MOBILE CLOUDFOR COLLABORATIVE SEARCH

By

Usman JamalKayani

Noor UlIslam

Muhammad ZeeshanKahoot

Submitted to the Faculty of Computer Science, Military College of SignalsNational University of Sciences and Technology,Rawalpindi in partial fulfillment for the requirement of a B.E Degree in Software Engineering

May 2015

CERTIFICATE OF CORRECTNESS AND APPROVAL

Certified that work contained in this thesis “Mobile Cloud for Collaborative Search” carried out by Usman JamalKayani , Noor Ul Islam and Muhammad ZesshanKahoot under the supervision of Asst. Prof. Dr. SarmadSadik for partial fulfillment of Degree of Bachelor of Software Engineering is correct and approved.

Approved by

_____

(Dr. SarmadSadik)

Department of CSE

MCS

Dated: _____

ABSTRACT

Most mobile search tools are designed for individuals, many mobile searches involve people searching with others. Recent research suggests that collaborative search is particularly common on mobile devices - for example, friends headed out to eat might work together on their smart phones to find a good restaurant. One study found that 93 percent of smart phones users have engaged in co-located collaborative search with multiple phones at some point, and another study found that 65 percent of all mobile searches take place with others.

Mobile search collaborators are usually co-located and searching for the same thing at the same time, but can feel distant nonetheless. One reason for this is that mobile device use can stifle direct verbal communication. Additionally, sharing mobile device screens can be awkward, particularly for certain group sizes, demographics, and relationships between collaborators. **CozeApp** addresses these issues by taking advantage of user proximity to augment the collaborative search experience.

People performing collaborative mobile search use body language to signal their willingness to share, usually by moving closer to each other and sharing their screens when they want to discuss results. The **CozeApp**application builds on this interaction by letting users physically signal collaboration intent to the device and those around them by using their phone orientation. **CozeApp**mitigates mobile-related antisocial behavior through features that facilitate and reduce unnecessary or uncomfortable coordination.

DECLARATION

We very solemnly declare that the work presented herewith is the result of sole effort of our group, comprising of Usman Jamal Kayani , Noor Ul Islam and Muhammad ZesshanKahoot, and is free of any kind of plagiarism in part or whole. We also declare that the dissertation has never been submitted previously in part or whole in support of another award or qualification either at this institution or elsewhere

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent, to our parents and teachers without whose unflinching support and unstinting cooperation, a work of this magnitude would not have been possible.

ACKNOWLEDGEMENT

There is no success without the will of Allah. We are grateful to Allah, for the strength, and guidance to accomplish this task. We owe to him in totality, whatever we have achieved. We are grateful to our parents, families and well-wishers for their splendid backing. We are thankful to Computer Science Department , Project Supervisor **Dr. SarmadSadiq**and all Faculty members who encouraged, supported and guided us timely andin rightdirection that lead the team towards the positive culmination and completion of the project.

PREFACE

The future of mobile applications is to provide the ability to harness the power of information technology by providing a centralized working environment for robust actionsthrough rapid sharing of information. Information sharing is easier as well as a difficult job. It is easier when one has access to the concerned authorities and difficult otherwise. Another blockade to information sharing is network availability (Internet). This problem is restrained by providing broadband facility in form of 3G/4G in Pakistan.

In today's era of ever-increasing user generated content and online sharing, computer and information scientists have a renewed interest in collaborative information seeking as an exciting area of research and development, with applications that range from education to e-commerce, with implications for areas that range from libraries to legal informatics.

Information seeking is a cognitive, psychological, and physical activity. It is more than just searching and retrieving information, and it can be performed with or without computing assistance. Information seeking typically takes place in the context of a broader task and involves looking for, collecting and analyzing information, as well as sense making and sharing.

Researchers in the fields of computer science, human computer interaction, information science and library science study how people engage in information seeking in hopes of building more efficient and effective systems to support these activities. To date, much of this research has focused on how individuals undertake these activities: how they search, collect, and make sense of information; and how they use that information in the broader context of a task. Algorithms and systems traditionally assume that one person is searching and reviewing the results; thus, user interfaces support the needs of individual searchers.

As a research area, collaborative information seeking focuses on how groups of people perform these activities based on the idea that information seeking is not always a solitary activity. Collaborative information seeking resembles individual information seeking, but with added dimensions such as the roles the collaborators assume, how they work together across time and space, their awareness of one another's actions, and the negotiations and sharing that must occur. All these examples involve people coming together with intention, looking for and sharing information, and making sense out of shared findings to attain their goals.

TABLE OF CONTENTS

List of Figures

Key to Abbreviations

ADT	Android Development Tool
JDK	Java Development Kit
MC^2S	Mobile Cloud for Collaborative Search
SDK	Software Development Kit
SDLC	Software Development Life Cycle
XML	Extensive Markup Language

Chapter 1

Introduction to Mobile Cloud for Collaborative Search

1.1 Introduction

Information restricted to single machine is of less or no use. It is difficult for the Government organization to have an eye on every inch of national land. Peoples are the major sources of information reporting and sharing for timely action. Information can benatural or manmade disasters bring destruction, chaos and misery to human life and society. Under such circumstance, need for effective and timely communication, collaboration of different organizations and people becomes very much important to carry our various activities of rescue, relief and rehabilitation. Mobile Cloud for collaborative Search is a content management and collaboration platform to support such activities. The contents shared can be used as evidence, incase incident reported is a crime and rescue and relief operations in case of disaster.

Information shared on centralized cloud can be accessed by any user of this app and the content of interest can be searched and download. Also user can upload the contents to the public cloud. Mobile Cloud for collaborative Search has been developed using open standards like XML, SOAP and Web services for interoperability and cross platform support to facilitate organizations in integrating their existing IT solutions

1.2 Background

Timely Information sharing is one of the most critical tasks. Many social apps e.g. Viber, Whatsappetc allows users to communicate and share information with each other. There is a dire need to have an information sharing system, which not only enable users to share information locally without having internet but also on internet. Information that are specific to some government organization will now be easier to disseminated with a message to the concern authorities to take appropriate action.Users can take out a printed copy of information of interest.

1.3 Purpose

The purpose of this document is to describe the implementation of the MC^2S Software described in the MC^2S Requirements specification Document. The MC^2S Software is designed to create and perform content sharing and searching activities. This document will present a detailed design and implementation of the MC^2S. It will explain the implementations of features, functionalities, interfaces and modules of system and what the system will do? And the constraints under which it must operate.

1.4 Product Scope

This project under development is a primarily a cloud based system for File Sharing, Searching, and DownloadingSystem in local proximity by establishing peer to peer cloud as well Printing and incident reporting on internet cloud. App will enable users to upload files to a server with intimation to the concerned authorities. These files can only be viewed by the authorities to take immediate action. This app will be using Wifi and Wifi Direct for communicationpuposes.

1.5 Goals And Objective

Mobile Cloud for collaborative search is a platform that will enable user to share contents robustly for immediate action and or to a cloud that can be accessed by any user to use the content to their requirement and also it enables users to install and print documents and images through google cloud print facility.

1.6 Deliverables

Deliveable Name	**Description**
Software Requirement Specification	Complete Description of **What** system will do, Who will use it. Detailed description of functional and non-functional requirements.
Analysis Document	Detailed requirement analysis and analysis model are included.
Design Document	Complete description of how the system will do and Design Model.

Code	Complete code with API.
Testing Document	Whole system is tested corresponding to the specifications. System is tested at all level of SDLC

1.7 **Overview of Document**

This document is organized keeping in view top-down approach. Big picture of the application is represented by the architecture pattern, style , network and detailed network diagram. And then the bigger architecture is elaborated with the help of use-case diagram. Sequence diagram and at last the class diagram.

Chapter 2

Overall Description

2.1 Product Perspective

Functioning of the system is to enhance file sharing capability of smartphones and to provide user an easy interface of interaction. Where user can view and see available contents uploaded to the cloud by app user and can download required ones. The content uploaded for incident reporting can be seen and downloaded by the authorities only. The system which is being built for an organization like a university (Faculty, students) and organization having closely working groups. It can facilitate user for sharing their personal Files over the internet for easy access and retrieval.

2.2 Product Functions

The major functions that will be provided by our product are:

2.2.1. Connection of local peers through Wifi Direct.

2.2.2. Share, Searching and Downloading files(***Images,Videos, Documents)*** in local proximity (Local Cloud) using WiFi Direct.

2.2.3. Share, Searching and downloading files (Images, Videos, Music, Documents) to and from Internet cloud.

2.2.4. Sharing contents with the Government Organization for immediate action.

2.3 Operating Environment

2.3.1. **Operating System**: Android based mobile operating system

2.3.2. **Hardware Platform**: desktop/laptop and android enabled device.

2.4 Design and Implementation Constraints

The system will be designed and developed under the Following constraints:

2.4.1. This app can be used only on Smartphones with Android OS.

2.4.2. Devices not enabled with WiFi Direct would not be able to use this app.

2.4.3. This app will not be compatible with iPhone.

2.4.4. This app will access all the apps and data on the smart phones to extract information.

2.4.5. Smartphones with this app must be customized to hide your personal information.

2.5 User Documentation

Following user documents shall be provided with the system on deployment:

2.5.1. User Manual.

2.5.2. A CD containing all the information about the system.

2.5.3. System Documentation.

2.6 Assumptions and Dependencies

2.6.1 This app is particularly based on peer to peer architecture for local cloud.

2.6.2 It primarily depends on internet (WiFi /3G /4G) and WiFi Direct. Using Wireless network, user can share and search for contents,So it is assumed that all the users either have WiFi connection or 3G/4G service enabled.

Chapter 3

System Features

3.1 Public Cloud for Information Sharing

3.1 File Sharing

3.1.1 Description and Priority

This feature will allow the user to select peer's files for sharing instantly or to make it available to be searched by other users.

3.1.2 Stimulus/Response Sequences

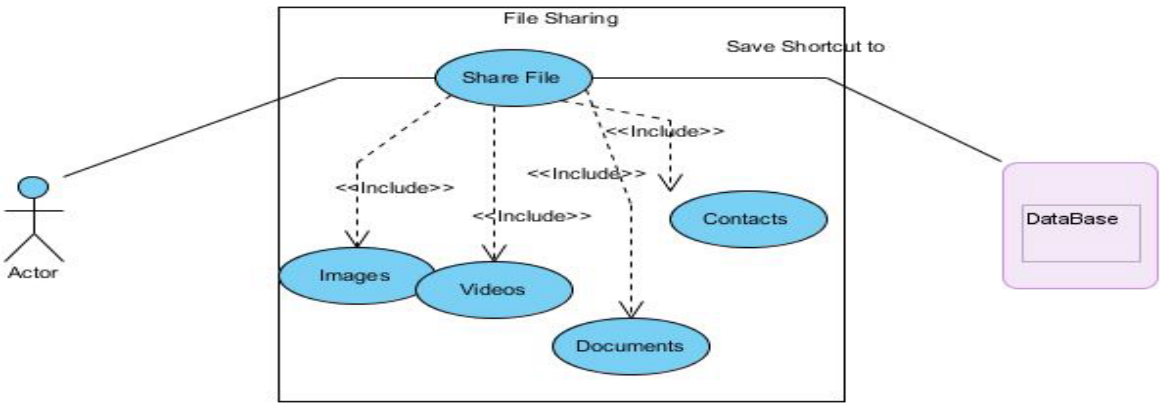
3.1.2.1 **Input**: User will select all the files to be shared.

3.1.2.2 **Output**: System will upload file to the cloud and make it visible to other user inform of thumbnails.

3.1.3 Functional Requirements

3.1.3.1 REQ-1: System should allow user to select files.

3.1.3.2 REQ-2: System should allow user to upload files to cloud.



3.1.4 File Searching

3.1.4.1 Description and Priority

This feature will allow the user to search for a specific file.

3.1.4.2 Stimulus/Response Sequences

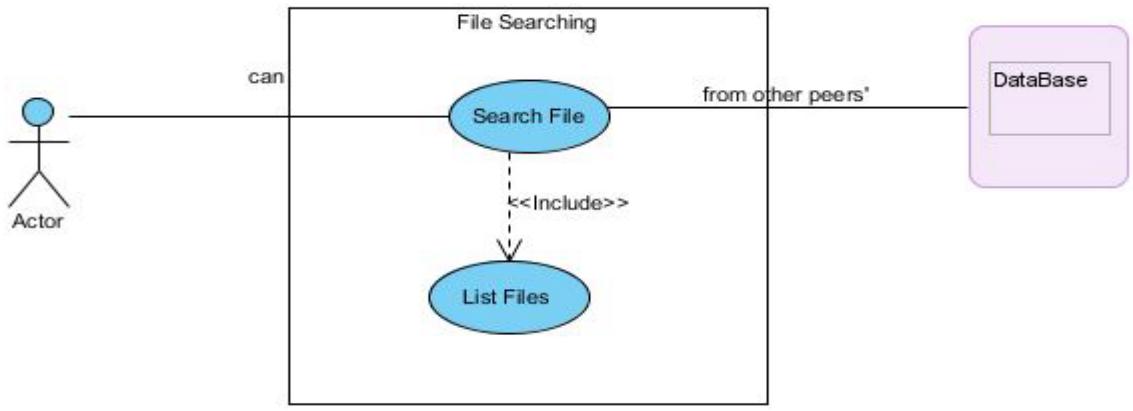
3.1.4.2.1 **Input**: User will tap on search button.

3.1.4.2.2 **Output**: System should list all available files uploaded to cloud.

3.1.4.3 Functional Requirements

3.1.4.4 **REQ-1:** System should allow user to tap search button.

3.1.4.5 REQ-2: System should list all files available on cloud.



3.1.5 File'sDownloading

3.1.5.1 Description and Priority

This feature will allow the user to download the specific file searched.

3.1.5.2 Stimulus/Response Sequences

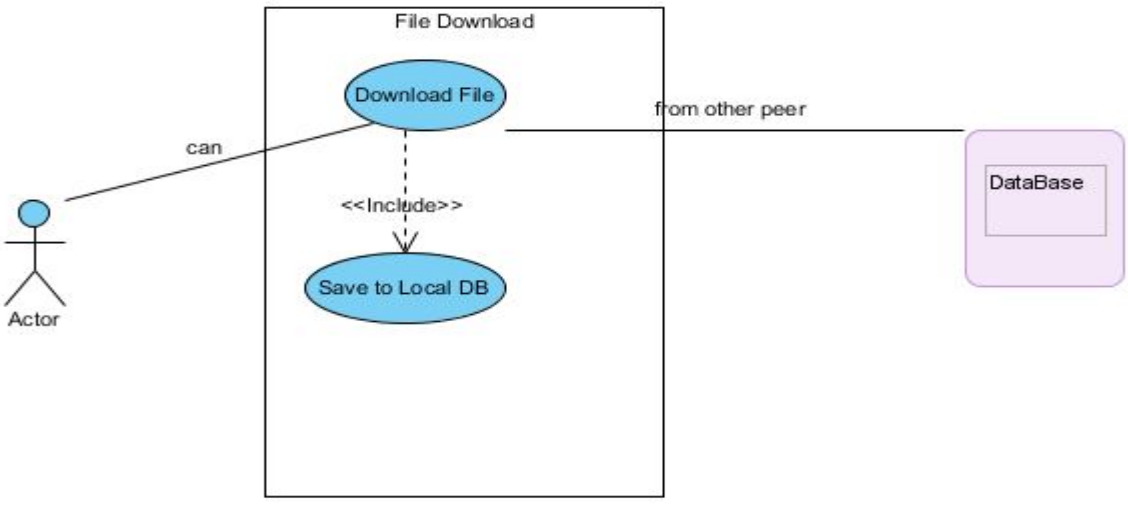
3.1.5.2.1 **Input**: User will select the file to be downloaded.

3.1.5.2.2 **Output**: System will download the selected file to local repository.

3.1.5.3 Functional Requirements

3.1.5.4 REQ-1: System should allow user to select from searched files.

3.1.5.5 REQ-2: System should allow user to download the selected files.



3.2 Private Cloud for Information sharing (Incident Reporting)

3.2.1 Incident Reporting

3.2.1.1 Description and Priority

This feature will allow the user to select peer's existing files or to capture image(s) or video(s) and upload it Private cloud (database) for further examination by concerned Government bodies. Files will not be accessible to all users.

3.2.1.2 Stimulus/Response Sequences

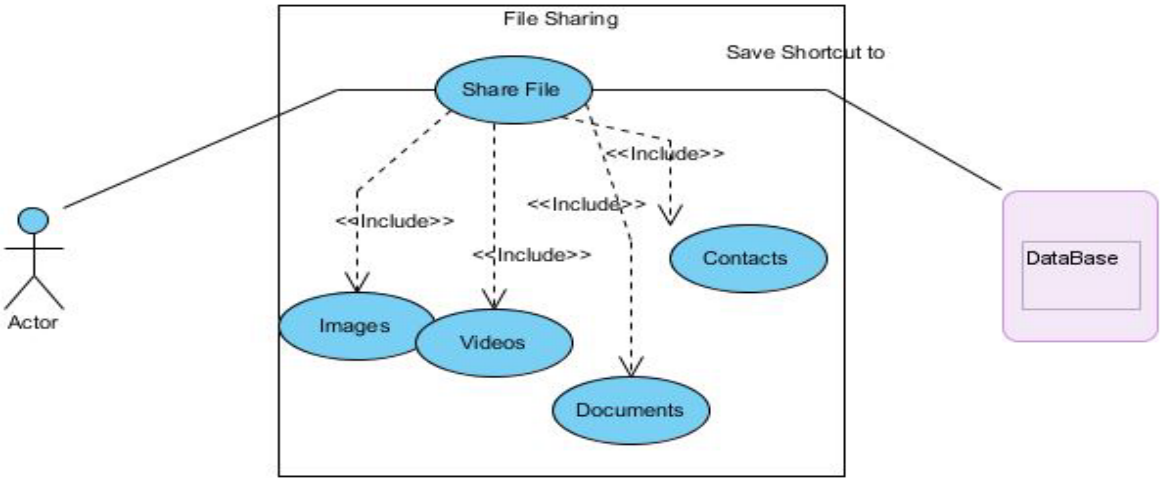
3.2.1.2.1 **Input**: User will select all the files to be shared or capture/Record it. User will enter few details about his/her name, mobile number, location and concern authority.

3.2.1.2.2 **Output**: System will upload file to the cloud and make it visible concerned user.

3.2.1.3 Functional Requirements

3.2.1.3.1 **REQ-1**: System should allow user to select files, capture image and video, and fill up the form.

3.2.1.3.2 **REQ-2**: System should allow user to upload files to cloud.



3.3 Local Cloud for Information sharing

3.3.1 Discover Peer

3.3.1.1 Description and Priority

This feature will allow the user to discover all the peers that have enabled Wifi Direct services on their phones and connect to them.

