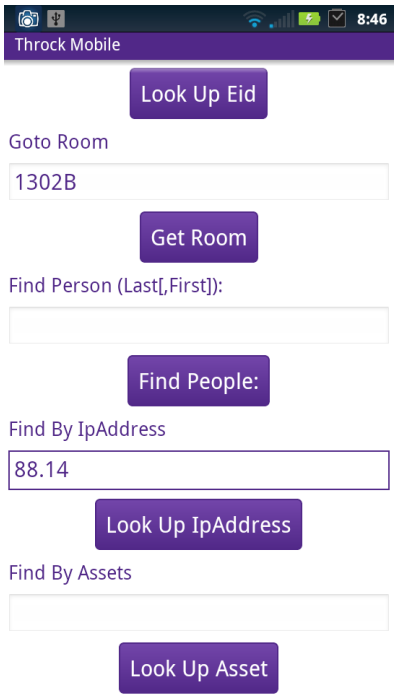

Figure35: LookUp Ipaddress

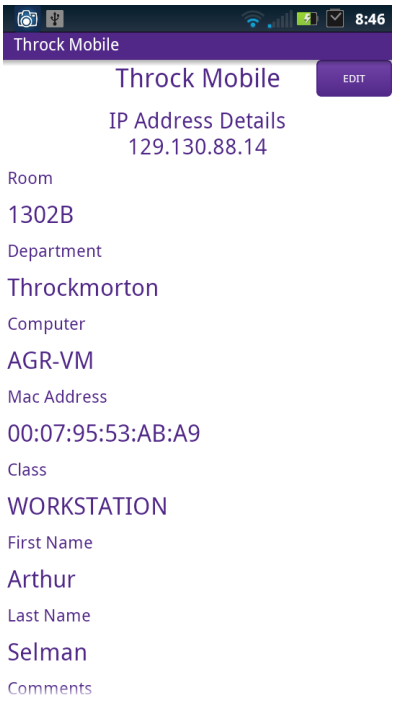

Figure36: Ipaddress Details

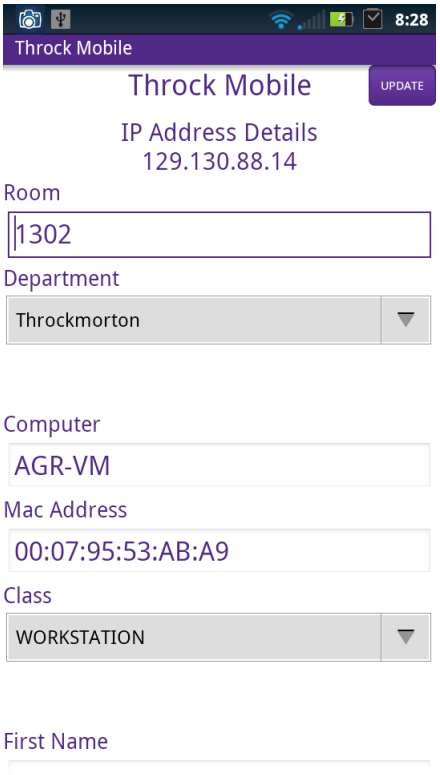

Figure37: Edit Ipaddress (topview**)**

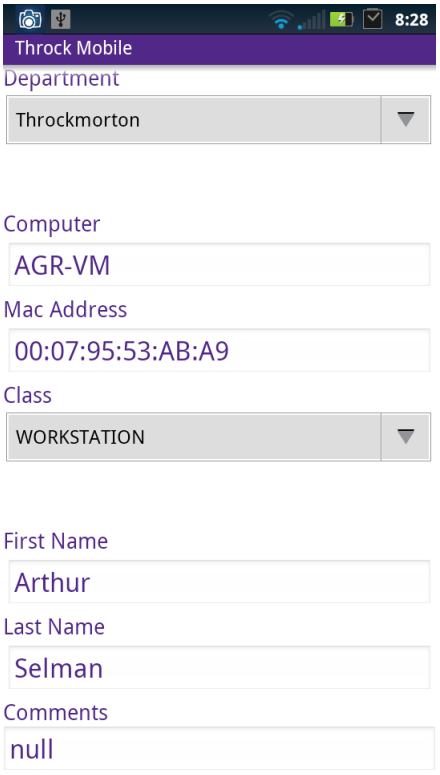

Figure38: Edit Ipaddress (bottomview**)**

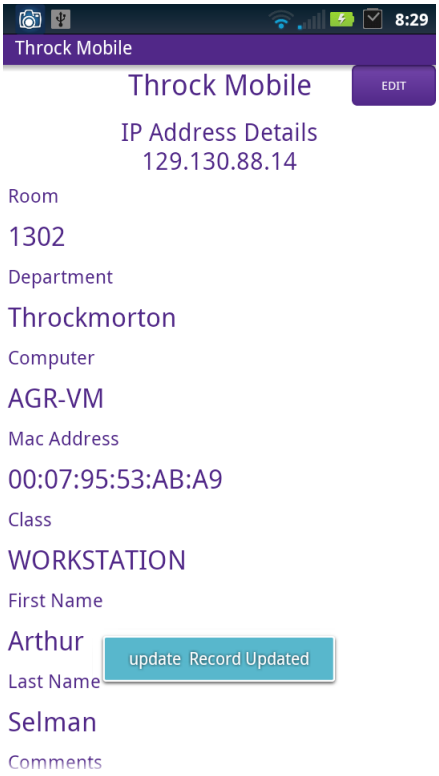

Figure39: IpAddress Details after update

On click of Edit button on Figure36 loads the Figure 37 and 38 so that user can edit the ipaddress details. From Figure 36, Room of Ipaddress is 1302B, I have updated it to 1302 as per the figures 37 and 38. Figure 39 is the resultant screen on successful update of Ipaddress Details.

6.1.9 Look up Asset:

Figure 40 and 41 lets you look up for an asset. The text field is auto populated with data that was fetched based on the database selected on the preferences screen. I have selected the agronomy department in the preferences screen from Figure 9. So the text field is populated with assets present in agronomy. Figure42 displays detailed information about the asset selected in Figure 40. Figure43, UpdateAsset Details, lets you update the information of an asset if needed.

