UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

WIRELESS PRODUCT-BROADCAST SYSTEM

BY

KENNEDY KIPKEMBOI CHONGWO
P56/61594/2013

SUPERVISOR: PROF ELIJAH OMWENGA

JANUARY 2015

Submitted in partial fulfilment of the requirements of Master of Science in Information Systems.

DECLARATION

This project, as presented in this report, is my original work and has not been presented for any other university award.

Sign: _____ Date: _____

Kennedy Kipkemboi Chongwo
P56/61594/2013

This project has been submitted as partial fulfillment of the requirements for the Master of Science degree in Information Systems of the University of Nairobi with my approval as the University supervisor.

Sign: _____ Date: _____

Prof. Elijah Omwenga.
Project supervisor,
School of Computing and Informatics,
University of Nairobi.

ABSTRACT

Many businesses lack advertising capacity for their products mostly because of high advertisement market rates offered by the industry players. For those who can afford newspaper and electronic media advertisement, they would also want to have day to day contact with customers. This research project explores a location based advertisement medium that can reach clients on their mobile phones within a radius of between 100 meters to 150 meters, the system will rely on Wi-Fi Direct, a peer-to-peer wireless standard that allows devices like some mobile phones (such as iPhones, Android phones) to communicate directly with each other rather than through a shared access point like a cell phone tower. Once a client's phone establishes connection with a server device hosting the adverts within the business premises, then it can load the adverts on demand. Wi-Fi direct works more like blue tooth but covers a much wider range of approximately between 100 meters to 150.

The above contribution was demonstrated by a prototype android application utilizing Wi-Fi direct to drill through sample advertisement data hosted on an android phone (Jelly Bean) acting as the server device.

ACKNOWLEDGEMENTS

During the process of this research, I was guided and aided by many professionals directly and indirectly through feedback, support and motivation. I would like to acknowledge and extend my gratitude to the following: my project supervisor and advisor; Prof. Elijah Omwenga for his guidance through all stages of this research project, the submission panel comprising Prof Peter Wagacha, Evans Miriti and Samuel Ruhiu for their valuable feedback on the structure and composition of this research work.

Table of Contents

List of Figures

List of Tables

1 CHAPTER 1 INTRODUCTION

1.1 Introduction

Many businesses lack advertising capacity for their products mostly because of high advertisement market rates offered by the industry players. For those who can afford newspaper and electronic media advertisement, they would also want to have day to day contact with customers.

By the time of this project newspaper rates for quarter page advertisement was KSH 150,000.00/= and prime time television rates for 30 seconds advert range between KSH 500,000.00/= to KSH 800,000.00 per week. These rates are on the higher side for small and medium size businesses and therefore the solution being proposed herein will be beneficial as it is affordable. For many years, advertising has focused primarily on traditional print advertising, along with television and radio media. But the Internet and social media have dramatically changed the role that print and television advertising plays whilst relying on paid internet connection (Chris McTiernan, 2009), According to Chris McTiernan, Some of the common advertising challenges include the following

1. **Planning marketing and advertising campaigns.** Limited access to information and budgets makes planning effective advertising programs a significant challenge, especially for smaller organizations. Trial and error not being an option, attractive alternatives are such lower-cost strategies as keyword placement and email marketing.

2. **Tracking postclick activity.** The tracking of customer activity beyond the initial acquisition event proves daunting without the data management infrastructure to do so. Many companies have chosen to outsource tracking user events (such as a purchase) to organizations such as DoubleClick. Although this certainly is a viable solution, concerns over sharing customer information with external vendors should be weighed in making such a decision.

3. **Structuring measurable agreements.** The lack of true accountability for post-click activity as it pertains to cost-per-click, -acquisition, or -conversion models is an issue even for the more established players. While commercial solutions exist to provide verifiable data, the contractual issues associated with payment are more difficult to address.

4. **Selecting the appropriate vendor.** Given the wealth of providers of everything from software solutions to email lists, determining the criteria for evaluating potential vendors is a challenge many mentioned.

The wireless instant product broadcast (WPB) system relies on Wi-Fi Direct, a peer-to-peer wireless standard that allows devices like some mobile phones (such as iPhones, Android phones) to communicate directly with each other rather than through a shared access point like a cell phone tower.

With WPB, users of Wi-Fi direct enable phones are able to contact another Wi-Fi direct enabled device resident within the business premises which then communicates back the company products on offer. This is a multi-user system that allows many users to connect at the same time.

The technology behind Wi-Fi Direct builds on traditional Wi-Fi strengths like performance, security, ease of use and ubiquity, and adds features that optimize it for consumer uses that don't require access to an infrastructure network. Rather than connecting first to an infrastructure network and then connecting to another networked device, users can connect directly to those devices offering the services they need. This allows, for example, a mobile device such as a cellular phone to connect directly to a laptop, regardless of whether an infrastructure network is available to the user (Wi Fi Alliance, 2010).

According to Wi Fi Alliance, roughly 10% of all people in the world use Wi-Fi to stay connected. There are more than 1 billion devices in use today, with an estimated 580 million units shipped in 2009. Shipments for 2011 are forecasted at 1 billion, growing to 1.5 billion by 2013.2 Wi-Fi is widely available in homes, Wi-Fi hotspots, and enterprise environments, and is found in many types of devices, including notebook computers, cameras, media players, photo frames, TVs, gaming devices, and mobile phones, among others. At the time of this paper's publication, the

Wi-Fi Alliance has more than 350 member companies and has completed more than 8,000 product certifications since March 2000. (Wi Fi Alliance, 2010). This implies therefore that any product utilizing Wi-Fi direct will be able to reach at most 10% of the world's population and hence worth pursuing.

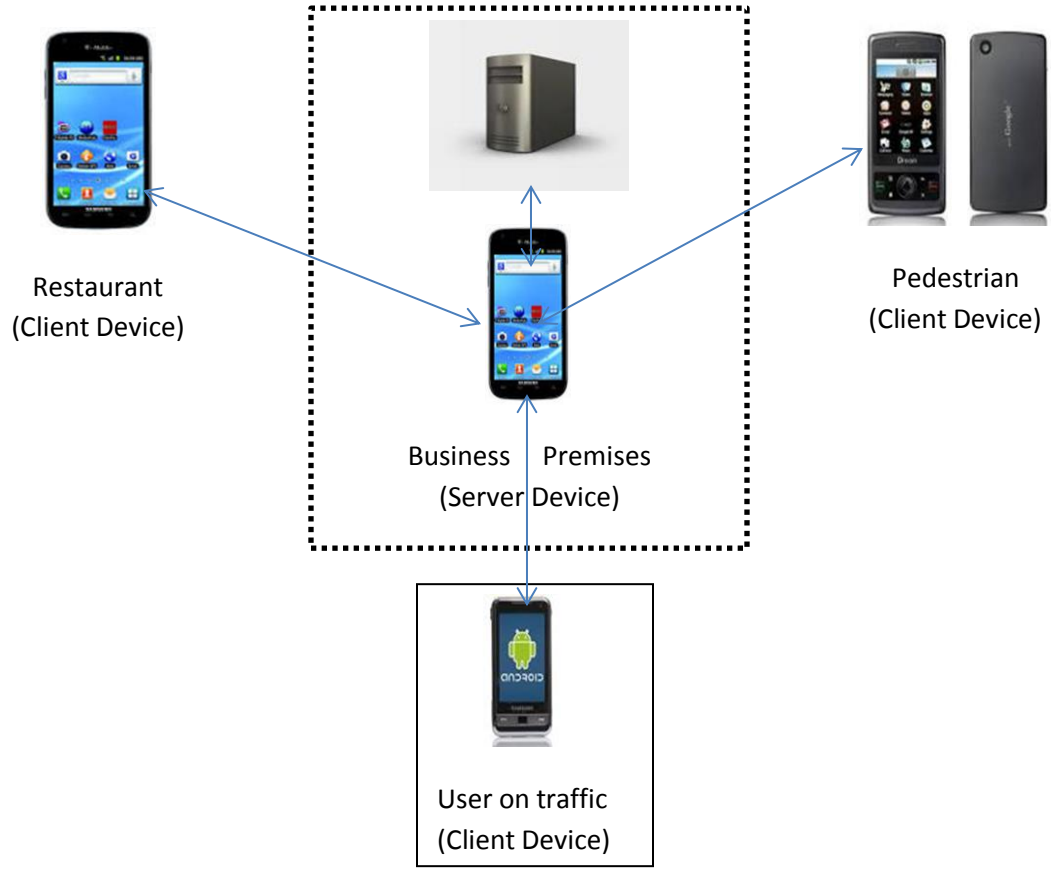


Figure 1.1 Conceptual Design

1.2 Problem statement

Businesses rely heavily on advertising to get their products and services known to consumers, however many lack advertising capacity for their products mostly because of high advertisement market rates offered by the industry players. For those who can afford newspaper and electronic media advertisement, they would also want to have day to day contact with customers. For consumers to come in contact with adverts, they either buy a newspaper or watch television or be connected to internet so as to view online adverts, this project however intends to develop a solution in which a consumers can access a company's adverts without having to buy a newspaper or be connected to the internet. Every interested consumer within reach of the android server device is able to activate his/her phone and drill through the company's products that have been made available at the server device residing in the company's premises.

1.3 Objectives

The general objectives of this project is to design and develop an affordable Wi-Fi direct application that will enable businesses to advertise their products as well as analyze Wi-Fi direct throughput in area with tall and wide buildings as well as bushy environments.

The specific objectives to be met by this project undertaking are.

1. To design and develop an affordable and easy to use wireless advertisement system that can reach phone users in the streets, offices, restaurants as well as travelers
2. To design and develop a client phone application that is able to wirelessly query data from a broadcast advertisement server using Wi-Fi direct
3. To study and analyze the performance of android Wi-Fi direct with distance (between the interacting devices).

1.4 Justification

The success of this project will provide an affordable alternative means by which companies and businesses can advertise their products and services either by way of follow up after a major print or TV advert or permanent site advertisement platform.

1.5 Significance

Once adopted, the system will significantly cut advertisement costs a business uses and will help businesses to broadcast their products to consumers without the need of internet, all a company needs to do is to designate an android phone (>= 2.0 GHZ, processing speed and >=2GB RAM) as their server and feed it with their product information which consumers can connect and view. Apart from the constant product advertisement, a company can use the system to display active product offers and discounted goods.

1.6 System Security

Since the server device is not connected to the host company's live business data, security for sensitive data will not be compromised besides the server device residing within the company's premises only broadcasts data locally loaded within SQL Lite resident on the sever device.

2 CHAPTER 2 LITERATURE REVIEW

2.1 Background

Societies are rapidly evolving towards information age where the creation, distribution, use, integration and manipulation of information is a significant economic, political, and cultural activity. The aim of the information society is to gain competitive advantage internationally, through using information technology (IT) in a creative and productive way (Wikipedia, 2009). With current trend every organization seeks to make available information on their products in every media (Television, Radio, Social media and print) so as to reach as many audience as possible, the downside of the traditional advertising (Television, Radio and Print) is that they are unaffordable to some businesses and much as social media advertising is affordable, not most audience are connected. This proposed project therefor intends to develop a system that wirelessly broadcasts the products of a business to be viewed by Wi-Fi users within proximity free of charge.

Wi-Fi Direct is a game-changing technology enabling Wi-Fi devices to connect directly, making it simple and convenient to do things like print, share, sync and display (Chris McTiernan, 2009). Products bearing the Wi-Fi CERTIFIED Wi-Fi Direct designation can connect to one another without joining a traditional home, office or hotspot network. Mobile phones, cameras, printers, PCs, and gaming devices will be able to connect directly to each other to transfer content and share applications quickly and easily. Wi-Fi Direct devices will be able to make a one-to-one connection, or several devices can connect simultaneously (Wi Fi Alliance, 2010).