3.3.1.2 Stimulus/Response Sequences

3.3.1.2.1 **Input**: User will tap the discover peers button and wait for the peers to be listed.

3.3.1.2.2 **Output**: System will connect to the available peers.

3.3.1.3 Functional Requirements

3.3.1.3.1 **REQ-1**: System should allow user to find peers.

3.3.1.3.2 **REQ-2**: System should allow user to connect to the peers.

3.4 Cloud File Printing

3.4.1 Description and Priority

This feature will allow the user to print selected document on the printer installed and registered with google cloud print.

3.4.2 Stimulus/Response Sequences

3.4.2.1 **Input:** User will search for available printers.

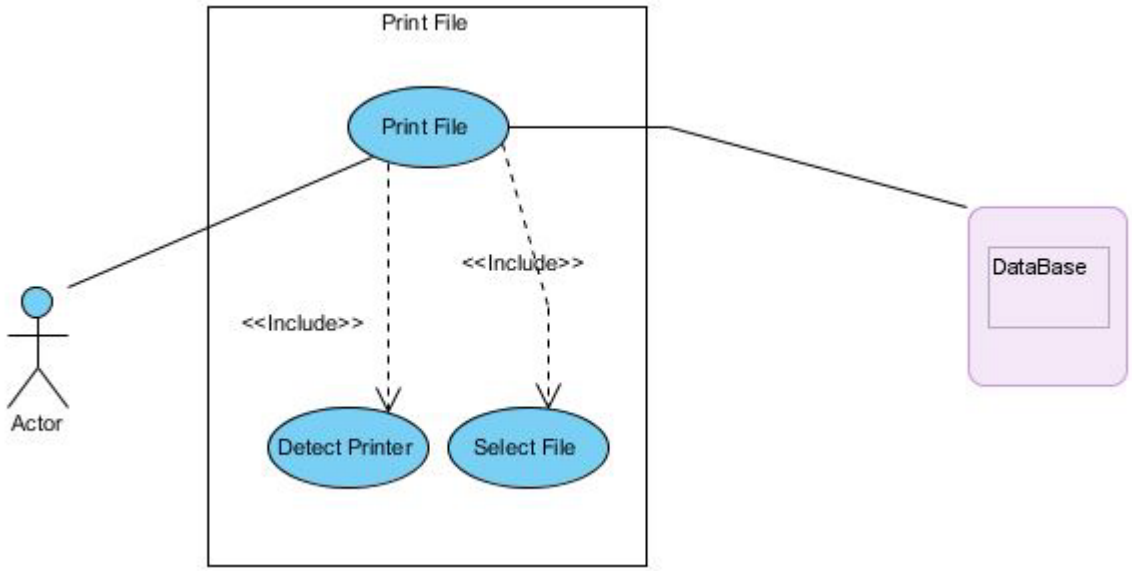
3.4.2.2 **Output:** After connecting to printer system will print the files chosen by user**.**

3.4.3 **Functional Requirements**

3.4.3.1 **REQ-1:** System should be able to list registered printer.

3.4.3.2 **REQ-2:** System should be able to connect to the printer.

3.4.3.3 **REQ-3:** System should allow user to select file(s) for printing.



Chapter 4

System Architecture

4.1 System Network Diagram

4.1.1. **Devices Connected through WiFi Direct (Left Side of Figure 1).** All devices are connected to each other using network protocols known as NFC (WiFi Direct) protocols.

4.1.2. **Devices Connected through WiFi (Right Side of Figure 1).** All the smart phone are connected to internet through a router/3G/4G. And can access the centralized cloud.

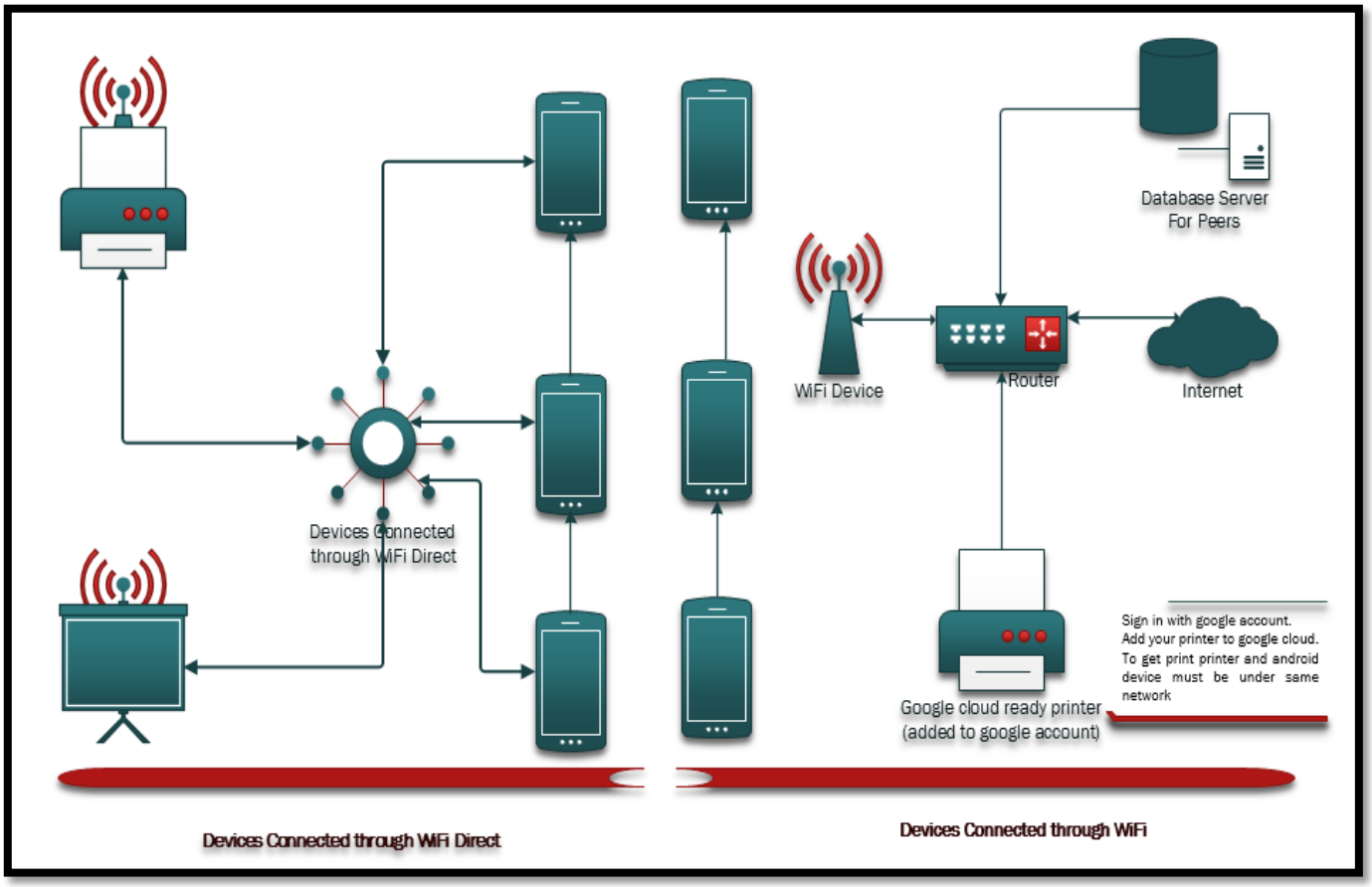


Figure 1: System Network Diagram

4.2 System Block Diagram

4.1.1 **Devices Connected through WiFi Direct (Left Side of Figure 2).** All the device are connected to each other using NFC(WiFi Direct) protocols. These smartphones can interact to transfer and share files.

4.1.2 **Devices Connected through WiFi (Right Side of Figure 2).** All the smart phones are connected through internet and they can interact with cloud to transfer files on internet and can print files as well.

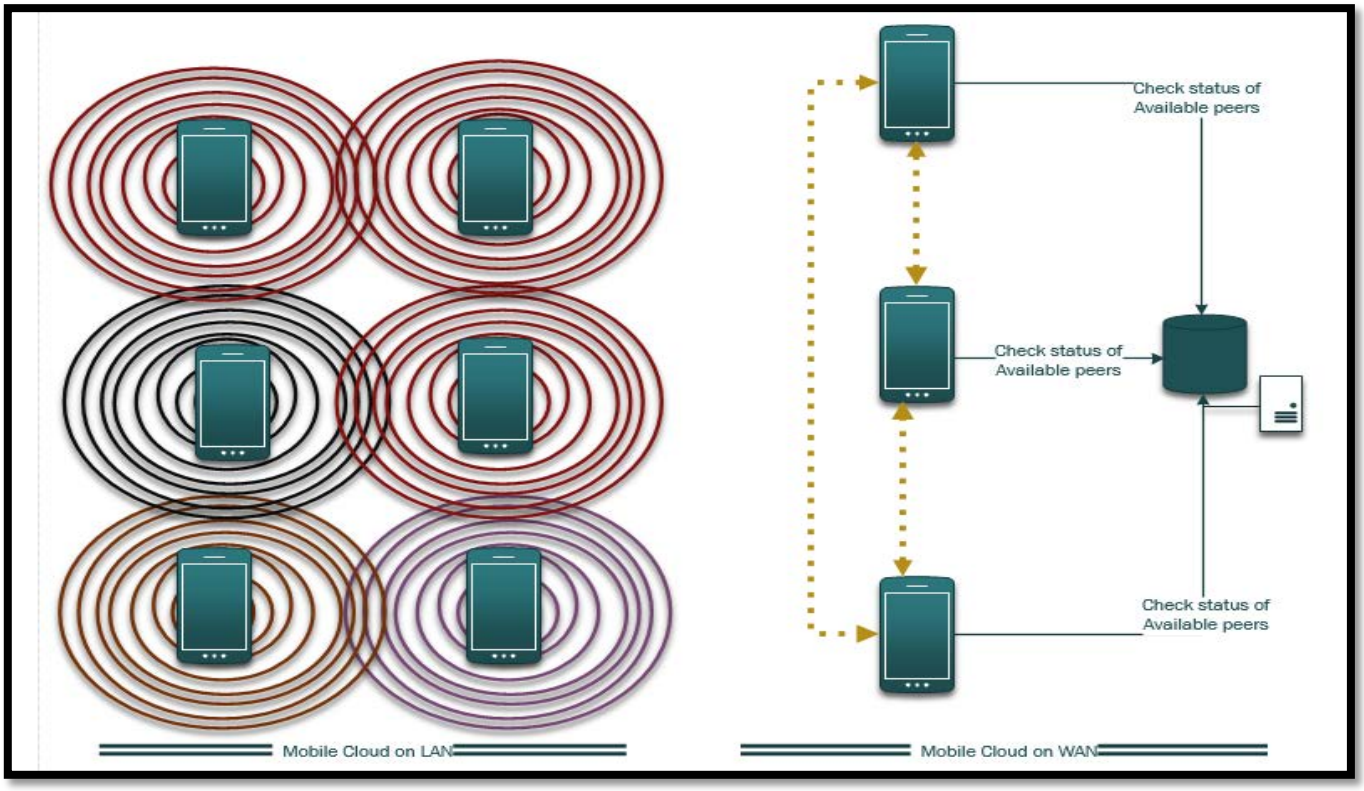


Figure 2: Block System Diagram

4.3 Architectural Style

4.3.1 The architectural style of MC^{2}S not only improves partitioning but also promotes design reuse by providing solutions to frequently recurring problems. Figure 3 describes the overall architectural style used for designing this application:

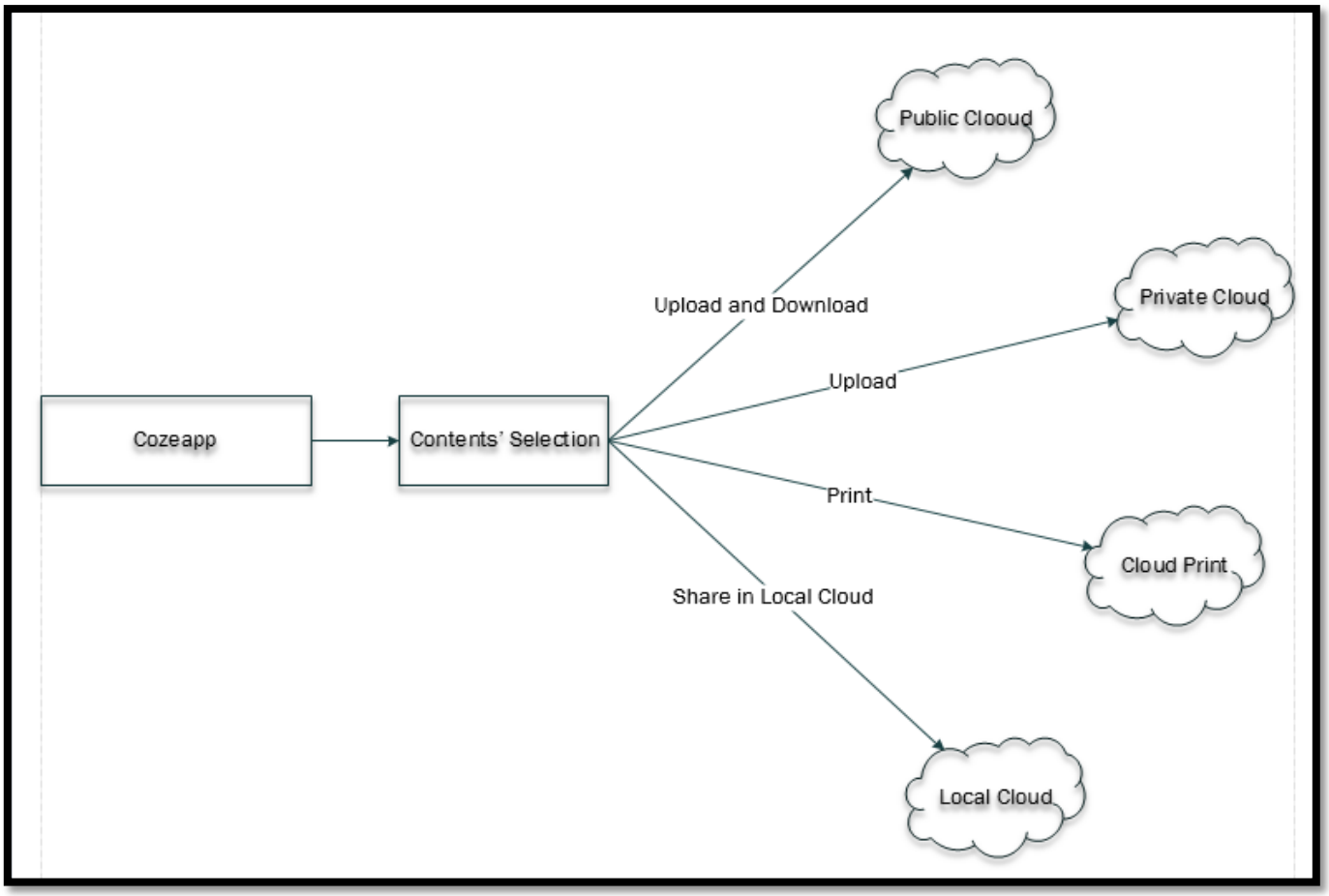


Figure 3: System Architecture Style

4.4 Architectural Pattern

4.4.1 Layered architecture focuses on the grouping of related functionality within into distinct layers, which are stacked vertically on top of each other. Functionality within each layer is related by a common role or responsibility.

4.4.2 The subsystems of this application shall be constructed and integrated in layers. Each subsystem constitutes a separate layer in the overall system design. The follow figure shows the division of system into 3 layers. Data communication is

only done between two consecutive layers. The 3 layers are briefly described as follows:

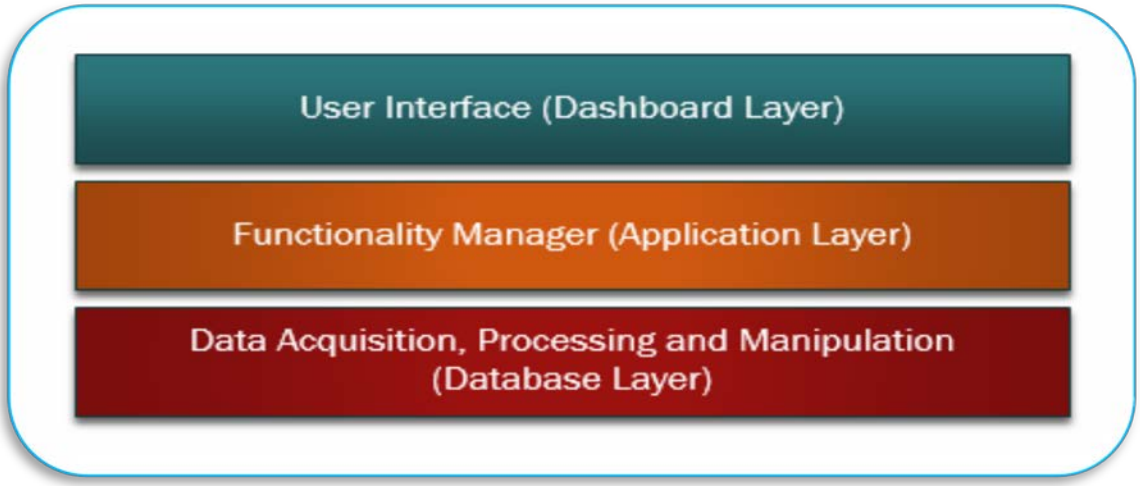


Figure 4: Layered Architecture of System

4.4.2.1 **Data Acquisition, Processing & Manipulation Layer (Database Layer).**

This Layer constitutes creation of data bases for various types of contents available on the user device. Fetching information about these contents and providing these contents based on the initiated queries. While uploading contents the devices act as a server and requesting devices act as clients. Sqlite databases will be created for all type of contents.

4.4.2.2 **Functionality Manager (Application Layer).** This Layer is responsible for displaying the data received from Data Acquisition, Processing & Manipulation Layer of the same device or other devices and finally transferring it for permanent storage or forwards it on demands of other peers. In case that any data is required for processing, report generation etc. this layer fetches previously stored data from Data Storage Layer.

4.4.2.3 **User interface (Dashboard Layer).** This Layer constitutes the user interface. It provides users over the intranet / internet to have access the functionalities provided by the system. User can swap around between various functionalities. Furthermore, this layer is responsible for

allowing the authorized viewers to view the data of users and share the information via email or download.

4.5 Overview of Modules/Components

4.5.1 The system is divided into three main modules as shown in Figure 5.

4.5.1.1.1 **Connectivity and Network Module**. This module will handle all the network related issues pertaining to the application. Which include peer to peer connectivity, load management on single peer.Type of protocols to be used on application, network and data link layer. Handling security related issues at each peer.

