

**A PROJECT REPORT
ON
“ANDROID SYNCHRONIZATION MANAGER”**

**Submitted to
UNIVERSITY OF PUNE**

In Partial Fulfilment of the Requirement for the Award of

**BACHELOR’S DEGREE IN
COMPUTER ENGINEERING**

BY

Parikshit Ghodake	B8404217
Imran Ahmed	B8404221
Akshay Khole	B8404237
Muneeb Shaikh	B8404259

**DEPARTMENT OF COMPUTER ENGINEERING
JAYAWANTRAO SAWANT COLLEGE OF ENGINEERING
HADAPSAR, PUNE - 28
2011-2012**

**AFFILIATED TO
UNIVERSITY OF PUNE**

**A PROJECT REPORT
ON
“ANDROID SYNCHRONIZATION MANAGER”**

**Submitted to
UNIVERSITY OF PUNE**

In Partial Fulfilment of the Requirement for the Award of

**BACHELOR’S DEGREE IN
COMPUTER ENGINEERING**

BY

Parikshit Ghodake	B8404217
Imran Ahmed	B8404221
Akshay Khole	B8404237
Muneeb Shaikh	B8404259



**DEPARTMENT OF COMPUTER ENGINEERING
JAYAWANTRAO SAWANT COLLEGE OF ENGINEERING
HADAPSAR, PUNE - 28
2011-2012**

AFFILIATED TO



UNIVERSITY OF PUNE

Jayawantrao Sawant College of Engineering

Department of Computer Engineering

Hadapsar, Pune 411028



CERTIFICATE

This is certify that the dissertation entitled

“ANDROID SYNCHRONIZATION MANAGER”

submitted by

Parikshit Ghodake B8404217

Imran Ahmed B8404221

Akshay Khole B8404237

Muneeb Shaikh B8404259

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Engineering) at Jayawantrao Sawant College of Engineering, Pune under the University of Pune. This work is done during year 2011-2012, under our guidance.

Date: / /

(Prof. H. A. Hingoliwala)
Project Guide
HOD, Computer Department

(Prof. M. D. Ingle)
Project Coordinator

(Dr. M. G. Jadhav)
Principal

External Examiner

Acknowledgements

We are profoundly grateful to **Prof. H. A. Hingoliwala** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We would like to express deepest appreciation towards **Dr. M. G. Jadhav**, Principal, Jayawantrao Sawant College of Engineering, **Prof. H. A. Hingoliwala**, Head of Department of Computer Engineering and **Prof. M. D. Ingle**, Project Coordinator whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped me directly or indirectly during this course of work.

Parikshit Ghodake
Imran Ahmed
Akshay Khole
Muneeb Shaikh

ABSTRACT

This project aims at developing the Synchronization Manager for Android based devices. Since there isn't any generic Synchronization Manger for android which works on all platforms (Linux, Mac and Windows); this would help lots of people who work on two or more platforms. This Synchronization Manager would be divided into two parts as server and client. The server would reside on PC and client would be on Android Device. Client will gather information which is to be synchronized.

The current scenario insists users to send information to be backed up to google servers. Once this information is present with google, they have the right to sell/use that data in any way they want according to their Terms of Service. This hampers the users privacy.

The primary purpose of this is to have offline synchronization. At later stages this can be extended to have online synchronization with services such as Ubuntu One for Linux.

For offline syncing USB and Bluetooth may be used. If possible Wi-Fi can also be used. For communicating between server and client SyncML (Synchronization Markup Language) would be used.

Data synchronization solution provides us a complete set of data security and data recovery tools to mop up with many problems. It also provides users and wireless carriers with a simpler, more connected means to accessing digital world.

It ensure us about our valuable data be always reliable, protected and transferable. Its transparent and unique approach seamlessly synchronizes our data, keep backup, and manipulate any data accumulated on our mobile to main or central server. This facilitates different events in case of lost, upgrade or stolen mobile phone. Users of android mobile devices often need to synchronize their mobile devices with desktop computers.

Keywords: Data Synchronization, SyncML

Contents

1	Introduction	2
1.1	Problem Statement	2
1.2	Project Objective	3
1.3	Architecture of the System	3
1.3.1	Current system	3
1.3.2	Proposed System	4
1.4	Licensing	5
1.4.1	GNU General Public License	5
1.4.2	Apache License	6
2	Literature Survey	8
2.1	Android	8
2.2	Synchronization	11
2.3	Software Tools and Language Constructs	13
2.3.1	Android SDK	13
2.3.2	Qt Framework	14
2.3.3	SQLite	16
2.3.4	SyncML	17
3	Software Requirements Specification	18
3.1	Introduction	18
3.1.1	Purpose	18
3.1.2	Document Conventions	18
3.1.3	Intended Audience	19
3.1.4	Product Scope	19
3.2	Overall Description	20
3.2.1	Product Perspective	20
3.2.2	Product Functions	20
3.2.3	Operating Environment	21

3.2.4	Design and Implementation Constraints	21
3.2.5	User Documentation	21
3.2.6	Assumptions and Dependencies	21
3.3	System Features	22
3.3.1	Synchronizing Contacts	22
3.3.2	Synchronizing Messages	22
3.3.3	Synchronizing files	22
3.4	Analysis of Data	22
4	Requirement Analysis	23
4.1	User Side Requirements	23
4.2	System Side Requirements	23
4.3	Hardware and Software Requirements	23
4.4	Non-functional Requirement	24
4.4.1	Security Requirements	24
4.4.2	Performance Requirements	24
4.4.3	Safety Requirements	24
4.5	External Interface Requirements	25
4.5.1	Hardware Interfaces	25
4.5.2	Software Interfaces	25
4.5.3	Communications Interfaces	25
5	System Design	26
5.1	Goals of Design	26
5.2	UML Diagrams	26
5.3	Feasibility Study	31
5.3.1	Current Implementation	31
5.3.2	The Proposed System	31
6	System Testing	32
7	Project Planning	39
7.1	Project Planning Model	39
7.2	Estimation and Efforts	41
7.3	Project Schedule	41
7.4	Timeline Chart	42
7.5	Source Code Management	44

8	Implementation	49
9	Screenshots of Client and Server	54
9.1	Server	54
9.2	Client	58
10	Conclusion and Future Scope	62
10.1	Conclusion	62
10.2	Future Scope	62
	References	62

List of Figures

1.1	Architecture of Current System	3
1.2	Architecture of Proposed System	4
2.1	System Architecture	10
2.2	Data Synchronization	12
2.3	Android SDK	14
2.4	Qt Framework	15
2.5	SQLite	16
3.1	Architecture of Android Synchronization Manager (Client) . . .	20
5.1	Class Diagram	28
5.2	Use Case Diagram	29
5.3	Sequence Diagram	30
7.1	Planning and Scheduling - Part A	43
7.2	Planning and Scheduling - Part B	43
7.3	AndroSync Repository at GitHub	44
7.4	Code Frequency	45
7.5	Commits	45
7.6	Contributors	46
7.7	Punchcard	46
7.8	Impact - Part A	47
7.9	Impact - Part B	47
7.10	Impact - Part C	48
7.11	Impact - Part D	48
8.1	Important State Paths of an Activity	50

Chapter 1

Introduction

1.1 Problem Statement

Today smart phones are not exclusive property of early adopters or IT professionals. Global Smart phone shipments grew a relatively healthy 43 per cent year-over-year to reach 600 million units in Q2 2010. According to comScores report, 234 million Americans subscribed to mobile phone plans in January 2010. Of these 42.7 million owned internet accessible smart phones, this represented an 18 per cent increase over the three months ended in October.

Important progress in mobile computing should start with straightforward local convergence of Smartphones and computer infrastructure via simplistic mobile use model. This use model would enhance the current synchronization schemes by bypassing the synchronized data over proprietary services and giving the user the full control and full accessibility of his/her own data both on the smart phone, as well on his very own synchronization server. It is not always advisable for users to use synchronization services provided by Google for android devices.

These services may not ensure full privacy of all the data of the user. This data is stored on the Google clouds. Also, today the security issues in cloud computing have not been fully resolved. Hence it cannot be said our data remains fully secure when it is on Google premises. Synchronization can be done by various means by the use of this product. Offline and online mode synchronization is possible. Online mode can be used when user is on-the-go and offline synchronization may be used when user is at the office or at home.

1.2 Project Objective

The project addresses the idea of android smartphones being synchronized in an online and offline manner. A server application in the form of a desktop application and a client application in the form of an android application is to be developed to achieve this objective.

The basic objectives of the project include:

- Setting up a secure online connection of the mobile phone to the server.
- Setting up interfacing techniques to connect the mobile phones to the PC via a USB cable, Bluetooth link or Wi-Fi.
- Devising a protocol which will control the authentication of the device, flow of data in and out of the smartphone and the server.
- Managing user account on the server application.
- Developing the application such that it is cross-platform compatible i.e. it can be deployed on a PC running any OS Linux, Mac OSX or Windows.

1.3 Architecture of the System

1.3.1 Current system

The current system being used for synchronization involves the data being synchronized to be synced via a Google account the user has to maintain. This means data will have to be given to a third-party provider like Google when it actually not really necessary to do so.

This scenario is illustrated in figure 1.1.



Figure 1.1: Architecture of Current System

It is not always possible to store sensitive information on third-party premises like Google. This sensitive information can be in the form of phone logs, SMS, contacts, e-mails, notes, calendar schedules, etc.

1.3.2 Proposed System

The proposed system enables Android mobile device to be synchronized by the user using this product such that all data may be synchronized in a private mutually exclusive manner. This is illustrated in figure 1.2.

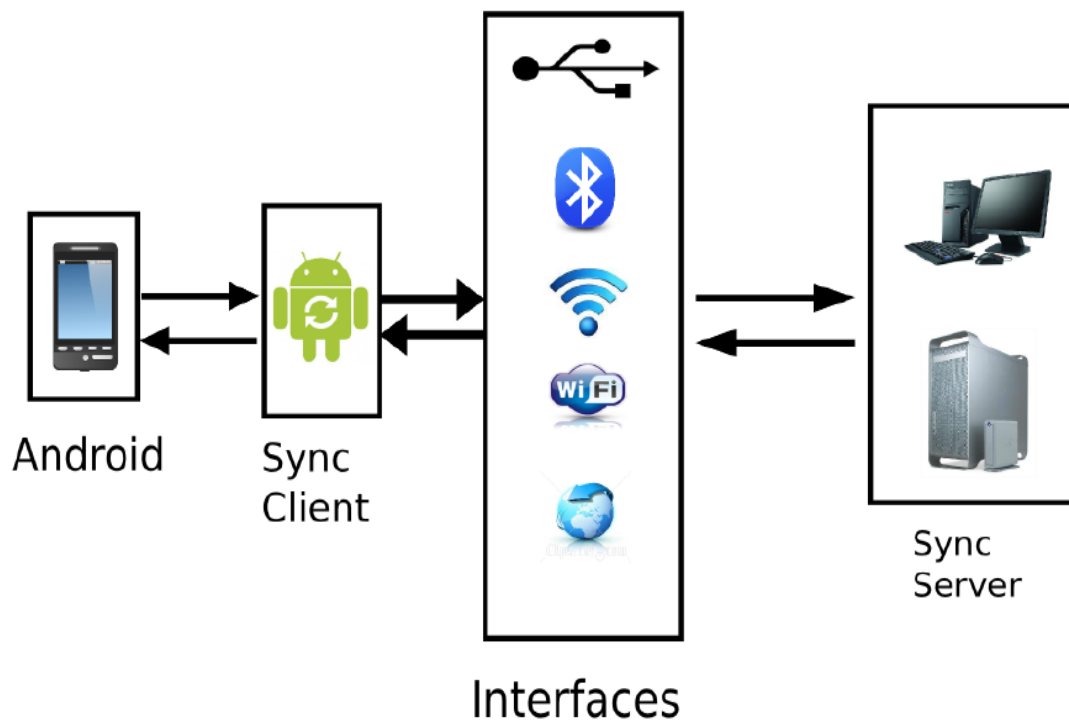


Figure 1.2: Architecture of Proposed System

In this system, the android smartphone is connected to the server and after the connections is established, synchronization may be started using any mode- offline or online i.e. using a Cable or Wi-Fi or over a secure internet connection.

1.4 Licensing

1.4.1 GNU General Public License

The GNU General Public License (GNU GPL or simply GPL) is the most widely used free software license, originally written by Richard Stallman for the GNU Project.

The GPL is the first copyleft license for general use, which means that derived works can only be distributed under the same license terms. Under this philosophy, the GPL grants the recipients of a computer program the rights of the free software definition and uses copyleft to ensure the freedoms are preserved, even when the work is changed or added to. This is in distinction to permissive free software licenses, of which the BSD licenses are the standard examples.

The text of the GPL is not itself under the GPL. The license's copyright disallows modification of the license. Copying and distributing the license is allowed since the GPL requires recipients to get "a copy of this License along with the Program". According to the GPL FAQ, anyone can make a new license using a modified version of the GPL as long as he or she uses a different name for the license, does not mention "GNU", and removes the preamble, though the preamble can be used in a modified license if permission to use it is obtained from the Free Software Foundation (FSF).

Version 2

According to Richard Stallman, the major change in GPLv2 was the "Liberty or Death" clause, as he calls it Section 7. This section says that if somebody has restrictions imposed that prevent him or her from distributing GPL-covered software in a way that respects other users' freedom (for example, if a legal ruling states that he or she can only distribute the software in binary form), he or she cannot distribute it at all. The hope is, that this will make it less tempting for companies to use patent threats to require a fee from the free software developers.

By 1990, it was becoming apparent that a less restrictive license would be strategically useful for the C library and for software libraries that essentially did the job of existing proprietary ones; when version 2 of the GPL (GPLv2) was released in June 1991, therefore, a second license the Library General Public License was introduced at the same time and numbered with version 2 to show that both were

complementary. The version numbers diverged in 1999 when version 2.1 of the LGPL was released, which renamed it the GNU Lesser General Public License to reflect its place in the philosophy.

Terms and Conditions

The terms and conditions of the GPL must be made available to anybody receiving a copy of the work that has a GPL applied to it ("the licensee"). Any licensee who adheres to the terms and conditions is given permission to modify the work, as well as to copy and redistribute the work or any derivative version. The licensee is allowed to charge a fee for this service, or do this free of charge. This latter point distinguishes the GPL from software licenses that prohibit commercial redistribution. The FSF argues that free software should not place restrictions on commercial use, and the GPL explicitly states that GPL works may be sold at any price.

The GPL additionally states that a distributor may not impose "further restrictions on the rights granted by the GPL". This forbids activities such as distributing of the software under a non-disclosure agreement or contract. Distributors under the GPL also grant a license for any of their patents practiced by the software, to practice those patents in GPL software.

The fourth section for version 2 of the license and the seventh section of version 3 require that programs distributed as pre-compiled binaries are accompanied by a copy of the source code, a written offer to distribute the source code via the same mechanism as the pre-compiled binary, or the written offer to obtain the source code that you got when you received the pre-compiled binary under the GPL. The second section of version 2 and the fifth section of version 3 also require giving "all recipients a copy of this License along with the Program". Version 3 of the license allows making the source code available in additional ways in fulfillment of the seventh section. These include downloading source code from an adjacent network server or by peer-to-peer transmission, provided that is how the compiled code was available and there are "clear directions" on where to find the source code.

1.4.2 Apache License

The Apache License is a free software license authored by the Apache Software Foundation (ASF). The Apache License requires preservation of the copyright notice and disclaimer.

All software produced by the ASF or any of its projects or subjects is licensed according to the terms of the Apache License. Some non-ASF software is also licensed using the Apache License. As of November 2010, over 6000 projects located at SourceForge.net were available under the terms of the Apache License. In a blog post from May 2008 Google mentioned that 25,000 out of the 100,000 projects then hosted on Google Code were using the Apache License.

Licensing Conditions

Like any free software license, the Apache License allows the user of the software the freedom to use the software for any purpose, to distribute it, to modify it, and to distribute modified versions of the software, under the terms of the license.

The Apache License is permissive, so it does not require modified versions of the software to be distributed using the same license. In every licensed file, any original copyright, patent, trademark, and attribution notices in redistributed code must be preserved (excluding notices that do not pertain to any part of the derivative works); and, in every licensed file changed, a notification must be added stating that changes have been made to that file.

If a NOTICE text file is included as part of the distribution of the original work, then derivative works must include a readable copy of these notices (again, excluding notices not pertaining to any part of the derivative work), in at least one of three places: within a NOTICE text file distributed as part of the derivative works, within the source form or documentation, or within a display generated by the derivative works (wherever such third-party notices normally appear). The contents of the NOTICE file do not modify the license, as they are for informational purposes only, and adding more attribution notices as addenda to the NOTICE text is permissible, provided that these notices cannot be understood as modifying the license. Modifications may have appropriate copyright notices, and may provide different license terms for the modifications.

Unless explicitly stated otherwise, any contributions submitted by a licensee to a licensor will be under the terms of the license without any terms and conditions, but this does not preclude any separate agreements with the licensor regarding these contributions.

Chapter 2

Literature Survey

2.1 Android

Google usually refers to the Android OS as a **software stack**. Each layer of the stack groups together several programs that support specific operating system functions. These layers are illustrated in Figure 2.1.