Connecting Wi-Fi Direct-certified devices is easy and simple, in many cases only requiring the push of a button. Wi-Fi Direct products are based upon an industry-wide technology defined in the Wi-Fi Alliance by computing, consumer electronics, and handset vendors from around the world. Wi-Fi Direct-certified devices have been tested and certified for this technology as well as interoperability and WPA2™ security, the current generation of security technology (Wi-Fi Alliance, 2010).

The technology behind Wi-Fi Direct builds on traditional Wi-Fi strengths like performance, security, ease of use and ubiquity, and adds features that optimize it for consumer uses that don't require access to an infrastructure network. Rather than connecting first to an infrastructure network and then connecting to another networked device, users can connect directly to those devices offering the services they need. This allows, for example, a mobile device such as a cellular phone to connect directly to a laptop, regardless of whether an infrastructure network is available to the user (Wi Fi Alliance, 2010).

Many consumers associate Wi-Fi with Internet connectivity. Devices certified for Wi-Fi Direct extend the technology's reach to include the simple, direct connections that many users may accomplish using cables today. Some of the benefits consumers will see from Wi-Fi Direct devices include the following

1. **Mobility & Portability:** Wi-Fi Direct-certified devices connect anytime, anywhere.
2. **Immediate Utility:** Users have the ability to create direct connections with the very first Wi-Fi Direct-certified device they bring home. For example, a new laptop certified for Wi-Fi Direct can create direct connections with the existing legacy Wi-Fi devices in the user's home.
3. **Ease of Use:** Wi-Fi Direct devices have features that allow users to identify available devices and services before establishing a connection.
4. **Simple Secure Connections:** Wi-Fi Protected Setup™ makes it simple to create security-protected connections between devices. Users in most cases will be able to connect at the push of a button.

Wi-Fi Direct device connections can happen anywhere, anytime - even when you don't have access to a Wi-Fi network. Your Wi-Fi Direct-certified device will signal to other devices in the area that it can make a connection. You can view available devices and ask them to connect. In some cases, you might receive an invitation to connect to another Wi-Fi Direct certified device. You can then decide whether to accept the invitation or not. In many devices the only action required is to push a connection button on the device, or accept a pop-up window request. Just like with traditional Wi-Fi networks, you should only connect with trusted devices or people.

Once a connection is confirmed using Wi-Fi Protected Setup, the seamless and easy to use protocol for Wi-Fi connections, the devices make a secure connection to help protect your communication. This secure connection is completed by industry-standard WPA2 security, the very latest and best form of Wi-Fi security.

Once two or more Wi-Fi Direct-certified devices connect directly, they have formed a Wi-Fi Direct Group. Now you can get started doing all the exciting things that Wi-Fi Direct enables!

Wi-Fi Direct devices can connect to each other on a one-to-one basis, or as a group. A one-to-one connection will be very common for Wi-Fi Direct devices. You may use one-to-one connections to exchange content or share applications with a friend or co-worker. You may often connect your camera to a printer, your phone to your PC to sync calendars, or a gaming device to another gaming device in order to compete real-time. You may also link your phone to a display device such as a TV in order to view downloaded videos (Wi Fi Alliance, 2010).

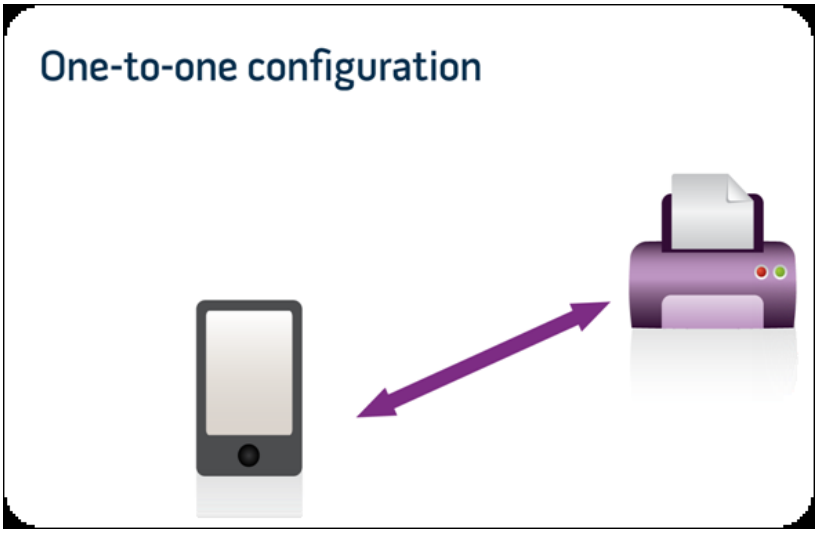


Figure 2.1 One-to-one Configuration (Taken from Wi-Fi Alliance)

Many Wi-Fi Direct devices can also connect with multiple devices at one time. This group connection is known as one-to-many because one device will act as the gate keeper to invite other devices and determine whether devices requesting to join the group will be allowed. Only devices that receive permission from the gatekeeper device are allowed to connect to other devices in the group.

A one-to-many connection allows many devices to connect together and replaces numerous cables between the devices. A one-to-many connection in your home may include PCs, printers, speakers, TVs, monitors and cameras.

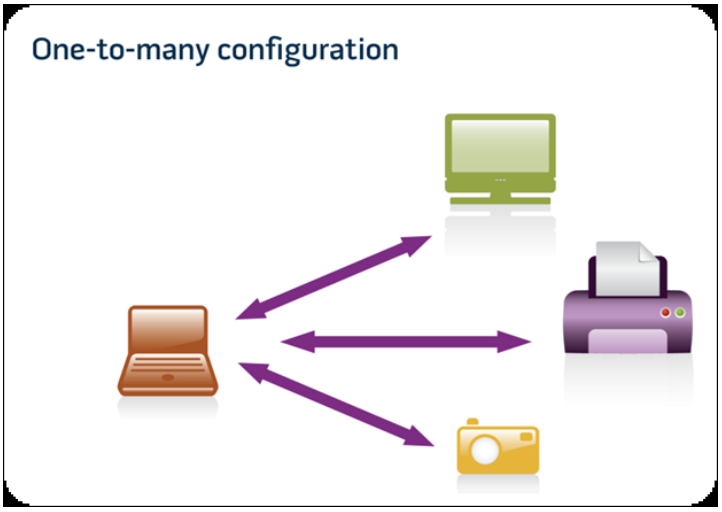


Figure 2.2 one-to-many configuration (Taken from Wi-Fi Alliance)

Wi-Fi Direct-certified devices are also able to make direct connections to most of your older Wi-Fi CERTIFIED equipment. In this way, Wi-Fi Direct brings direct connection capabilities to the hundreds of millions of legacy Wi-Fi CERTIFIED devices already in our homes and workplaces. As long as one device in a connection is Wi-Fi Direct-certified, you can connect all devices without a Wi-Fi home network or hotspot. You'll never be without a network again, as the network can travel with you – wherever you go, whenever you need it (Wi Fi Alliance, 2010).

2.2 Related work

In the year 2012, General Motors started working on a way to use Wi-Fi Direct to alert drivers to the presence of pedestrians and cyclists so as to avoid collision. The technology relies on Wi-Fi Direct, a peer-to-peer wireless communication system that lets devices connect directly with each other rather than through a shared access point like a cell phone tower (Mike Milikin, (2012). This makes the transfer of data much quicker, a crucial factor in a potential accident situation where seconds count. This wireless pedestrian detection system is a part of GM's development efforts for vehicle-to-infrastructure and vehicle-to-vehicle communication

systems for advanced warnings about hazards and other danger situations on the road (Mike Milikin, (2012).

Using Wi-Fi Direct, devices are able to connect in roughly one second compared to needing 6 to 8 seconds to acquire location information when connecting to a network as opposed to forcing the car to connect to a mobile network first which will take longer time.

GM researchers have determined Wi-Fi Direct can be integrated with other sensor-based object detection and driver alert systems already available on production vehicles to help detect pedestrians and bicyclists carrying smartphones also equipped with Wi-Fi Direct.

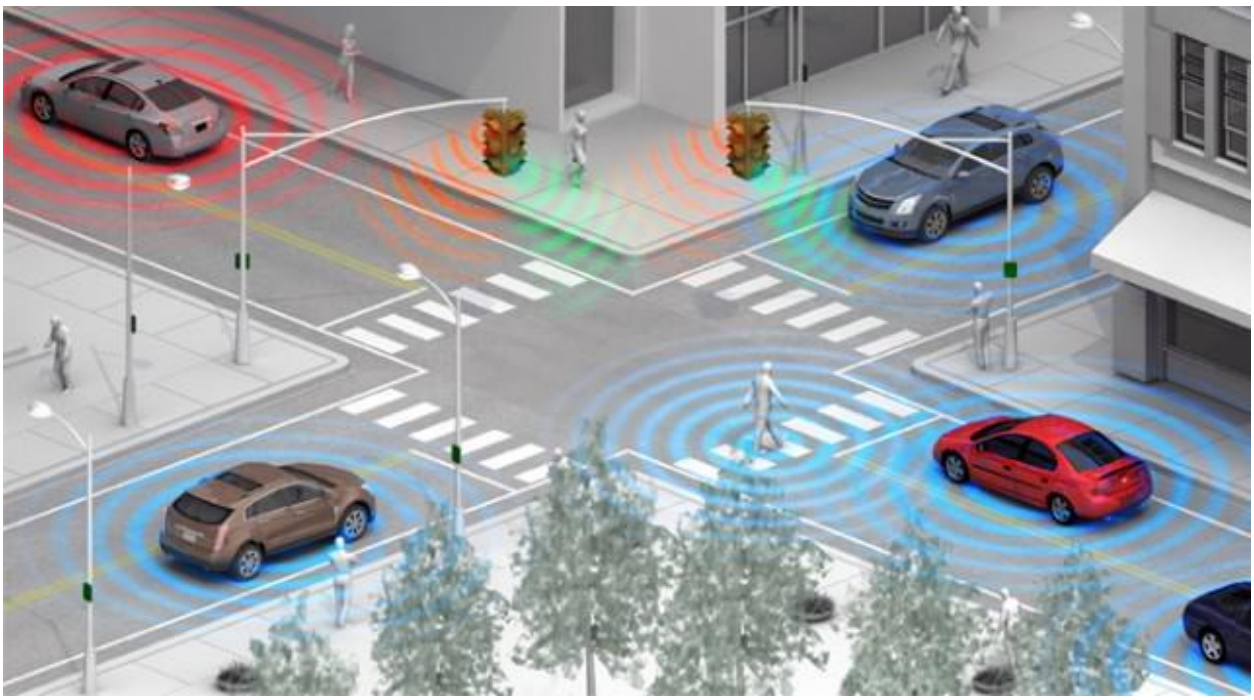


Figure 2.3 GM pedestrian detection system

GM is looking to develop a complementary app for Wi-Fi Direct-capable smartphones that can be downloaded by frequent road users such as “bike messenger” or “construction worker” that will help Wi-Fi Direct-equipped vehicles identify them (Mike Milikin, 2012).

There are many android applications hosted in google play (android applications repository) that use Wi-Fi direct to share files (photos & documents) between Wi-Fi direct enabled devices, these include

1. SuperBeam
2. Wi-Fi Shoot
3. WiFiShare

These applications take advantage of Wi-Fi direct capability of directly connecting devices without requiring an access point.

The capabilities that enable Wi Fi direct to be used to transfer files between android devices as well as a pedestrian detection system by General Motors, can be harnessed to facilitate advertisement of products and services, this is the focus of this project and the result is a server and client applications that are able to exchange the data being broadcast. The project will address the advertisement gaps discussed in section 1.0 as well as determine the performance of android Wi Fi direct as a measure of distance versus repose.