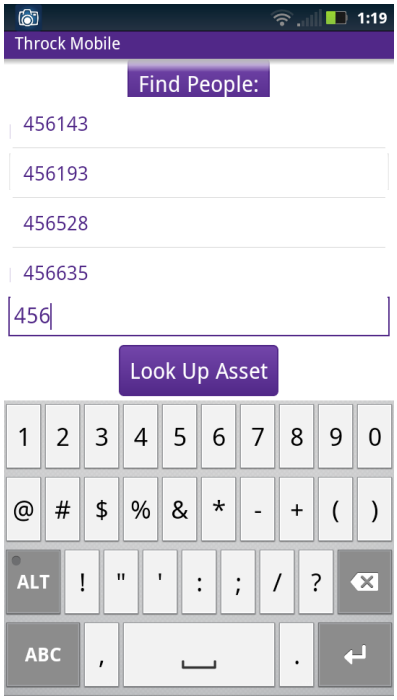

Figure 40: LookUp Asset 1

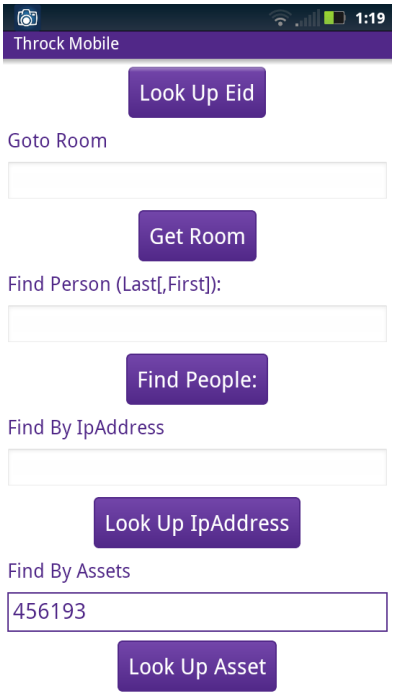

Figure 41: LookUp Asset 2

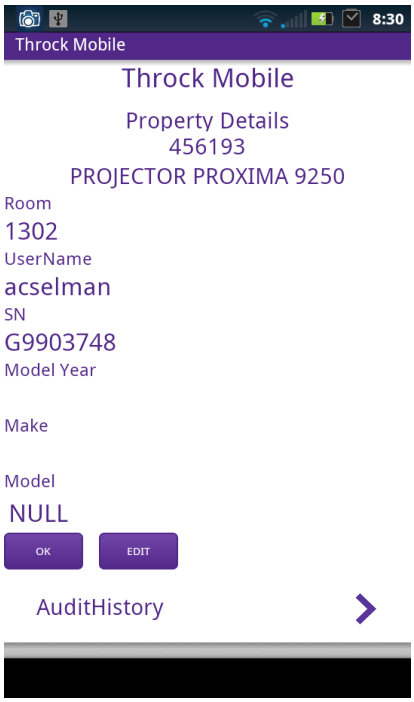

Figure 42: Asset Details

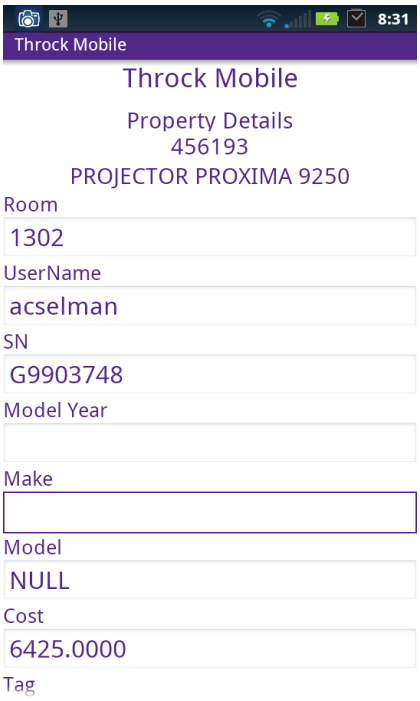

Figure 43: UpdateAsset Details(Top)