4.5.1.1.2 **Functionality Module**. This module will handle all the functionalities provided by the application.

4.5.1.1.3 **Server (Cloud) Module**. This module will maintain the records of all the data uploaded by the users and make it readily available as and when the peers gets connected in form of thumbnails.

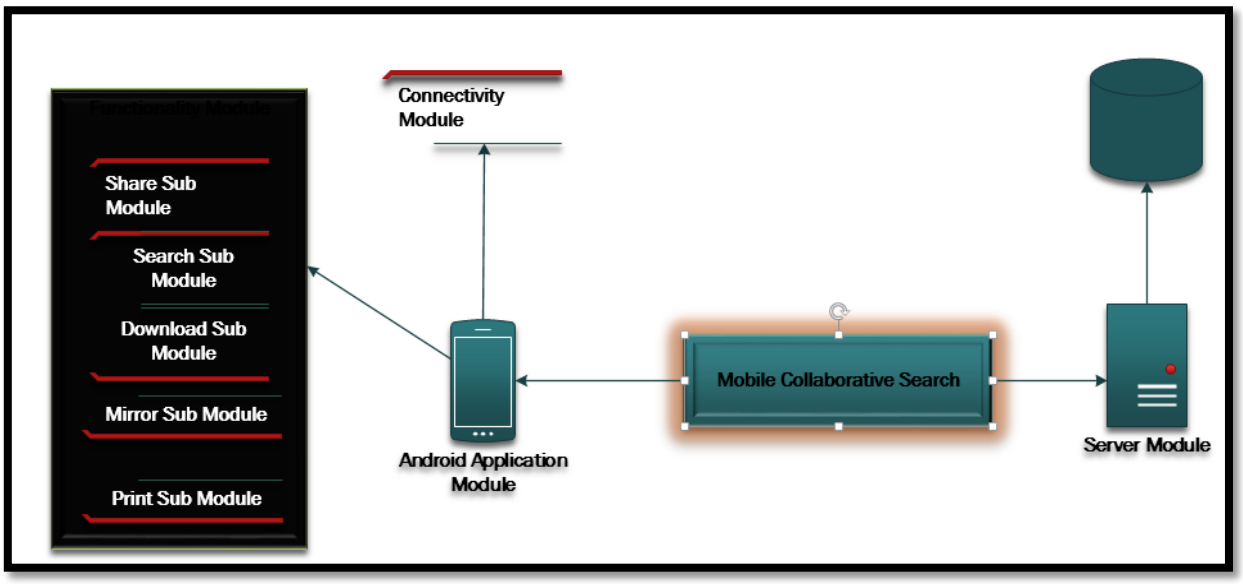


Figure 5: System Modules

4.6 Structure and Relationship

4.6.1 Structural View

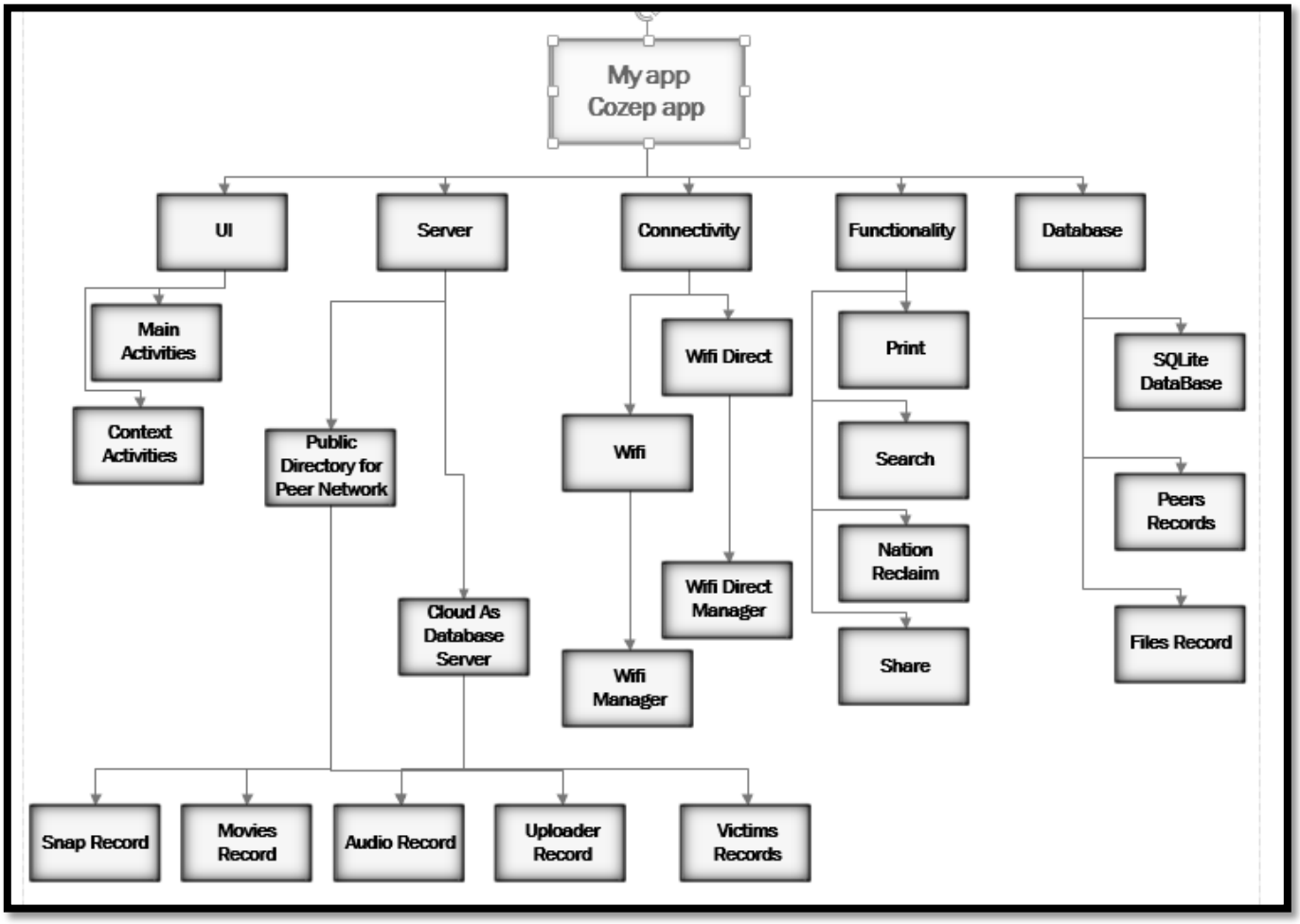


Figure 5a: Structure

4.6.2 Relationship in Functionalities

4.6.2.1 Finite state machine shows the various states of the application for the given inputs. The inputs change the application from one state to another. Inputs that transit application from one state to another are represented by I_n where (n=0,1,2,……n) and states are represented by S_n where (n=0,1,2,……n) as shown in Figure 6.

4.6.2.2 Transition Inputs of the application are as follow:

4.6.2.2.1 I0 : Power button pressed

4.6.2.2.2 I1 : Open App

4.6.2.2.3 I2 : close App

4.6.2.2.4 I3 : Return back

4.6.2.2.5 I4 : Share Document

4.6.2.2.6 I5 : Print Document

4.6.2.2.7 I6 : Mirror Document

4.6.2.2.8 I6a : Search Document

4.6.2.2.9 I7 : Find Friends

4.6.2.2.10 I8 : Mark Location

4.6.2.2.11 I9: connection Fails

4.6.2.2.12 I10 :Wifi

4.6.2.2.13 I11 :Wifi Direct

4.6.2.2.14 I12 : Logout

4.6.2.2.15 I13 : Connection or Authentication Fails

4.6.2.2.16 I14 : Authentic User

4.6.2.2.17 I15: Send mail

4.6.2.3 States of the application are as follow:

4.6.2.3.1 S0 : Phone OFF

4.6.2.3.2 S1 : Phone ON

4.6.2.3.3 S2 : App Activated

4.6.2.3.4 S3 : App Deactivated

4.6.2.3.5 S4 : Document Shared

4.6.2.3.6 S5 : Document Printed

4.6.2.3.7 S6 : Document Mirrored

4.6.2.3.8 S7 : Search Result

4.6.2.3.9 S8 : Friends found list

4.6.2.3.10 S9 : Location Marked

4.6.2.3.11 S10: Main Apps Screen

4.6.2.3.12 S11: Connected to server

4.6.2.3.13 S12: Email sent to group

4.6.2.3.14 S13: Connected to paired devices

4.6.2.4 Transition of states of the application for the given inputs shows various operation that can be performed as follow:

4.6.2.4.1 In **STATE S0 (PHONE OFF)** on input *I0 (POWER BUTTON PRESSED)* the phone goes to **STATE S1 (PHONE ON)**.

4.6.2.4.2 In **STATE S1 (PHONE ON)** on input *I0 (POWER BUTTON PRESSED)* the phone goes to **STATE S0 (PHONE OFF)**.

4.6.2.4.3 In **STATE S1 (PHONE ON)** on input *I1 (OPEN APP)* the application goes to **STATE S2 (APP ACTIVATED)**.

4.6.2.4.4 In **STATE S2 (APP ACTIVATED)** on input *I0 (POWER BUTTON PRESSED)* the application goes to **STATE S0 (PHONE OFF)**.

4.6.2.4.5 In **STATE S2 (APP ACTIVATED)** on input *I2 (CLOSE APP)* the application goes to **STATE S3 (APP DEACTIVATED) and STATE S10 (MAIN APPS SCREEN)**.

4.6.2.4.6 In **STATE S2 (APP ACTIVATED)** on input *I3 (RETURN BACK)* the application goes to **STATE S10 (MAIN APPS SCREEN)**.

4.6.2.4.7 In **STATE S2 (APP ACTIVATED)** on input *I10 (WIFI)* the application goes to **STATE S11 (CONNECTED TO SERVER)**.

4.6.2.4.8 In **STATE S2 (APP ACTIVATED)** on input *I11 (WIFI DIRECT)* the application goes to **STATE S13 (CONNECTED TO PAIRED DEVICES)**.

4.6.2.4.9 In **STATE S11 (CONNECTED TO SERVER)** on input *I14 (AUTHENTIC USER)* the application goes to **STATE S11 (CONNECTED TO SERVER)**.

4.6.2.4.10 In **STATE S11 (CONNECTED TO SERVER)** on input *I12 (LOGOUT)* **or** *I13 (CONNECTION OR AUTHENTICATION FAILS)* the application goes to **STATE S2 (APP ACTIVATED)**.

4.6.2.4.11 In **STATE S11 (CONNECTED TO SERVER) or STATE S13 (CONNECTED TO PAIRED DEVICES)** on input *I8 (MARK LOCATION)* the application goes to **STATE S9 (LOCATION MARKED)**.

4.6.2.4.12 In **STATE S11 (CONNECTED TO SERVER) or STATE S13 (CONNECTED TO PAIRED DEVICES)** on input *I7 (FIND FRIEND)* the application goes to **STATE S8 (FRIENDS FOUND LIST)**.

4.6.2.4.13 In **STATE S11 (CONNECTED TO SERVER) or STATE S13 (CONNECTED TO PAIRED DEVICES)** on input *I6a (search Document)* the application goes to **STATE S7 (SEARCH RESULT)**.

4.6.2.4.14 In **STATE S11 (CONNECTED TO SERVER) or STATE S13 (CONNECTED TO PAIRED DEVICES)** on input *I6 (MIRROR DOCUMENT)* the application goes to **STATE S6 (DOCUMENT MIRRORED)**.

4.6.2.4.15 In **STATE S11 (CONNECTED TO SERVER) or STATE S13 (CONNECTED TO PAIRED DEVICES)** on input *I5 (PRINT DOCUMENT)* the application goes to **STATE S5 (DOCUMENT PRINTED)**.

4.6.2.4.16 In **STATE S11 (CONNECTED TO SERVER) or STATE S13 (CONNECTED TO PAIRED DEVICES)** on input *I4 (SHARE DOCUMENT)* the application goes to **STATE S4 (DOCUMENT SHARED)**.

4.6.2.4.17 In **STATE S11 (CONNECTED TO SERVER) or STATE S13 (CONNECTED TO PAIRED DEVICES)** on input *I14 (SEND MAIL)* the application goes to **STATE S12 (EMAIL SENT TO GROUP)**.

4.6.2.4.18 In **STATE S9 (LOCATION MARKED)** on input *I8 (MARK LOCATION)* the application goes to **STATE S9 (LOCATION MARKED)**.

4.6.2.4.19 In **STATE S8 (FRIENDS FOUND LIST)** on input *I7 (FIND FRIEND)* the application goes to **STATE S8 (FRIENDS FOUND LIST)**.

4.6.2.4.20 In **STATE S7 (SEARCH RESULT)** on input *I6a (search Document)* the application goes to **STATE S7 (SEARCH RESULT)**.

4.6.2.4.21 In **STATE S6 (DOCUMENT MIRRORED)** on input *I6 (MIRROR DOCUMENT)* the application goes to **STATE S6 (DOCUMENT MIRRORED)**.

4.6.2.4.22 In **STATE S5 (DOCUMENT PRINTED)** on input *I5 (PRINT DOCUMENT)* the application goes to **STATE S5 (DOCUMENT PRINTED)**.

4.6.2.4.23 In **STATE S4 (DOCUMENT SHARED)** on input *I4 (SHARE DOCUMENT)* the application goes to **STATE S4 (DOCUMENT SHARED)**.

4.6.2.4.24 In **STATE S12 (EMAIL SENT TO GROUP)** on input *I15* the application goes to **STATE S12 (EMAIL SENT TO GROUP)**.

4.6.2.4.25 In **STATE S9 (LOCATION MARKED), STATE S8 (FRIENDS FOUND LIST), STATE S7 (SEARCH RESULT), STATE S6 (DOCUMENT MIRRORED), STATE S5 (DOCUMENT PRINTED), STATE S4 (DOCUMENT SHARED), STATE S12 (EMAIL SENT TO GROUP)** on input *I3 (RETURN BACK)* the application goes to **STATE S10 (MAIN APPS SCREEN)**.

4.6.2.4.26 In **STATE S10 (MAIN APPS SCREEN)** on input *I0 (POWER BUTTON PRESSED)* the application goes to **STATE S0 (PHONE OFF)**.

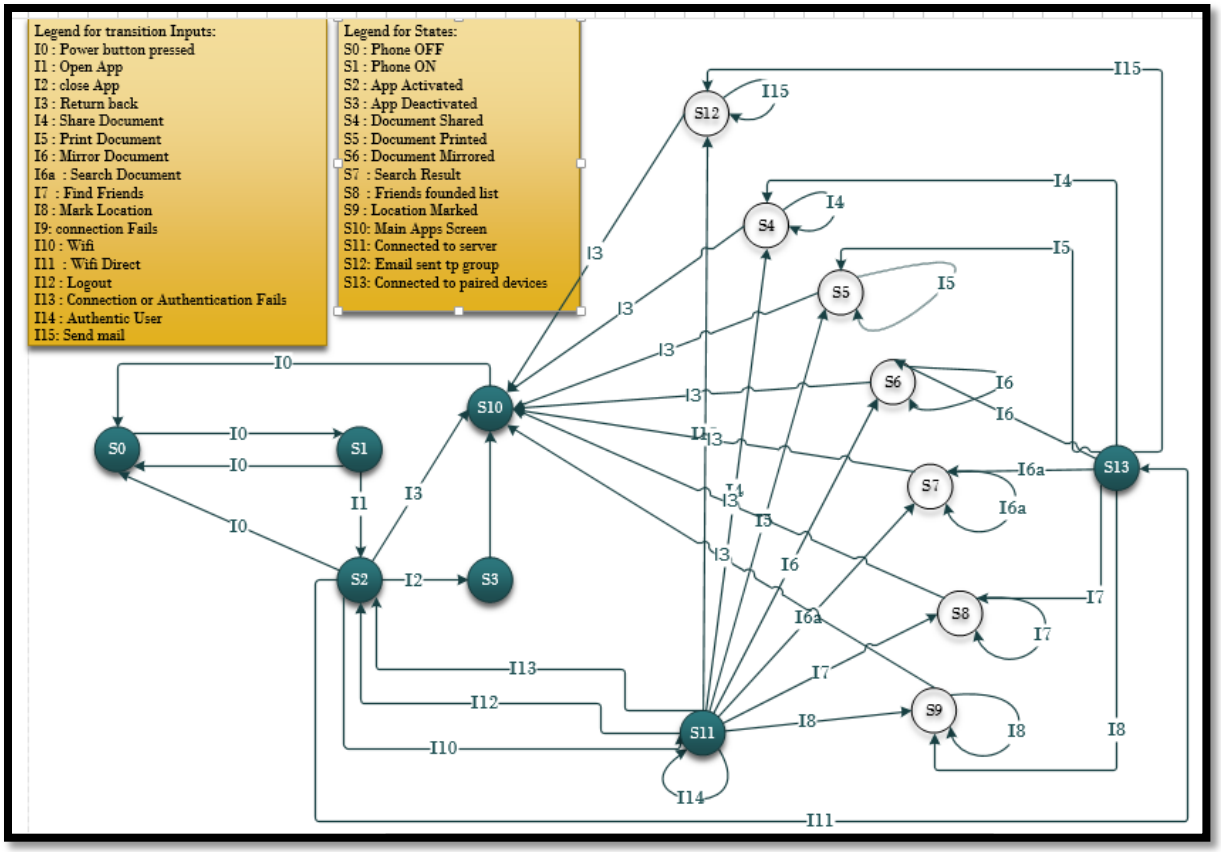


Figure 6: Finite State Machine Diagram of the System

4.7 User Interface Issues

4.7.1 When user launches this app from the main apps panel of android phones. It will automatically push back all other opened apps and seeks highest priority of launching. On receiving a voice call this app will automatically go into background and allow the user to receive a call. And at the end of call the app will resume its status. Messages and other notifications will have no effects on the application. However their usual notification sounds will be played. On closing the app, all the instances and memory utilized by the app will be freed up and app will go to least memory utilization state. User will interact with the google gmail interface and printer interface to complete its two essential task. The overall functioning of the system is to enhance file sharing capability of smartphones and to provide user an easy interface of interaction. Where user can view and see available contents on all the connected peers and can download required ones. The system which is being built for an organization like a university (Faculty, students) and organization having

closely working groups. It can facilitate user for sharing their personal Files over the internet for easy access and retrieval.