3 CHAPTER 3 METHODOLOGY

3.1 Introduction

Every android application runs in its own Linux process. Android starts the process when any of the application's components need to be executed, then shuts down the process when it's no longer needed or when the system must recover memory for other applications (Android developers, 2013).

Wi-Fi Direct is a platform that allows Wi-Fi devices to connect to each other with no need for a wireless access point. In order to be able to broadcast product details as this project aims, a server application running on an android device designated as a server was developed so as to accept connections from multiple clients hosting the client application responsible for initiating active socket communication with the server device as depicted by the image below.

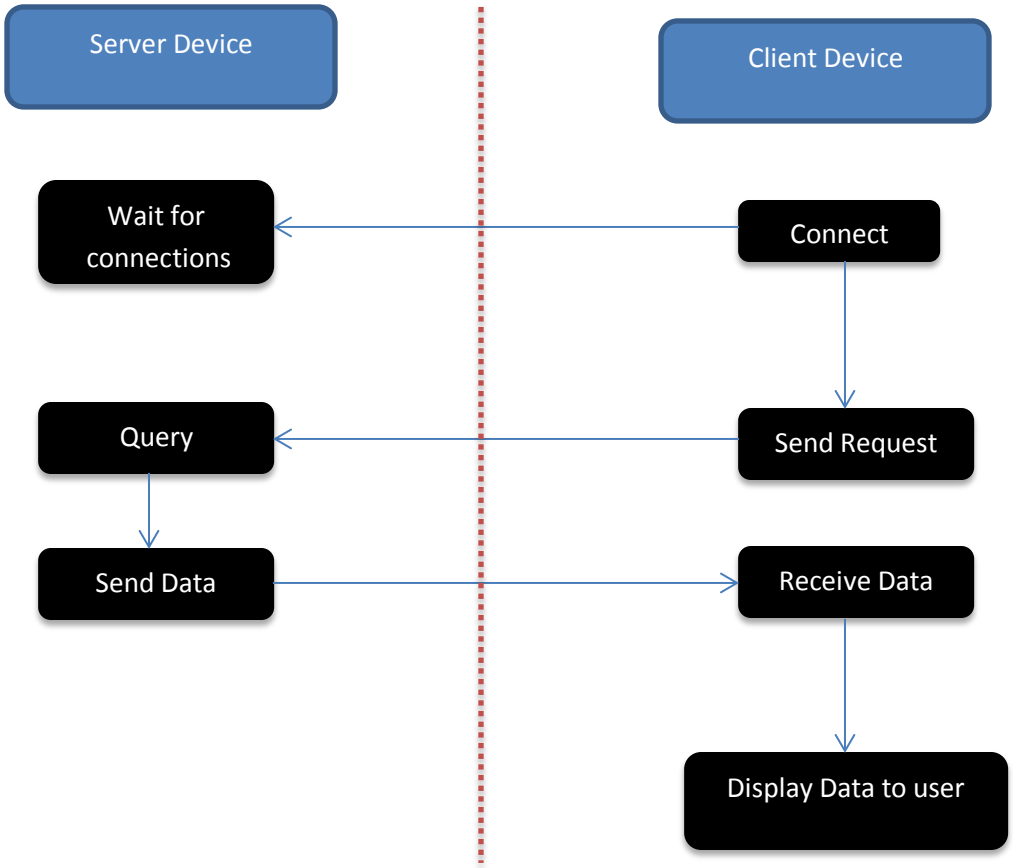


Figure 3.1 Client-Server interaction

For this project I adopted one-to-many connection which allows many devices to connect to a single server multi-threaded to serve multiple connections at the same time.

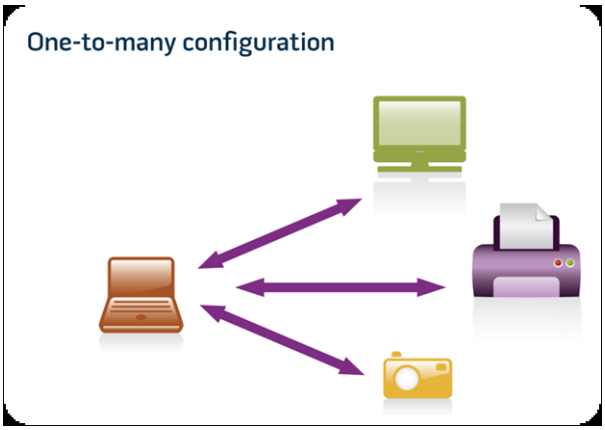


Figure 3.2 Wi-Fi direct one-to-many configuration (Taken from Wi-Fi Alliance)

For Wi-Fi Direct, once the connection is set up between two devices, we can get an IP address, and then use this IP address to create server socket as well as client socket that will facilitate communication between the client device and the server device.

3.2 Research Design

3.2.1 Conceptual Design

This project required the design and development of two applications, one that was installed on the server device and the other on the client device.

Server device application is responsible for broadcasting a company's products while on the other hand the client device application is responsible for querying the server device and display content to a user (phone's owner) as depicted by the context diagram below.

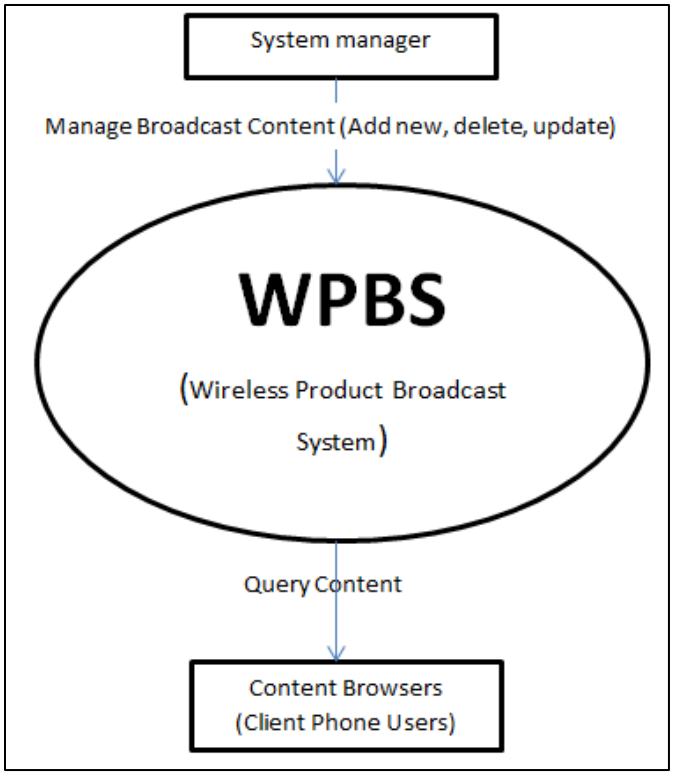


Figure 3.3 Design Context diagram

Once the advertisement data have been populated on the server device, many client applications can wirelessly connect simultaneously and query the data. The data flow diagram below show how adverts are produced and consumed.

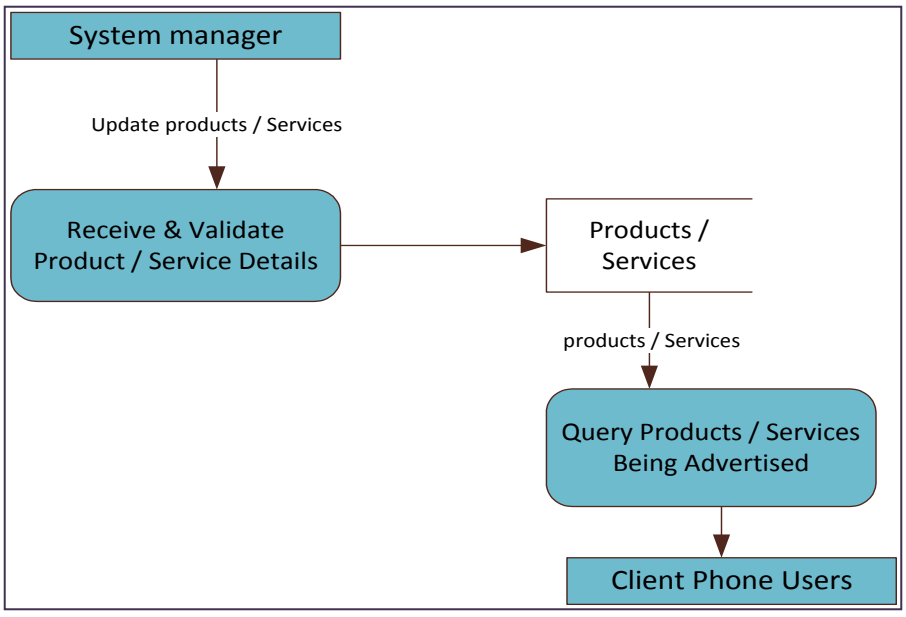


Figure 3.4 Conceptual Data flow diagram

3.2.2 Programming tools

The programming tools that were used are

1. Android SDK
2. Eclipse IDE
3. SQL lite - *SQLite* is an Open Source database. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements.

Versions to be used are

- Android- 4.2 Jelly bean
- IEEE 802.11 a/g/n Wi-Fi CERTIFIED gear.
- SAMSUNG Galaxy Note II, and SAMSUNG Galaxy Nexus phones
- Eclipse IDE - Luna Release

Figure 3.3 below depicts data flow in an android application installed in an android device. The application opens a data flow socket to WI Fi Direct which can then relay the data to other listening WI FI direct devices.

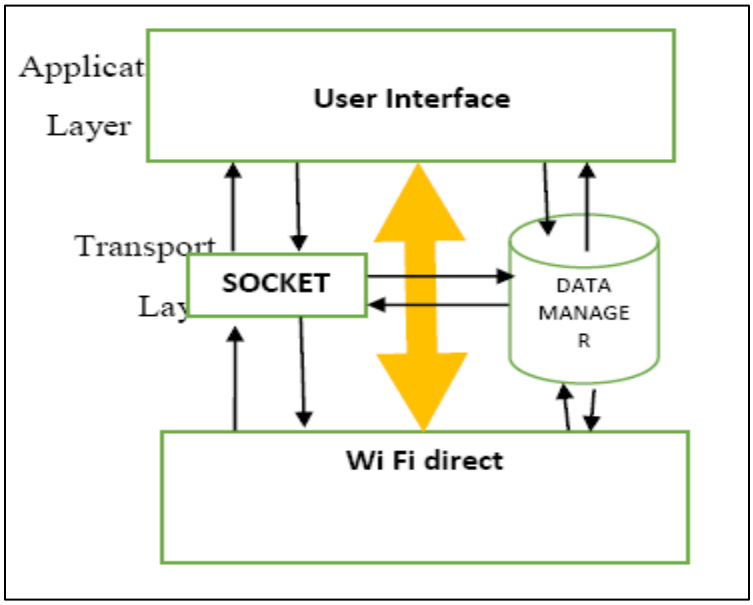


Figure 3.5 Wi-Fi direct Data Transport

User interface is used to connect to the bottom layers. Connection is established between the two ends. The Data Manager maintains the information about the connecting client devices. Once the connection is established the two devices, data exchange begins with the client device sending queries which are then replied by the server device.

3.3 Design and Implementation

Since Wi-Fi direct is a new technological aspect of wireless networks, I opted to use the iterative model derived from the waterfall model where the different activities are proceeded sequentially through various phases as depicted by the image below.