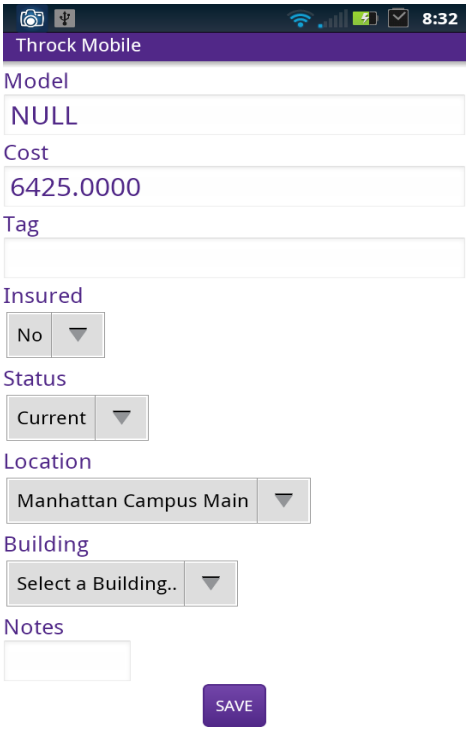

Figure 44: UpdateAsset Details(Bottom**)**

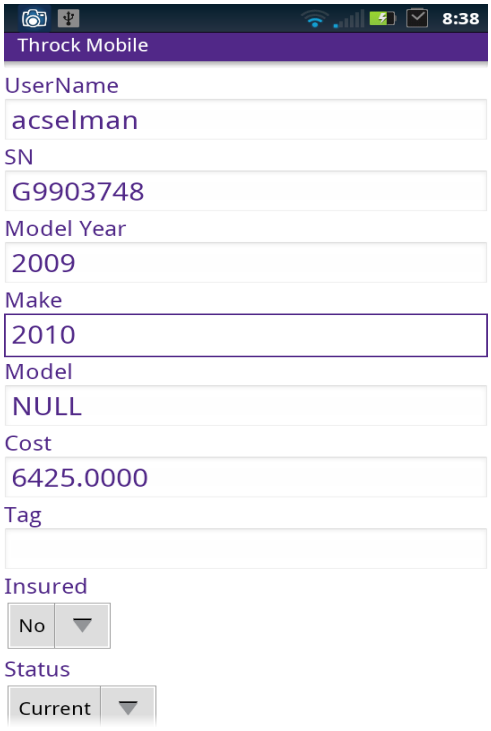

Figure 45: Updating Make and Model of Asset

Figure 46: Successful Update of Asset

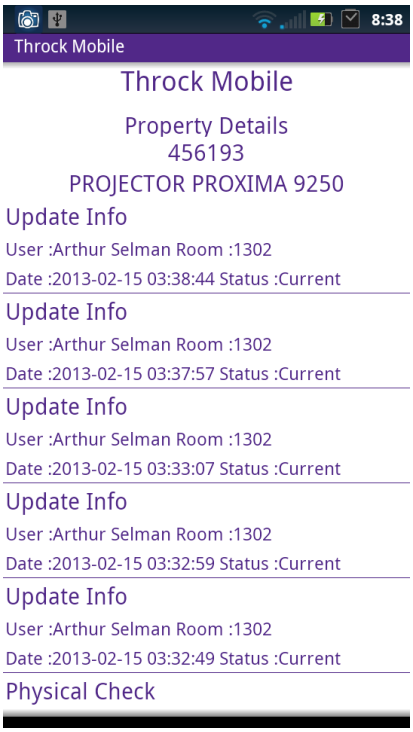

Figure 47: Audit History of Asset

Figure47 gives an overview of audit history of an asset which contains data about when the asset was last updated(Update info) or checked(Physical check).

6.1.10 Bar code Scanning:

Figure 48 lets you scan the barcode of an asset and check for details. You can even enter the details present in the barcode manually. Please see below for flow of activities of barcode scanning of an asset. Figure48 is loaded on click of BarcodeScanning on Menu Screen of Figure 7.

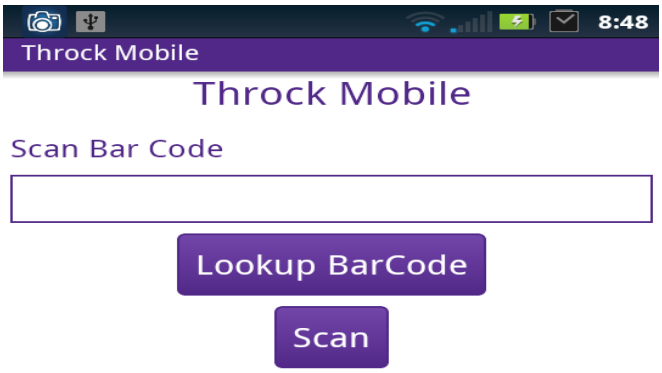

Figure 48: BarCode Activity

Figure 49: Scanning a barcode(129964)

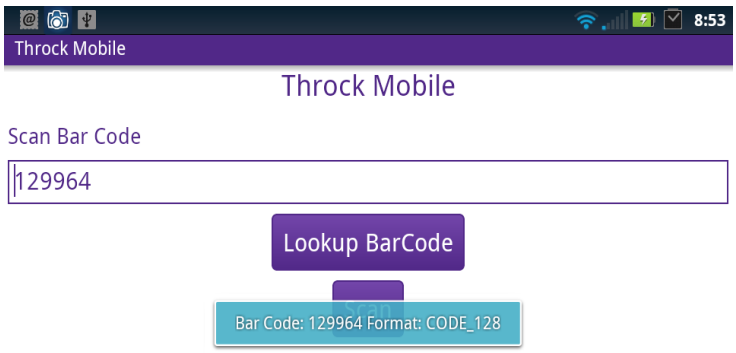

Figure 50: Barcode Result is fetched on to the textview

Figure 51: Asset Details of scanned barcode

6.1.11 Add new asset:

When the admin buys any new inventory, he needs to update it in the system, add new asset accomplishes this task of adding a new asset. He can choose whether it is a departmental asset or a university related asset as in Figure52.