The base of the stack is the **kernel**. Google used the Linux version 2.6 OS to build Android's kernel, which includes Android's memory management programs, security settings, power management software and several hardware drivers. The next level of software includes Android's **libraries**. Libraries are a set of instructions that tell the device how to handle different kinds of data. Android runtime layer includes a set of core **Java** libraries – Android application programmers build their apps using the Java programming language. It also includes the Dalvik Virtual Machine. The next layer is the **application framework**. This includes the programs that manage the phone's basic functions like resource allocation, telephone applications, switching between processes or programs and keeping track of the phone's physical location. Application developers have full access to Android's application framework.

- a. Application Framework is used to write applications for Android. Unlike other embedded mobile environments, Android applications are all equal, for instance, applications which come with the phone are no different than those that any developer writes. The framework is supported by numerous open source libraries such as openssl, sqlite and libc. It is also supported by the Android core libraries. From the point of security, the framework is based on UNIX file system permissions that assure applications have only those abilities that mobile phone owner gave them at install time.

- b. Dalvik virtual machine is extremely low-memory based virtual machine, which was designed especially for Android to run on embedded systems and work well in low power situations. It is also tuned to the CPU attributes. The Dalvik VM creates a special file format (.DEX) that is created through build time post processing. Conversion between Java classes and .DEX format is done by included dx tool.
- c. Integrated browser, WebKit is chosen as an open source web browser. Google added a two pass layout and frame flattening. Two pass layout loads a page without waiting for blocking elements, such as external CSS or external JavaScript and after a while renders again with all resources downloaded to the device. Frame flattening converts founded frames into single one and loads into the browser. These features increase speed and usability browsing the internet via mobile phone.
- d. Optimized graphics as Android has 2D graphics library and 3D graphics based on OpenGL ES 1.0, great applications like Google Earth and spectacular games like Second Life are seen, which come on Linux version. At this moment, the shooting legendary 3D game Doom was presented using Android on the mobile phone.
- e. SQLite is used, which is extremely small (500kb) relational database management system that is integrated in Android. It is based on function calls and single file, where all definitions, tables and data are stored. This simple design is more than suitable for a platform such as Android.

There are a number of hardware dependent features, for instance, a huge media and connections support, GPS, improved support for Camera and simply GSM telephony. A great work was done for the developers to start work with Android using device emulator, tools for debugging and plugin for Eclipse IDE.

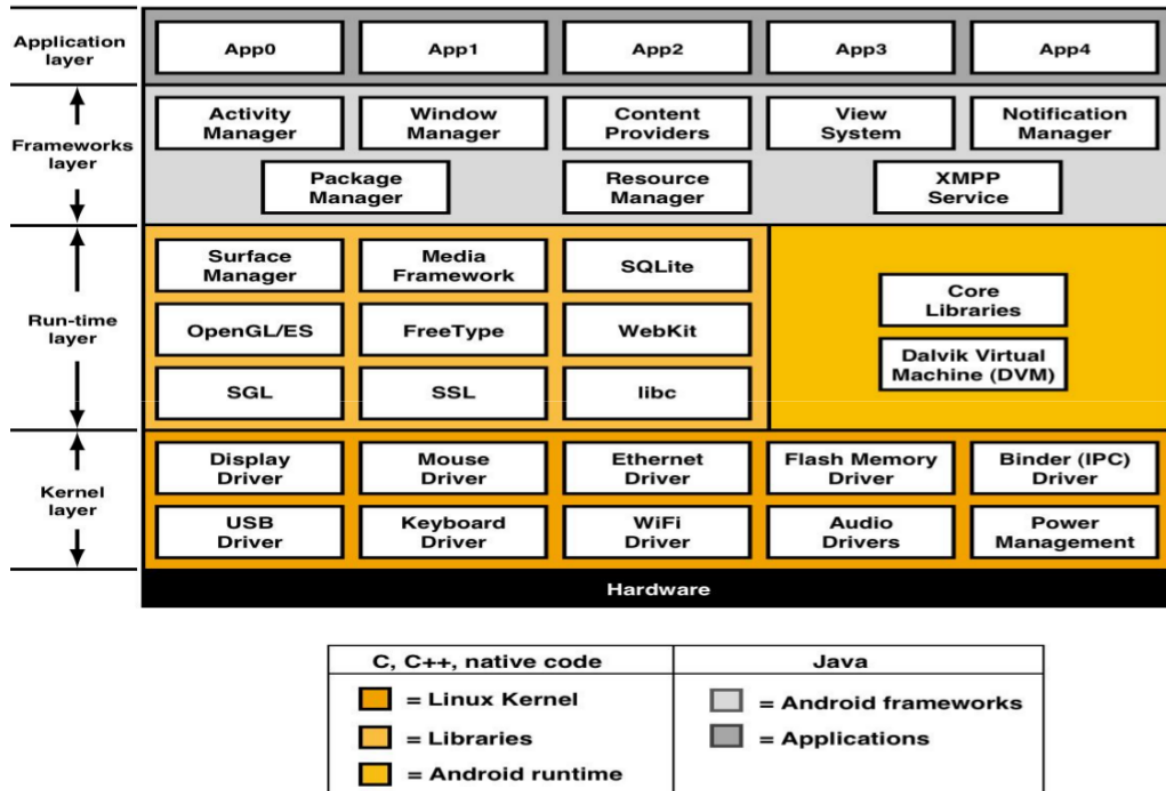


Figure 2.1: System Architecture

Android Architecture is based on Linux 2.6 kernel. It helps to manage security, memory management, process management, network stack and other important issues. Therefore, the user should bring Linux in his mobile device as the main operating system and install all the drivers required in order to run it. Android provides the support for the Qualcomm MSM7K chipset family. For instance, the current kernel tree supports Qualcomm MSM 7200A chipsets with stable version Qualcomm MSM 7200, which includes major features:

- WCDMA/HSUPA and EGPRS network support
- Bluetooth 1.2 and Wi-Fi support
- Digital audio support for mp3 and other formats
- Support for Linux and other third-party operating systems
- Java hardware acceleration and support for Java applications
- Qcamera up to 6.0 megapixels
- gpsOne solution for GPS
- and lots of other features.

2.2 Synchronization

The fast paced technology has changed the world as well as its inhabitants. In Today's global village there is no need to grab the telephone receiver and dial a specific number to transmit voice through cables merely to hear the voice of a beloved. Now each of us carries our own handsets with a built-in phonebook and text messages. The facilities like Wi-Fi and internet have further improved the standard of communications by cutting down expenditure and increasing availability.

Regardless of the location, the world of web can be accessed through handsets, laptops and tablet PCs. These modern times give us the necessity to maintain highly important data on these portable devices so that we can access them on the go. But carrying around this important data brings with it the risk of data getting corrupted due to hardware failure, natural elements like rain spoiling our gadgets or the devices hard drive failing due to shocks during transit.

Hence to be safe and sure against such incidents it has become extremely to back up our data after short intervals. Currently, the features of synchronization are available in android devices, but they have a catch. These services are provided by third-party software managers for e.g., Google.

To back up our data, the data must be first synchronized with our Google account. This means sending our extremely important data to the Google Clouds. This directly affects the security and privacy aspects of the user. The data being stored on the Google cloud may not be guarded securely. It is not guaranteed data will be safe and not private.

Hence to overcome these restrictions, we have proposed a system where each user can synchronize his/her phone with his/her own private server running anywhere. This will ensure that all the data remains on our own private devices and no third-party can get hold of this data.

This means after synchronizing our private data like emails, SMS messages, contacts, calendar entries, notes, documents, folders etc., all the data remains on the users server itself. The server can be a Desktop Computer or a laptop connected to the internet. Also the android devices can be synchronized in an offline manner.

Offline sync can be done using Wi-Fi, Bluetooth or USB cable. Online sync will be done via a secure connection over the internet. Using this product, the android

user can be ensured all their data remains on their own premises, no unnecessary sharing will be done to third-parties, multiple copies of data can be maintained over multiple servers for additional backup, if necessary.

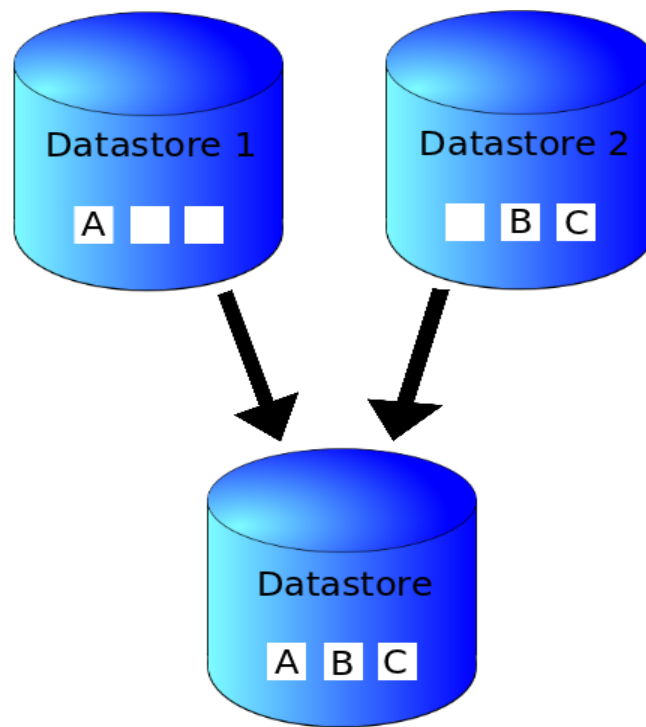


Figure 2.2: Data Synchronization

2.3 Software Tools and Language Constructs

Android synchronization manager will be designed with the following software tools and language constructs-

2.3.1 Android SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.4.9 or later, Windows XP or later. The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit Java and XML files then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible only to root user for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executables), resource files, etc.

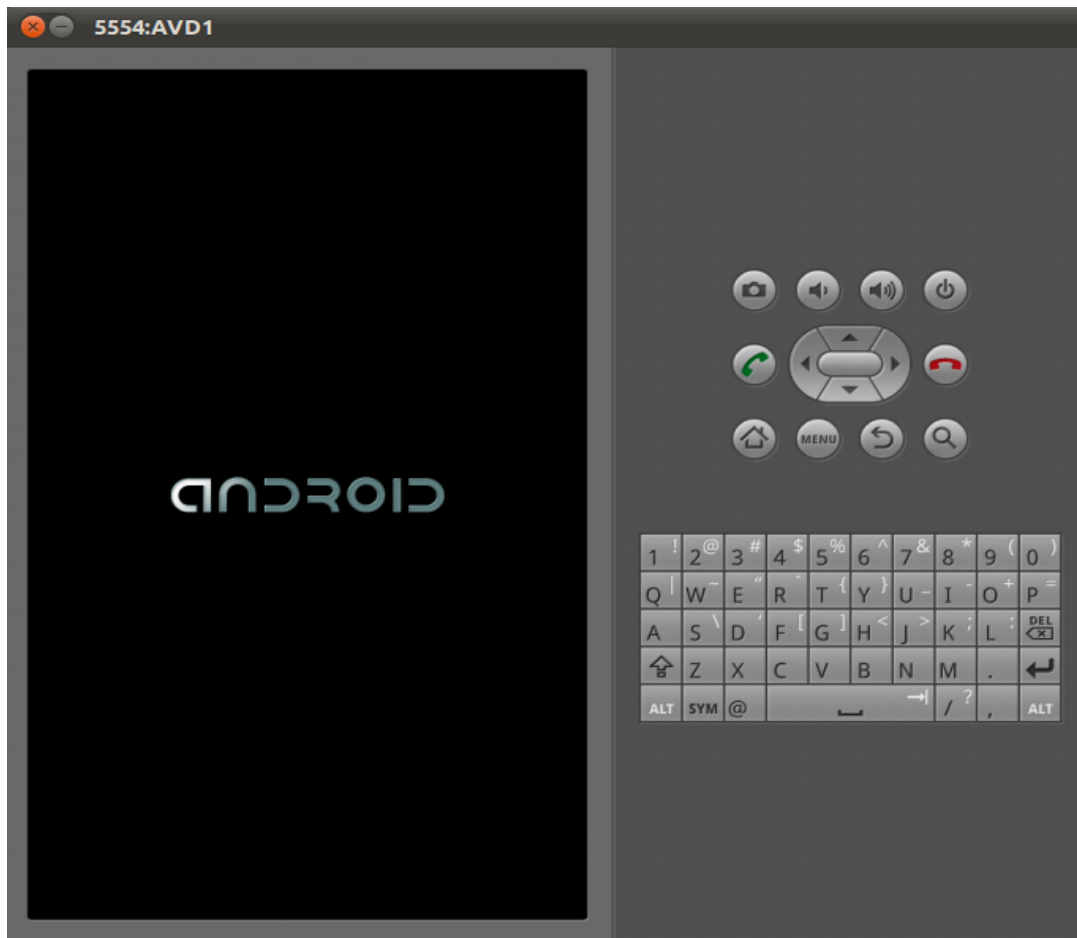


Figure 2.3: Android SDK

2.3.2 Qt Framework

Qt is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI) (in which cases Qt is classified as a widget toolkit), and also used for developing non-GUI programs such as command-line tools and consoles for servers.

Qt is developed by an open source project, the Qt Project, involving developers as individuals and from firms working to advance Qt, such as Nokia, Digia, and others. Before the launch of the Qt Project, it was produced by Nokia's Qt Development Frameworks division, which came into being after Nokia's acquisition of the Norwegian company Trolltech, the original producer of Qt. In February 2011 Nokia announced its decision to drop Symbian technologies and base their future smartphones on Microsoft platform instead. One month later Nokia announced the sale of Qt's commercial licensing and professional services to Digia PLC, although Nokia will remain the main development force behind the framework. On May 9, it was announced on the Qt Labs website that the groundwork was being laid for the

next major version of Qt, with the expectation that Qt 5 would be released in 2012.

Qt uses standard C++ but makes extensive use of a special code generator (called the Meta Object Compiler, or moc) together with several macros to enrich the language. Qt can also be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, thread management, network support, and a unified cross-platform application programming interface (API) for file handling.

Distributed under the terms of the GNU Lesser General Public License (among others), Qt is free and open source software. All editions support many compilers, including the GCC C++ compiler and the Visual Studio suite.

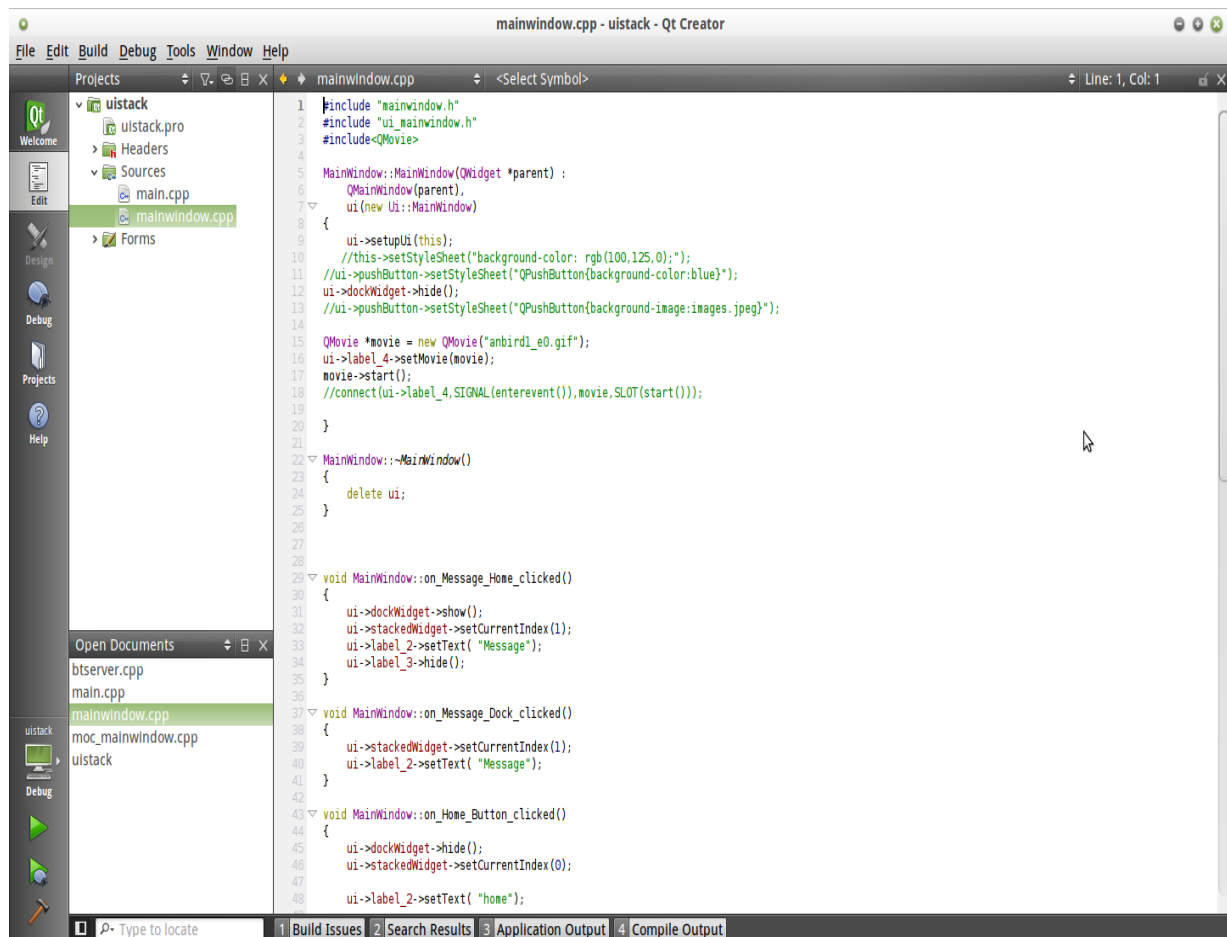


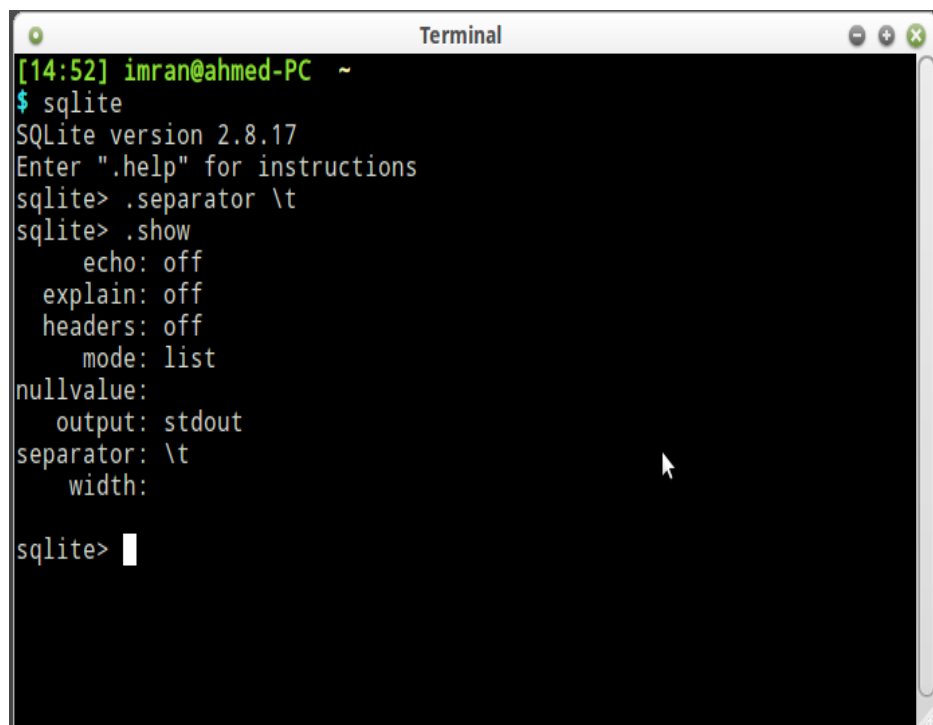
Figure 2.4: Qt Framework

2.3.3 SQLite

SQLite is an ACID-compliant embedded relational database management system contained in a small (275 kB) C programming library.