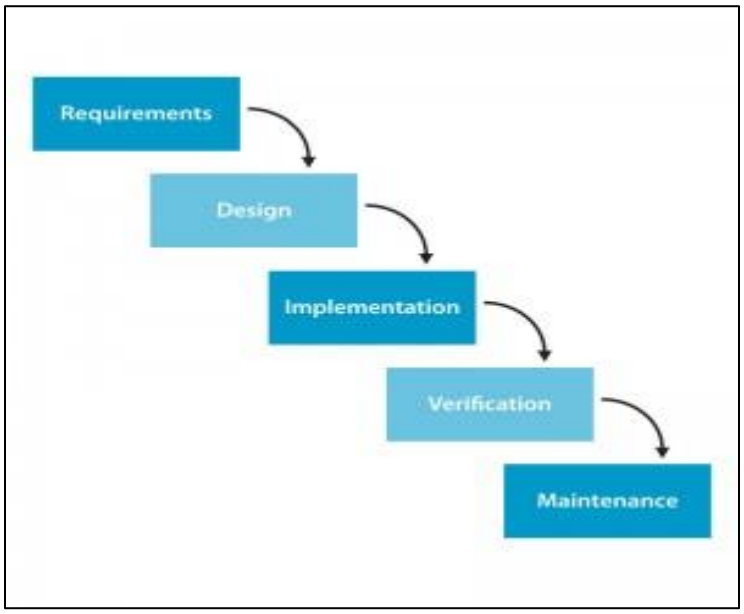


Figure 3.6 Waterfall model - adapted from [6]

As noted by Shelly and Rosenblatt (2010), the result of each phase in the waterfall model is called a deliverable, or end product, which flows into the next phase.

The waterfall model does not provide a way of going back to a previous phase to address new system requirements that come up during the development process and therefore, for this project I adopted an approach that allowed me to alternate between model design and code implementation through testing and evaluation. The approach iteratively incorporates all the waterfall model phases as depicted by figure 3.7 below.

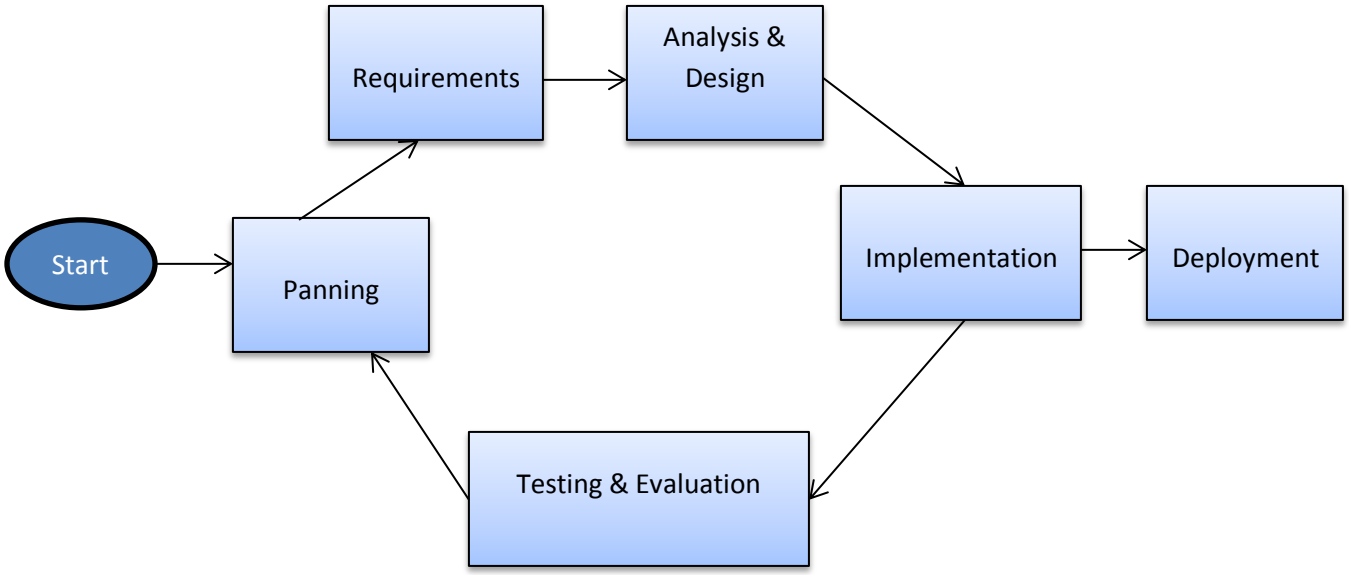


Figure 3.7 Iterative model

The planning largely involves the careful choosing of the various modules and the suitable artifacts required to complete their implementation.

3.4 Data Analysis

The server and client application were developed and deployed to their respective devices, a test was carried out to ensure that connectivity between the devices is reliable and that the devices are able to exchange data. The server device was be able to broadcast adverts while the client device was be able to connect to the server device and drill down on the products being advertised so as to display the product in detail.

In order to determine the optimal performance distance (Meters) between the server device and client device, an experiment was carried out by moving the client device away from the server device as response time is measured. The data were tabulated as shown in Table 3.3.0

Distance Between Server & Client (Meters)	Average Response Time (Milliseconds)
10	25
20	20
30	70
50	27
100	34.5
150	63
200	73

Table 3-1 Sample Data Form

3.5 Deliverables

1. Android application deployable on the server device (android mobile phone) responsible for broadcasting company's products and services.
2. Android application that serves as the client program deployable on a client device (user's android mobile phone).

4 CHAPTER 4 SYSTEM ANALYSIS AND DESIGN

4.1 User Requirements

The development and implementation of this system focused on a bank that needs to advertise its products to nearby potential customers via their mobile phones as depicted by the requirements narrative below

National Commercial bank is a local bank offering various banking products to its clientele; NCB however has a challenge maintaining continuous contact with their potential and current customers within or out of their premises particularly outside working hours when the offices are closed. NCB would like a system which can interact with users in close proximity to their premises, the system will act as an advertisement platform which users can query from their mobile phones and get up to date information about the products offered by the bank.

The system is expected to have a back-end interface where a bank official can add, delete and update products being advertised, furthermore the products are grouped into categories and when a category id deleted, the products within that category are also deleted.

Then client application running on a user's phone is expected to locate and connect to the bank's server device hosting the advertised products after which it begins to interrogate it by way of drilling down on the products being advertised. The medium of connection between the client application and the server application should be wireless so as to facilitate flexibility of reach.

This translates to the system requirements are as follows

1. Development of an application that runs on the customers phone that can reach and query the device within the bank's premises
2. Development of an application that runs on the server device (located within the bank's premises) listening for client connections and passing data to the client application through the established connection.
3. Develop an interface which the bank's official can populate product details to be advertised, this should include adding, updating and deleting of advertised products details.

4.2 Requirement Analysis use cases

To adequately understand the system, I represented all the programmable modules in the form of use cases.

4.2.1 Adverts Management system

This is the application that will be running on the banks server device.

There are two major tasks for user of this system; manage adverts and start as depicted in fig 4.1.0 below

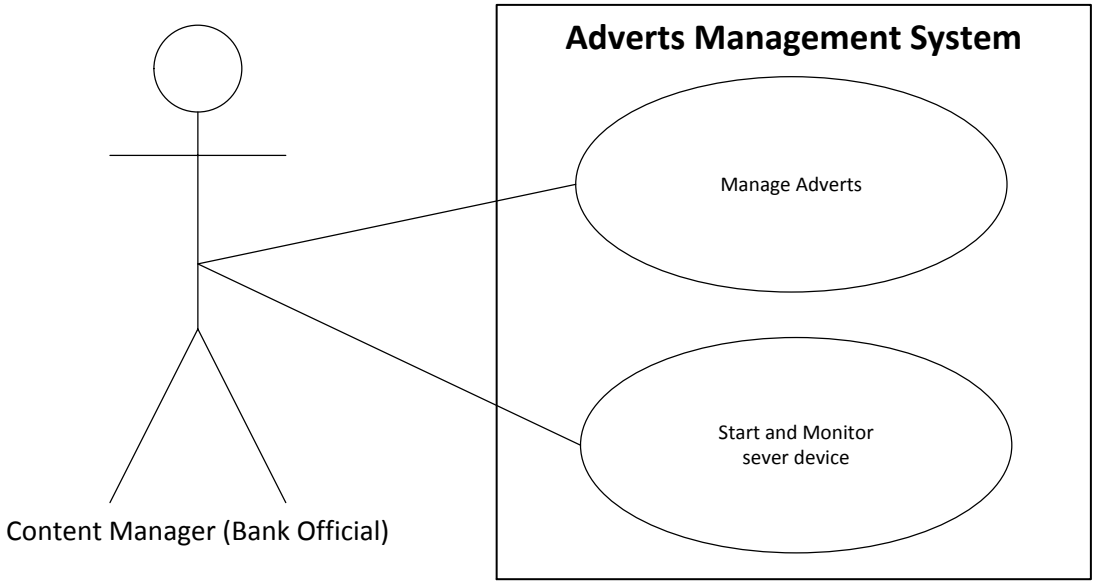


Figure 4.1 Server application main use case

This can further be broken down to activity particular use cases

Manage Adverts

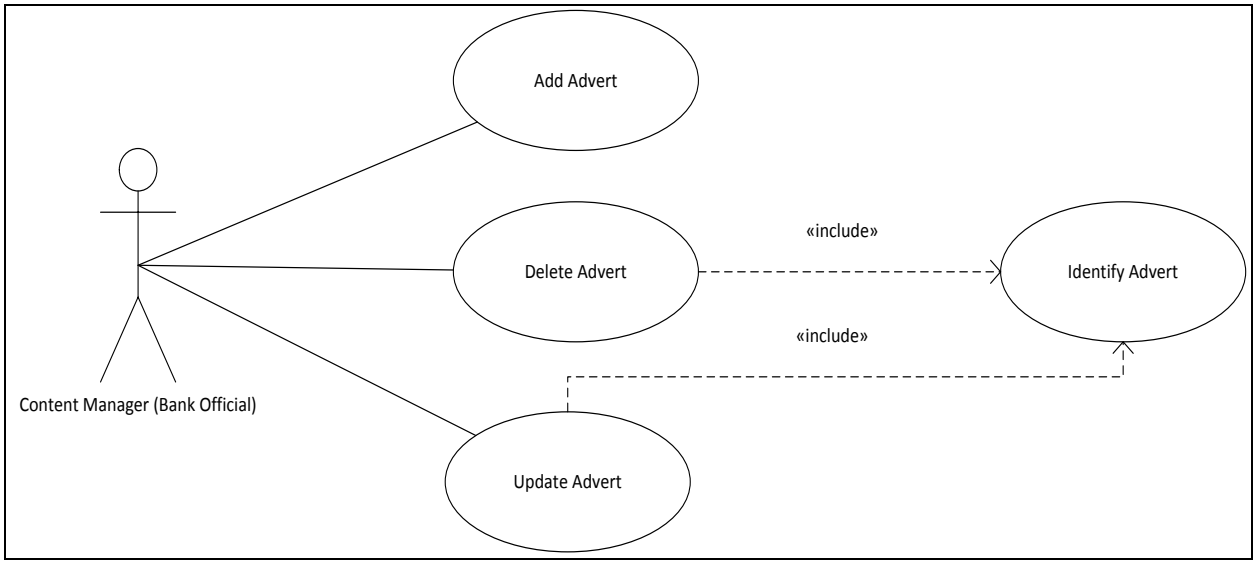


Figure 4.2 Advets management use case

Start and monitor server device

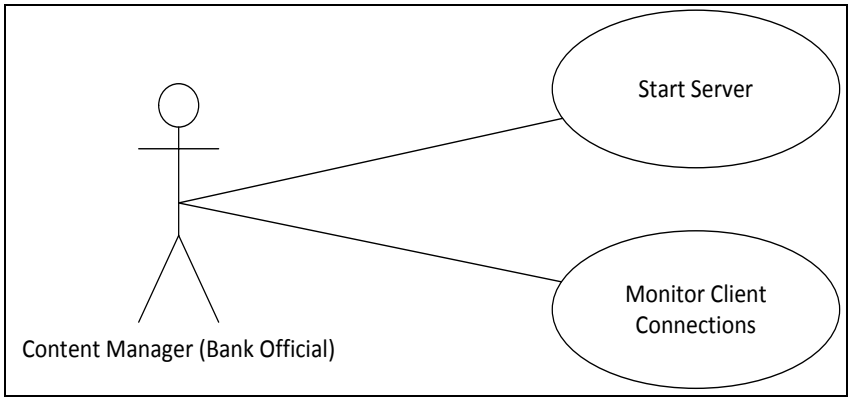


Figure 4.3 start server device use case

Adverts Management system use case documentation

Primary use case scenario	
The beginning	The use case begins when the bank's official needs to alter the content being broadcast to users
The Middle	The bank official navigates to the list of already existing adverts then selects either to add a new advert or delete existing advert or update