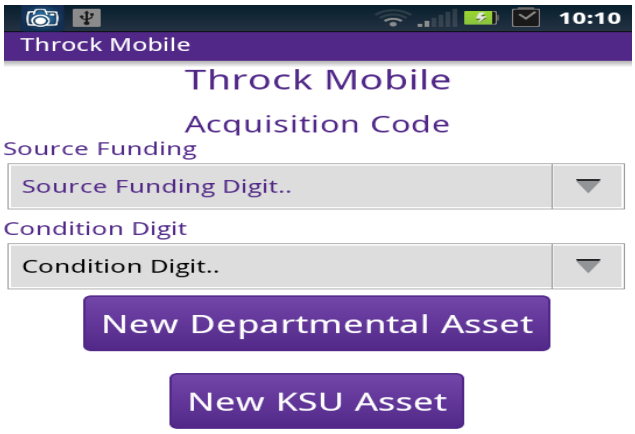

Figure 52: Add New Asset

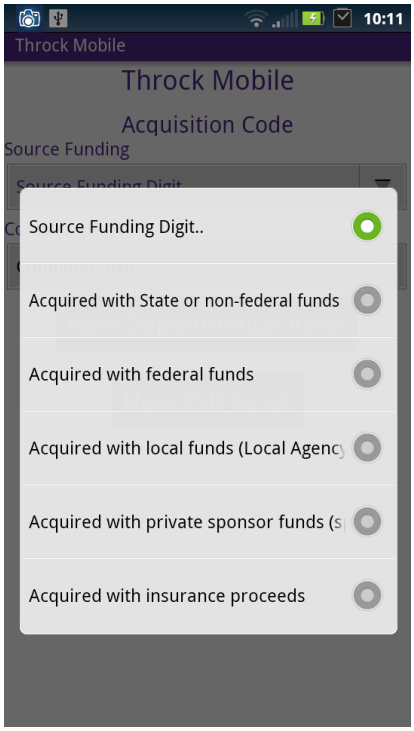

Figure 53: Select Funding Source of Asset

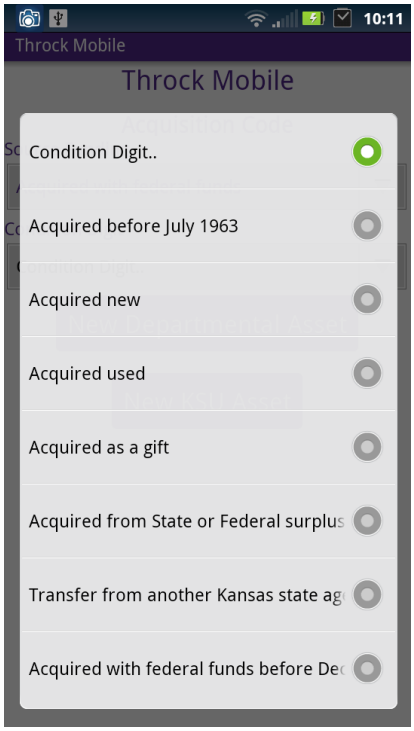

Figure 54: Condition of Asset

Figure55: Asset Details Page with empty fields after creating a new Departmetnal Asset.

7. Testing

7.1 Unit Testing

Unit Testing is a method by which individual units of source is tested for correctness And it is used for testing each and every functionality of an activity without interacting with other activities. Below is the detailed explanation of different unit tests.