SQLite implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it. SQLite read operations can be multitasked, though writes can only be performed sequentially. The source code for SQLite is in the public domain.

SQLite is a popular choice for local/client storage on web browsers. It has many bindings to programming languages. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems, among others.



```
Terminal
[14:52] imran@ahmed-PC ~
$ sqlite
SQLite version 2.8.17
Enter ".help" for instructions
sqlite> .separator \t
sqlite> .show
      echo: off
      explain: off
      headers: off
      mode: list
      nullvalue:
      output: stdout
      separator: \t
      width:

sqlite> 
```

Figure 2.5: SQLite

2.3.4 SyncML

SyncML (Synchronization Markup Language) is the former name for a platform-independent information synchronization standard. The project is currently referred to as Open Mobile Alliance Data Synchronization and Device Management. The purpose of SyncML is to offer an open standard as a replacement for existing data synchronization solutions, which have mostly been somewhat vendor-, application- or operating system specific.

SyncML is most commonly thought of as a method to synchronize contact and calendar information (personal information manager) between some type of handheld device and a computer (personal, or network-based service), such as between a mobile phone and a personal computer. The new version of the specification includes support for push email, providing a standard protocol alternative to proprietary solutions like BlackBerry.

Several major companies such as Samsung, Motorola, Nokia, Sony Ericsson, LG, IBM and Siemens AG already support SyncML in their products. Some products by now use SyncML for more generic information synchronization purposes, such as to synchronize project task information across a distributed group of team members. SyncML may also be used as a base for backup solutions.

Chapter 3

Software Requirements Specification

3.1 Introduction

3.1.1 Purpose

This project aims at developing the Synchronization Manager for Android based devices. Since there isn't any generic Synchronization Manager for android which works on all platforms (Linux, Mac and Windows); this would help lots of people who work on two or more platforms. This Synchronization Manager would be divided into two parts as server and client. The server would reside on PC and client would be on Android Device. Client will gather information which is to be synchronized.

3.1.2 Document Conventions

The following are the list of conventions and acronyms used in this document and the project as well, privileges to the software:

- **Client:** Intended users for the software.
- **SQL:** Structured Query Language; used to retrieve information from a database
- **SQL Server:** A server used to store data in an organized format
- **IEEE:** Institute of Electrical and Electronics Engineers
- **Layer:** Represents a section of the project
- **SYNC:** Synchronization
- **User Interface Layer:** The section of the assignment referring to what the user interacts with directly

- **Application Logic Layer:** The section of the assignment referring to the Web Server. This is where all computations are completed
- **Data Storage Layer:** The section of the assignment referring to where all data is recorded
- **Data flow diagram:** It shows the data flow between the entities
- **Use Case:** A broad level diagram of the project showing a basic overview
- **Boolean:** A true/false notation
- **Interface:** Something used to communicate across different mediums

3.1.3 Intended Audience

The intended audiences for this document are:

- The team members of Innovative Android Synchronization Manager.
- The different Android users who are the clients.

This document will be reviewed frequently by the above audiences to check if the different phases of the project are being completed by meeting the given requirements. If there are any changes in the requirements in the course of the project they must be included in this document by making the necessary changes.

3.1.4 Product Scope

The primary purpose of this is to have offline synchronization. At later stages this can be extended to have online synchronization with services such as Ubuntu One for Linux. The software can be made available as a paid service for corporations, users who want to backup their data.

3.2 Overall Description

3.2.1 Product Perspective

Since at the moment there does not exist any generic android client for synchronizing data off-line, this product will simplify synchronizing data across multiple mobile device manufacturers. This is a new self-contained product. It is basically for android users who would like to have a generic client for off-line and on-line synchronization of their data. This data may include information like contacts, calendar entries, messages, notes, audio, video files. It is particularly useful for the careful user who prefers synchronizing data on their own servers rather than a third-party service provider. Also, it will benefit users of android who are concerned about sending data on third-party servers.

The following diagram represents the proposed architecture:

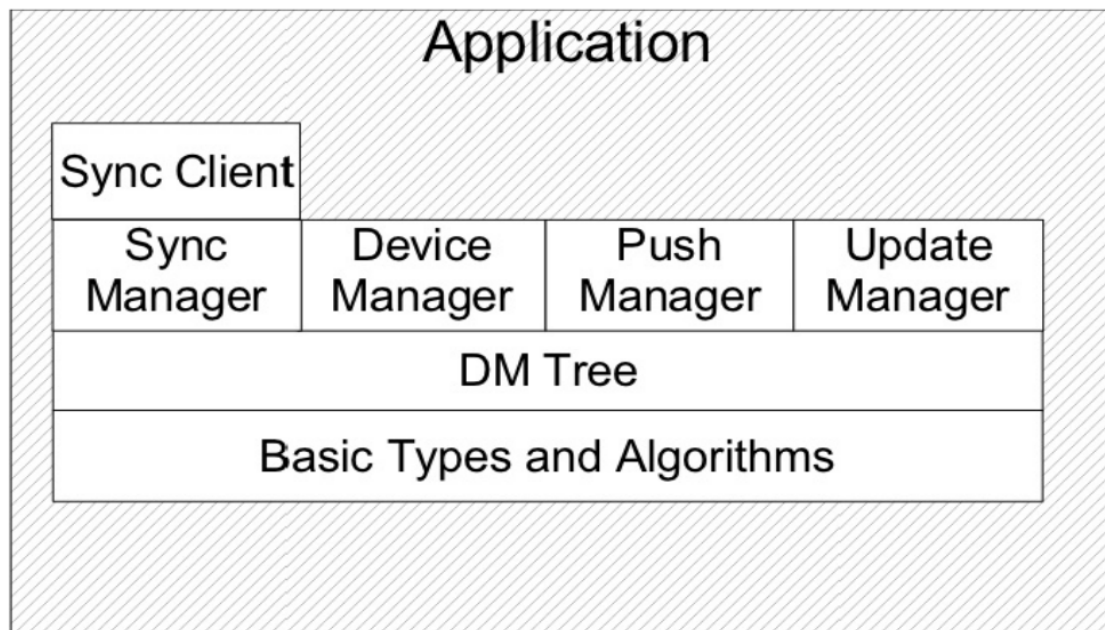


Figure 3.1: Architecture of Android Synchronization Manager (Client)

3.2.2 Product Functions

- Contact synchronization.
- Message synchronization.
- Calendar synchronization.
- Generating Reports.

- Managing logs of various activities in the phone (received/missed/dialed numbers).
- Backup of images, videos and other files.

3.2.3 Operating Environment

- **For Client:** Android OS with v2.1 and above
- **For Server:** Qt 4.7+, SQLite

3.2.4 Design and Implementation Constraints

This is a product primarily being developed for users of a specific mobile platform i.e. Android. Hence users of mobile phones apart from android cannot use the features of this product. Also it may be a challenge to create synchronization of data between an android device and a device running on some other platform. Hence these constraints cannot be overlooked. Fortunately, this product will be designed in a way that it will be extensible for integrating synchronization feature for other platforms of mobile devices.

But taking into consideration the huge number of mobile phone manufacturers, adding this feature may take a considerable amount of time. These additions can be added keeping in view the popularity of the platform this product is being ported into. Also, online synchronization might be slower for additional authentication features like use of SSL and other security features.

3.2.5 User Documentation

User documentation would include the built-in help which will give a step by step guidance to users of the software on how to effectively use this product. Apart from this, an online help system through e-mails or mailing lists can also be set up. The product will be developed with out-most importance being given to simplicity so that minimum developer intervention would be needed for maintenance.

3.2.6 Assumptions and Dependencies

While developing this product it is being assumed that the android platform version used would be v2.1 [and above]. The usage of the same on a lower version

may give unexpected results. But the product will be developed to keep it as cross-platform compatible as possible.

3.3 System Features

3.3.1 Synchronizing Contacts

This feature is primarily why the product is being developed. This feature lets the user backup contacts to the server. It will enable users to keep their server constantly updated with the newly added contacts and their respective data.

3.3.2 Synchronizing Messages

This feature is basically used to backup SMSs and other drafts in the phone on to the server. This feature is included as many users are keen on wanting to save all their conversations.

3.3.3 Synchronizing files

Along with all the data mentioned above, all the files on the users device can also be backed up on the server. This will enable the user to store important information on the server and all the information will be synchronized on a secure communication link.

3.4 Analysis of Data

All the information collected on Android operating system is obtained from developer.android.com. Since Android is an open source, many of the codes are readily available for application development, which has rapidly added to the success of Android Systems. It gives budding engineers a platform for learning and development. The data mentioned on the website developer.android.com explains the changes to be made in the manifestation file for installing a new application. Moreover it explains the procedure of connecting a USB device to Android supported phone. The two modes of android device viz. Host mode and Accessory mode are mentioned on this website. These modes can be used for offline synchronization of the data. The data obtained from this website proves to be tremendously helpful in the android application development for the synchronization process.

Chapter 4

Requirement Analysis

4.1 User Side Requirements

1. User should have an android mobile device running OS version 2.1 or above.
2. User should have Android Application installed within the mobile.
3. User must be able to connect the device using USB cable
4. User must be able to connect device using Wi-Fi.
5. User must have internet plan enabled on his device or use Wi-Fi connection for online synchronization.
6. Interface must be user friendly and attractive.

4.2 System Side Requirements

1. System must have Android based drivers for USB and Wi-Fi adapters.
2. The server side application must be installed on the standard PC.
3. System must be connected to the internet for online synchronization.
4. Interface must be user friendly and attractive.

4.3 Hardware and Software Requirements

1. Android Based Mobile with minimum of:
 - a. 128KB non-volatile memory to run Mobile Information Device (MID).
 - b. 8KB of non-volatile memory for storage of persistent application data.

- c. 2KB of volatile memory to run JVM
- 2. Device should have Wi-Fi connectivity.
- 3. 600mhz processor
- 4. 150 MB of RAM

4.4 Non-functional Requirement

4.4.1 Security Requirements

In our product, security is of out-most importance. Hence we need to implement communication over HTTPS. For that it is required that the user should be in an environment where the HTTPS port is not blocked. In case these ports are not enabled, an http connection will have to be used. This means using http instead of https would lead to vulnerability of the users data to be sniffed by third-party. Hence this should be considered as a security requirement of the product.

4.4.2 Performance Requirements

The main requirement here is a good communication. If the communication lags, the purpose of the software is defeated. However, this requirement is mostly a issue at the client end.

4.4.3 Safety Requirements

This product requires some safety concerns. The handshaking between client and server in bluetooth mode should be secure.

4.5 External Interface Requirements

4.5.1 Hardware Interfaces

Typical hardware interfaces that would be required would be the standard USB data cable provided with the device. No other additional hardware interfacing would be needed to be bought by the user. Also Bluetooth and/or Wi-Fi adapter on the server machine would be needed for communication (in case USB cable is not available). Communication protocol would be developer implemented and the user need not worry about communicating using this protocol. Hence this product will be extremely user-friendly.

4.5.2 Software Interfaces

The product featured, requires a database to be maintained at both the ends - client and server. Hence we will be deploying two different databases on both of them which will be in sync on demand.

The client DB will be implemented using SQLite and the server DB will be standard PostgreSQL. The UI on client will be a typical android user interface using the default options available at the time of development. UI on server will be implemented using Qt and python at the backend.

4.5.3 Communications Interfaces

This product basically works on the client-server architecture. Hence a communication mechanism between server and client has to be developed. This communication will be established over https or local Bluetooth or Wi-Fi. USB cable is another option. E-mails can be used to send/receive reports, log files.

Chapter 5

System Design

5.1 Goals of Design

1. The project design should help us to visualize the system as it is or want it to be.
2. It should permit us to specify the structure or behavior of the system.
3. To give us template that guides us in the construction of the system.
4. To document the decision we have made.

5.2 UML Diagrams

The OMG's Unified Modeling Language (UML) helps specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements. Using any one of the large number of UML-based tools on the market, one can analyze future application's requirements and design a solution that meets them, representing the results using UML's twelve standard diagram types. One can model just about any type of application, running on any type and combination of hardware, operating system, programming language, and network, in UML. Its flexibility helps model distributed applications that use just about any middleware on the market. Built upon the MOF metamodel which defines class and operation as fundamental concepts, it's a natural fit for object-oriented languages and environments such as C++, Java and the recent C#, but one can use it to model non-OO applications as well in, for example, Fortran, VB, or COBOL.

Architects design buildings. Builders use the designs to create buildings. The

more complicated the building, the more critical the communication between architect and builder. Blueprints are the standard graphical language that both architects and builders must learn as part of their trade.

Writing software is not unlike constructing a building. The more complicated the underlying system, the more critical the communication among everyone involved in creating and deploying the software. In the past decade, the UML has emerged as the software blueprint language for analysts, designers, and programmers alike. It is now part of the software trade. The UML gives everyone from business analyst to designer to programmer a common vocabulary to talk about software design.

The UML is applicable to object-oriented problem solving. Anyone interested in learning UML must be familiar with the underlying tenet of object-oriented problem solving it all begins with the construction of a model. A model is an abstraction of the underlying problem. The domain is the actual world from which the problem comes.

Models consist of objects that interact by sending each other message. Think of an object as "alive." Objects have things they know (attributes) and things they can do (behaviors or operations). The values of an object's attributes determine its state.

Classes are the "blueprints" for objects. A class wraps attributes (data) and behaviors (methods or functions) into a single distinct entity. Objects are instances of classes.

At the center of the UML are its nine kinds of modeling diagrams, which we describe here.

- Use case diagrams
- Class diagrams
- Object diagrams
- Sequence diagrams
- Collaboration diagrams
- Statechart diagrams
- Activity diagrams
- Component diagrams

- Deployment diagrams

To model our system we have used the following diagrams:

Class Diagram

A Class diagram is type of structure diagram that describes structure of our system by showing system classes, their attributes and relationships between them. Class diagrams are important for visualizing, specifying and documenting structural models.

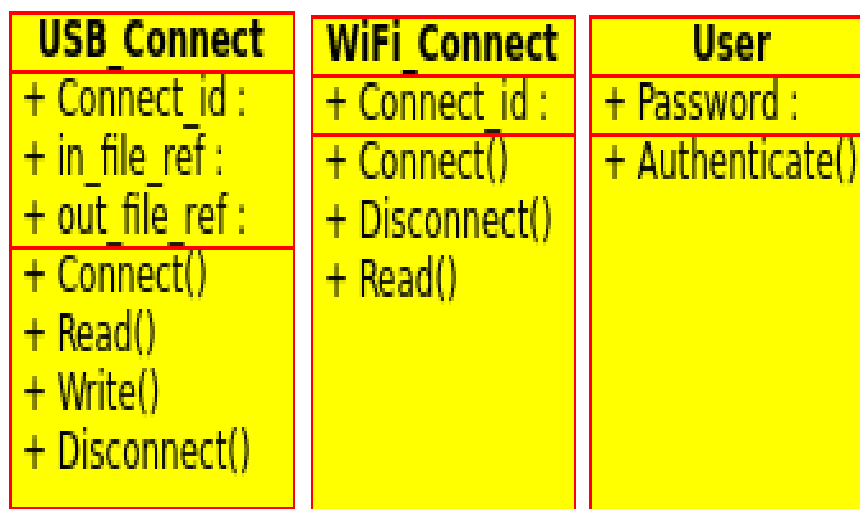


Figure 5.1: Class Diagram

Use Case Diagram

Use case diagram are central to modeling the behavior of a system or class each one shows a set of use cases and actors and their relationships. These model the users expectation for using the system.