4.7.1.1. Creating activities, xmls, manifest, icons.

4.7.1.2. Supporting multiple sizes of screens.

4.7.1.3. Supporting multiple android SDKs.

4.7.1.4. Maintaining Activity for each independent task.

4.7.1.5. Keeping context activities.

4.7.1.6. Maintaining xml, resources etc.

4.7.1.7. Putting interfaces on stacks when new activity is generated.

4.7.1.8. Providing user with dialog boxes during interactions.

4.7.1.9. Asking for user permissions if user wants to undo the previous selected activity.

4.7.1.10. Maintain and updating the manifest files for each activity.

4.7.1.11. Maintaining and xml files for both landscape and portrait mode.

4.7.1.12. Making application sensitive to the orientation.

4.7.1.13. Selection and maintaining themes for each activity and for the base application.

4.7.1.14. Notification to user when new user gets online

4.8 Operating Environment

Our project contains following operating environment components:

4.8.1 Operating System:

4.8.1.1 Android OS

4.8.1.2 Windows XP or above (for Simulation & Implementation)

4.8.1.3 Cloud database

4.8.2 Platforms:

4.8.1.1 Android SDK(Software Development Kit)

4.8.1.2 Eclipse IDE(Integrated Development Environment)

4.8.1.3 Java SDK 1.6

4.8.3 Hardware:

4.8.1.1 Laptop/Desktop

4.8.1.2 Smartphones

4.8.1.3 Tablets etc

Chapter 5

Detailed Description of Components

5.1 **Component Template Description:-** This section will describe following components one by one and then the most essential activities will be elaborated with the help of SDS component template.

5.2 **Detailed Architectural Pattern:-** The detailed architectural pattern shows the functionality layer in detail as shown in Figure. Other two layers are already discussed in the Architeucre.

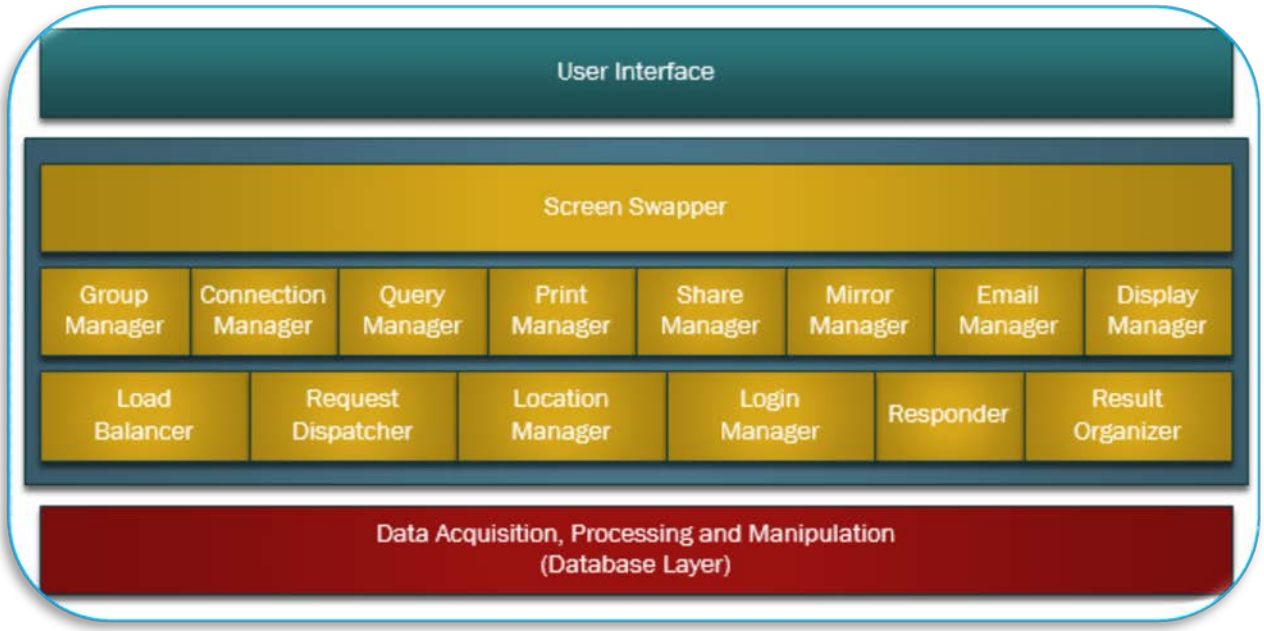


Figure 7: Architectural Pattern

5.2.1 **Group Manager:-** User can manage groups in local proximity by allowing or restricting the groups.

Identification	**Group Manager**
Type	Class
Purpose	To organize contacts in groups both on Wifi and Wifi direct
Function	1. To enable user to create group(s). 2. To enable user to set preference of the group. 3. To enable user to add contacts to group.

	4. To enable user to remove contacts from a group. 5. To enable user to send mail to a group.
Dependencies	1. This components uses class connection manager. 2. Group manager is user by other classes like search and share and email.

5.2.2 **Connection Manager:-** Usedto enable user to connect through Wifi or Wifi direct.

Identification	Connection Manager
Type	Class
Purpose	To enable user to connect through Wifi or Wifi direct.
Function	To enable user to select the network. To get the status of current connection. To search available peers in local proximity.
Dependencies	Connection manager is user by other classes like search, share and email.

5.2.3 **Query Manager:-** Usedto enable user to input query and fetch the results.

Identification	Query Manager
Type	Class
Purpose	To enable user to get results from connected peers for the typed query.
Function	To enable user to write a query. To parse the query. To send query to request dispatcher for broadcasting. To get the results from Display manager.
Dependencies	It depends on Request dispatcher for dispatching the query. It depends on display manager to get the results.

5.2.4 **Print Manager:-** Usedto enable user to select images or documents for printing through the printer installed on google cloud.

Identification	Print Manager
Type	Class
Purpose	To enable user to get the desired document printed.
Function	To enable user to select a File for printing. To detect printer connected through Wifi or Wifi Direct. To send document for printing.
Dependencies	It depends on google print cloud API for printing on internet.

5.2.5 **Share Manager:-** Usedto enable user to select images or documents for Sharing onto the cloud as well as in local cloud.

Identification	Share Manager
Type	Class
Purpose	To enable user to get the desired document shared.
Function	To enable user to select a File for sharing. To mark the file as shared. To share document with closing working peers connected through WiFi Direct.
Dependencies	It depends on Request Dispatcher for sending request to the peers for sharing documents. It depends on display manager to get the results.

5.2.6 **Load Balancer:-** Usedto work in background to look for the system not being overburdened.

Identification	Load balancer
Type	Class
Purpose	To restrict device being overloaded by too many request.
Function	To limit the number of user to connect to the device for searching.

	To maintain waiting queue. To limit number of peers for mirroring the files.
Dependencies	It is used by mirror manager and all other classes.

5.2.7 **Login Manager:-** Usedto enable the user to login to the account on which the printer is installed.

Identification	Login Manger
Type	Class
Purpose	To enable user to login to server and get status of connected peers.
Function	To enable user to enter login credentials. To authenticate user on network. To enable user to use WiFi services to connect to other peers.
Dependencies	It uses connectivity manager.

5.2.8 **Responder:-** Thiswill respond when the query is received by mobiles working in local cloud.

Identification	Responder
Type	Class
Purpose	To get the response from other peers.
Function	To receive the response messages. To receive the results. To forward the results to results organizer.
Dependencies	It uses connectivity manager.

5.2.9 **Result Organizer:-** This is used to organize results for display to the user.

Identification	Result Organizer
Type	Class
Purpose	To organize the results received by the responder.
Function	To receive and organize results.

	To pass results to display manager.
Dependencies	It uses connectivity manager and responder. It is used by display manager.

5.3 **System Flow Diagram**. This Diagram shows the overall flow of the activities from loading the application to closing the application.

5.3.1 The system is activated and when the application is selected by the user.

5.3.2 Then it prompts user to select the connectivity type i.e. WiFi or WiFi Direct.

5.3.3 On selection of WiFi the user is prompted to enter his login credentials. These credentials are verified by the server and then user can carry out normal operations and functionalities provided by the system. And can also check other connected peers.

5.3.4 On selection of WiFi-Direct the devices go on synchronization state, where all other peers are listed and these peers can be added to a group or can be removed as well from group.

5.3.5 User can perform normal functions provided to the system.

5.3.6 After closing the application the app is deactivated and the user is taken back to the main apps panel of android phone.

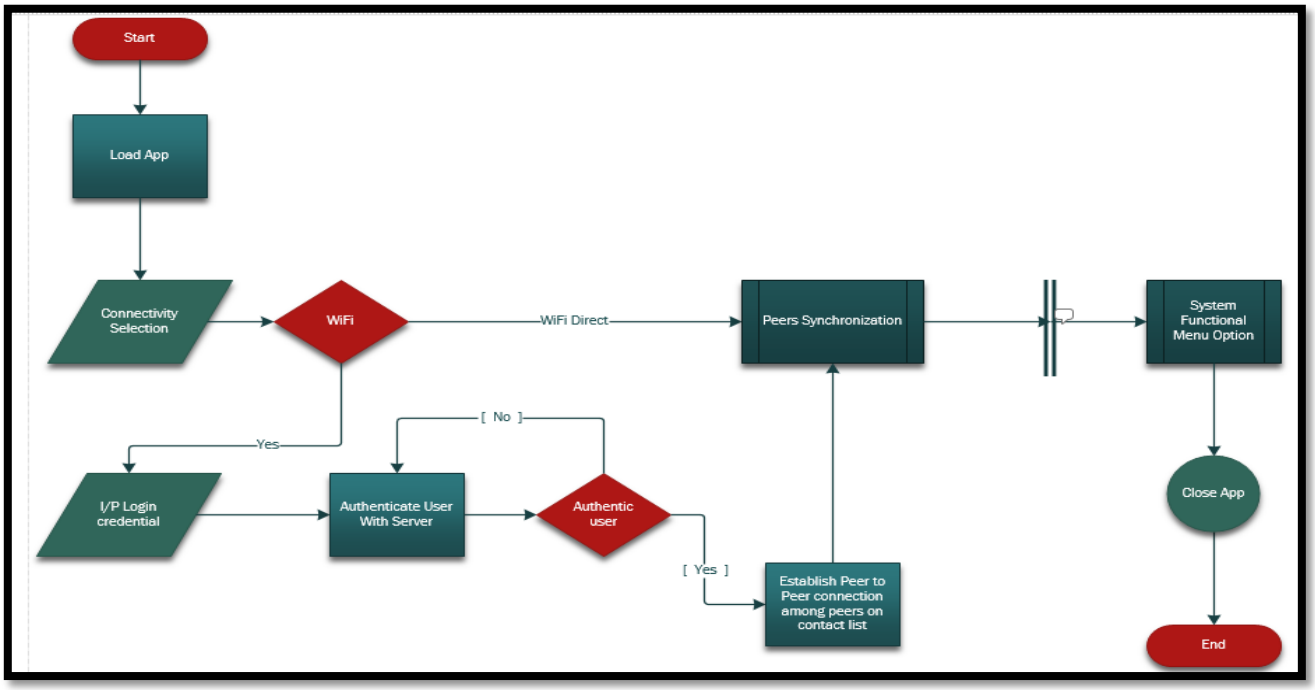


Figure 8: **System Flow Diagram**

5.4 **Activity Diagram**. The activity diagram shows the detailed workout of system flow diagram.

5.4.1 The system is activated and when the application is selected by the user.

5.4.2 Then it prompts user to select the connectivity type i.e. WiFi or WiFi Direct.

5.4.3 On selection of WiFi the user is prompted to enter his login credentials. These credentials are verified by the server and then user can carry out normal operations and functionalities provided by the system. And can also check other connected peers. If the user is already registered, he will only be verified by the server, otherwise he will presented with dialogue to get registered. Incase the login credentials are incorrect; he will be presented with dialogue to renter again his credentials. Successful login will take him to use the system functionalities.

5.4.4 On selection of WiFi-Direct the devices go on synchronization state, where all other peers are listed and these peers can be added to a group or can be removed as well from group. Following activities are performed during synchronization.

5.4.5 Scan active devices.

5.4.6 Join already subscribed groups.

5.4.7 Connect to the previously paired devices if available.

5.4.8 View online devices (paired or to be paired).

5.4.9 User can perform normal functions provided to the system. These functions are as follow:-

5.4.9.1 Search

5.4.9.2 Share

5.4.9.3 Print**.**

5.4.10 After closing the application the app is deactivated and the user is taken back to the main apps panel of android phone.

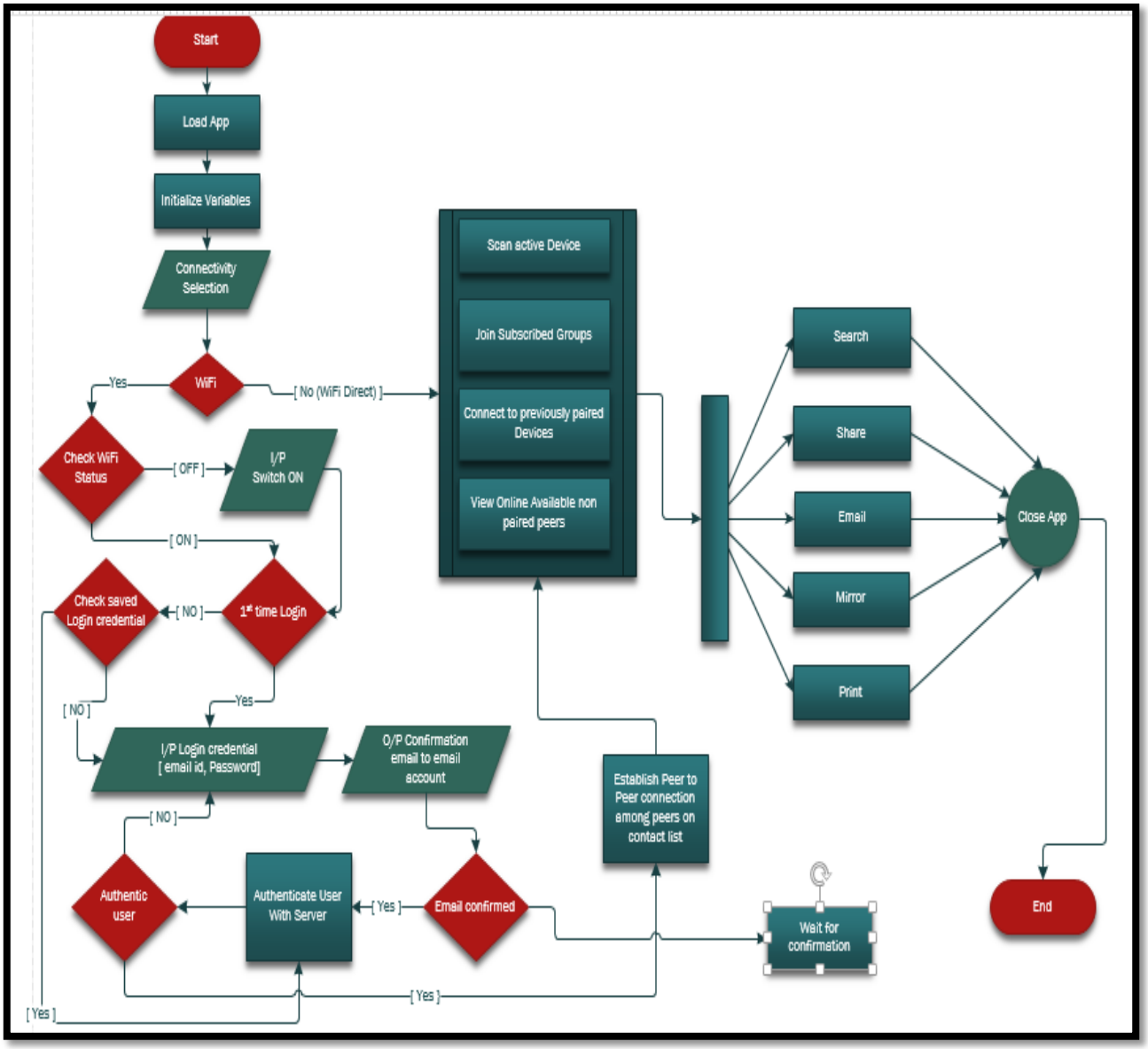


Figure 9: Activity Diagram

5.5 **Use case Diagram**:- Use case for overall system is shown here, which points the basic functionalities carried out by the user interacting with the application. Detailed Use Case for Share File, Search File,Print File, DownloadFile and ManageConnection are attached as Appendix C

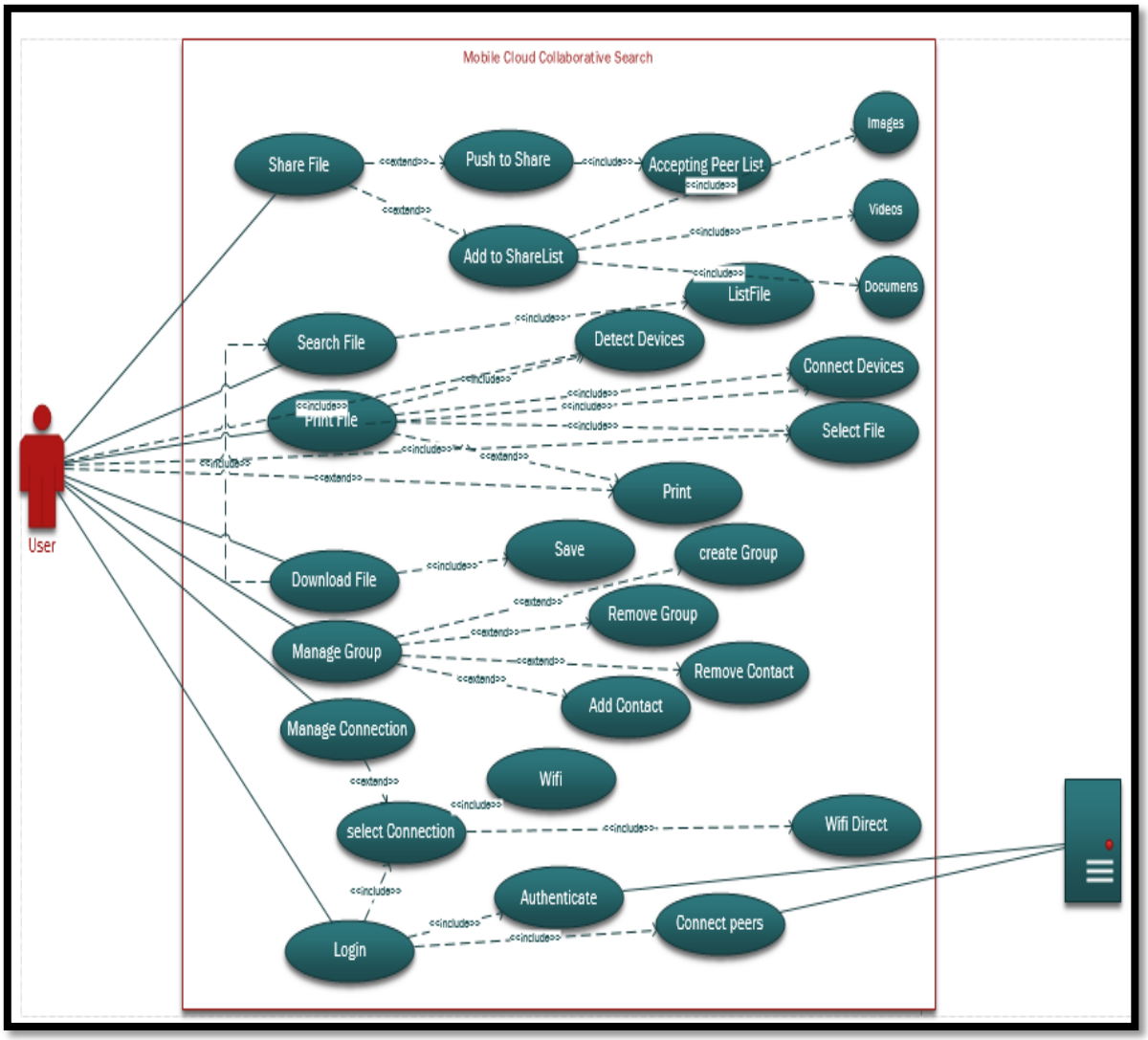


Figure 10: Use case Diagram

5.6 **Class Diagram**

The class diagram is shown in Figures below. It is consist of following classes. Which are the main components of application..