Sr. No	Test Case	Expected Result	Pass/Fail
1	On load of Menu Screen	List view of features are displayed	Pass
2	On click of Rooms in UserDetails Page	Display list of rooms in a pop up window	Pass
3	On click of Assets in UserDetails Page	Display list of assets in a pop up window	Pass
4	On click of Ipaddresses in UserDetails Page	Display list of ipaddresses in a pop up window	Pass
5	On click of Phones in UserDetails Page	Display list of phone numbers along with icons	Pass
6	On click of button Ok in AssetChecking Details	Audit History of the asset will be updated with physical check info	Pass
7	On click of Save in Update AssetChecking Details	Displays a toast about successful update.	Pass
8	On click of Save in EditIpDetails	Displays a toast about successful update.	Pass
9	On click of Computers in	Show a popup window with list of	Pass

	RoomDetails Page	computers allocated to the room.	
10	On click of Users in RoomDetails Page	Show a popup window with list of users allocated for the room.	Pass
11	On click of database in preferences screen	Opens a spinner with two databases like agronomy and horticulture	Pass
12	On click of server in preferences screen	Opens up a pop window which has a text field to enter a ip address of server	Pass
13	On click of Webservice in preferences screen	Opens up a pop window which has a text field to enter a location of webservice	Pass

Table 2: Unit Test Cases

7.2 Integration Testing

Integration testing depicts the navigation of activity to another activity with help intents . Below is the detailed explanation of different integration test cases.

Sr. No	Test Case	Expected Result	Pass/Fail
1	On click of BarCode Scanning in Menu	Navigates to BarCode Scanning Activity where you can lookup barcode and open up barcode scanner	Pass
2	On Click of Scan button in	Opens up a barcode scanner to scan a	Pass

	BarcodeSaccning Activity	barcode and fetch the result	
3	On Click of LoopUp BarCode in BarCode Scanning Activity	Navigates to AssetChecking Details Activity which has data about a barcode which was scanned	Pass
4	On click of LookUp: User/Room in Menu	Navigates to LookUp Page where you look up eid, room, asset and ipaddress	Pass
5	On click of LookUp: User/Room in Menu	Navigates to LookUp Page where you look up eid, room, asset and ipaddress	Pass
6	On Click of LookUp Eid in LookUp page	Navigates to UserDetails Page with data fetched based on the field entered in textview	Pass
7	On Click of LookUp Room in LookUp page	Navigates to RoomDetails Page with data fetched based on the field entered in textview	Pass
8	On Click of LookUp Ipadress in LookUp page	Navigates to IpAddress Details Page with data fetched based on entered ipaddress	Pass

9	On Click of LookUp Asset in LookUp page	Navigates to Asset Details Page with data fetched based on the asset number entered in textview	Pass
10	On Click of Edit in Asset Details Page	Navigates to Update AssetDetails Page where you can edit the fields	Pass
11	On Click of Edit in Ip Address Details Page	Navigates to Update IpAdress Details Page where you can edit the fields	Pass
12	On Click of AddRoom in User Details Page	Navigates to AddRoom page where you can mentioned the details	Pass
13	On Click of Add Phone in User Details Page	Navigates to AddPhone Page	Pass
14	On Click of AddUser in Room Details Page	Navigates to Add User Page	Pass
15	On Click of Preferences in Menu Page	Navigates to Preferences Page where you can edit preferences	Pass
16	On Click of Reuest Access in Menu	Opens up an email client to send email	Pass
17	On Click of AddNew Asset	Navigates to Add new Asset Page	Pass

Table 3: Integration Test Cases

7.3 Performance Testing

Performance testing has been done using JMeter to measure the responsiveness of the server that is located in ThrockMorton building to users request. It is tested with varying users requests at a given time. The test has been done on a WIFI network with a speed of 3MBPS.

The configuration of server where the database is located:

Operating System: Windows XP

Processor: Intel® Pentium D CPU @ 3.00GHz

RAM: 2GB

JMeter is installed in the client system where the application is running on an emulator.