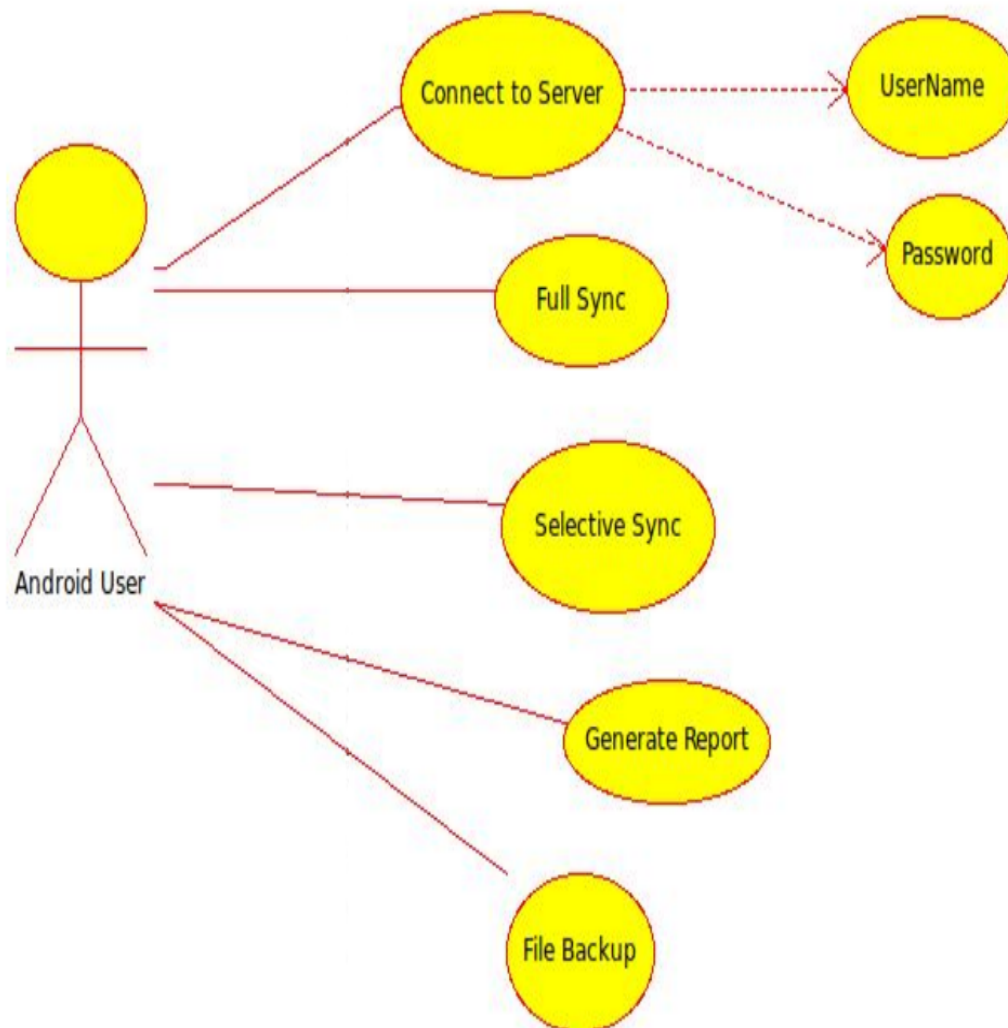


Figure 5.2: Use Case Diagram

Sequence Diagram

A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development.

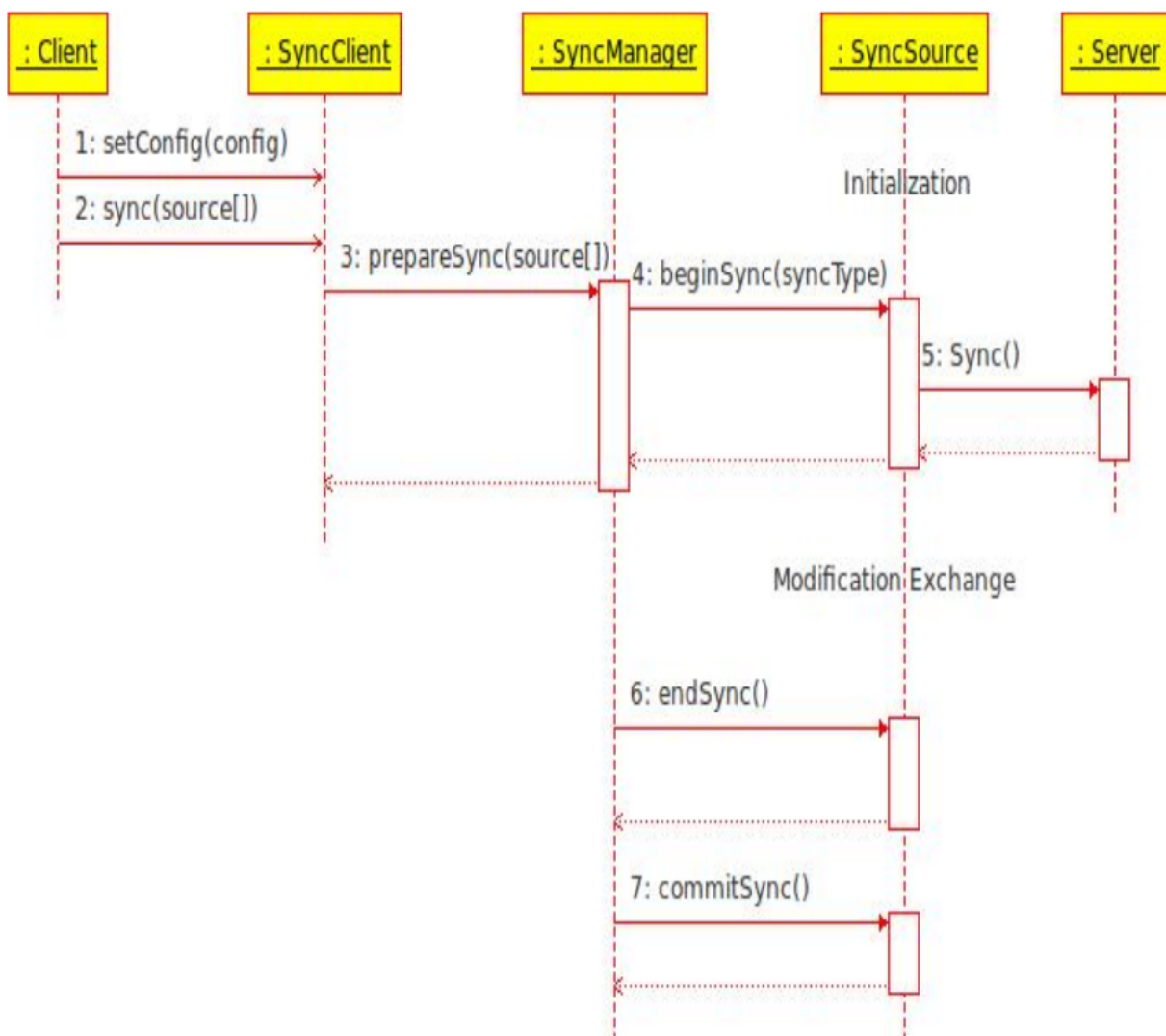


Figure 5.3: Sequence Diagram

5.3 Feasibility Study

As we are developing our system we would like to consider the feasibility aspects of this system. We are trying to develop a system which is an improvised version of the current implementation. So while carrying out a feasibility study, we would like to compare our proposed system with the current scenario.

5.3.1 Current Implementation

The current scenario is as follows:

1. User logs onto his Google account.
2. Synchronization starts and data are sent to the cloud.
3. This data remains on third-party servers (Google)
4. This data can be later retrieved from the cloud.
5. This data can be later retrieved from the cloud.

5.3.2 The Proposed System

As opposed to the current implementation, we want to increase the privacy of the users by enabling them to save their data to be synchronized on their own private servers. These private servers can be Desktop PCs, Laptops, etc.

The steps would include:

1. Ensure server is on
2. Log onto the application on the server (for online mode)
3. For offline mode, connect the device using USB cable, Wi-Fi, Bluetooth
4. Start synchronization (full or selective sync)
5. Log off
6. This system is being developed keeping in mind the basic synchronization necessities of the average android user. Hence it can be used by both average users as well as large corporations to keep their employee data (on their android mobile devices) safe.

Chapter 6

System Testing

Software Testing

Software testing is the process used to assess the quality of computer software. Software testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. this includes, but is not limited to, the process of executing a program or application with the intent of finding software bugs.

There are many approaches to software testing. Reviews, walkthroughs or inspection are considered as static testing, whereas actually running the program with a given set of test cases in a given development stage is referred to as dynamic testing.

Software testing is used in association with verification and validation:

- **Verification:** Have we built the software right (i.e., does it match the specification)?
- **Validation:** Have we built the right software (i.e., is this what the customer wants?)

Software testing can be done by software testier. until the 1950s the term software tester was used generally, but later it was also seen as a separate profession. Regarding the periods and the different goals in software testing there have been established different roles: test lead/manager , tester, test designer, test automater/automation developer, and test administrator.

Test plan:

It is a systematic approach to testing a system such as a machine or software. The plan typically contains a detailed understanding of what the eventual workflow will be.

Test bed:

It is a platform for experimentation for large development projects. Test beds allow for rigorous, transparent and replicable testing of scientific theories, computational tools, and other new technologies.

Test environment:

In software, the hardware and software requirements are known as the test bed. This is also known as the test environment.

Scenario testing:

Scenario testing is a software testing activity that uses scenario tests, or simply scenarios, which are based on a hypothetical story to help a person think through a complex problem or system. they can be as simple as a diagram for a testing environment or they could be a description written in prose.

Test case:

Test case in software engineering is a set of conditions or variables under which a tester will determine if a requirement or use case upon an application is partially or full satisfied. It may take many test cases to determine that a requirement fully satisfied. Test cases are often incorrectly referred to as test scripts. Test scripts are lines of code used mainly in automation tools. written test cases are usually collected into test suites.

Unit testing:

Unit testing tests the minimal software component or module. each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented. In an object oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

Integration testing exposes defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and testing until the software works as a system. System testing tests a completely in-

egrated system to verify that it meets its requirements. System integration testing verifies that a system is integrated to any external or third party systems defined in the system requirements.

Black box testing:

Black box testing treats the software as a black box without any understanding of internal behavior. it aims to test the functionality according to the requirements. Thus the tester inputs data and only sees the output from the test object. This level of testing usually requires through test cases to be provided to the tester who then can simply verify that for a given input, the output value (or behavior), is the same as the expected value specified in the test case. Black box testing methods include: equivalence partitioning , boundary value analysis, all - pairs testing, fuzz testing , model-based testing, traceability matrix etc.

White box testing:

White box testing, however is when the tester has access to the internal data structures, code, and algorithms. White box testing methods include creating tests to satisfy some code coverage criteria. For Example, the test designer can create tests to cause all statements in the program to be executed at least once. Other examples of white box testing are mutation testing and fault injection methods. White box testing includes all static testing.

Alpha testing:

Alpha testing is simulated or actual operational testing by potential users customers or an independent test team at the developer's site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing. Before the software goes into beta testing.

Beta testing:

Beta testing comes after alpha testing. Version of the software , known as beta versions , are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta version are made available to the open public to increase the feedback field to a maximal number of future users.

Regression testing:

After modifying the software, either for a change in functionality or the fix defects, a regression test re-runs previously passing tests on the modified software to

ensure that the modifications haven't unintentionally caused a regression of previous functionality. Regression testing can be performed at any or all of the above test levels. These regression tests are often automated.

Sanity test:

Sanity test is a basic test to quickly evaluate the validity of a claim or calculation. In mathematics, for example, when dividing by three or nine, verifying that the sum of the digits of the result is a multiple of 3 or 9 (casting out nines) respectively is a sanity test.

Smoke testing:

Smoke testing is a term used in plumbing, woodwind repair, electronics and computer software development. It refers to the first test made after repairs or first assembly to provide some assurance that the system under tests will catastrophically fail. After a smoke test proves that the pipes will not leak, the keys seal properly, the circuit will not burn, or the software will not crash outright, the assemble is ready for more stressful testing.

6.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	Application Installed on Server	Check application is installed or not on Server	1. Application is not shown in list of installed packages 2. Application is shown in list of installed packages	1. Application is not installed 2. Application is installed
T02	Application Installed on Client	Check application is installed or not on client	1. Application is not shown in list of installed packages 2. Application is shown in list of installed packages	1. Application is not installed 2. Application is installed
T03	Bluetooth is ON/OFF	Check bluetooth is on/off	1. Bluetooth ON is shown by system 2. Bluetooth OFF is shown by system	1. Bluetooth is ON 2. Bluetooth is OFF
T04	WiFi is ON/OFF	Check WiFi is on/off	1. WiFi ON is shown by system 2. WiFi OFF is shown by system	1. WiFi is ON 2. WiFi is OFF
T05	USB cable is connected	Check USB cable is connected or not	1) Device is not shown by system 2) Device is shown by system	1) USB cable is connected 2) USB cable is not connected
T06	Check mobile is connected to server or not	Check for list of bluetooth devices connected to server	List of connected devices is shown on server	Name of device is shown in connected devices
T07	Is mobile device visible	Check mobile visibility is ON/OFF	1) Mobile visibility is OFF 2) Mobile visibility is ON	1) Device is not shown by bluetooth 2) Device is shown by bluetooth

T08	Check mobile exist in list of paired devices	Check list of paired devices	1) Mobile Device doesn't exist in list of paired devices 2) Mobile Device exist in list of paired devices	1) Add device in list of paired devices 2) Start next process
T09	Enter correct passphrase	Check entered passphrase is correct	1) Entered passphrase is Wrong 2) Entered passphrase is Correct	1) Device will not get added on Server 2) Device will be added to Server
T10	Check mobile exist in list of paired devices	Check list of paired devices	1) Mobile Device doesn't exist in list of paired devices 2) Mobile Device exist in list of paired devices	1) Add device in list of paired devices 2) Start next process
T11	User account exist or not for server	Check for User Login	1) User account Exist 2) User account does not exist	1) Start next process 2) Create new account
T12	Enter Username	Check username is correct or not 1) Username is not correct 2) Username is correct	1) Enter username again 2) password is focused	
T13	Enter Password	Check password is correct or not	1) Password is not correct 2) Password is correct	1) Enter password again 2) user get verified

T14	Check for new data on mobile device	Check new data exist on mobile device or not	1) New data does not exist on mobile device 2) New data exist on mobile device	1) Do nothing 2) Start Synchronization
-----	-------------------------------------	--	---	---

Note: Testing should be performed manually

Chapter 7

Project Planning

7.1 Project Planning Model

Development occurs as a succession of releases with increasing functionality. Testing and feedback on each release is used in deciding requirements and improvements for next release. There is no "maintenance" phase - each version includes both problem fixes as well as new features. this may also include "re-engineering" - changing the design and implementation of existing functionality, for easier maintainability.

We have followed a spiral model like approach for our project as it has a large no. of units. So, prototyping at each stage is necessary.

The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. Also known as the spiral lifecycle model, it is a systems development method(SDM) used in information technology (IT). This model of development combines the features of the prototyping model and the waterfall model. The spiral model is intended for large expensive and complicated projects.

The steps in the spiral model can be generalized as follows:

1. The new system requirements are defined in as much detail as possible. A preliminary design is created for the new system.
2. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled down system, and represents an approximation of the characteristics of the final product.
3. A second prototype is evolved by a fourfold procedure:

- (a) Evaluating the first prototype in terms of its strengths, weaknesses, and risks;
 - (b) Defining the requirements of the second prototype
 - (c) Planning and designing the second prototype
 - (d) Construction and testing the second prototype
4. At the customer's option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could in the customer's judgment, result in a less than satisfactory final product.
 5. The existing prototype is evaluated in the same manner as was the previous prototype, and, if necessary another prototype is developed from it according to the fourfold procedure outlined above. The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
 6. The final system is constructed, based on the refined prototype.
 7. The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.

Advantages:

- Estimates(i.e. budget, schedule, etc) become more realistic as work progresses, because important issues are discovered earlier.
- It is more able to cope with the (nearly inevitable) changes that software development generally entails.
- Software engineers (who can get restless with protracted design processes) can get their hands in and start working on a project earlier.

7.2 Estimation and Efforts

- **Technical:** As all the technical knowledge required for developing the system is available through books, it is technically feasible.
- **Economical:** As no extra investment is required for the implementation of the system, it is economically feasible.
- **Time requirements:** 400 hours.

7.3 Project Schedule

Software project scheduling is an activity that distributes estimated effort across the planned project duration. By allocation the effort to specific software engineering task like all other areas of software engineering, number of basic principles guide software project scheduling:

1. **Compartmentalization:**

It is accomplished by decomposing both the product and the process into manageable activities and tasks. This is called work structure break down.