5.6.1 **MainComponent**

Identification	Main
Type	Class/Activity
Purpose	This is the class/activity required to display the initial user interface when the app is created.
Function	1. It holds UI widgets to go to the specific function/Activity. 2. To return back to this class from each activity being performed. 3. To display the result in user friendly interface.
Dependencies	1. Itself it is independent class. 2. All other activities depend on this class. 3. When this class is destroyed the application will fail to launch. And all other classes would be destroyed.

5.6.2 **File Component**

Identification	File
Type	Class/Activity
Purpose	This class will be used by its derived classes to get information about the documents.
Function	1. To let derived classes implements its functions. 2. To get information about existence of file. 3. To get information about file's attributes. 4. To get information about file's Size. 5. To get information about file's Last Modified. 6. To get information about file's path.
Dependencies	1. Itself it is abstract class. 2. It has main two derived classes Share and Search. 3. It holds strong composition relationship with **Main class**.

5.6.3 **Connection Manager**

Identification	Connection Manager
Type	Class/Activity
Purpose	To enable user to connect through Wifi or Wifi direct.
Function	1. To enable user to select the network. 2. To get the status of current connection. 3. To search available peers in local proximity.
Dependencies	1. Connection manager is user by other classes like search, share and email.

5.6.4 **Cloud Database**

Identification	Server
Type	Class
Purpose	To enable user to login to server and get status of connected peers.
Function	1. To confirm the login credentials. 2. To authenticate user on network. 3. To enable user to use WiFi services to connect to other peers.
Dependencies	1. It uses connectivity manager.

5.6.5 **Share**

Identification	Share Manager
Type	Class
Purpose	To enable user to get the desired document shared.
Function	1. To enable user to select a File for sharing. 2. To mark the file as shared. 3. To share document with closing working peers connected through WiFi Direct.
Dependencies	1. It depends on Request Dispatcher for sending request to the peers for sharing documents. 2. It depends on display manager to get the results.

5.6.6 **Search**

Identification	Query Manager
Type	Class
`Purpose	To enable user to get results from connected peers for the typed query.
Function	1. To enable user to write a query. 2. To parse the query.

	3. To send query to request dispatcher for broadcasting. 4. To get the results from Display manager.
Dependencies	1. It depends on Request dispatcher for dispatching the query. 2. It depends on display manager to get the results.

5.6.7 **Download File**

Identification	Download File
Type	Class
Purpose	To enable user to download contents.
Function	1. To enable user to select select File for downloading. 2. To download and save file to device. 3. To update database.
Dependencies	1. It depends on request dispatcher to broadcast message. 2. It depends on search class to get the list of available files to be selected.

5.6.8 **Print**

Identification	Print Manager
Type	Class
Purpose	To enable user to get the desired document printed.
Function	1. To enable user to select a File for printing. 2. To detect printer connected through Wifi or Wifi Direct. 3. To send document for printing.
Dependencies	1. It depends on google print cloud API for printing on internet.