	existing record. In the case of add new, the banker enters the advert details to be broadcast to phone users by keying in a complete description of the product or service being provided by the banker (ID, Product or Service description), then submit the record for storage. The system then checks for duplication and informs the banker accordingly. Once the record is saved the user is taken back to existing list of adverts. For the case of delete the user selects the advert to delete from the list then submits it for deletion. The user is then prompted to confirm the deletion action he/she has chosen, if yes, the advert is permanently deleted from the system including it's sub items. If the user chooses to update then he/she is taken to product update page where he/she can alter the advert details and submit for storage.
The End	The system Assigns a unique ID to the new advert which can be used to attach to it's sub items.
Secondary use case scenario	
The beginning	The use case begins when the bank's official needs to alter the content being broadcast to users
The Middle	The bank official navigates to the list of already existing adverts then selects either to add a new advert or delete existing advert or update existing record. For the case of add new, the banker enters the advert details that are to be broadcast to phone users, a complete description of the product or service will be provided by the banker (ID, Product / Service description) then submit the record for storage. The system validates the entry and if there is a duplicate a message to that effect is displayed to the user and the advert is not saved, the user is then taken back to the adverts list screen. For the case of advert update, the user selects the advert to update then enters the new details which are validated and if a duplication condition is reached a message to that effect is displayed to the user

	and the advert is not saved, the user is then taken back to the adverts list screen.
The End	The system contents remain unchanged.

Table 4-1 Adverts Management system use case documentation

Use Case Form:

Form Element	Description
Use Case Name	Adverts Management system
Description	The bank official in charge of adverts being broadcast to the public via the broadcast device can either add new adverts, delete or update existing adverts
Actors	Primary Actor: Bank Official
Priority	Must Have: Highly essential to this system
Risk	Low: adverts management simply involves adding new, deleting or updating the content being advertised
Pre-conditions and Assumptions	None
Extension Points	None
Extends	None
Trigger	The bank official in charge of the adverts broadcast system needs to make changes to the advertised content.
Flow of Events	1.0 The use case starts when the bank official launches the adverts management screen 2.0 If the bank official chooses to add a new advert then an advert entry screen is opened where the user types the advert details while choosing the advert category. The user then submits the record which is then permanently saved 3.0 If the user chooses to delete an existing advert, then he /she selects the record from the list then chooses delete and after conforming his/her action via a confirmation prompt, the record and its sub items are permanently deleted from the database 4.0 If the user chooses to update then an update screen is opened where the user can edit the record and finally submit for permanent

	storage.
Alternate Flows	1.0 The use case starts when the bank official launches the adverts management screen 2.0 If the bank official chooses to add a new advert then an advert entry screen will be opened where the user types the advert details while choosing the advert category. The advert will not be saved if it violates the business rules set which also apply to advert update scenario
Post-conditions	At least one advert item is removed, added or updated
Business Rules	1. The advert field describing the product should have a maximum number of characters (e.g 200) since we are using a resource limited device to store the adverts 2. Every advert must have a parent category to facilitate grouping of adverts.
Non-Functional Requirements	The size of storage space used by the adverts software should not exceed the capacity of the phone which serves as the server device.
Notes	The server program host only 2 major functions: 1. The modules that facilitates creation, deletion and updating of adverts that will be broadcast to connecting clients 2. The module used to start the advert broadcast service through WI-FI

Table 4-2 Adverts Management system use case form

4.2.2 Broadcast services search system

This is the application that runs on the customer's phone querying the back end server.

This use case has three major tasks; search broadcast servers, connect to server and load advertised items and depicted in fig 4.1.3 below

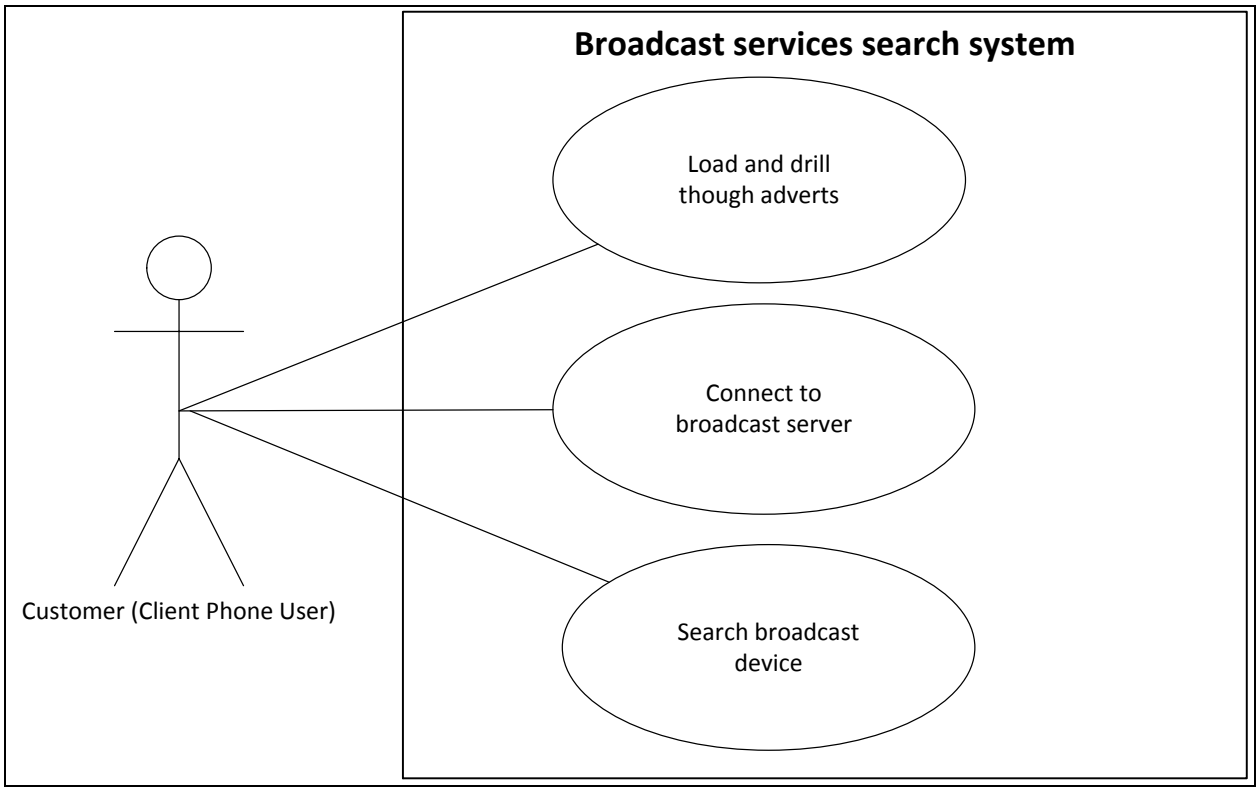


Figure 4.4 Client application use case

Since loading adverts depends on the connection created between the client device and the server device, fig 4.1.3 above can be reduced to fig 4.1.4 below

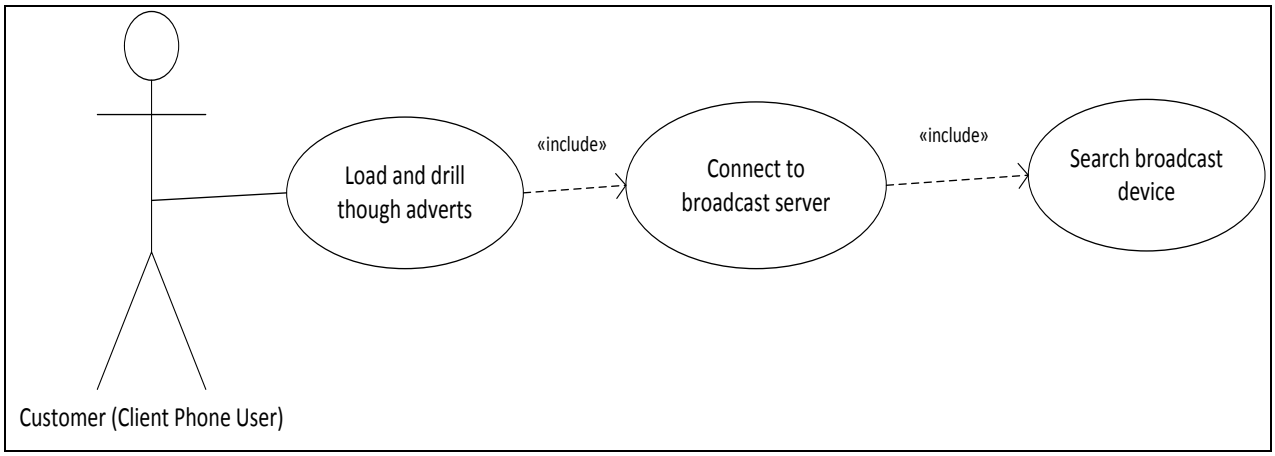


Figure 4.5 Client application user interaction use case

Broadcast services search system use case documentation

Primary use case scenario	
The beginning	The use case begins when a client user launches the application in his / her phone so as to search for nearby broadcast servers.
The Middle	The user launches the application and selects search services. The search operation ends with a list of all nearby service broadcast servers being displayed to the user. The user then selects the server device he / she would like to query then taps on connect button. Once the connection has been established between the user's phone and the selected broadcast server device, then server device forwards a list of adverts which are then displayed on the user's phone. The user can then decide to select and advert of his / her choice to view its details. If the selected advert has more sub items then the user can further drill down on the content until he / she finds the information he/she is looking for, thereafter he / she can choose to disconnect.
The End	The system ends when the user disconnects from the broadcast server device, the servers content remain unchanged.
Secondary use case scenario	

The beginning	The use case begins when a client user launches the application in his / her phone so as to search for nearby broadcast servers.
The Middle	The user launches the application and selects search services. If the search fails then the user is notified accordingly. If the search does not find any broadcast server device then the user will be shown an empty list and he / she can choose to invoke search again. If the search succeeds the user goes ahead and selects the server device he/she is interested in then initiates a connection to the selected device. The server device refuses new connections if its thread pool is already used up and the client device fails to establish a connection, the user is then unable to retrieve adverts from the broadcast server.
The End	The client device fails to retrieve the servers content and the user is unable to view adverts broadcast by the bank.