The configuration of client:

Operating System: Windows 7

Processor: Intel® Core 2 Duo CPU T6500@ 2.15GHz

RAM: 4GB

The fetching of Asset Details by scanning a barcode is main feature of the application. This Asset Details fetching also happens during Lookup Asset, Lookup Asset of particular user or room functionalities. As Asset Details plays an important role, JMeter testing is done on Asset lookup.

The web service (php) is tested for measuring the performance as it interacts with database. The performance of the application solely depends on how fast data is being fetched as the display of it will happen in less time. The Performance is measured by varying number of users requesting data of Asset Details by sending an asset through a query string to web service. The application makes a HTTP POST request to the server. So in Jmeter, HTTP request is being sent by Thread group of users.

The graph below depicts the variations of webservice script for 2 Http users request.

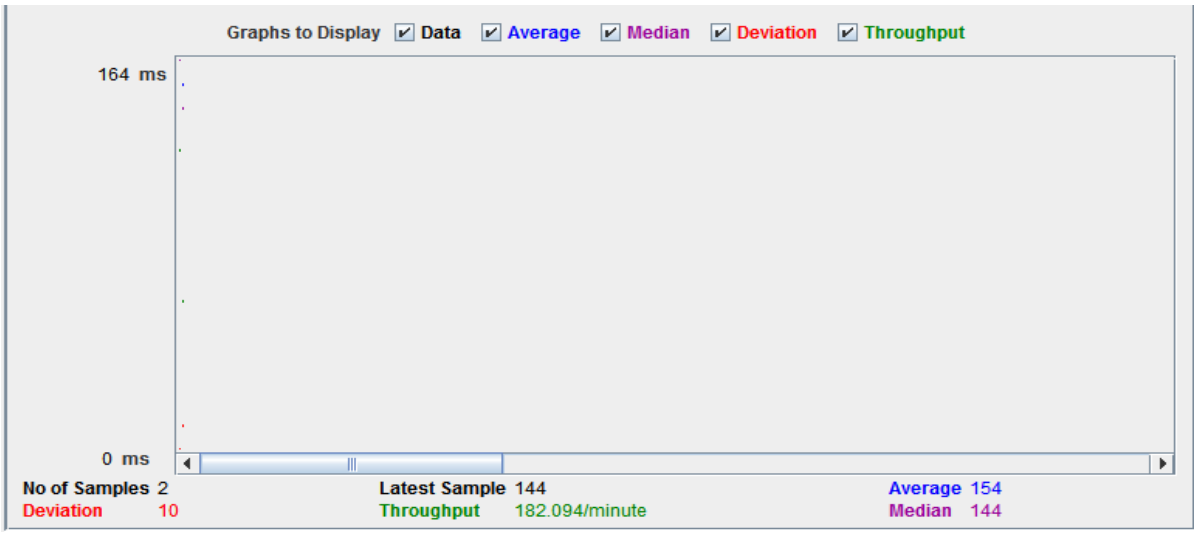

Figure56: Performance Testing of LookUpAsset on 2 requests

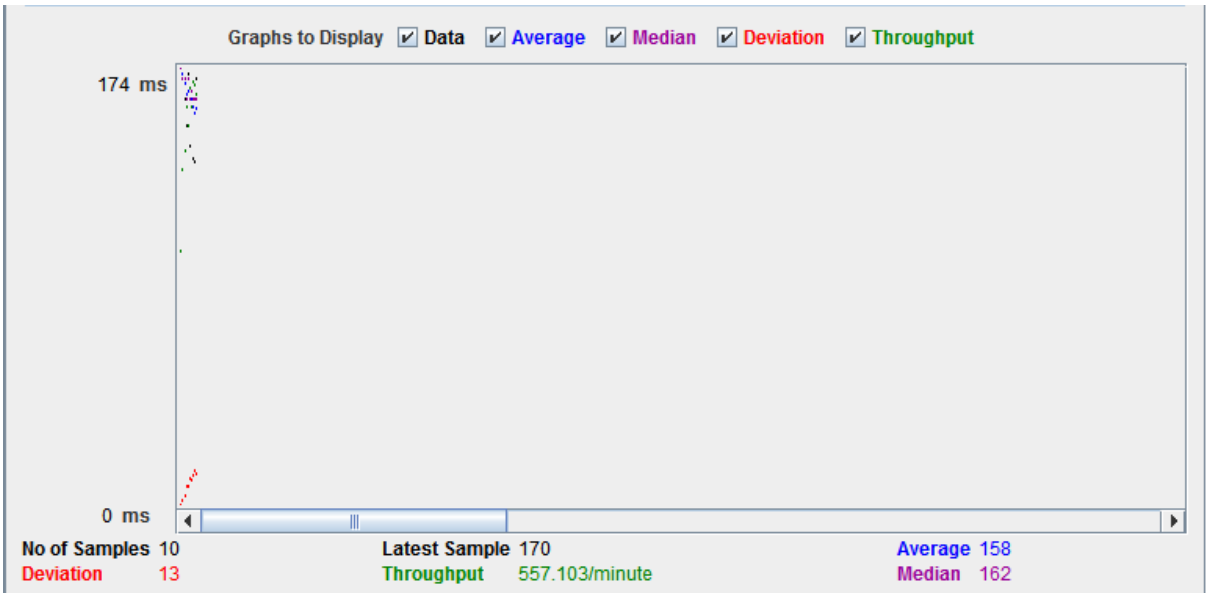

Figure57: Performance Testing of LookUpAsset on 10 requests

From the figure56, we can say that it takes approximately .15 seconds to service a user request with a throughtput of 182 requests per minute.

Figure57 shows the graph for 10 requests. It approximately takes around .14 seconds to service a user request with a throughtput of 557 requests per minute.

The above performance is good and consistent for 2-10 users. Each and every request was served properly. The application might take little longer when users vary.

7.3.1 Testing Analysis

The average performance of the application is because of the accessing multiple tables from multiple databases for a single request. Every request sent to the web service is serviced following the below strict procedure:

- Data fetched from the query string of the request is stored in local variables.
- After this the device id is validated by checking its existence in database.
- Then a function name is resolved based on the command in the query string.