2. **Interdependency:**

Some task must occur in a sequence while some must occur in parallel. The interdependencies of each compartmentalized act must be determined.

3. **Time allocation:**

Each task to be scheduled must be allocated some number of work units.

4. **Effort Validation:**

Every project has a defined no. of team members and project leader should allocate the number of people that are scheduled at any given time for a particular task.

5. **Defined responsibilities:**

Every task that is scheduled should be assigned to a specific team member.

6. **Defined outcomes:**

Every task that is scheduled should have a defined outcome , normally a work product. Work product is often combined in deliverables.

7. **Defined milestones:**

Every task or group of task should be associated with project milestones.A

milestone is accomplished when one or more work product has been reviewed for quality and has been approved.

Our project was scheduled to follow spiral model. We planned the project in following way:

We spent the first two months i.e. July and August in Requirements Gathering and Surveying for software available and those to used for the project. Once we divided on it out next task was to design the front end of the system.

7.4 Timeline Chart

Timeline chart are user items that create a chart where a series of events are arranged on a bar graph . Each event can be single point in time or a date range. the contents of a timeline chart are defined in groups of events. Each group is considered a timeline entry. Each group is assigned a title, a color, and other properties that vary by the timeline entry type.

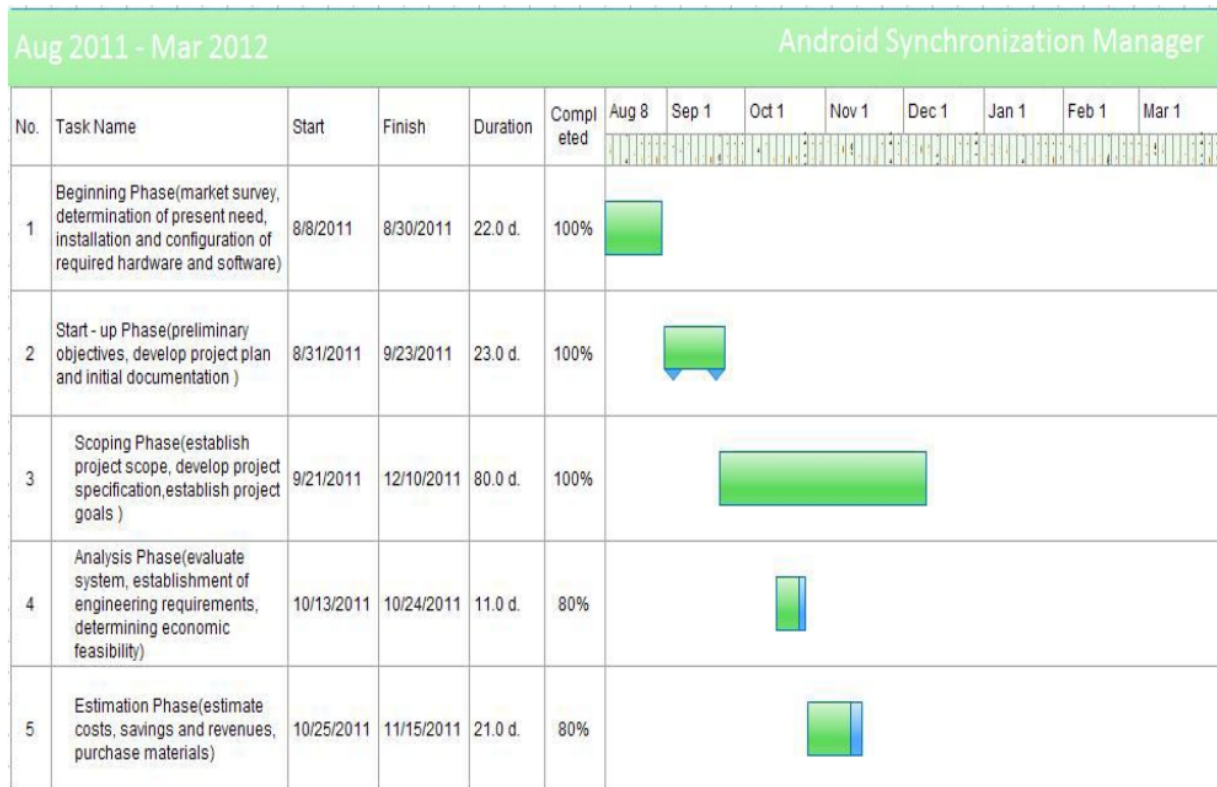


Figure 7.1: Planning and Scheduling - Part A

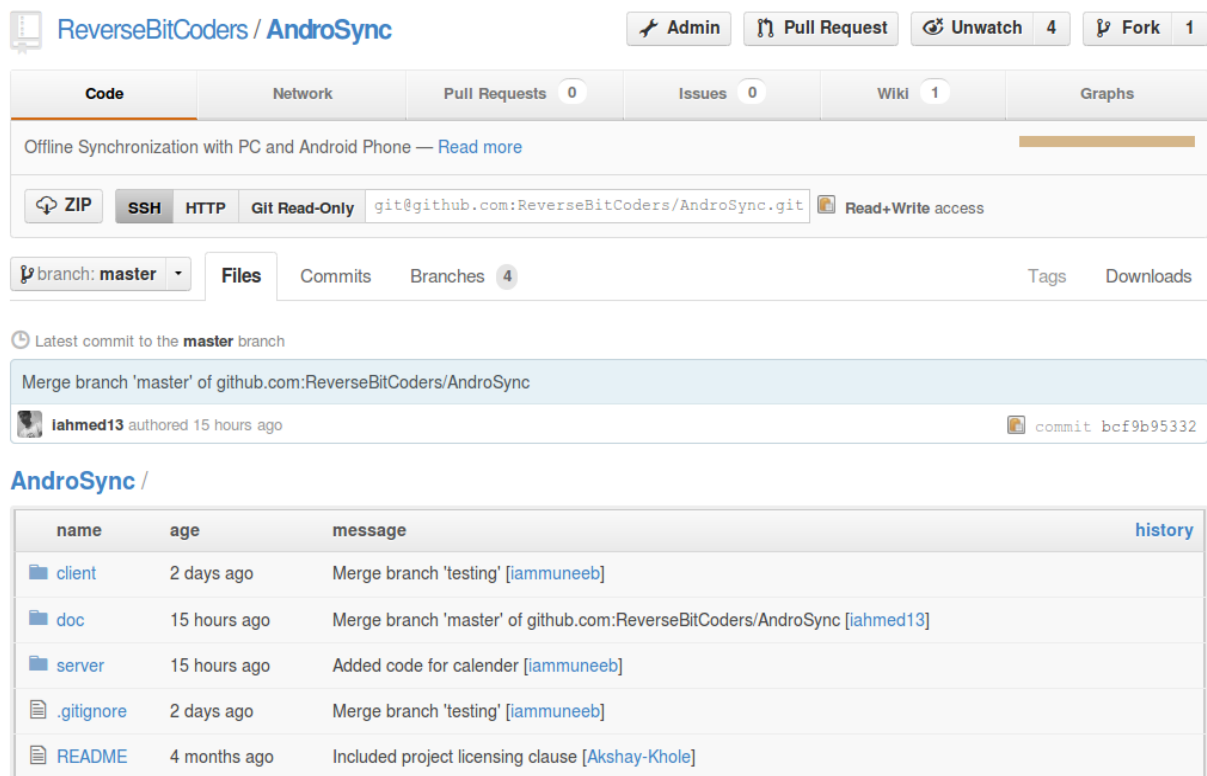


Figure 7.2: Planning and Scheduling - Part B

7.5 Source Code Management

For source code management/version control system, we are using GIT. The source code repository are located at GitHub. URL of our project for watching or forking is <https://github.com/ReverseBitCoders/AndroSync>

GitHub is a web-based hosting service for software development projects that use the Git revision control system. GitHub offers both commercial plans and free accounts for open source projects.



ReverseBitCoders / AndroSync

Admin Pull Request Unwatch 4 Fork 1

Code Network Pull Requests 0 Issues 0 Wiki 1 Graphs



Offline Synchronization with PC and Android Phone — [Read more](#)

ZIP SSH HTTP Git Read-Only `git@github.com:ReverseBitCoders/AndroSync.git` Read+Write access

branch: master Files Commits Branches 4 Tags Downloads

Latest commit to the master branch

Merge branch 'master' of github.com:ReverseBitCoders/AndroSync

 jahmed13 authored 15 hours ago  commit bcf9b95332

AndroSync /

name	age	message	history
client	2 days ago	Merge branch 'testing' [jammuneeb]	
doc	15 hours ago	Merge branch 'master' of github.com:ReverseBitCoders/AndroSync [jahmed13]	
server	15 hours ago	Added code for calender [jammuneeb]	
.gitignore	2 days ago	Merge branch 'testing' [jammuneeb]	
README	4 months ago	Included project licensing clause [Akshay-Khole]	

Figure 7.3: AndroSync Repository at GitHub

Progress Graphs generated by GitHub:

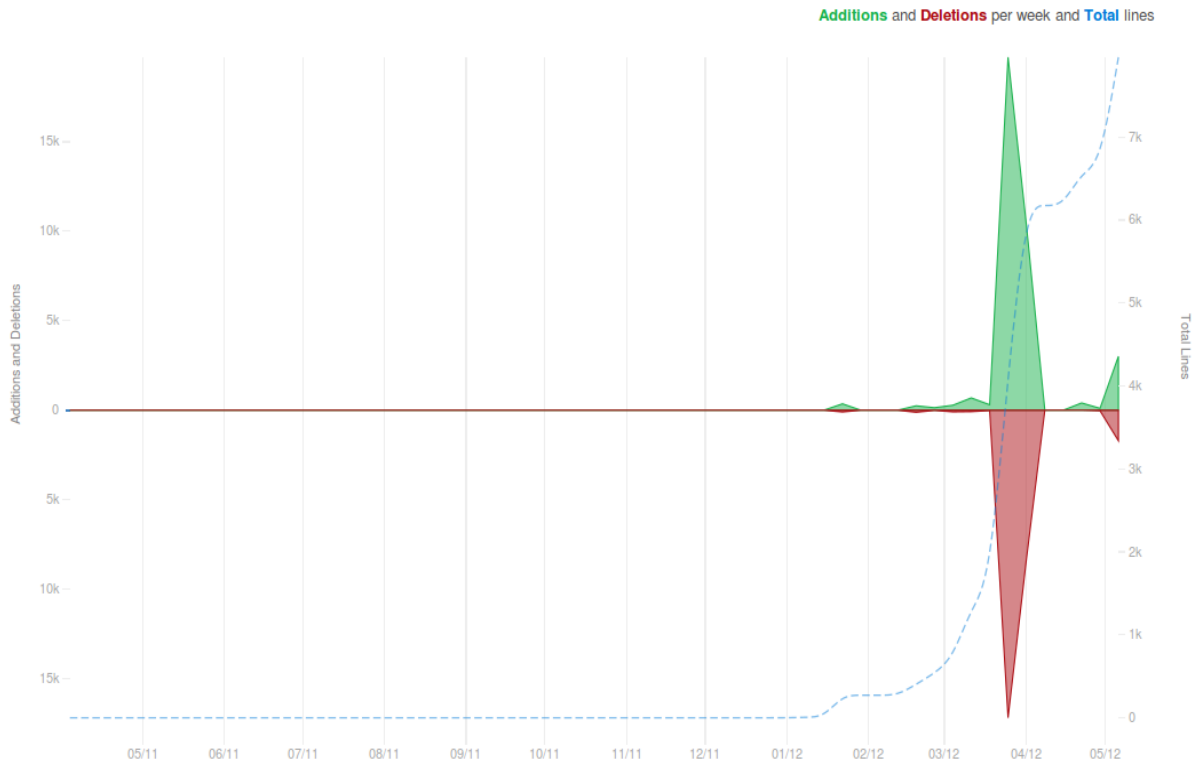


Figure 7.4: Code Frequency

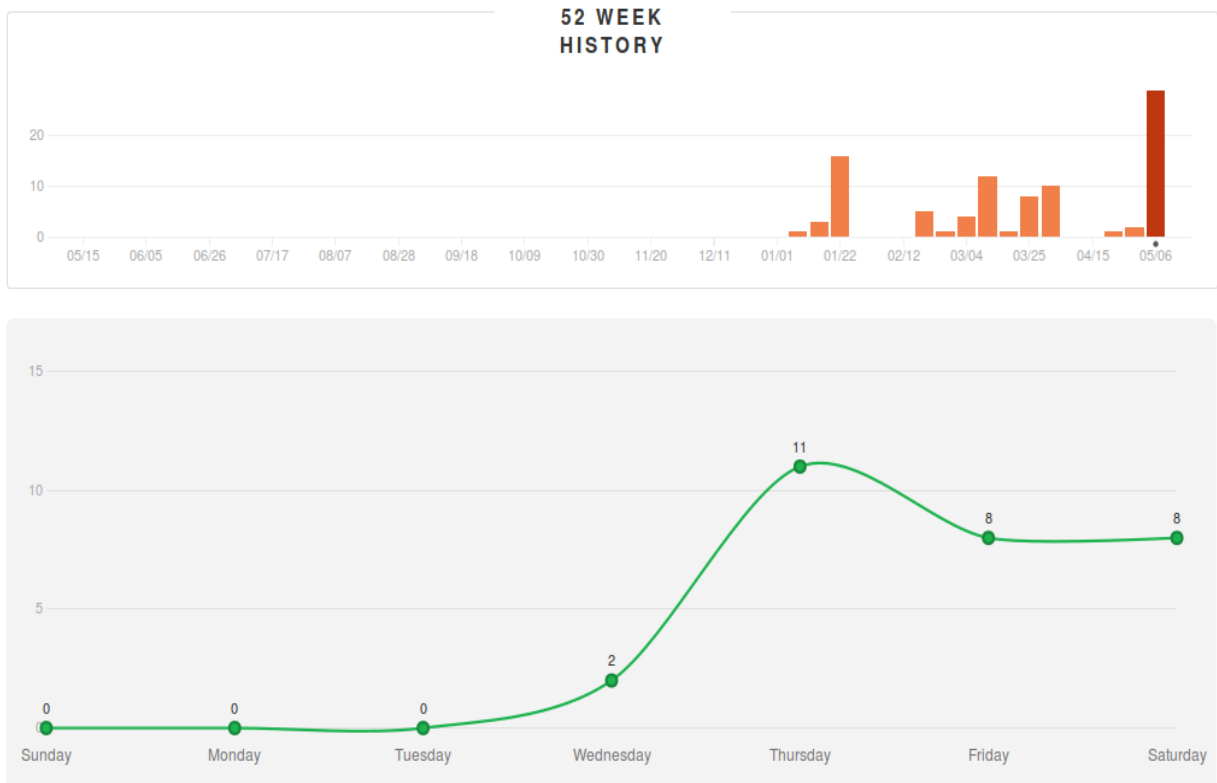


Figure 7.5: Commits



Figure 7.6: Contributors

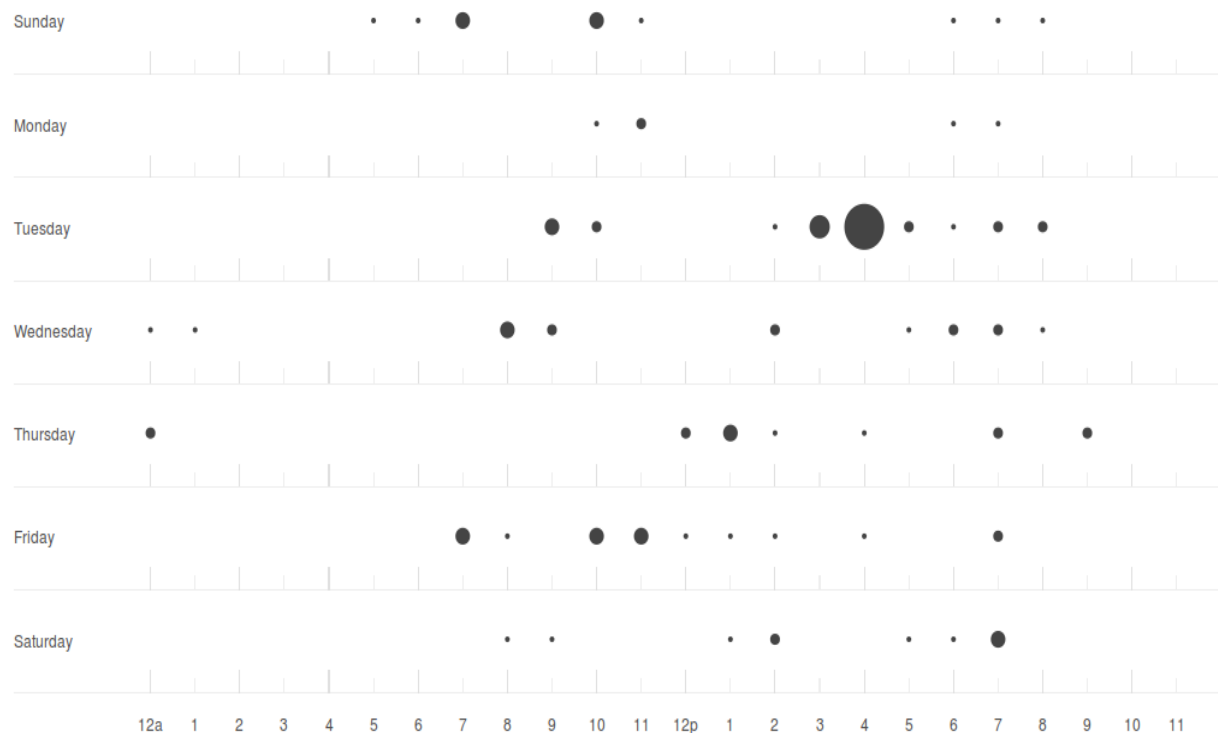


Figure 7.7: Punchcard

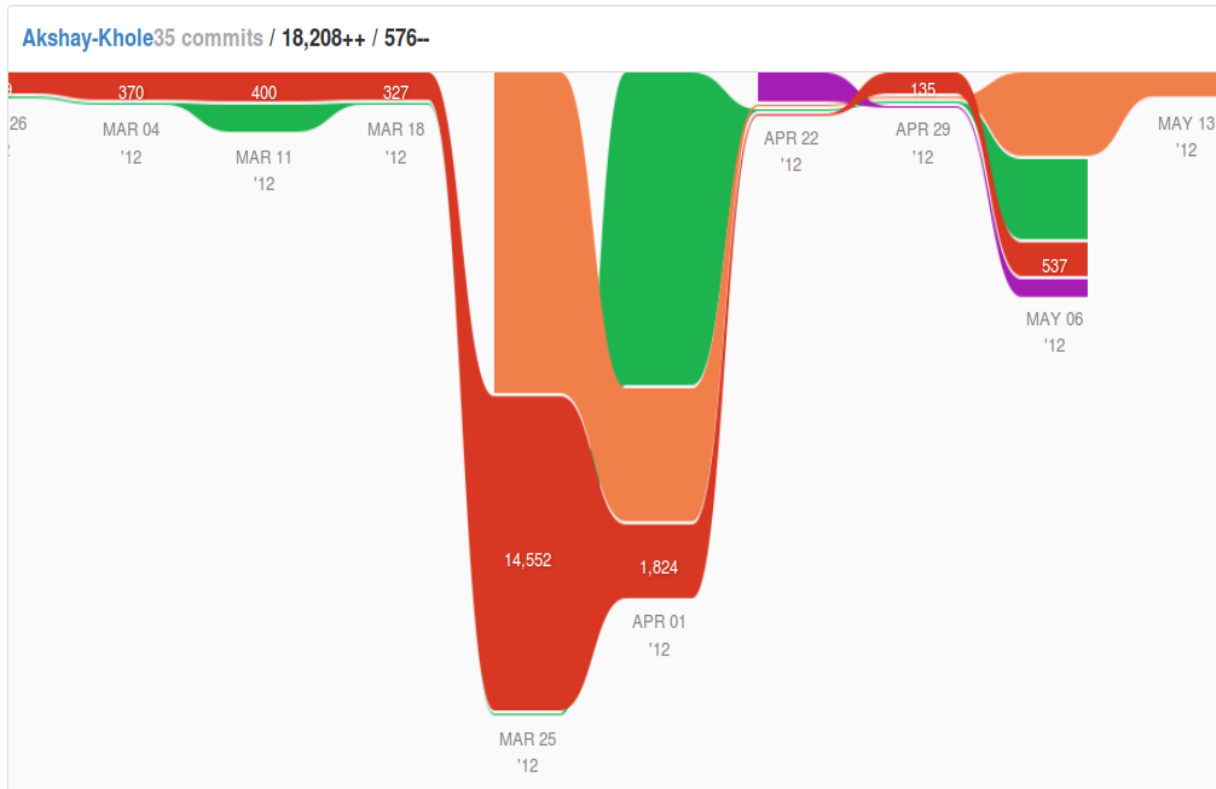


Figure 7.8: Impact - Part A

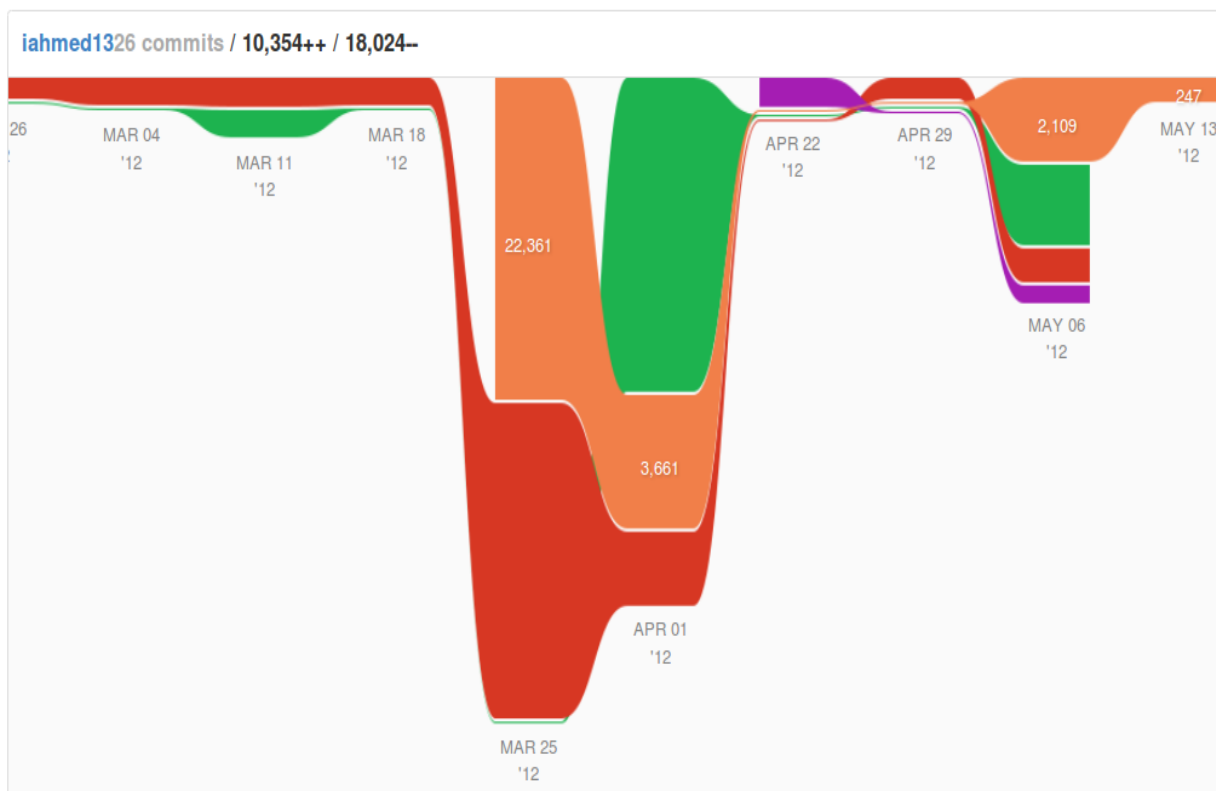


Figure 7.9: Impact - Part B

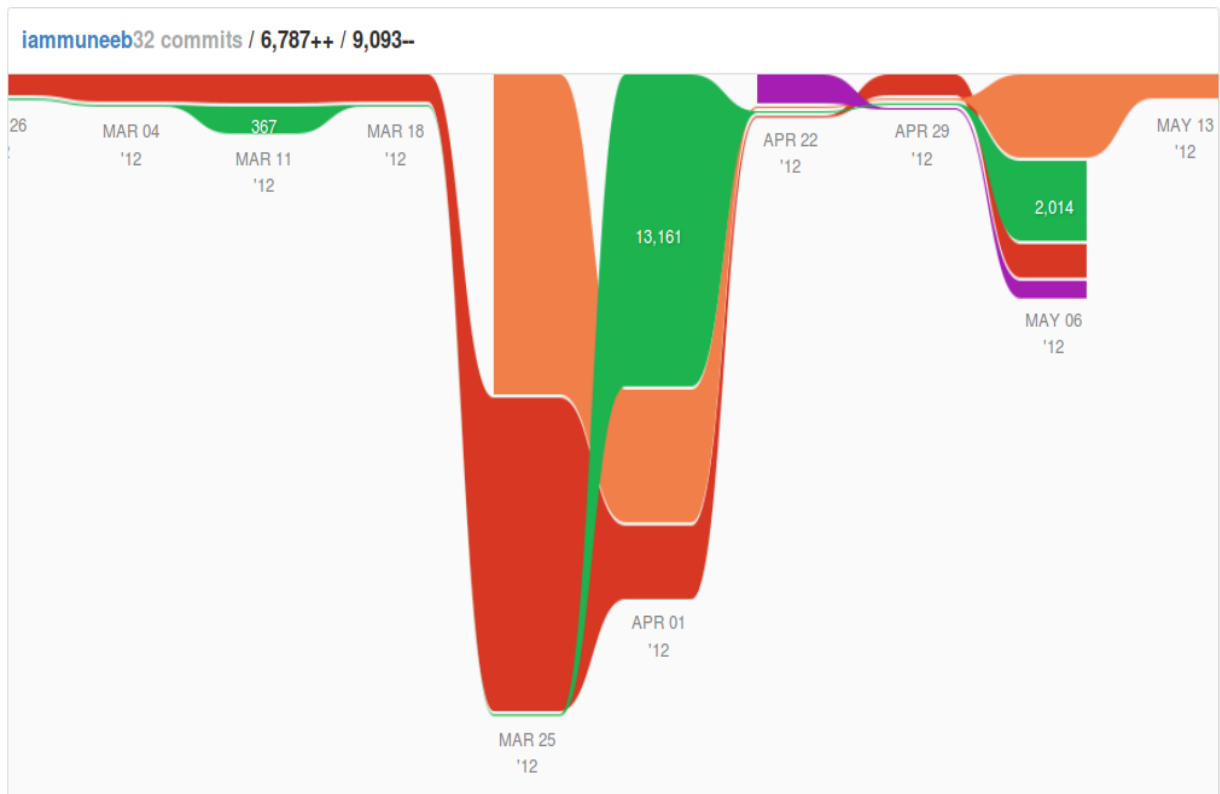


Figure 7.10: Impact - Part C

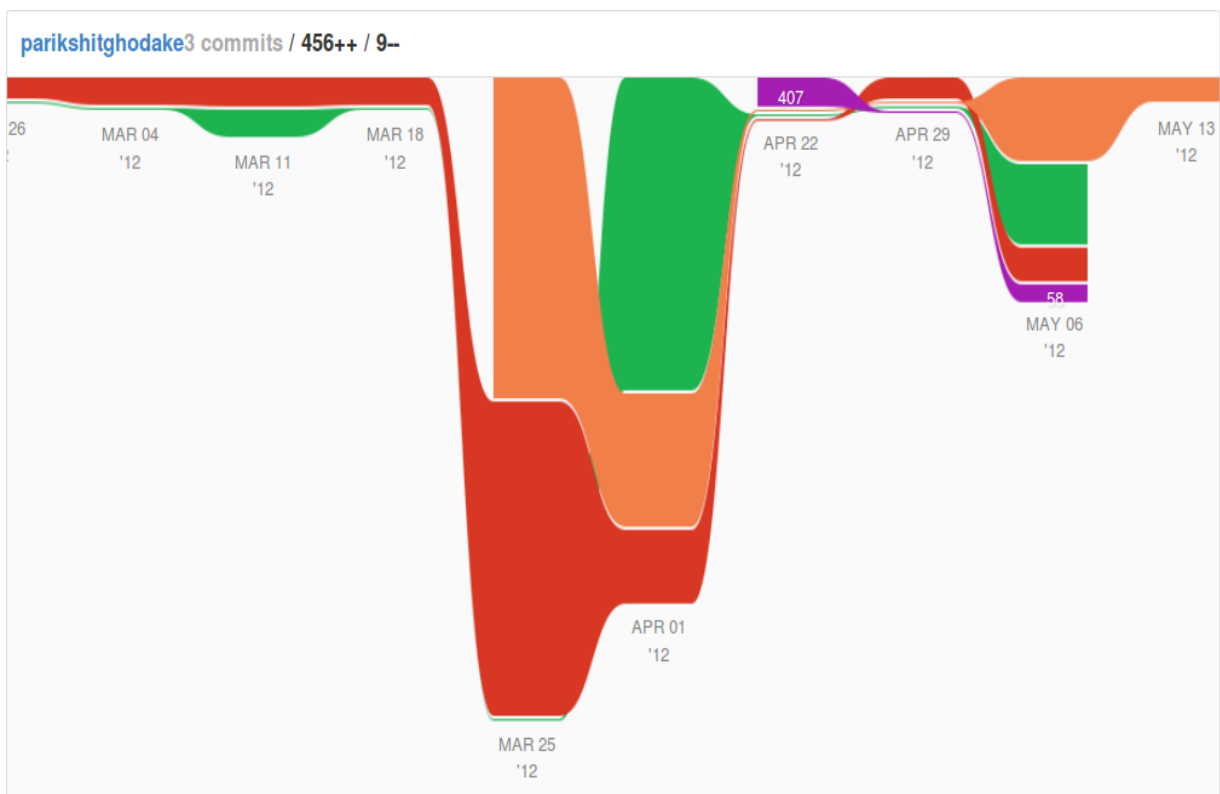


Figure 7.11: Impact - Part D

Chapter 8

Implementation

Activity:

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with `setContentView(View)`. While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with `windowIsFloating` set) or embedded inside of another activity (using `ActivityGroup`). There are two methods almost all subclasses of Activity will implement:

- **onCreate(Bundle)** is where you initialize your activity. Most importantly, here you will usually call `setContentView(int)` with a layout resource defining your UI, and using `findViewById(int)` to retrieve the widgets in that UI that you need to interact with programmatically.
- **onPause()** is where you deal with the user leaving your activity. Most importantly, any changes made by the user should at this point be committed (usually to the `ContentProvider` holding the data).

To be of use with `Context.startActivity()`, all activity classes must have a corresponding `<activity>` declaration in their package's `AndroidManifest.xml`.

```
1 public class Activity extends ApplicationContext {  
2     protected void onCreate(Bundle savedInstanceState);  
3     protected void onStart();  
4     protected void onRestart();  
5     protected void onResume();  
6     protected void onPause();  
7     protected void onStop();  
8     protected void onDestroy();  
9 }
```

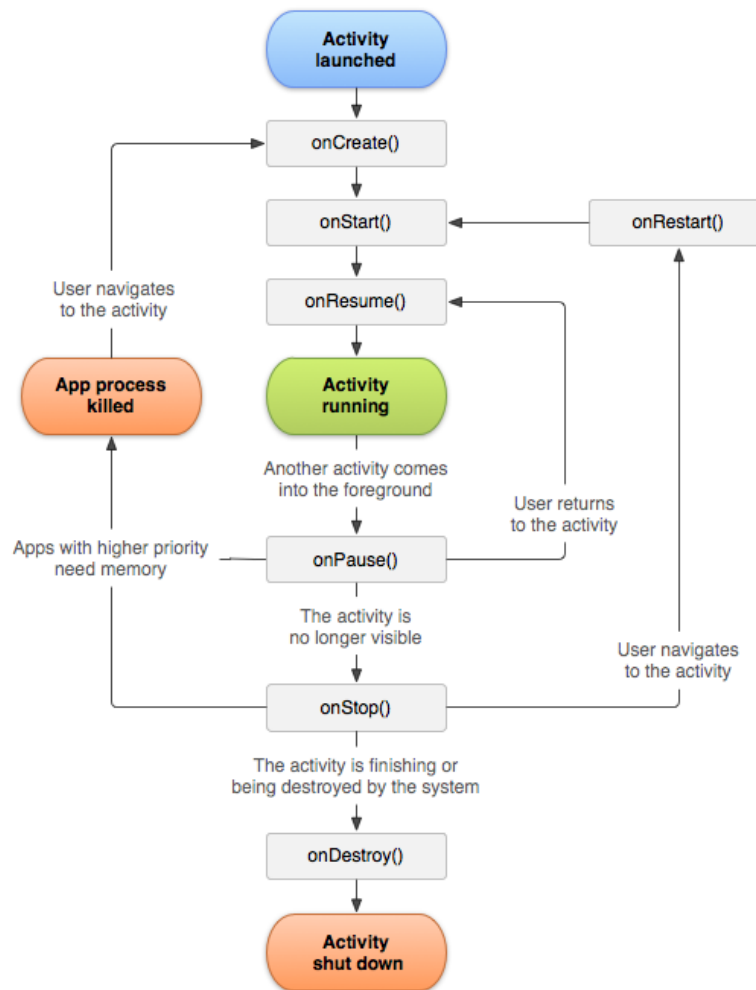


Figure 8.1: Important State Paths of an Activity

Bluetooth Adapter

The `BluetoothAdapter` lets you perform fundamental Bluetooth tasks, such as initiate device discovery, query a list of bonded (paired) devices, instantiate a `BluetoothDevice` using a known MAC address.

Create a `BluetoothServerSocket` to listen for connection requests from other devices. To get a `BluetoothAdapter` representing the local Bluetooth adapter, call the static `getDefaultAdapter()` method. Fundamentally, this is your starting point for all Bluetooth actions. Once you have the local adapter, you can get a set of `BluetoothDevice` objects representing all paired devices with `getBondedDevices()`; start device discovery with `startDiscovery()`.

Create a `BluetoothServerSocket` to listen for incoming connection requests with `listenUsingRfcommWithServiceRecord(String, UUID)`.