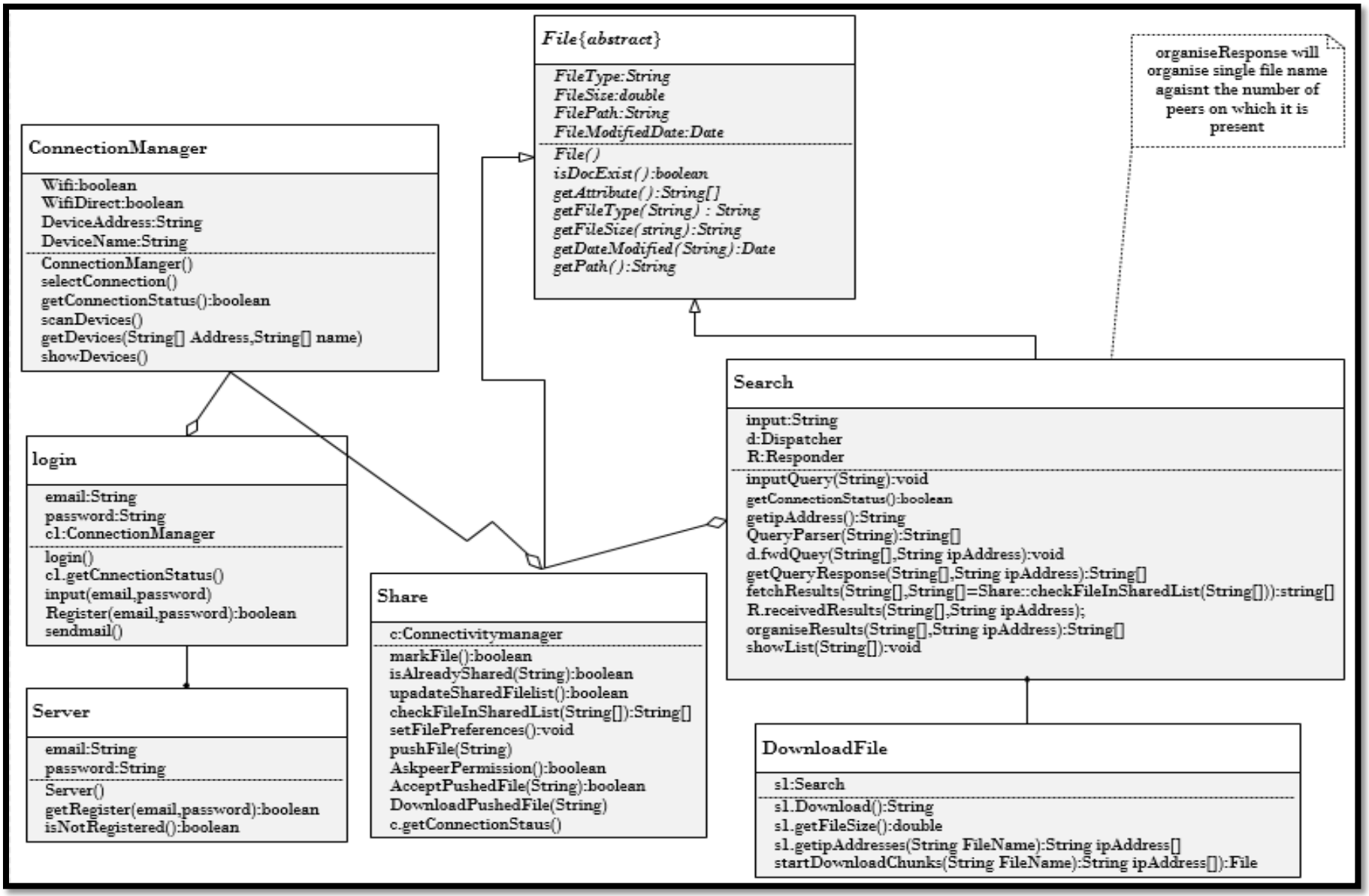


Figure 22: Class Diagram 1

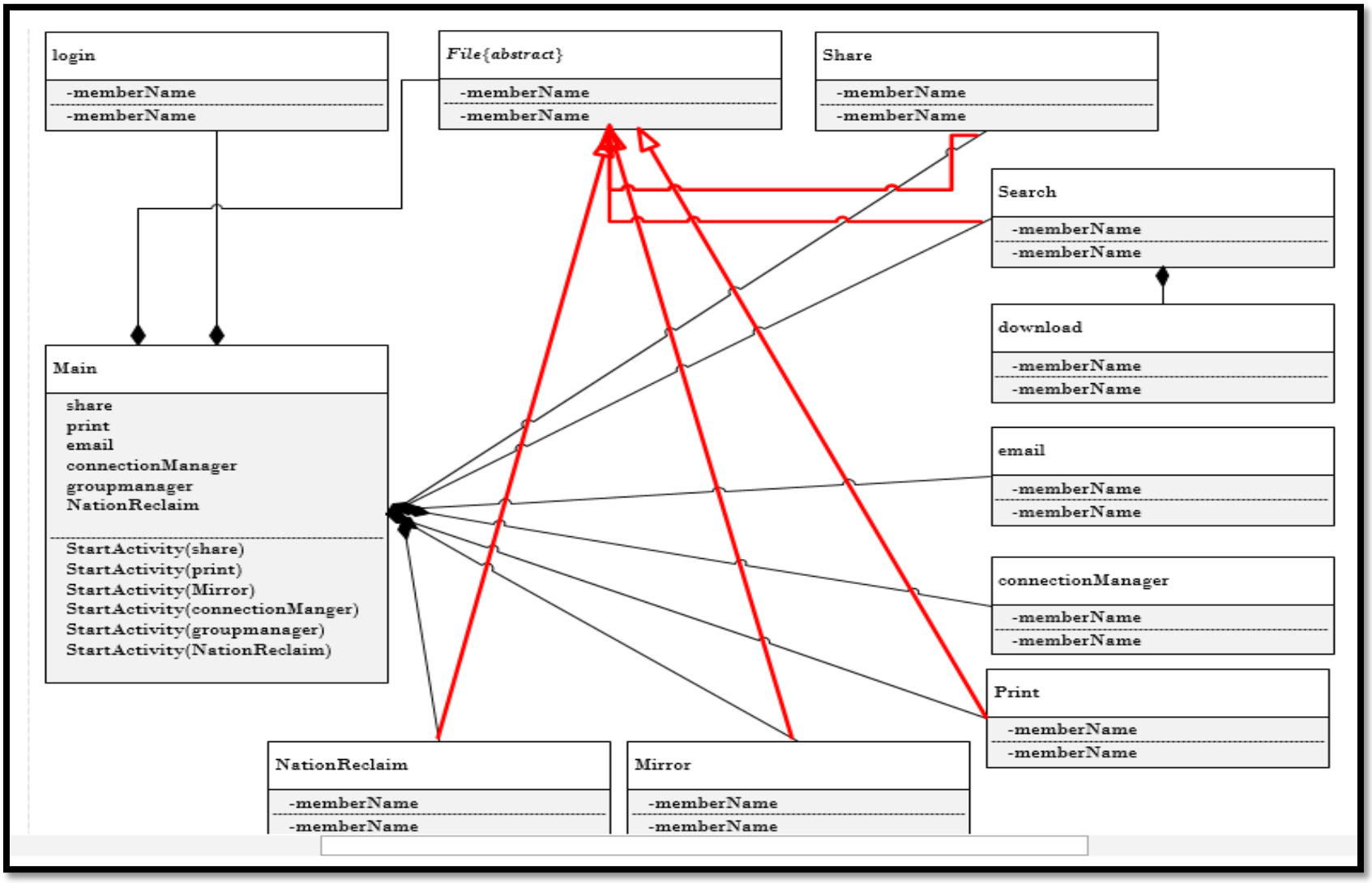


Figure 23: Class Diagram 2

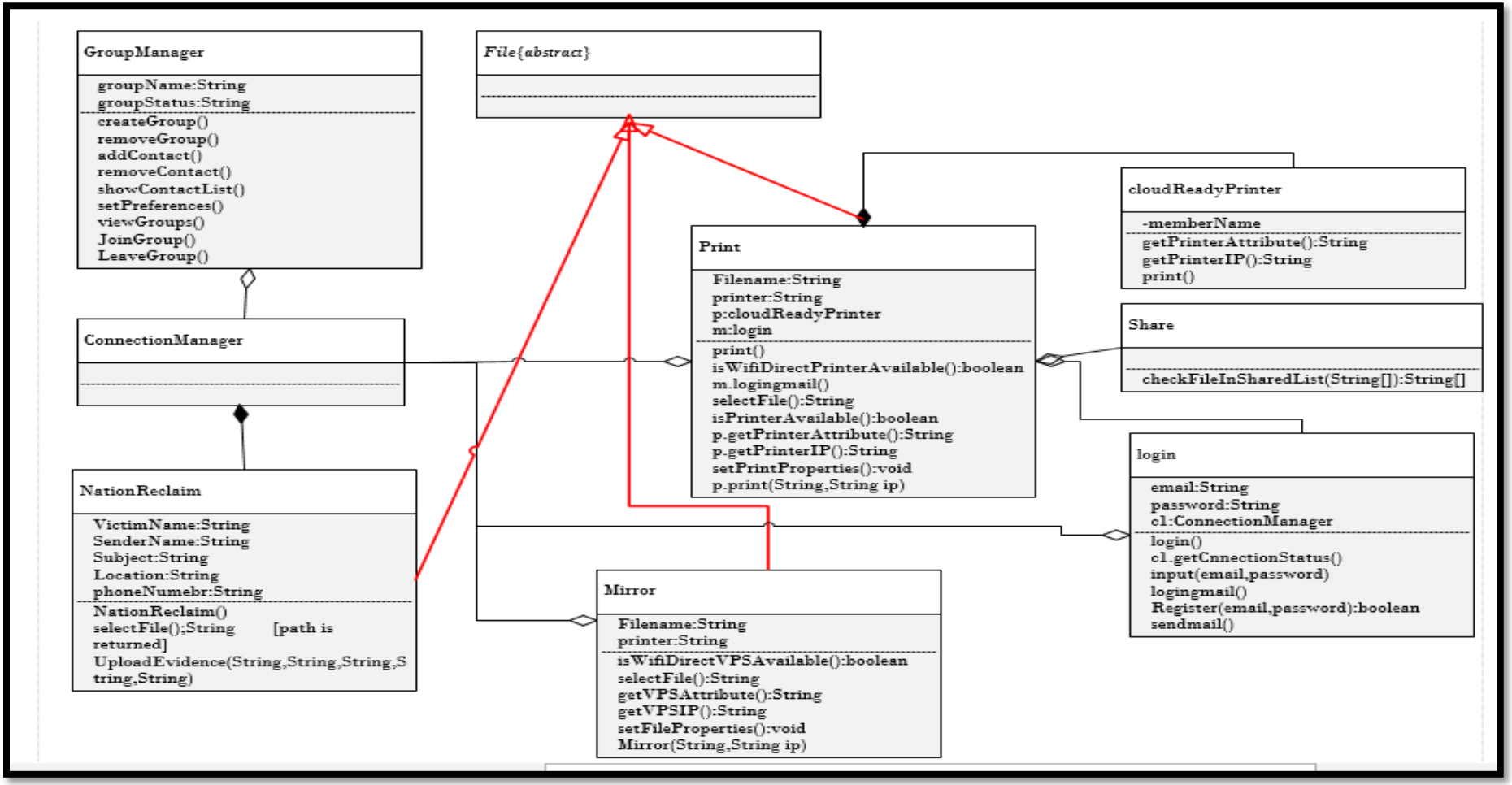


Figure 24: Class Diagram 3

5.7 Sequence Diagram

5.7.1 Login Sequence Diagram

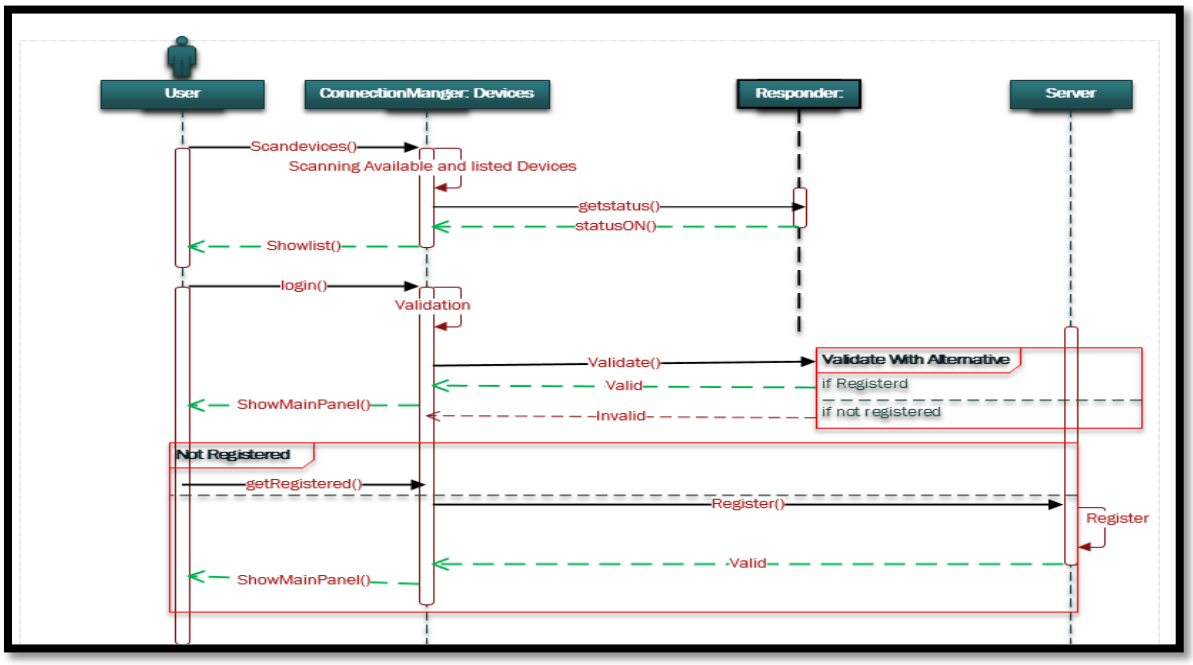


Figure 19: Login and Peers connection Sequence Diagram

5.7.2 Search Sequence Diagram

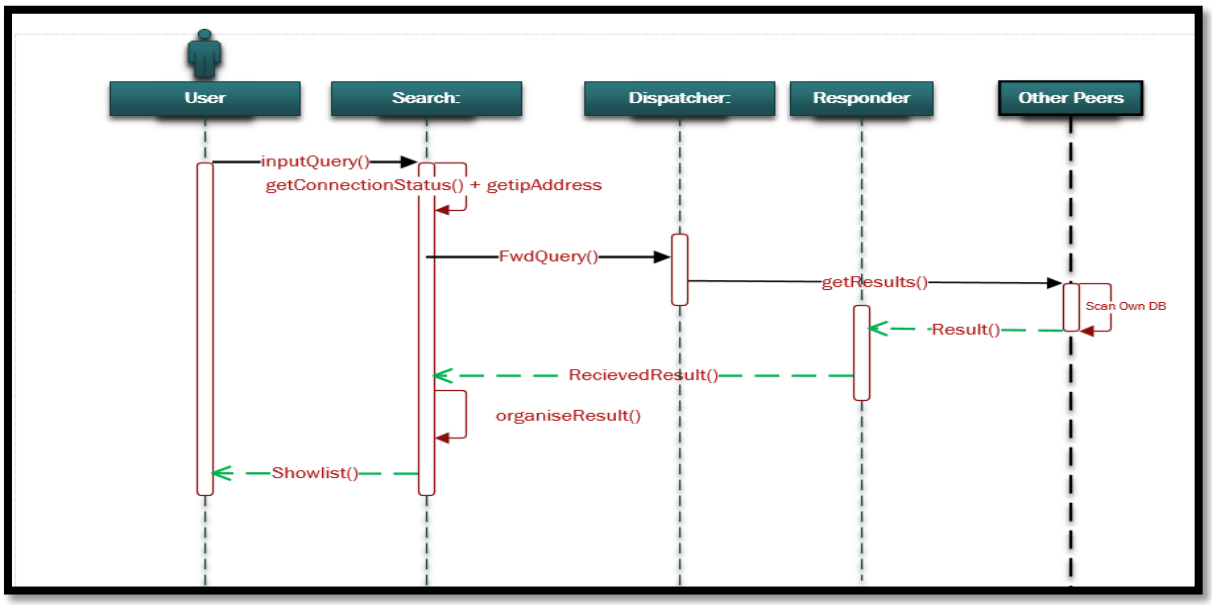


Figure 20: Search Sequence Diagram

5.7.3 **Print Sequence Diagram**

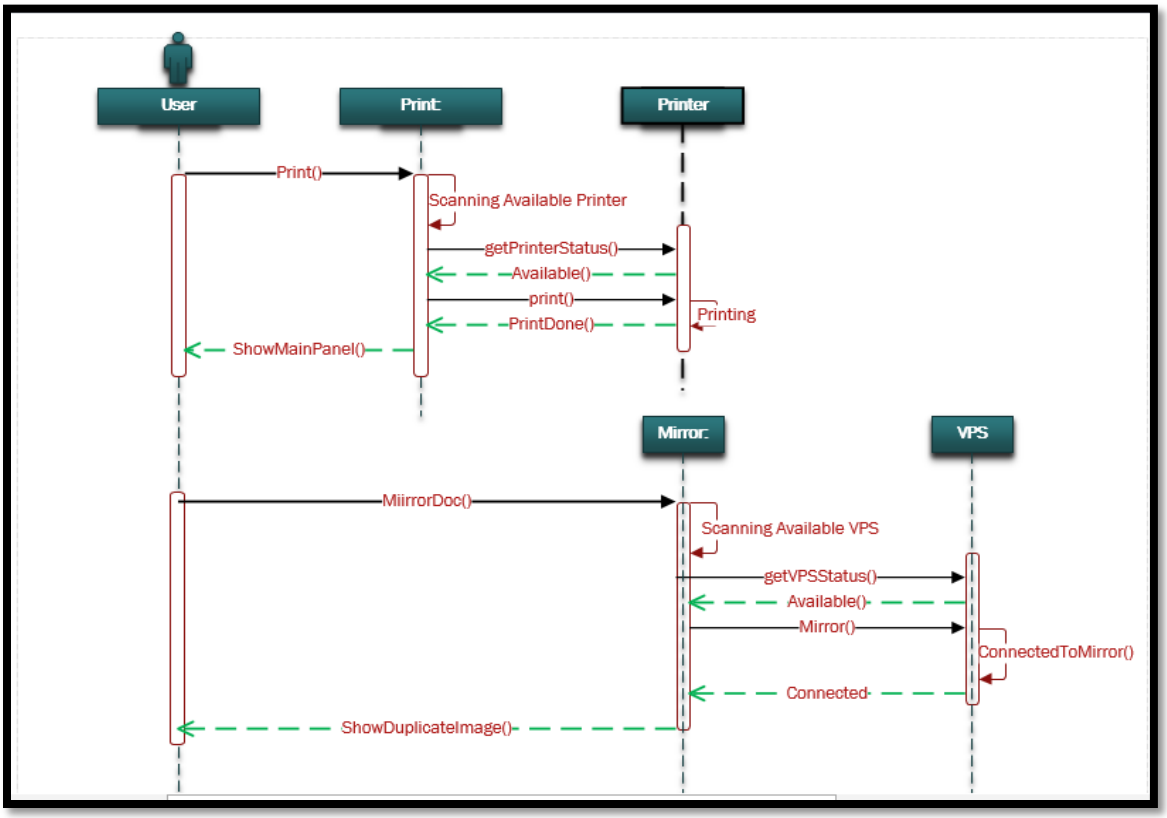


Figure 21: Print Sequence Diagram

5.8 Design Decisions and Tradeoff

5.8.1 Creation and maintaining SQLite Database will be implemented in code. It is not part of the design.

5.8.2 Different permission of android SDK are not included in design. For example WiFi_STATUS_PERMISSION, P2P_PERMISSION etc.

5.8.3 Displaying contents in user friendly user interface and its programmatic implementation is not discussed here.

5.8.4 New classes if required can be created based on the requirement. Major classes at higher level architecture will remain same.

5.9 Reuse and Relationship to other products

5.9.1 It is a standalone application not using any of existing projects.

5.9.2 This project shares some similarities with Viber, Whatsapp and Line. These apps are focused on content sharing over internet and do not cater for searching and printing document facility.

5.9.3 This app will be completely different from in respect that it will be providing user with search, print and instant sharing capabilities on both Wifi and Wifi Direct. It will also allow user to mirror screen on wifi-direct enabled devices. Moreover it will allow user to record hateful speeches images, videos along with location and details which causes imbalance to our country and report it to a centralized crime controlled server.

Chapter 6

Implementation

6.1 Pseudo Code for Components

6.1.1 Connectivity Manager

6.1.1.1 **Wifi-Direct:-**This will enable wifi direct and peers will searched continuously. This feature is used for local cloud where users will be able to use the power of wifi Direct to rapidly share file with each other. Pseudo code is attached as Appendix C

6.1.1.2 **Wifi:-** This feature will enable the user to get connected to the cloud, where user have placed their data at centralized cloud. Users that login first time will get all the thumbnails of the files uploaded onto the cloud. These files can be downloaded to local machine (mobile phone). Pseudo code is attached as Appendix C

6.1.2 **Search and download:-** This feature will enable the user to search files in the cloud and subsequently download it to personal phone. Pseudo code is attached as Appendix C

6.1.3 **Print:-** This feature will enable the user to print files on a printer attached on a cloud. Pseudo code is attached as Appendix C

6.2 Source Code

6.2.1 Major Packages

6.2.1.1 Com.icloud.download (Attached as Appendix C)

- 6.2.1.1.1 Config.java
- 6.2.1.1.2 CustomAdapter.java
- 6.2.1.1.3 FileCache.java
- 6.2.1.1.4 FileDownloadAdapter.java
- 6.2.1.1.5 ImageLoader.java
- 6.2.1.1.6 MainActivity.java
- 6.2.1.1.7 MainMenu.java

6.2.1.1.8 MemoryCache.java
6.2.1.1.9 Utils.java

6.2.1.2 **Com.icloud.upload(Attached as Appendix C)**

6.2.1.2.1 MainActivity.java
6.2.1.2.2 AndroidMultiPartEntity.java
6.2.1.2.3 CaptureActivity.java
6.2.1.2.4 Config.java
6.2.1.2.5 IncidentDetails.java
6.2.1.2.6 IncidentUpload.java
6.2.1.2.7 JSONParser.java
6.2.1.2.8 LocationGet.java
6.2.1.2.9 UploadCapturedData.java

6.2.1.3 **Com.mcs.cs.wifidirect (Attached as Appendix C)**

6.2.1.3.1 WifiDirectMainActivity.java
6.2.1.3.2 WiFiDirectBroadcastReceiver.java
6.2.1.3.3 UserInputClass.java
6.2.1.3.4 SharedPreferenceManager.java
6.2.1.3.5 serverClass.java
6.2.1.3.6 GenerateFileRequest
6.2.1.3.7 FileMetaData.java
6.2.1.3.8 DownloadButton
6.2.1.3.9 Discovery
6.2.1.3.10 DeviceActionListener
6.2.1.3.11 DeviceActionListener

6.2.1.4 **Student.mcs.cs.cloudprint (Attached as Appendix C)**

6.2.1.4.1 CloudClick.java
6.2.1.4.2 PrintDialogActivity.java

6.2.1.5 **Nust.mcs.cs.student (Attached as Appendix C)**

6.2.1.5.1 CaptureMedia.java
6.2.1.5.2 CloudFragment.java
6.2.1.5.3 FileFragment.java
6.2.1.5.4 FindPeopleFragment.java
6.2.1.5.5 HomeFragment.java
6.2.1.5.6 MainActivity.java
6.2.1.5.7 UploadImages.java
6.2.1.5.8 VideosFragment.java
6.2.1.5.9 WifiDirectFragment.java

6.2.1.6 **Nust.mcs.cs.student.adapter**

6.2.1.6.1 NavDrawerListAdapter.java

6.3 Testing

6.3.1 Introduction

To ensure quality of the product, testing is conducted. Accuracy of functions performed by MC2S has to be tested and maintained to improve quality of software.

6.3.2 TestingLevels

Separate modules are developed to provide different functionalities of MC2S. All of these modules are tested at different levels in their development and after integration. Different levels at which MC2S has been tested and results obtained are described in this section.

6.3.3 Unit Testing

Each module is developed and tested individually. Different sets of sample data are used to test all functionalities. The module for user login was developed first. This module is tested for both user login and organization login. Setting up different organizations and user's and admin the login module was tested to see if each type of system user gets has the correct responsibilities assigned and each user type should only be able to view system functionalities he has been assigned to. All the tests confirmed that data integrity and confidentiality is maintained and user(s) only see information intended for them. Another aspect is that user shouldn't be able to access any page and session handling was done at all levels. Different tests confirmed that accessing a restricted page without authentication is not possible and user is redirected to login page.

Project creation is the second module developed which has sub modules of workflow management; appoint tasks, communication system, and resource management. All these sub modules are developed and tested using relevant sample data and then tested at integration level to complete the whole project module. Regression testing is used so that at any level of project creation no wrong data is fed into the system. The expected output was that once all steps of project creation are followed, the system should show all data related to a project and shouldn't miss any relevant data.

6.3.4 **Integration Testing**

MC2S's different modules which were developed and tested independently were also tested during integration to ensure system stability. Integration testing helped in ensuring that different modules when combined give complete functionality and nothing is missed or some functionality doesn't give error when integrated with other modules. Integration testing gave us more than 90% results ensuring that most modules were integrated with other MC2S as well as compatible. This shows that errors were minimized during integration testing.

6.3.5 **System Testing**

System testing was performed at the end of development and integration of MC2S. Complete system was tested using sample data. User registration, user roles and responsibilities, creating new project and monitoring project all sub modules were tested as a whole using sample data. Almost 90% of test cases were successful ensuring that most of errors and bugs in the system were removed and system was stable enough to perform optimally.

6.3.6 **Box Approach**

To test whether MC2S functionality is in accordance with the code written box approach testing was done. The two box testing approaches used to test software were black box testing and white box testing. Black box testing is done at various stages of testing by inserting sample data in various components, checking outputs and removing errors. White box testing is done when a separate module is developed (unit level), integration of different modules and sub-modules and at system level. Once the system is developed white box testing was the most performed testing technique.

6.3.7 **Test Cases**

The system is thoroughly checked for consistency and for errors using different test cases. Overall system is checked for the different attributes and resources are displayed correctly in given scenarios, the forms are validated correctly, session handling is done correctly, user interface is easy to use and user of MC2S get features and options defined by their roles.*Test cases are attached as Appendix B*

Chapter 7

Results and Analysis

7.1 Introduction

MC2S has been developed to work in a real time environment. Since incidents can happen any second, MC2S should always be available to help organizations to track disaster MC2S and plan rescue and relief operations efficiently. The data integrity and confidentiality should be maintained at all levels.

7.2 Results

MC2S has been developed to facilitate organizations in post disaster activities or rescue or relief. The idea was to develop a web application which is at the center of disaster management activities of rescue and relief and helps organization to execute these activities efficiently and effectively. MC2S performs all the functionalities defined in chapter 3 system requirements. All the functionalities have been achieved using the most advanced tools and technologies for development of Web services.

7.3 Analysis

Since MC2S is a web application performance, robustness and usability are important features. MC2S code is optimized so that page loading time is minimum and has been tested using multiple connections to server to test its load and stress and how system will perform. The results are more than 90% accurate showing it will perform fairly well with multiple client connections.

Usability is an important aspect of web application and in MC2S from the design phase this issue was paid special attention by developing navigation model for different web pages to show how information will be displayed and how different pages will link to each other.

Another important part of analysis is how MC2S compares with existing system. One of the existing systems that exist is SAHANA FOSS Disaster Management System. The functionalities that MC2S has and SAHANA doesn't are a central database of resources for all organizations. SAHANA also was developed for rehabilitation efforts and then extend to rescue and relief activities.

7.4 **Summary**

MC2S performs all the functionalities functional and nonfunctional provided in the system requirements. All the important nonfunctional requirements that are essential for a web application to perform effectively are present in the system.

Other systems that are similar to MC2S are built on older web technologies and do not use SOA. Also the scope or the disaster management activities which those systems perform are different from MC2S

Chapter 8

Future Work and Conclusion

8.1 Future Work

Research shows that people are frequently try to search for information with other people. The fact that no user interface for collaborative searching has yet caught fire suggests that the best parts of the design space have yet to be investigated. The question is, what should the next generation of collaborative search tools be focused on? Below we discuss three use cases in which collaboration is likely to be needed in a search task.

8.1.1 Scenario 1: Selecting a few from many similar choices

Sometimes when people collaborate, it is because they need to make a decision together to meet the various group members' desires. The classic example is travel planning where people demand various things like one requires a hotel with a good food, the other wants to stay close to a beach and so on. Or some people engage in "competitive shopping" in which they split up the work when looking at many similar items offered at different venues to try to negotiate best prices.

For this case, a collaborative search tool should offer structure to let individuals define what their personal constraints or preferences are in some manner, as they search.

8.1.2 Scenario 2: Covering a topic thoroughly

A student researching his or her dissertation work or a paralegal looking through a document trove trying to find all instances of some concept exemplify classic information retrieval challenge problems. Collaborative search is especially useful for this problem. The current search tools lack certain fundamental functionalities. For one the tool should be aware both of what has already been stashed away in the bibliography and what has been viewed by anyone in the group of

searchers. It should re-rank based on this information, hiding what has already been assessed (but allowing users to override this setting).

8.1.3 **Scenario 3: Discovering unknown information**

By far the most challenging search task is that of trying to help people make a new discovery, such as solving the mystery of why honeybees are dying in North America. For a problem like this, search over known materials is only part of a multifaceted, wide ranging collaborative effort. But it is worth considering the role of collaborative search for this type of problem.

8.2 Conclusion

Mobile collaborative search is an interestingly common place, yet unsupported, experience. People regularly use their mobile devices to engage in co-located collaborative search by moving between individual searching and shared exploration of results. To support this behavior, **CozeApp** uses phone orientation to physically represent what mode of search a person is in.

By focusing on interpersonal interaction, we believe that **CozeApp** provides insight into ways to approach co-located collaborative search that are different from traditional collaborative search support and can significantly improve the co-located mobile experience. Applications like **CozeApp** is one such step toward building mobile tools that rather than remove us from our surroundings, connect us with the people around us. And further inform us or the concerned department (15, rescue 1122 etc) with the incidents that commonly take place in the society so that necessary action can be taken against the defaulters.

Appendix A: User Manual

1. **System Overview:-** Most mobile search tools are designed for individuals, many mobile searches involve people searching with others. Recent research suggests that collaborative search is particularly common on mobile devices - for example, friends headed out to eat might work together on their smart phones to find a good restaurant. One study found that 93 percent of smart phones users have engaged in co-located collaborative search with multiple phones at some point, and another study found that 65 percent of all mobile searches take place with others.

Mobile search collaborators are usually co-located and searching for the same thing at the same time, but can feel distant nonetheless. One reason for this is that mobile device use can stifle direct verbal communication. Additionally, sharing mobile device screens can be awkward, particularly for certain group sizes, demographics, and relationships between collaborators. **CozeApp** addresses these issues by taking advantage of user proximity to augment the collaborative search experience.

The aim and novelty of this document is to develop and evaluate **CozeApp**.
The system receives the instructions from the user through user interface, which consist of a log in screen, and a log out screen.

1.1 **System Features and Functionalities:-**System summary provides the general overview of the system. The summary outlines the uses of the system's software requirement, system's configuration, user access levels and system's behavior in case of any contingencies.

1.2 **System Home Screen**:- The system will show the user through the user interface and a welcome screen is shown to the user as shown and the user can click on the desired feature to enjoy the features of CozeApp . the user can click on any of the feature such as incident reporting, images, documents, share, print through cloud etc.

1.3 **User Clicks on *Share* Button from Navigation Bar**:-

User is prompted to the gallery, Where selected file is uploaded to the cloud. This file is now available to all user at centralized place.

1.4 **User clicks on *Image / Videos / Documents*from Navigation Bar:-** User view a screen showing the respective files' thumbnails (images, videos, documents). If the files are not already downloaded onto the user system. When user makes a selection from files' thumbnails, that file is downloaded to the phone memory. These files are now available offline after downloading.