- There are five databases which the webservice refers to fetch the data from. Resolving which database to access is done initially in the function based on the dbPref field from the query string. Some functions might access other databases and fetch data from its tables.
- Null checks are done on the data in respective function before executing a query to fetch the data.
- Data is fetched from the database
- Encoding the data to JSON
- Send back the JSON object.

8. Conclusion

The application has been successfully implemented and tested on emulator as well as android phones which runs on Android Ginger Bread and IceCream Sandwich operating system. Improving the GUI to include the K-State Colours has been very interesting. This interest has motivated me to design new UI components as well using a tool called GIMP. This application development has also increased my skills in UI design, mobile development and adapting quickly to a Software Development Life Cycle.

9. Future Work

The application can further be enhanced to include the features as mentioned below.

- It can be extended to make it compatible with new and upcoming android operating systems such as Jelly bean and KeyLime Pie.
- A camera can be integrated with the application so as to take picture of an asset and store it in the database as an asset information and be able to view the picture in AssetDetails Page.
- The application can further be enhanced to more department by including additional databases in preferences screen and fetch data respectively.
- I would maintain a specific schema for this application to store/fetch data, thus improving performance of the application.

10. References

1. Developer guide
   http://developer.android.com/guide/index.html
2. Android tutorials
   http://www.vogella.com/android.html
3. Android Blog
   http://android-er.blogspot.com/
4. Android Architecture
   http://www.ecnmag.com/articles/2011/07/get-started-android
5. Using Zxing for Barcode Scanning
   http://damianflannery.wordpress.com/2011/06/13/integrate-zxing-barcode-scanner-into-your-android-app-natively-using-eclipse/
6. Using GIMP for UI Design
   http://www.gimp.org/