Bluetooth Server Socket

The interface for Bluetooth Sockets is similar to that of TCP sockets: `Socket` and `ServerSocket`. On the server side, use a `BluetoothServerSocket` to create a listening server socket. When a connection is accepted by the `BluetoothServerSocket`, it will return a new `BluetoothSocket` to manage the connection. On the client side, use a single `BluetoothSocket` to both initiate an outgoing connection and to manage the connection.

The most common type of Bluetooth socket is RFCOMM, which is the type supported by the Android APIs. RFCOMM is a connection-oriented, streaming transport over Bluetooth. It is also known as the Serial Port Profile (SPP).

To create a listening `BluetoothServerSocket` that's ready for incoming connections, use `BluetoothAdapter.listenUsingRfcommWithServiceRecord()`. Then call `accept()` to listen for incoming connection requests. This call will block until a connection is established, at which point, it will return a `BluetoothSocket` to manage the connection. Once the `BluetoothSocket` is acquired, it's a good idea to call `close()` on the `BluetoothServerSocket` when it's no longer needed for accepting connections. Closing the `BluetoothServerSocket` will not close the returned `BluetoothSocket`.

`BluetoothServerSocket` is thread safe. In particular, `close()` will always immediately abort ongoing operations and close the server socket.

Bluetooth Socket

The interface for Bluetooth Sockets is similar to that of TCP sockets: `Socket` and `ServerSocket`. On the server side, use a `BluetoothServerSocket` to create a listening server socket. When a connection is accepted by the `BluetoothServerSocket`, it will return a new `BluetoothSocket` to manage the connection. On the client side, use a single `BluetoothSocket` to both initiate an outgoing connection and to manage the connection.

The most common type of Bluetooth socket is RFCOMM, which is the type supported by the Android APIs. RFCOMM is a connection-oriented, streaming transport over Bluetooth. It is also known as the Serial Port Profile (SPP).

To create a `BluetoothSocket` for connecting to a known device, use `BluetoothDevice.createRfcommSocketToServiceRecord()`. Then call `connect()`

to attempt a connection to the remote device. This call will block until a connection is established or the connection fails.

Once the socket is connected, whether initiated as a client or accepted as a server, open the IO streams by calling `getInputStream()` and `getOutputStream()` in order to retrieve `InputStream` and `OutputStream` objects, respectively, which are automatically connected to the socket.

`BluetoothSocket` is thread safe. In particular, `close()` will always immediately abort ongoing operations and close the socket.

Contacts Contract

`ContactsContract` defines an extensible database of contact-related information. Contact information is stored in a three-tier data model:

- A row in the `ContactsContract.Data` table can store any kind of personal data, such as a phone number or email addresses. The set of data kinds that can be stored in this table is open-ended. There is a predefined set of common kinds, but any application can add its own data kinds.
- A row in the `ContactsContract.RawContacts` table represents a set of data describing a person and associated with a single account (for example, one of the user's Gmail accounts).
- A row in the `ContactsContract.Contacts` table represents an aggregate of one or more `RawContacts` presumably describing the same person. When data in or associated with the `RawContacts` table is changed, the affected aggregate contacts are updated as necessary.

Content Provider

Content providers are one of the primary building blocks of Android applications, providing content to applications. They encapsulate data and provide it to applications through the single `ContentResolver` interface. A content provider is only required if you need to share data between multiple applications. For example, the contacts data is used by multiple applications and must be stored in a content provider. If you don't need to share data amongst multiple applications you can use a database directly via `SQLiteDatabase`.

When a request is made via a `ContentResolver` the system inspects the authority of the given URI and passes the request to the content provider registered with the authority. The content provider can interpret the rest of the URI however it wants. The `UriMatcher` class is helpful for parsing URIs.

The primary methods that need to be implemented are:

- `onCreate()` which is called to initialize the provider
- `query(Uri, String[], String, String[], String)` which returns data to the caller
- `insert(Uri, ContentValues)` which inserts new data into the content provider
- `update(Uri, ContentValues, String, String[])` which updates existing data in the content provider
- `delete(Uri, String, String[])` which deletes data from the content provider
- `getType(Uri)` which returns the MIME type of data in the content provider

Requests to `ContentResolver` are automatically forwarded to the appropriate `ContentProvider` instance, so subclasses don't have to worry about the details of cross-process calls.

Telephony Manager

Provides access to information about the telephony services on the device. Applications can use the methods in this class to determine telephony services and states, as well as to access some types of subscriber information. Applications can also register a listener to receive notification of telephony state changes.

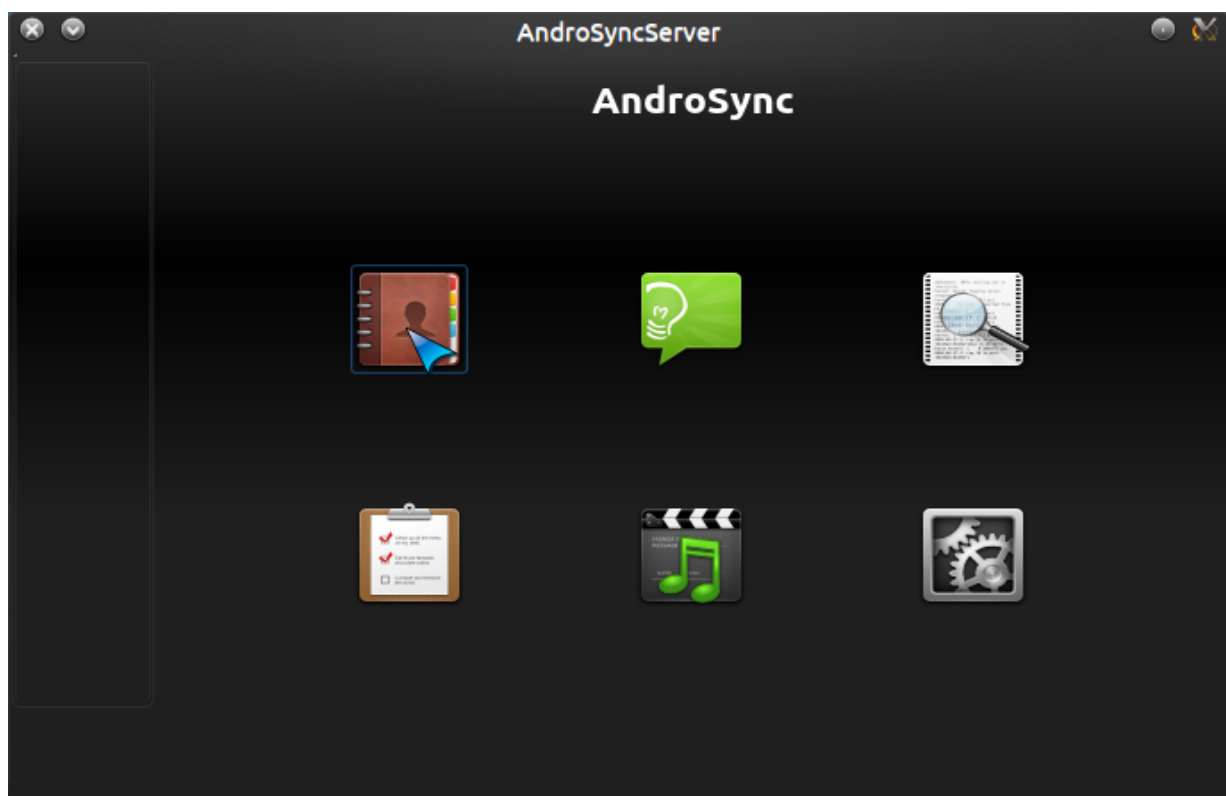
You do not instantiate this class directly; instead, you retrieve a reference to an instance through `Context.getSystemService(Context.TELEPHONY_SERVICE)`.

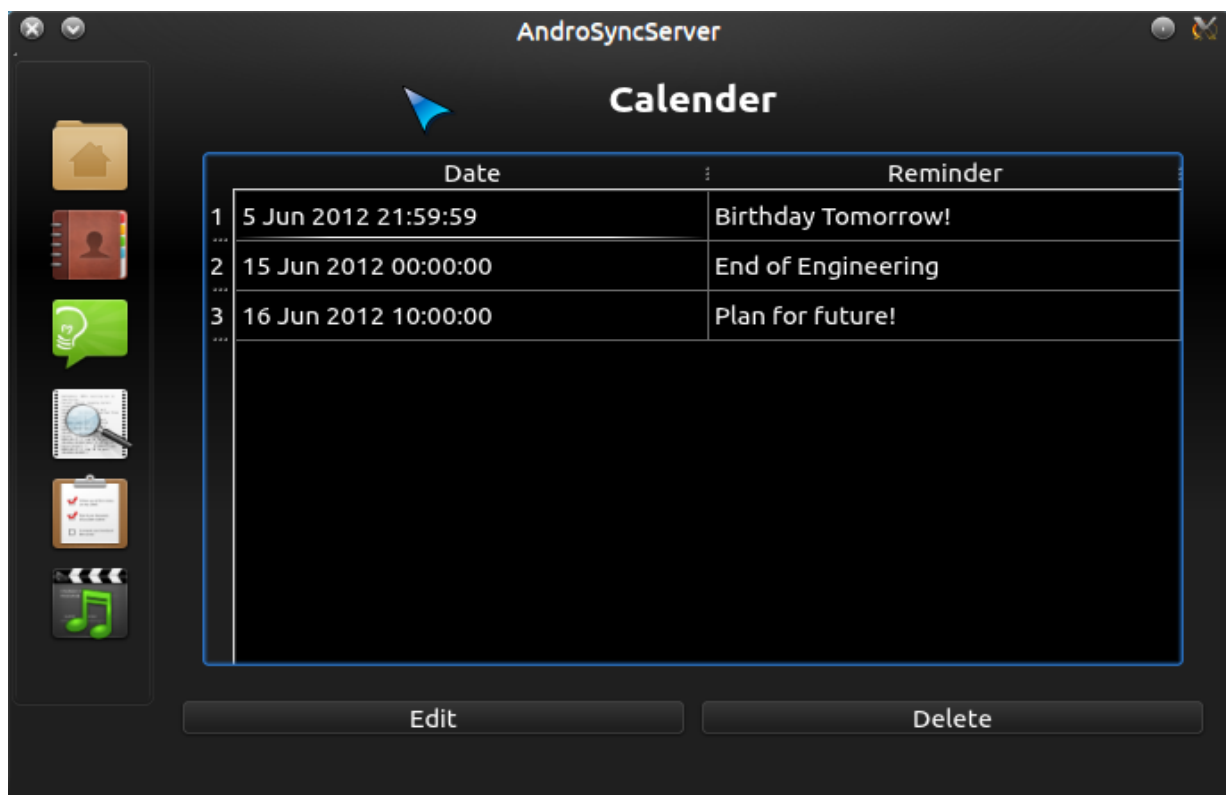
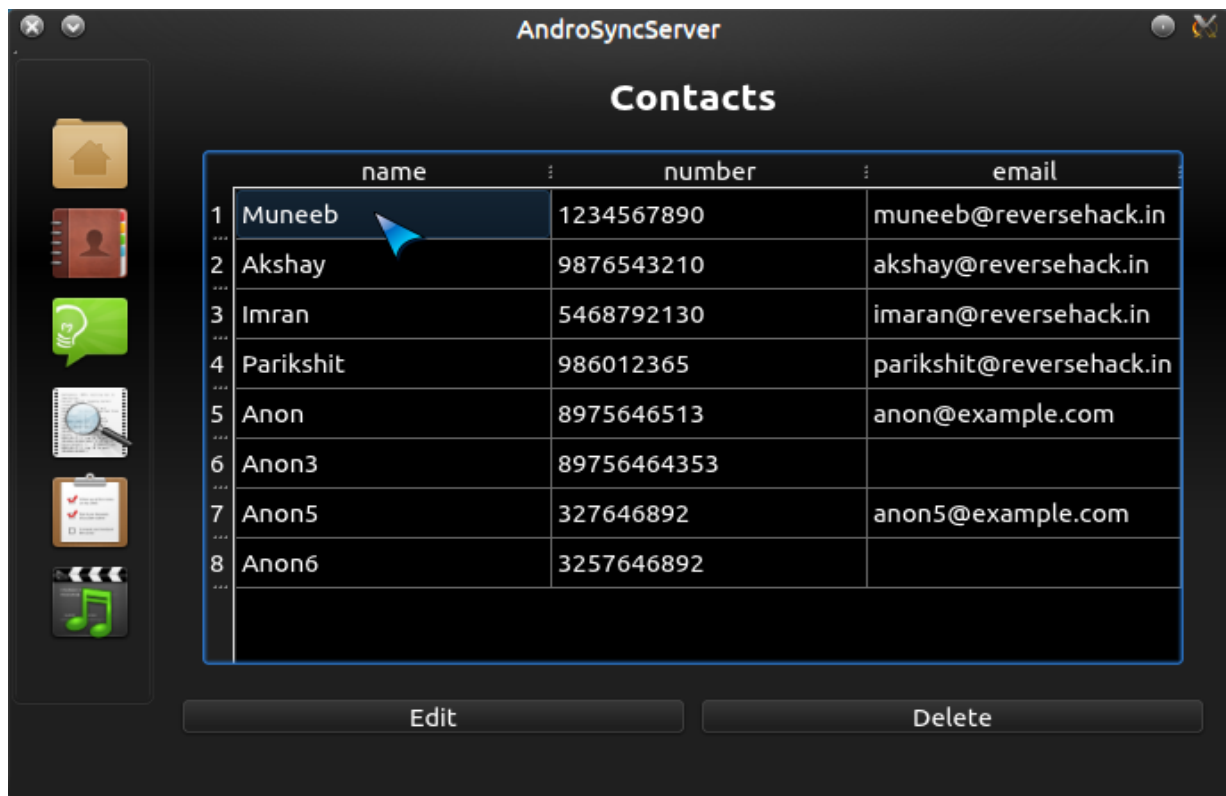
Note that access to some telephony information is permission-protected. Your application cannot access the protected information unless it has the appropriate permissions declared in its manifest file. Where permissions apply, they are noted in the the methods through which you access the protected information.

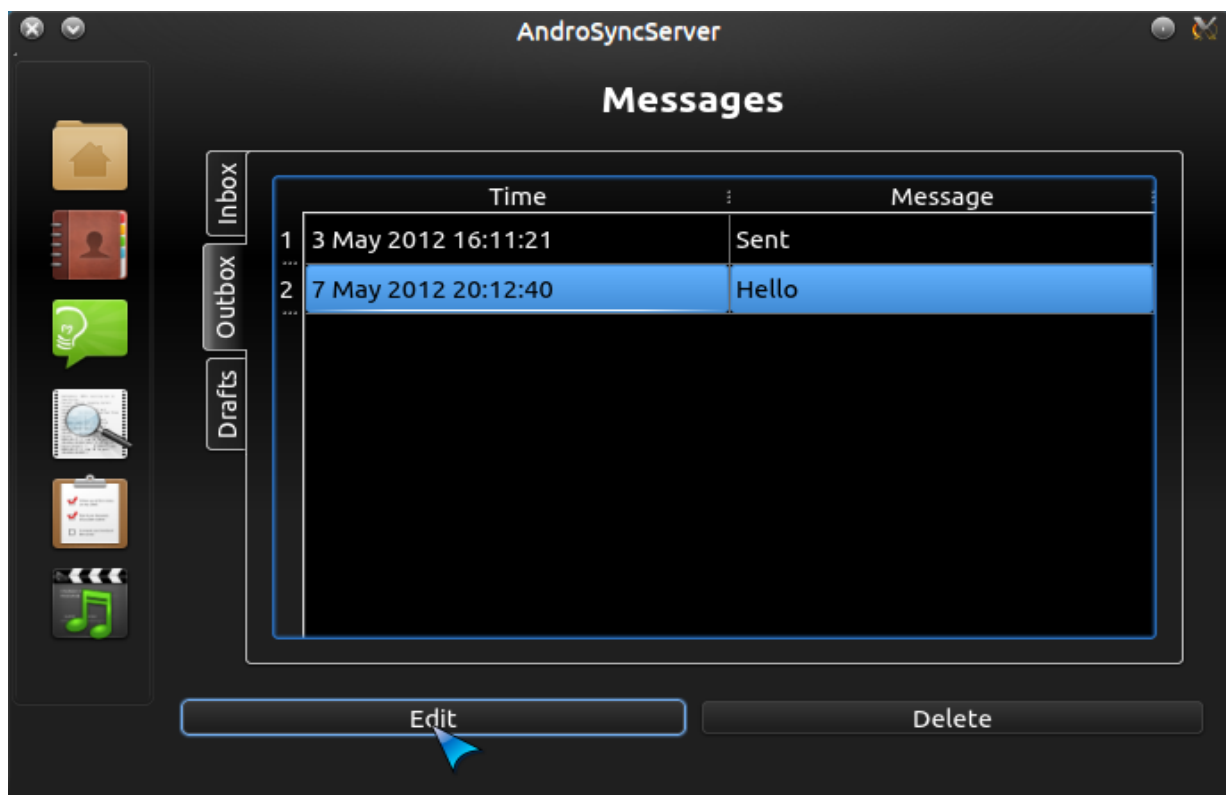
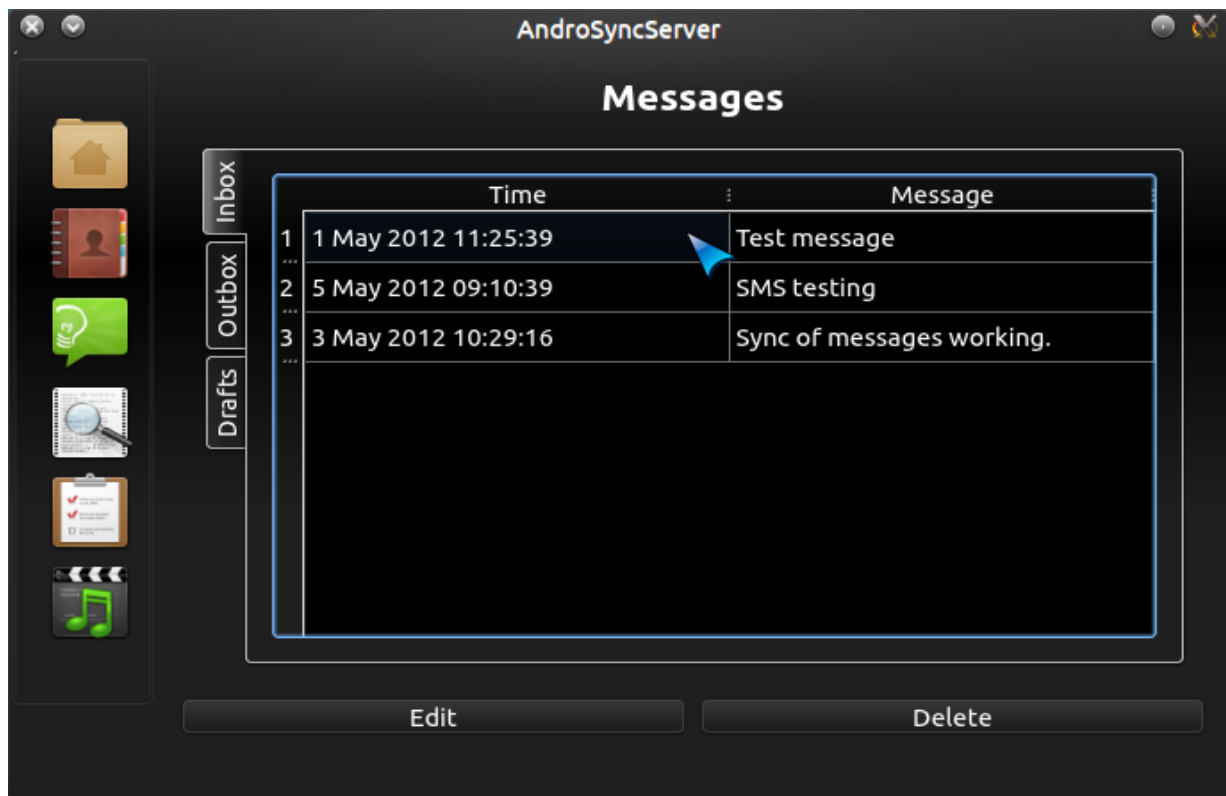
Chapter 9