1.5 **User clicks on *Cloud Print* Button from Navigation Bar:-**

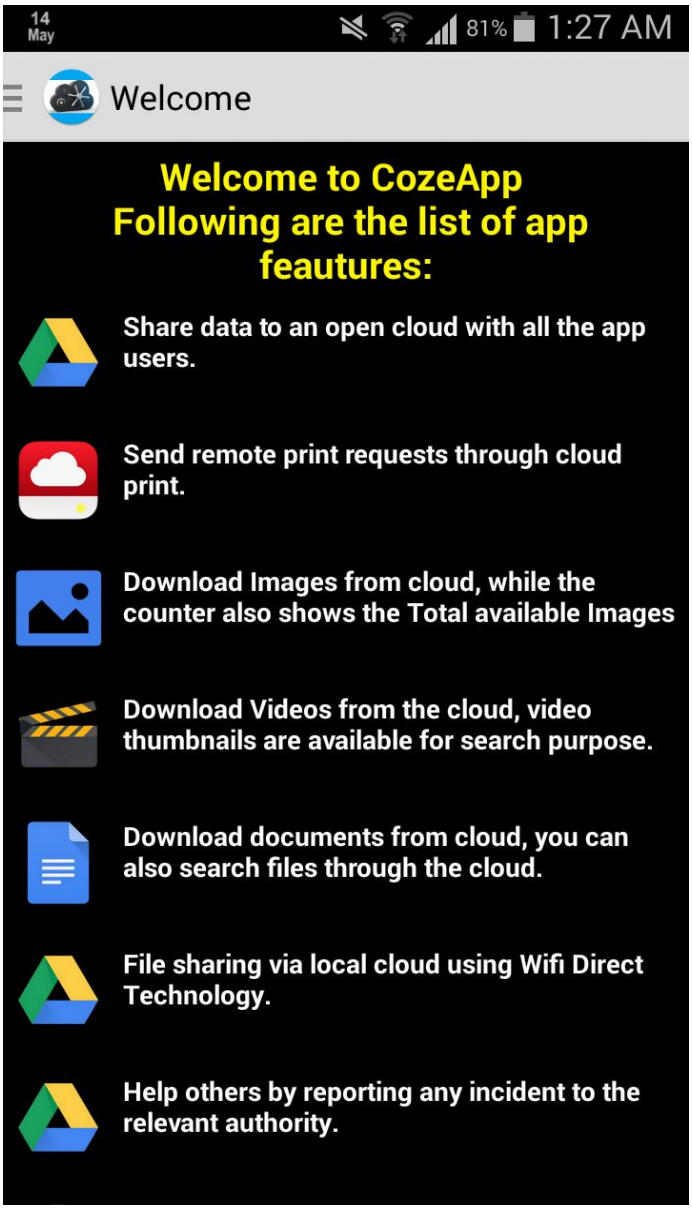
User clicks on the cloud print and files are opened to be selected to print. On selection user is taken to next screen to select a printer already added to gmail account. And the settings are presented to the user if he wants to alter. Then he can print it remotely.

1.6 **User clicks on *Local Cloud* Button from Navigation Bar:-**

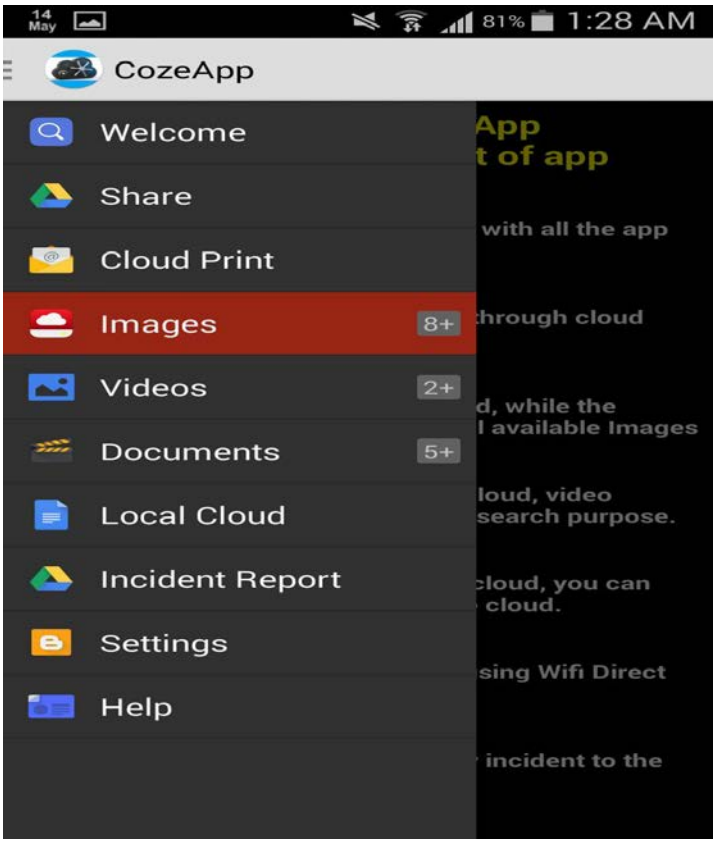
User view a screen where Discover peers can be clicked, once peers are discovered, they gets connected. And search query can be written to search files in public folders of other peers. The selected files can then be downloaded.

2. **Various Screen**

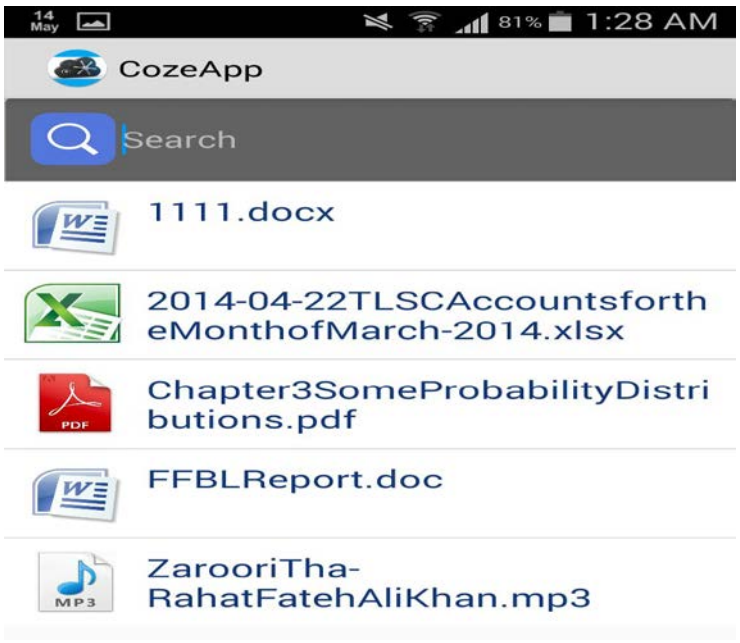
2.1 **Welcome Screen:-** Guide the new User about the features.



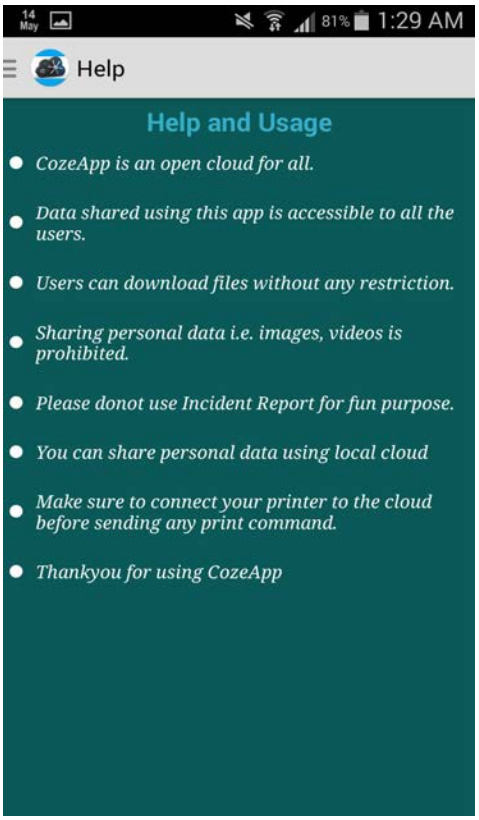
2.2 User can select any function from Navigation bar shown below.



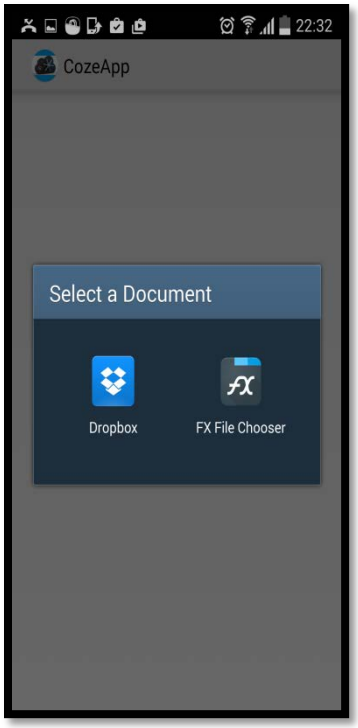
2.3 When User selects ***documents*** button from the Navigation Bar, the screen shown below is displayed.

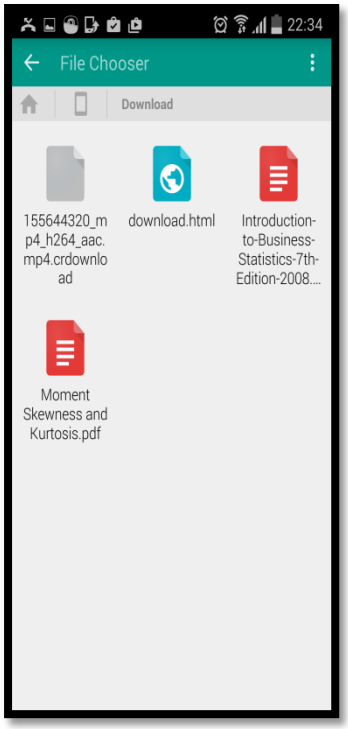


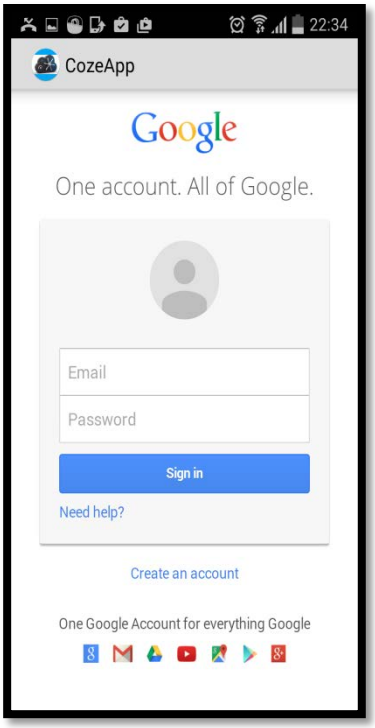
2.4 Help button shows following screen.

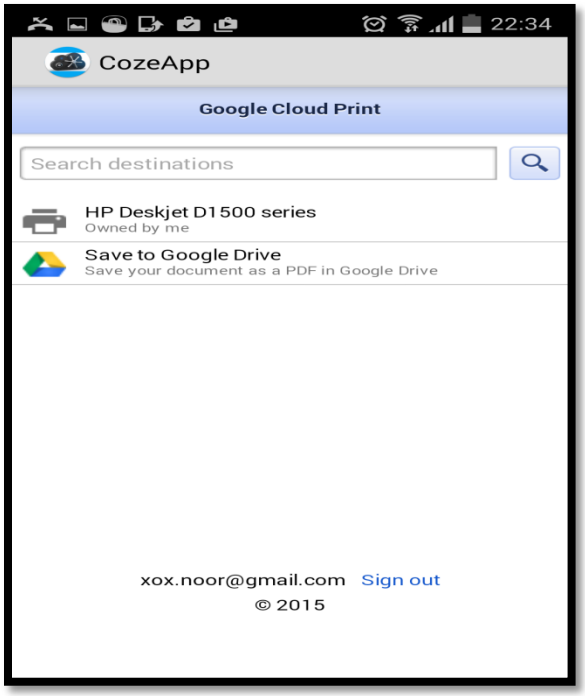


2.5 When cloud print is selected the sequence of activity is shown as following. First file explorer is selected by user, which displays all the files to user. Then user is asked to enter the email id with which the printer is associated or added. Once the credentials are entered, user is taken to next screen where he/she can select the printer(s) for printing. After this the last step is to set up printing page. Then user can click on print icon at right top of the screen.


22:32
CozeApp
Select a Document
Dropbox
FX File Chooser


22:34
File Chooser
Download
155644320_mp4_h264_aac.mp4.crdownload
download.html
Introduction-to-Business-Statistics-7th-Edition-2008....
Moment Skewness and Kurtosis.pdf


22:34
CozeApp
Google
One account. All of Google.
Email
Password
Sign in
Need help?
Create an account
One Google Account for everything Google


22:34
CozeApp
Google Cloud Print
Search destinations
HP Deskjet D1500 series
Owned by me
Save to Google Drive
Save your document as a PDF in Google Drive
xox.noor@gmail.com Sign out
© 2015


22:35
CozeApp
« Printers
Google Cloud Print
Print
Document:
Printer: HP Deskjet D1500 series
Owned by me
Collate
Colour
Colour
Copies
1
Duplex
Off
Fit to Page
Shrink to Page
Paper size
Letter
Orientation
Auto
All
Pages
e.g. 1-5, 8, 11-13
Print quality
600 dpi
Reverse order

Appendix B: Detailed Use Cases

Detailed Use Case for Share File

Name	Share File
Summary	User can share image, video or document with the accepted peer list
Actor	User
Dependency	Include Add to Share List and Push to Share abstract Use Cases
Pre-condition	Mobile cloud collaborative search system should be connected with Peers through WiFi or WiFi Direct
Flow of Events	1. User selects the file whether image, video or document to be shared 2. User adds the selected file to the add to share list 3. Include Add to Share List abstract Use Case 4. User selects the peer, with whom he wants to share, from the accepting peer list 5. Include Push to Share abstract Use Case 6. System displays a successful message
Alternative	If the system determines that the selected peer with whom user wants to share a file, is not connected through WiFi or WiFi Direct, it will display that the selected peer is not connected and a request to share to connected peer.
Post-condition	File whether image, video or document will be shared with the accepted peer list.

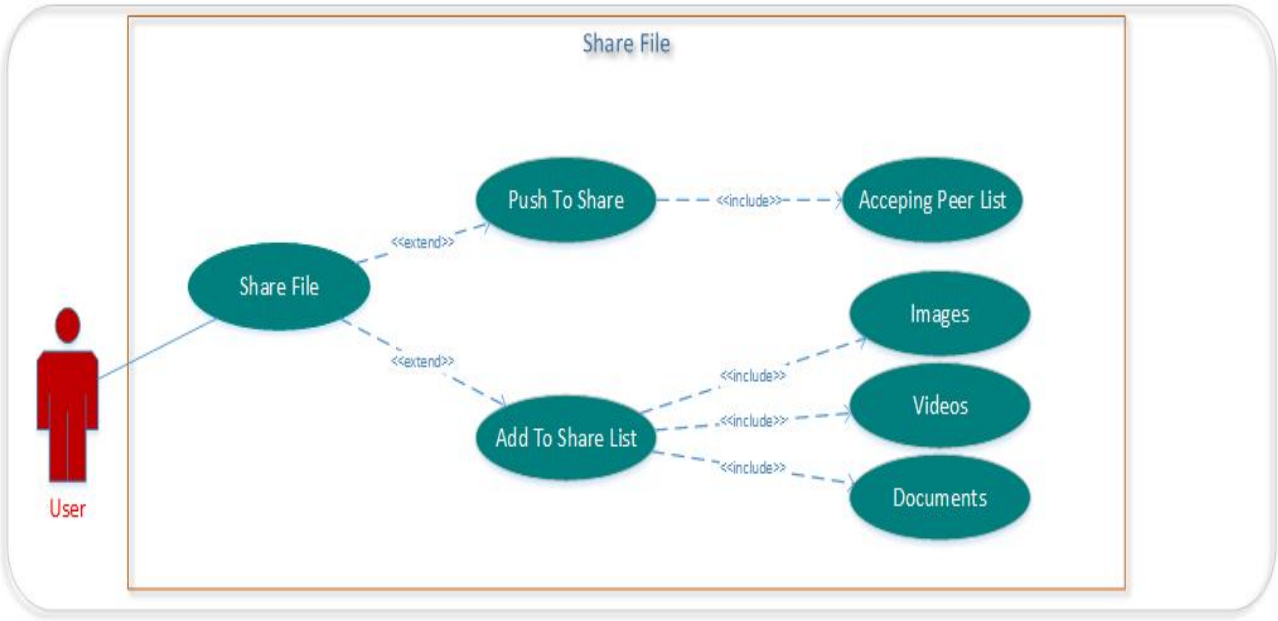


Figure 11: Share Use case Diagram

Detailed Use Case for Search File

Name	Search File
Summary	User can search a file
Actor	User
Dependency	Include List File abstract Use Case
Pre-condition	Mobile cloud collaborative search should be connected with WiFi or WiFi Direct
Flow of Events	1. Include List File abstract Use Case 2. User selects the file from the list of the files on own device 3. User selects the file from the list of the files on shared devices connected through WiFi or WiFi Direct 4. The respective file is searched and the system displays a successful message
Alternative	If the system determines that the selected File is not located on devices connected through WiFi or WiFi Direct, it will display an error message.
Post-condition	Selected File will be searched.