Table 4-3 client application use case documentation

Use case form

Form Element	Description
Use Case Name	Broadcast services search system
Description	This use case illustrates the adverts services search system where phone users running the client application can search and connect to broadcast devices so that they can view adverts
Actors	Primary Actor: Client phone user
Priority	Must Have: Highly essential to this system
Risk	High: since the client and the sever device connect to each other wirelessly then care must be taken to ensure proper management of the connection
Pre-conditions and Assumptions	Clients phone's WI-FI is turned on
Extension Points	None
Extends	None
Trigger	When a user interested in viewing the banks advertised products and offers he / she launches the client application and begins to locate the

	bank's broadcast device to connect to.
Flow of Events	1.0 The use case begins when a client user launches the application in his / her phone so as to search for nearby broadcast servers. 2.0 The user selects on search to begin the searching for nearby broadcast servers which ends in a list of servers being displayed to the user 3.0 The user chooses the broadcast server of his / her interest and selects connect to initiate connection 4.0 Once connected, the user selects load data operation which will load the server's adverts via the established connection, the user can subsequently drill down on the list of adverts to load the sub items. 5.0 The user disconnects
Alternate Flows	1.0 The use case begins when a client user launches the application in his / her phone so as to search for nearby broadcast servers. 2.0 The user triggers the search which ends in no servers found or a list of broadcast severs are displayed to the users. If no severs are found then the user can start searching again or terminates the application. 3.0 If the search was successful and a list of servers displayed, then the user will select a server and chooses connect. The client fails to connect may because the server device suddenly becomes unavailable or the server's thread pool is already used up and therefore cannot except anymore connections. 4.0 The user terminates the application.

Post-conditions	No change on the broadcast device's (server) content.
Business Rules	1. The client application can connect to only one broadcast device at a time 2. The client application can only read the server's broadcast content, it cannot delete or update
Non-Functional Requirements	Multi-threading: The server device should be able to handle simultaneous client connections
Notes	The server application continuously listens to client connection and once there is a connection, it is assigned a thread from an existing thread pool. Once the client has been served with the data it requested, the connection terminates.

Table 4-4 client application use case form

4.3 System Design

The general configuration of the final system is such that several client phones can connect to the central server device and query the data as per user request. The figure below depicts the end result.

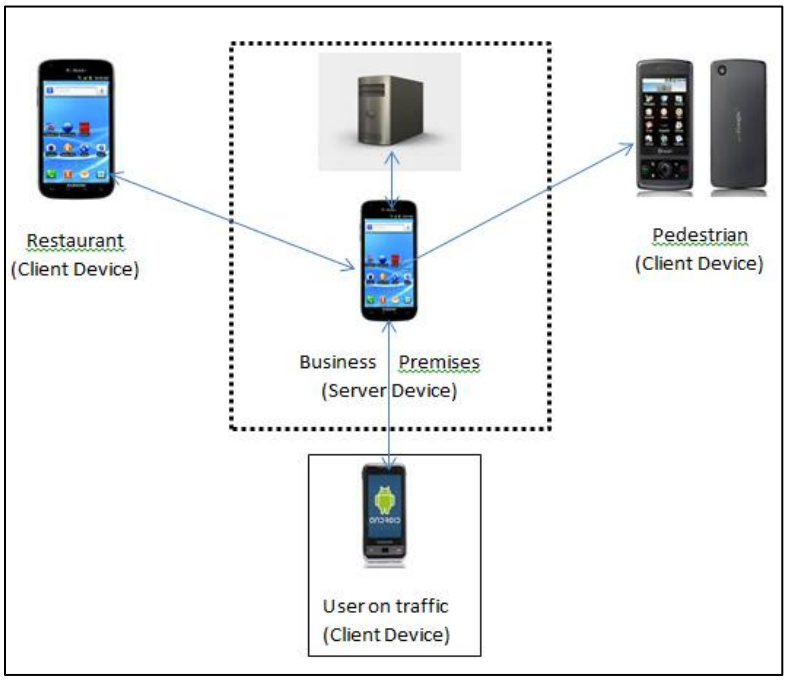


Figure 4.6 proposed conceptual design

A set of Wi-Fi Direct certified devices connect by forming an ad-hoc network with one of the devices designated as the group owner and any ordinary Wi-Fi device can discover and connect to Wi-Fi Direct groups. The group owner appears as an access point for legacy equipment and for this project the group owner is the server device (Android Phone) resident within the bank's premises which accepts client connections and passing data to clients through the established socket connections.

Once a client device has located the server device, it requests to join the group and if granted the client application can drill through the adverts. The figure below is a summary of the communication model between the client device and the server device.

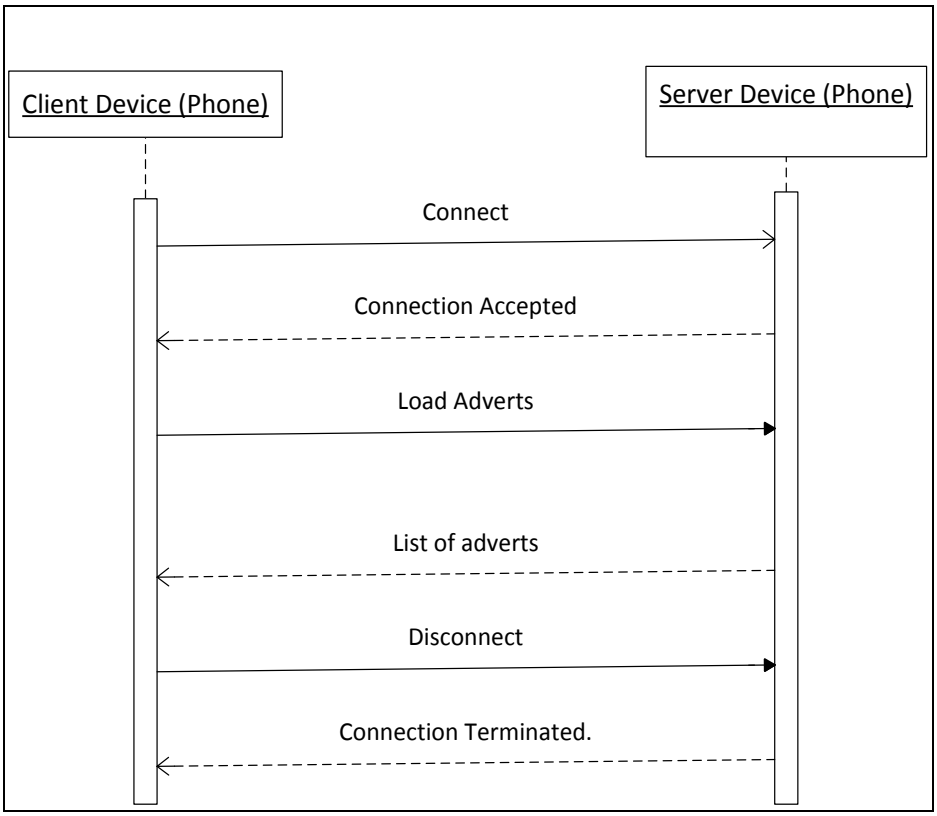


Figure 4.7 Client and server device interaction

4.3.1 Server application design

4.3.1.1 Server application activity diagram

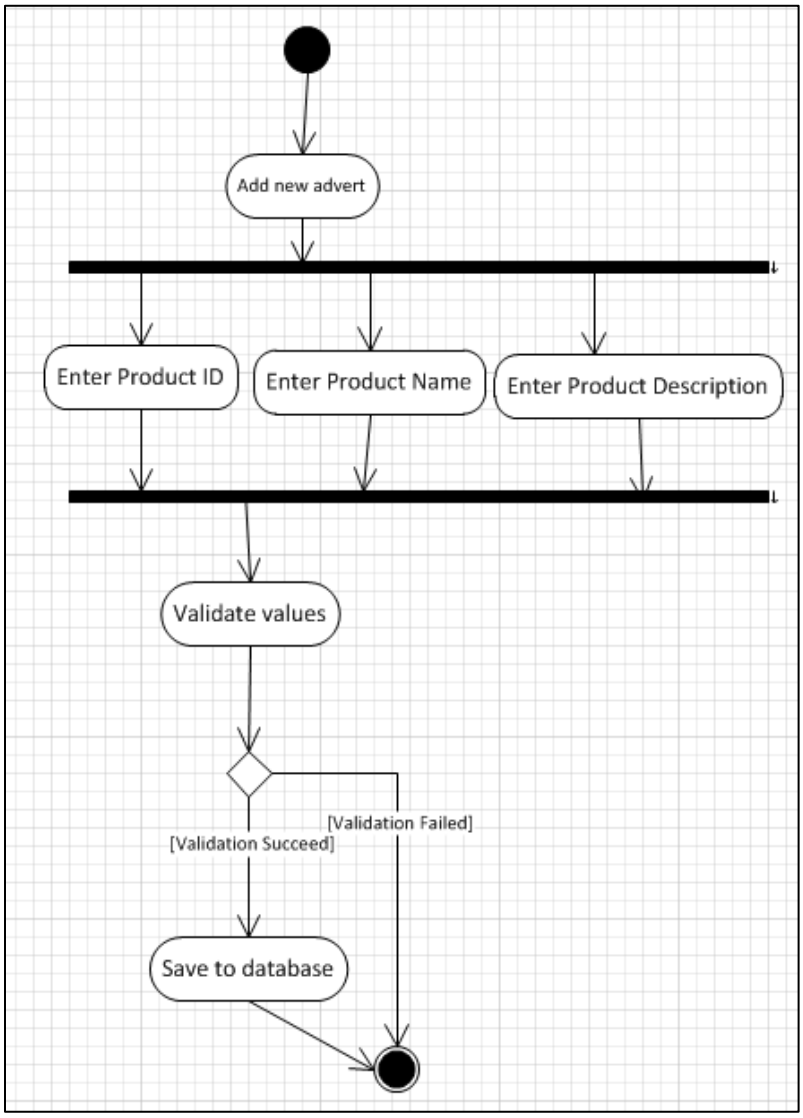


Figure 4.8 Server application activity diagram

The activities performed at the server involves managing the content being advertised (i.e adding new advert, updating adverts and deleting adverts).

These activities are as depicted in the sequence diagram below.