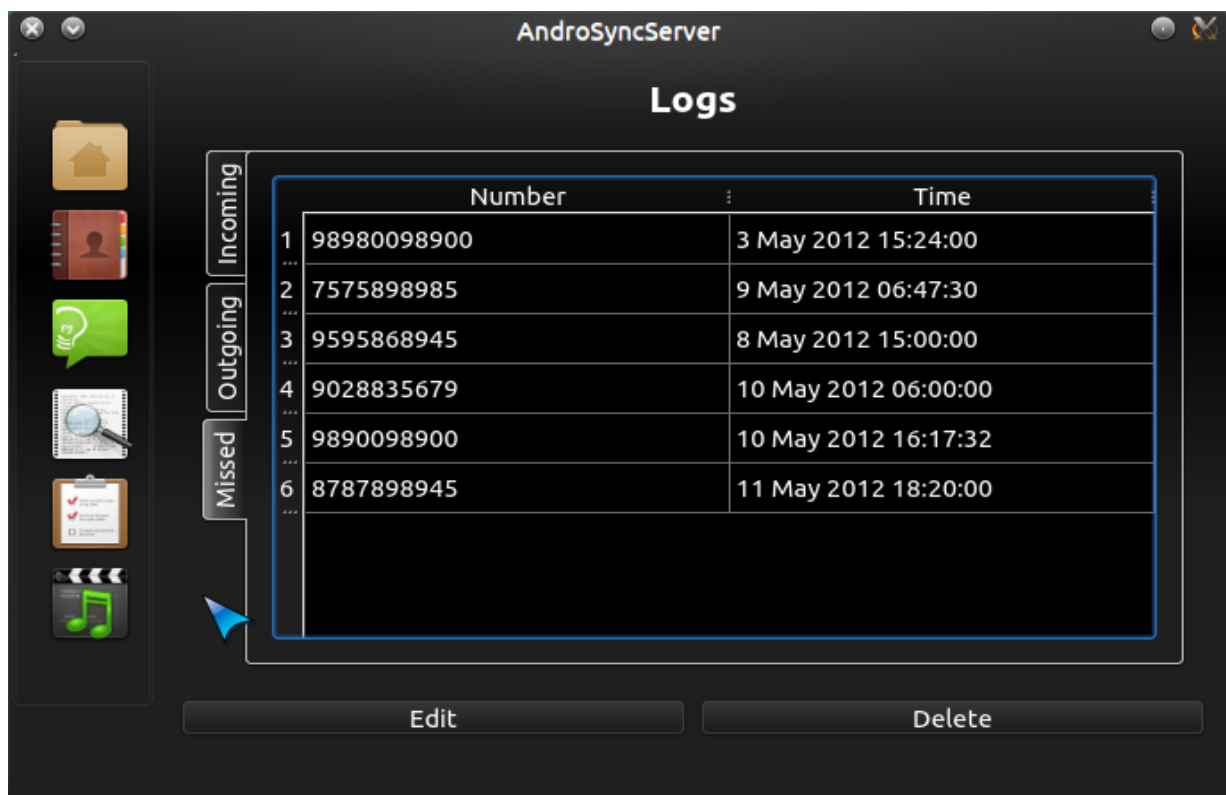
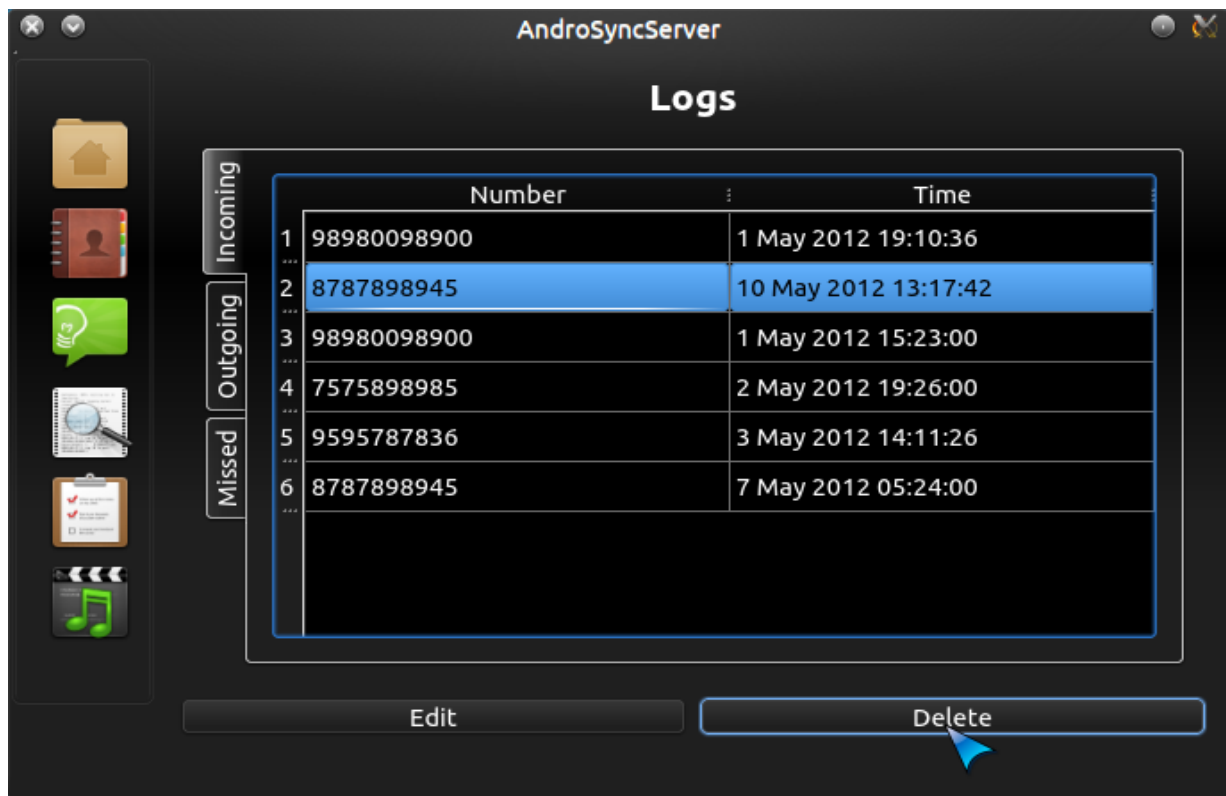
Screenshots of Client and Server

9.1 Server

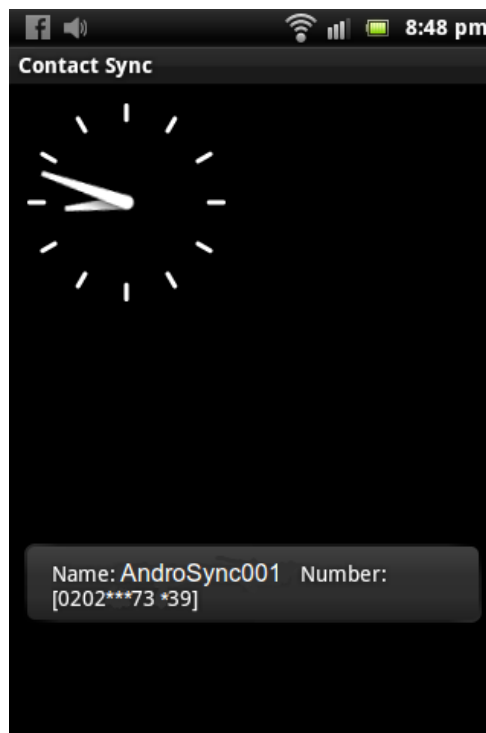
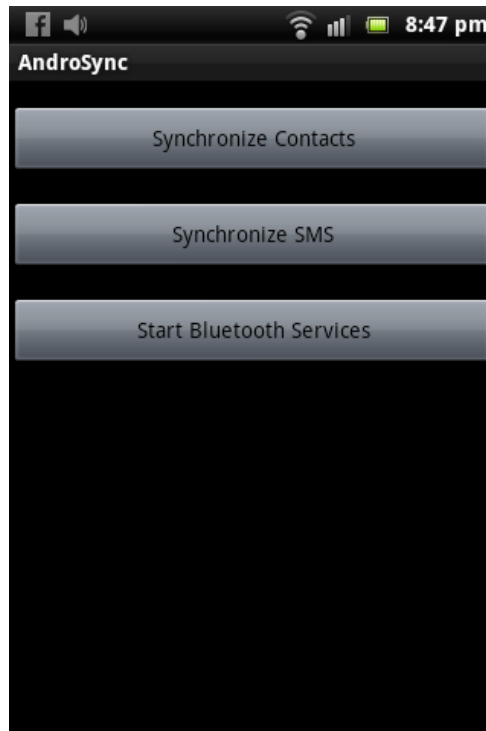


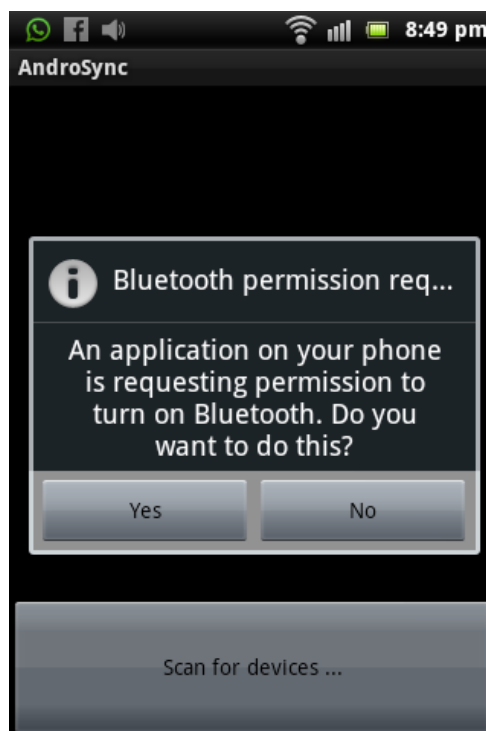
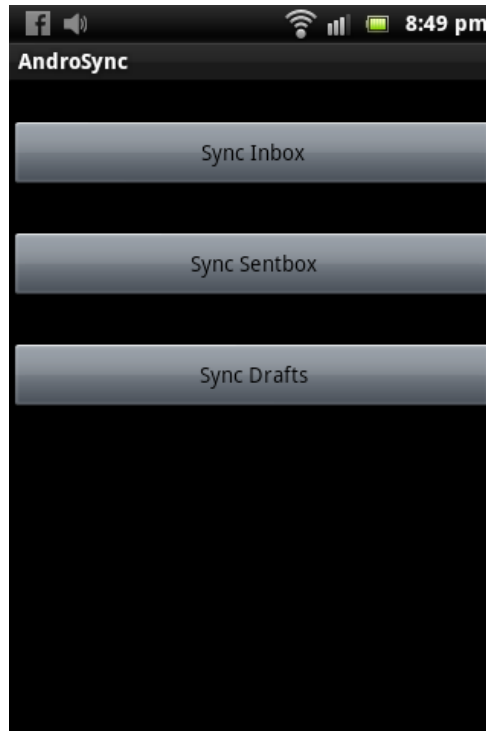


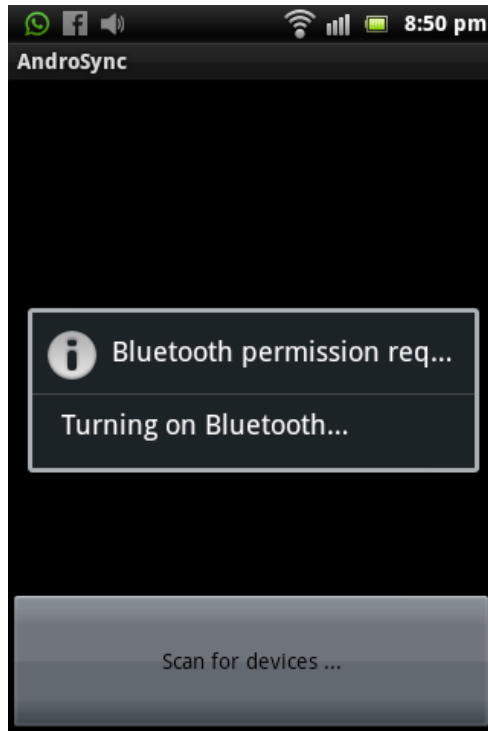


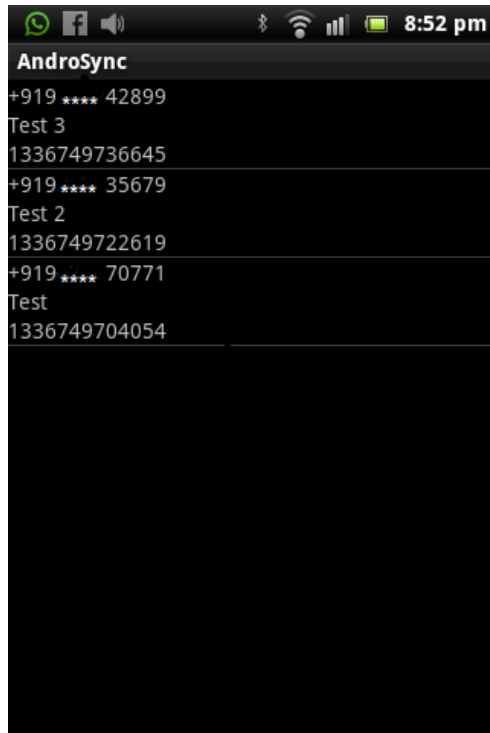


9.2 Client









Chapter 10

Conclusion and Future Scope

10.1 Conclusion

Android operating system is still in its infancy despite of some tremendous progress. The proposed system focuses on some important and fruitful technologies which may assist human race in easier interaction with the digital world. This system has ample scope in the android market where millions of users download and use software on a daily basis. Furthermore the future work could draw in research on similar operations using Java- powered devices.

10.2 Future Scope

- Support Java Platforms: Create application for Java supported phones due to be wide availability.
- Making system compact to increase flexibility, portability
- Extend application to support J2ME devices.
- Make similar server application that can be run another android phone instead of PC.
- Animation of user interactions for simplified use.
- Integrate other features like controlling android device from PC.

References

- [1] *MoSync: A synchronization scheme for cellular wireless and mobile multimedia systems*; Azzendine Boukerche, Sungbum Hong and Tom Jacob
- [2] *Data Synchronization Protocol in Mobile Computing Environment Using SyncML*; Byung-Yun Lee, Tae-Wan Kim, Dae-Woong Kim and Hoon Choi
- [3] <http://developer.android.com>
- [4] <http://funambol.org>
- [5] <http://opensync.org>
- [6] <http://en.wikipedia.org/wiki/SyncML>
- [7] <http://doc.qt.nokia.com/latest/tutorials.html>