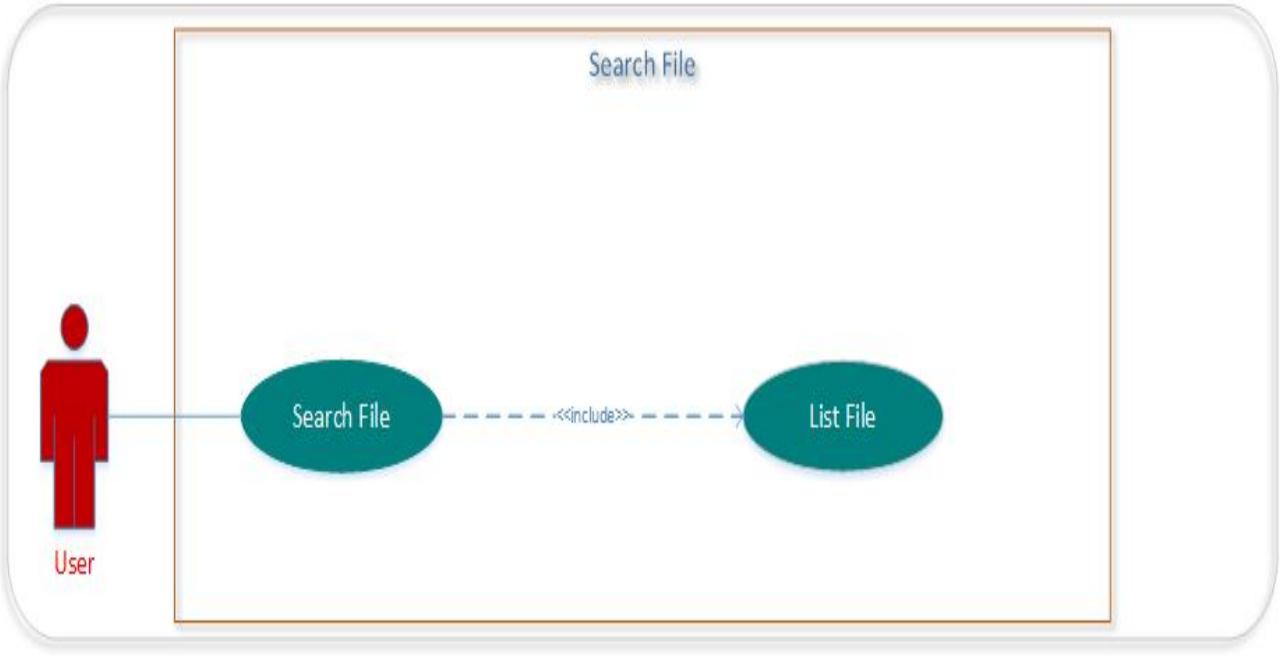


Figure 12: Search Use case Diagram

Detailed Use Case for Print File

Name	Print File
Summary	User prints the selected file through the devices available
Actor	User
Dependency	Include detect devices, connect devices, select file and print abstract use cases
Pre-condition	Mobile cloud collaborative search system should be connected with WiFi or WiFi Direct
Flow of Events	1. Include Detect devices abstract Use Case 2. Include Connect devices abstract Use Case 3. Include Select file abstract Use Case 4. Include Print abstract Use Case 5. User selects the file from the list of the files on own device or other devices connected through WiFi or WiFi Direct 6. User detects the devices available. 7. User connects the system to the printing devices available. 8. User prints the selected file and the system displays a successful message
Alternative	1. If the system determines that the selected File is not located on devices connected through WiFi or WiFi Direct, it will display an error message. 2. If the system determines that the selected device is not attached to the system, it will display an error message and prompts for re-connection to the device
Post-condition	File will be printed to the device.

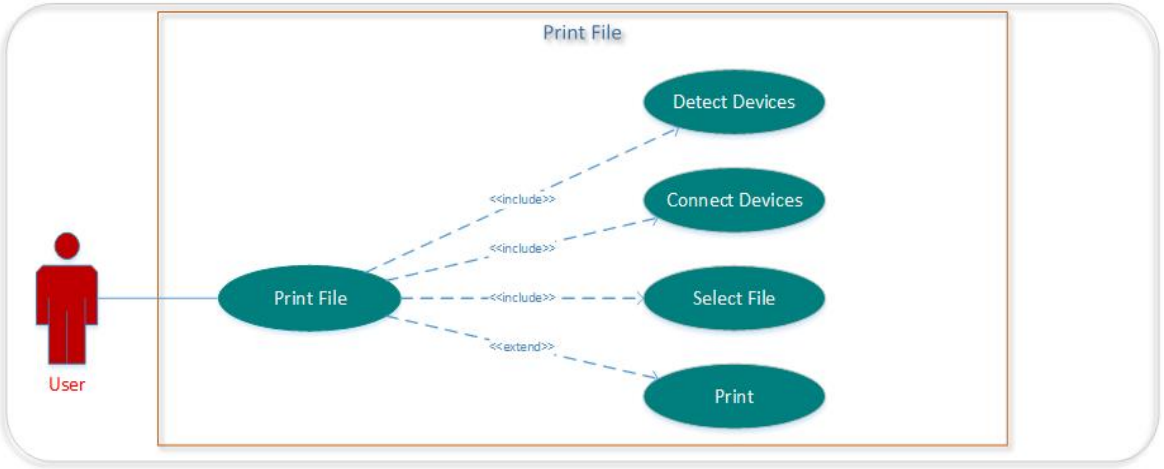


Figure 13: Print Use case Diagram

Detailed Use Case for Download File

Name	Download File
Summary	User downloads the selected file through the connected devices and saves on his device
Actor	User
Dependency	Include Search File And Save abstract Use Cases
Pre-condition	Mobile cloud collaborative search system should be connected with WiFi or WiFi Direct
Flow of Events	1. Include Search File abstract Use Case 2. User searches the selected file from the list of the files on own device 3. User searches the file from the files in the shared list on shared devices connected through WiFi or WiFi Direct 4. Include Save File abstract Use Case 5. The respective file is then saved on the system and the system displays a successful message
Alternative	If the system determines that the selected File is not located on devices connected through WiFi or WiFi Direct, it will display an error message.
Post-condition	File will be downloaded.

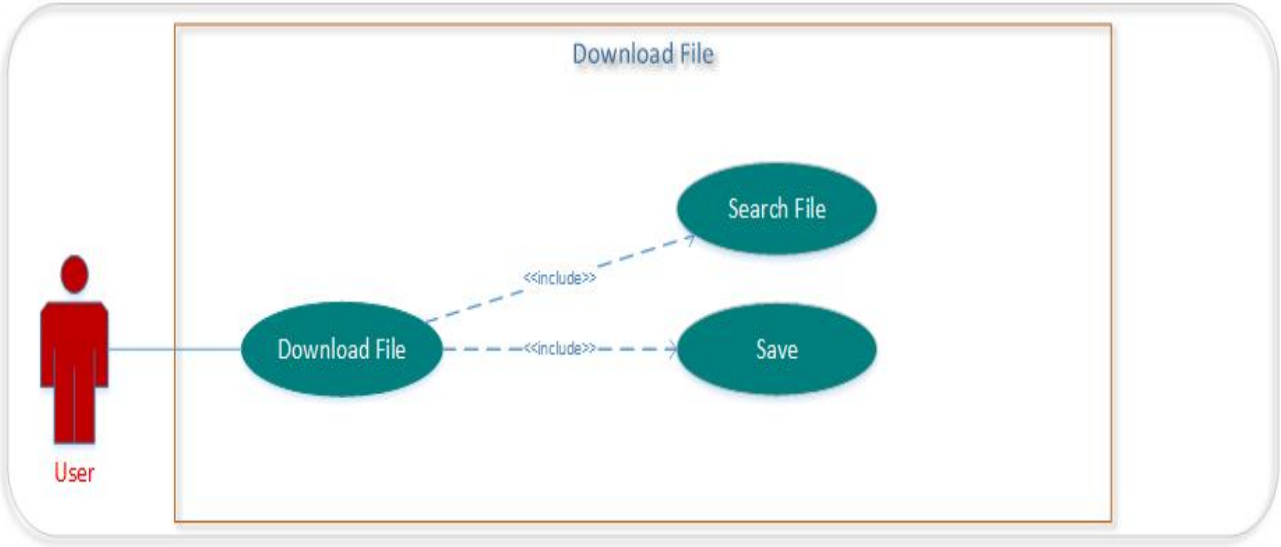


Figure 15: Download Use case Diagram

Detailed Use Case for Manage Connection

Name	Manage Connection
Summary	User manages various connections by selecting WiFi or WiFi Direct connection
Actor	User
Dependency	Extend Select Connection Use Case and Include WiFi and WiFi Direct abstract Use Cases
Pre-condition	WiFi or WiFi Direct Connection should be available
Flow of Events	1. Extend Select Connection Use Case 2. User selects a connection 3. Include WiFi abstract Use Case 4. User selects WiFi connection 5. Include WiFi Direct abstract Use Case 6. User selects WiFi Direct connection
Alternative	If the system determines that no connection exist, it will generate an error message
Post-condition	Connection will be managed.

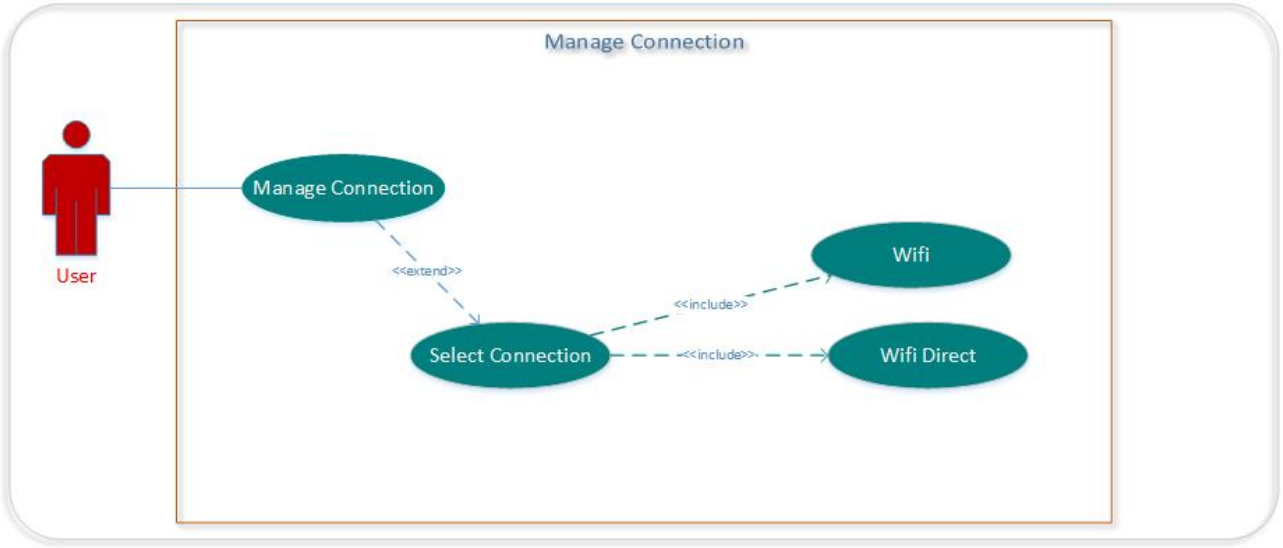


Figure 17: Manage Connection Use case Diagram

Appendix C: Pseudo Code

Wifi Direct

IF *wifi disabled*

Enable Wifi.Direct

Acquire lock.

End

While *peers not found and new pairs appears*

Begin

Search for connected peers on internet

IF*peers found*

Connect to peers.

Update each peer of newly arrived peer.

Update local database

Break

End

IF *user exit application*

Close connection.

Else*Maintain Connection*

End

End

Wifi

IF *wifi disabled*

Enable Wifi.

Acquire lock.

End

While *peers not found and new pairs appears*

Begin

Search for ***Wifi- enabled*** *peers*

IF *peers found*

Connect to peers.

Add newly discovered peers.

Update local database

End

End

IF *user exit application*

Close connection.

Else *Maintain Connection*

End

Search and download

IF *Search is selected.*

Take user to a new window

IF *query is entered*

Split and remove meaningless words from query

Search local database for the query.

IF *Result found*

Show available results.

Send query to dispatcher.

Listen to responder for response.

IF *response is not null*

Show updated list along with the peers address

End

Else

Send query to dispatcher.

Listen to responder for response.

IF *response is not null*

Show updated list along with the peers address

End

IF *any listed file is selected*

Divide the file chunks

Download chunks from other peers.

Save chunks to the database.

End

Print

***IF** print is selected*

***IF** internet is enabled.*

***IF** set properties is selected*

Open print property setting.

Send document to google cloud print.

Empty queue.

Else

Send document to google cloud print.

Empty queue.

End

Else** open wifi manager Activity**.

*Enable Wifi**.***

Return** back to the stacked activity**.

End

***Else if** document is selected*

***IF** internet is enabled.*

***IF** set properties is selected*

Open print property setting.

Send document to google cloud print.

Empty queue.

Else

Send document to google cloud print.

Empty queue.

End

Else** open wifi manager Activity**.

*Enable Wifi**.***

Return** back to the stacked activity**.

End

Appendix D: Source Code

Com.icloud.download

MainActivity.java

```
1   package com.icloud.download;
2
3   import java.io.BufferedInputStream;
43  public class MainActivity extends Activity {
44      GridView gv;
45       Context context;
46          ArrayList prgmName;
47          String fileType="";
48          EditText etSearch;
49          private ProgressDialog pDialog;
50          public static final int progress_bar_type = 0;
51          ArrayList<String> ImagesThumbURL = new ArrayList<String>();
52          ArrayList<String> ImagesNamesList = new ArrayList<String>();
53          ArrayList<String> ImagesExtentions = new ArrayList<String>();
54          ListView lv;
55          Activity activity;
56          String ActualFileName="";
57          Integer[] imageId = {
65      protected void onCreate(Bundle savedInstanceState) {
79      protected Dialog onCreateDialog(int id) {
94
95      private class getFileNames extends AsyncTask<Void, Void, String> {
97          protected String doInBackground(Void... params) {
100
102         private String uploadFile() {
172
174         protected void onPostExecute(String result) {
252     }
253
254     class DownloadFileFromURL extends AsyncTask<String, String, String> {
255
256         /**
257          * Before starting background thread
258          * Show Progress Bar Dialog
259          * */
261         protected void onPreExecute() {
265
266         /**
```

Com.icloud.upload

MainActivity.java | MainActivity.java

```
1   package com.icloud.upload;
2
3
4   import java.io.File;
24
25  public class MainActivity extends Activity implements View.OnClickListener {
26
27      // LogCat tag
28      private static final String TAG = MainActivity.class.getSimpleName();
29
30
31      // Camera activity request codes
32      private static final int CAMERA_CAPTURE_IMAGE_REQUEST_CODE = 100;
33      private static final int CAMERA_CAPTURE_VIDEO_REQUEST_CODE = 200;
34      private static final int FILE_SELECT_REQUEST_CODE = 300;
35
36      public static final int MEDIA_TYPE_IMAGE = 1;
37      public static final int MEDIA_TYPE_VIDEO = 2;
38      public static final int MEDIA_TYPE_FILE = 3;
39
40      String contentType="";
41      String ImagePath="";
42      private Uri fileUri; // file url to store image/video
43
44      private Button btnCapturePicture, btnRecordVideo, btnUploadFile;
45
47      protected void onCreate(Bundle savedInstanceState) {
65
66
67      private void captureImage() {
76
77      /**
78       * Launching camera app to record video
79       */
80      private void recordVideo() {
95
96      /**
97       * Here we store the file url as it will be null after returning from camera
```

Com.mcs.cs.wifidirect

MainActivity.java | MainActivity.java | WifiDirectMainActivity.java

```
1  package com.mcs.cs.wifidirect;
2
3  import java.io.File;
47
48
49 public class WifiDirectMainActivity extends Activity implements ChannelListener, DeviceActionListener{
50     public final int segmentSize = 512;
51      public List<FileMetaData> requestList= new ArrayList<FileMetaData>();
52      public List<FileMetaData> myFiles = new ArrayList<FileMetaData>();
53      public final int MAXREQ =3;
54      private boolean retryChannel = false;
55     final Lock listlock=new ReentrantLock();
56     final Lock filelock=new ReentrantLock();
57     final Condition listCV= listlock.newCondition();
58     final Condition fileCV= filelock.newCondition();
59     final Lock socketLock=new ReentrantLock();
60     final Condition socketCV= socketLock.newCondition();
61     final Lock eventQLock = new ReentrantLock();
62     final Condition eventQCV = eventQLock.newCondition();
63     public List<String> eventQ = new ArrayList<String>();
64     public Context myContext;
65     private boolean isWifiP2pEnabled = true;
66     public WifiP2pManager manager;
67     public String myMacAddress;
68     private final IntentFilter intentFilter = new IntentFilter();
69     public Channel channel;
70     public WiFiDirectBroadcastReceiver receiver = null;
71     public UserInputClass UI;
72     public generateFileRequest GFR ;
73     public IntentFilter eventIntent = new IntentFilter();
74     private AsyncTask<WifiDirectMainActivity, Void, Integer> dd;
75     private AsyncTask<WifiDirectMainActivity, Void, String> ss;
76     public ServerSocket server = null;
77     public Socket client = null;
78     public boolean isConnected = false;
79     public TextView tv16;
80     SharedPreferenceManager spm;
81     public Button DiscoverPeersBtn;
```

Student.mcs.cs.cloudprin

MainActivity.java | MainActivity.java | WifiDirectMainActivity.java | PrintDialogActivity.java

```
1  package noor.mcs.cs.cloudprint;
2
3  import java.io.ByteArrayOutputStream;
19
20
21 public class PrintDialogActivity extends Activity {
22   private static final String PRINT_DIALOG_URL = "https://www.google.com/cloudprint/dialog.html";
23   private static final String JS_INTERFACE = "AndroidPrintDialog";
24   private static final String CONTENT_TRANSFER_ENCODING = "base64";
25
26   private static final String ZXING_URL = "http://zxing.appspot.com";
27   private static final int ZXING_SCAN_REQUEST = 65743;
28
29
30   private static final String CLOSE_POST_MESSAGE_NAME = "cp-dialog-on-close";
31
32
33   private WebView dialogWebView;
34
35
36   Intent cloudPrintIntent;
37
39   public void onCreate(Bundle icicle) {
55
57   public void onActivityResult(int requestCode, int resultCode, Intent intent) {
62
63   final class PrintDialogJavaScriptInterface {
64     public String getType() {
67
68     public String getTitle() {
71
72     public String getContent() {
95
96     public String getEncoding() {
99
100    public void onPostMessage(String message) {
105  }
106
```

Nust.mcs.cs.student

MainActivity.java | MainActivity.java | WifiDirectMainActivity.java | PrintDialogActivity.java

```
1   package nust.mcs.cs.noor;
2
3
4⊕  import java.util.ArrayList;
34
35  public class MainActivity extends Activity {
36      private DrawerLayout mDrawerLayout;
37      SharedPreferenceManager spm;
38      private ListView mDrawerList;
39      private ActionBarDrawerToggle mDrawerToggle;
40      // nav drawer title
41      private CharSequence mDrawerTitle;
42
43      // used to store app title
44      private CharSequence mTitle;
45
46      // slide menu items
47      private String[] navMenuTitles;
48      private TypedArray navMenuIcons;
49
50      private ArrayList<NavDrawerItem> navDrawerItems;
51      private NavDrawerListAdapter adapter;
52
55⊕     protected void onCreate(Bundle savedInstanceState) {
127
128⊖    /**
129     * Slide menu item click listener
130     * */
131⊖    private class SlideMenuClickListener implements
132             ListView.OnItemClickListener {
134⊕         public void onItemClick(AdapterView<?> parent, View view, int position,
139     }
140
142⊕    public boolean onCreateOptionsMenu(Menu menu) {
146
148⊕    public boolean onOptionsItemSelected(MenuItem item) {
161
162⊖    /* *
```

Bibliography

[1] Sommerville – Software Engineering- 9th Edition 2010.

[2] Alan Shalloway, James R. Troit -Design Patterns Explained (A New Perspective on Object Oriented Design) in 2011

[3] Roger S. Pressman -Software Engineering (A practitioner's Approach) -5th Edition- 2011

[4] Tutorialspoint online:["http://www.tutorialspoint.com/uml"] accessed in:2015

[5] Androidtutorial online:["http://developer.android.com/"] accessed in: 2015