4.3.1.2 *Server application sequence diagram*

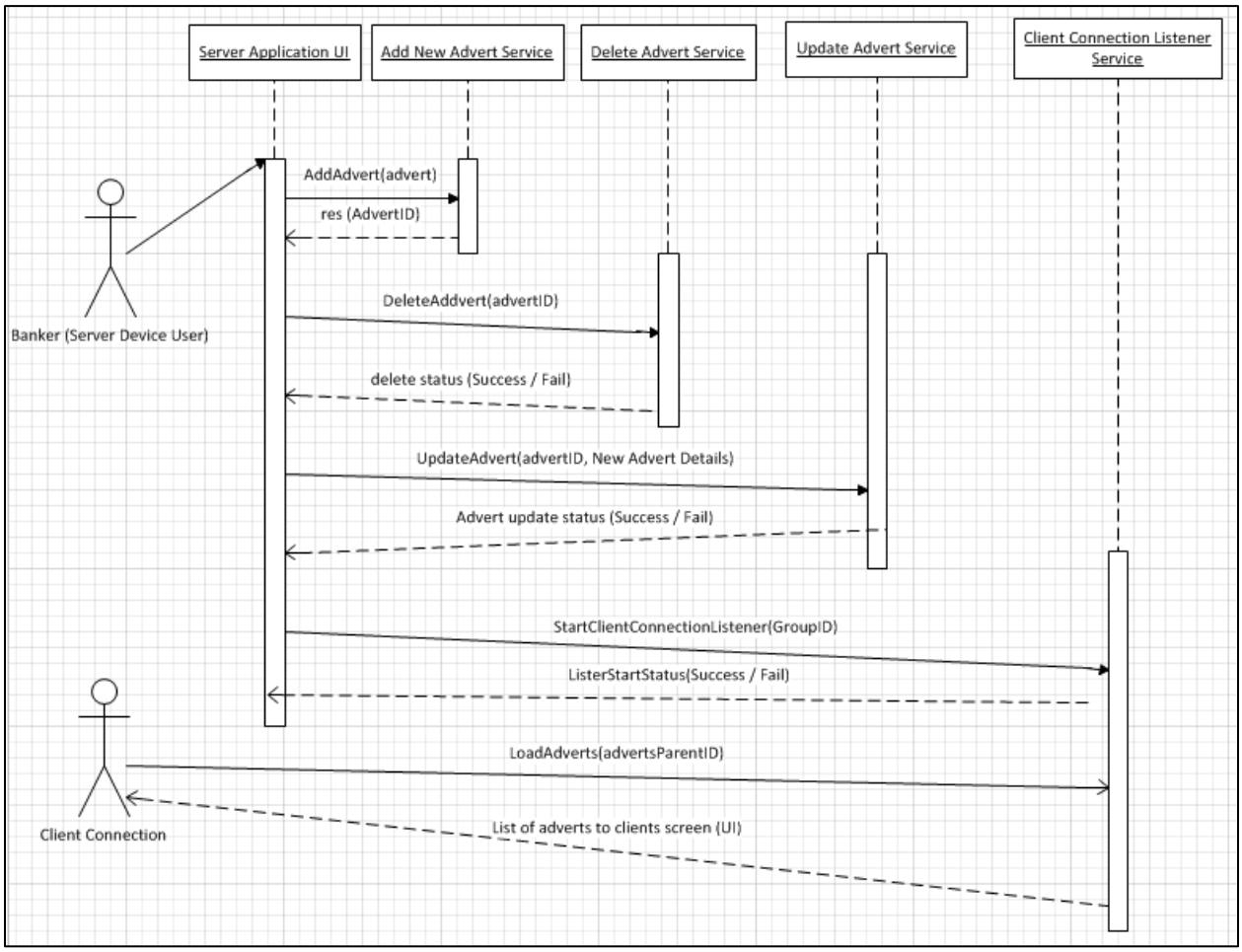


Figure 4.9 Server application sequence diagram

4.3.1.3 *Server application key abstractions*

From the user requirements artifacts contained in the use cases as well as the user requirement narrative, they key abstractions can be pointed out from the nouns contained.

User Requirement Narrative:

National Commercial **bank** is a local bank offering various banking products to its clientele, NCB however has a challenge maintaining continuous contact with their potential and current customers within or out of their premises particularly outside working hours when the offices are closed. NCB would like a system which can interact with users in close proximity to their premises, the system will act as an advertisement platform which users can query from their mobile phones and get up to date information

about the products offered by the bank through the adverts broadcast wirelessly.

The system is expected to have a back-end interface where a bank official can add, delete and update products being advertised, furthermore the adverts are grouped into categories and when a category is deleted, the products within that category are also deleted.

Then client application running on a user's phone is expected to locate and connect to the bank's server device hosting the advertised products after which it begins to interrogate it by way of drilling down on the products being advertised. The medium of connection between the client application and the server application should be wireless so as to facilitate flexibility of reach.

Key abstractions validation using CRC (class responsibility collaborators) Card.

SeverDevice	
Responsibilities Hosts the database of advertised content	**Collaborators**

Advert	
Responsibilities Item being advertised	**Collaborators** Adverts (sub items)

Candidate Key Abstractions Form

Candidate Key Abstraction	**Eliminated for the Following Reason**	**Selected Component Name**
Bank	External to the system	
Advert		Advert
Customers	External to the system	
System	The Whole system	
Users	External to the system	
Server Device		ServerDevice
Bank official	System Operator: External to the system	

WI-Fi	System Communication media	

Table 4-5 Candidate Key Abstractions Form

4.3.1.4 Server Application Domain Model

Based on the identified key abstractions in the previous section the problem domain model comes out as shown in the figure below.

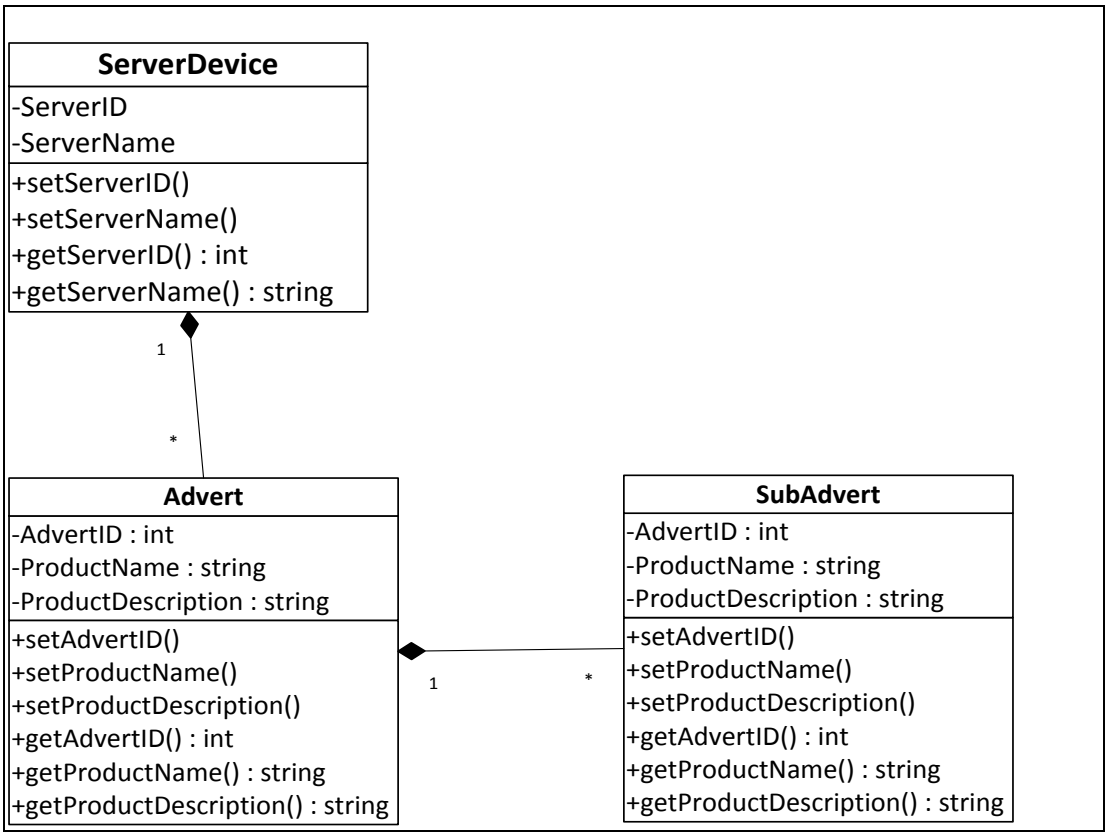


Figure 4.10 server application domain model

Since the advert and SubAdvert Classes have the same fields, they can be matched to depict a *self join* as shown in the domain model below

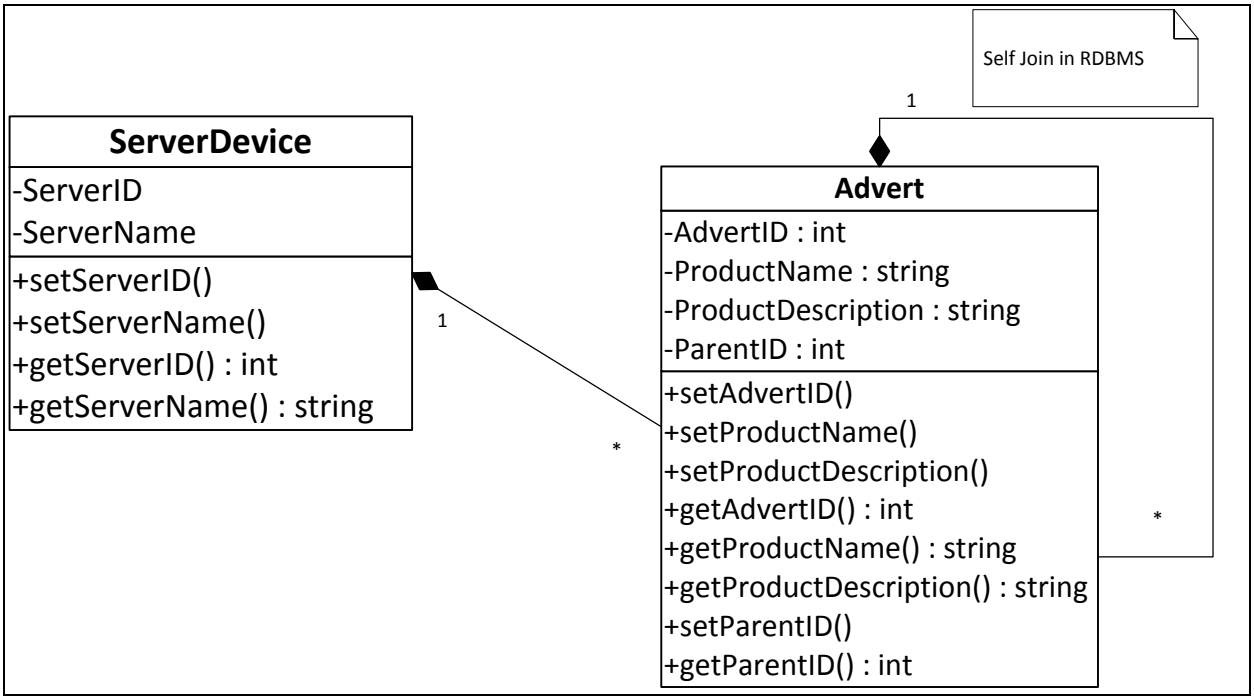


Figure 4.11 refined server application class diagram

By adding a parent ID field in adverts, we are able to attach sub adverts to parent advert which also serve as the category. Outer most parent adverts have their parent ID set to a value of -1

4.3.2 Client application design

4.3.2.1 Client application activity diagram

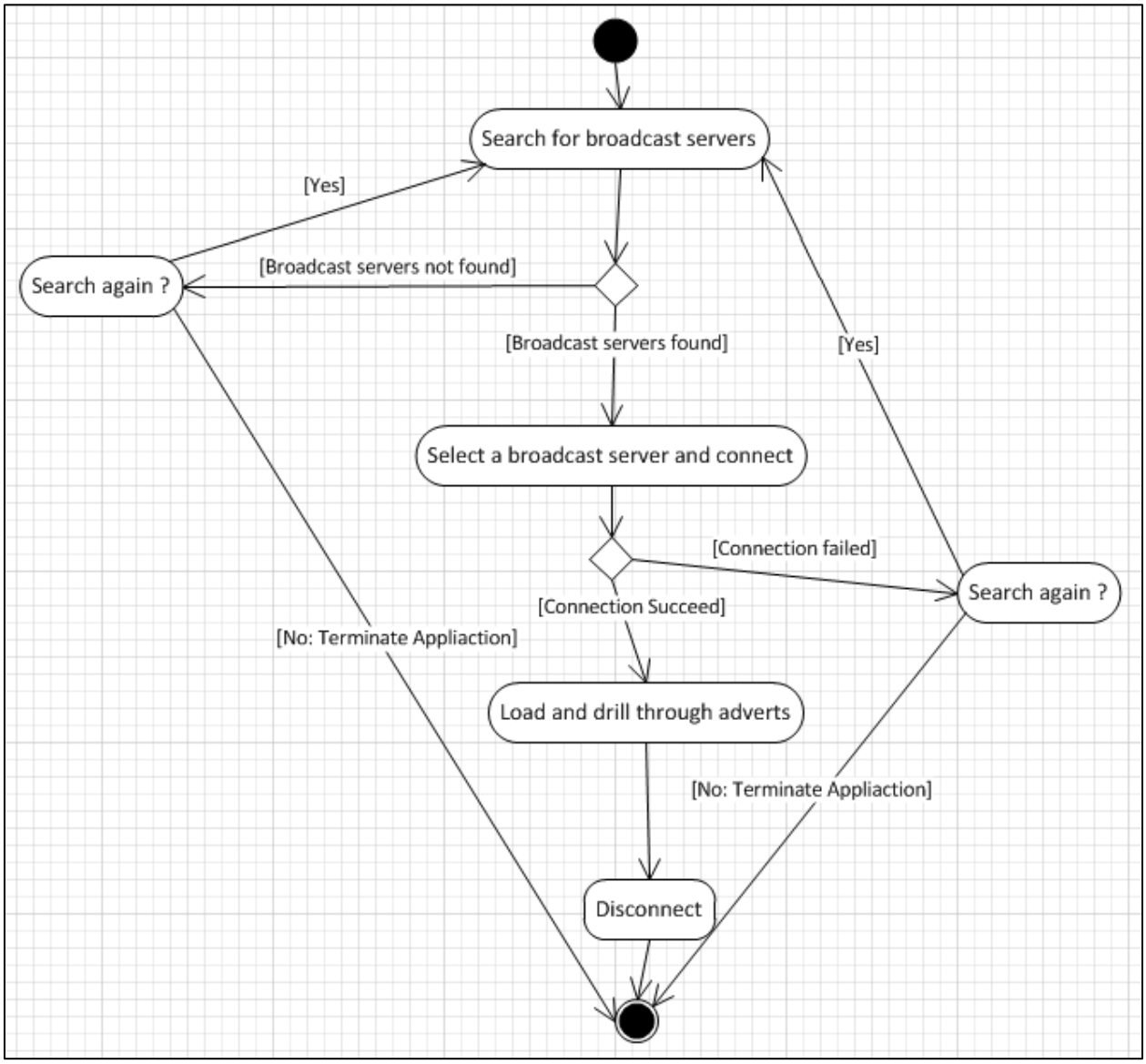


Figure 4.12 Client application activity diagram

Based on the client application use case flow of events, the tasks can be sequenced as shown in the sequence diagram below.

4.3.2.2 Client Application sequence diagram

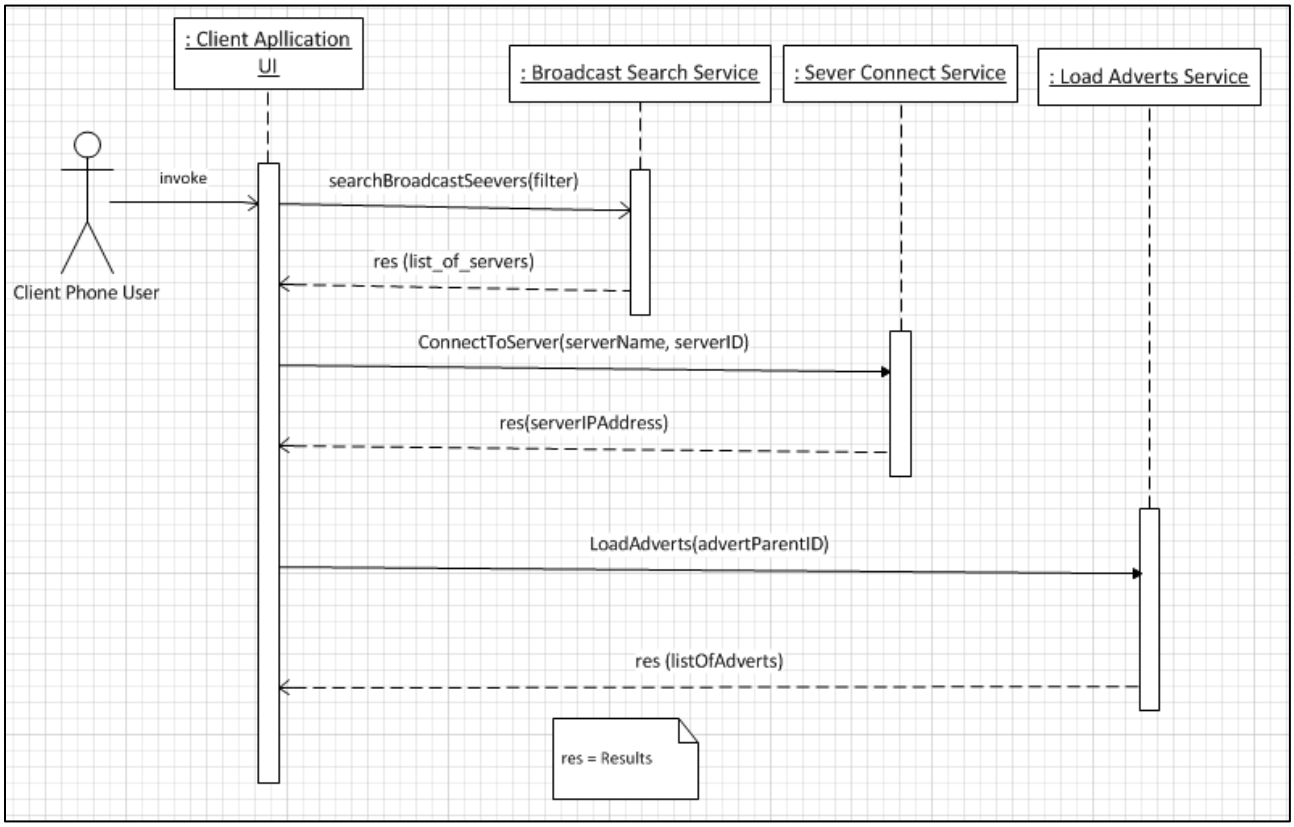


Figure 4.13 Client application sequence diagram

4.4 Implementation

The development of both the server and client application has been done using standard eclipse IDE (Luna release) loaded with Android development packages with the following development details chosen.

Application	SDK	Minimum SDK	Target SDK	Compiled With	Database
Server Application	Android	API 17: Android 4.2 (Jelly Bean)	API 17: Android 4.2 (Jelly Bean)	API20 Android 4.4 (KitKat)	SQL Lite
Client Application	Android	API 17: Android 4.2 (Jelly Bean)	API 17: Android 4.2 (Jelly Bean)	API20 Android 4.4 (KitKat)	Not Applicable

Table 4-6 Development platform details

N/B) Since the system do not need to store any data on the client side, we not need any database implementation.

4.4.1 Server application implementation

Once the server application has been populated with adverts, its next task is to continuously listen for client connection then assign a thread to each connection that has been accepted

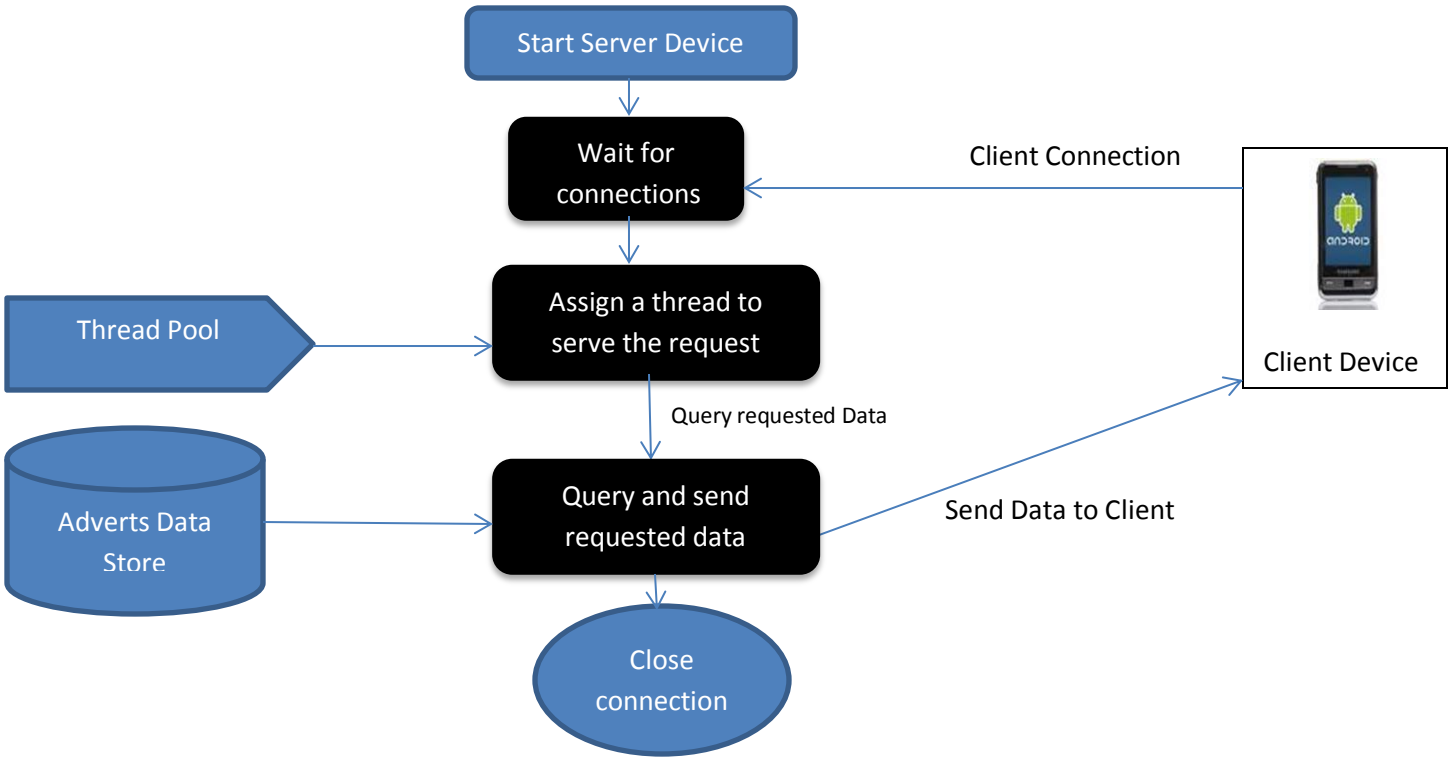


Figure 4.14 Implementation plan

Wi-Fi direct provides Wi-Fi P2P API that allow applications to connect to nearby devices without needing to connect to a network or hotspot, it enables applications to quickly find and interact with nearby devices, at a range beyond the capabilities of Bluetooth. For this server application broadcast receiver registration and on connection available code has been implemented as below.

4.4.1.1 Broadcast Registration

```
private WifiP2pManager manager;

public void searchPeer() {
		// check if wifi direct is enabled
		if (!isWifiP2pEnabled) {
			new AlertDialog.Builder(this)
					.setIcon(R.drawable.ic_action_discover)
					.setTitle("WiFi Direct is Disabled!")
					.setPositiveButton("Setting",
							new DialogInterface.OnClickListener() {
								public void
onClick(DialogInterface dialog,
										int whichButton) {
									startActivity(new Intent(

	Settings.ACTION_WIRELESS_SETTINGS));
								}
							}).show();
			return;
		}


		// Display all devices using fragment class
		final DeviceListFragment fragment = (DeviceListFragment)
getFragmentManager()
				.findFragmentById(R.id.devicelist);
		fragment.onInitiateDiscovery();
		fragment.getView().setVisibility(View.VISIBLE);
		//Broadcast sever availability
		manager.discoverPeers(channel, new WifiP2pManager.ActionListener() {
			@Override
			public void onSuccess() {

				return;
			}

			@Override
			public void onFailure(int reasonCode) {
				Toast.makeText(CONTEXT, "Search Failed: " + reasonCode,
						Toast.LENGTH_SHORT).show();
				return;
			}
		});
	}
```

The above code is executed when the broadcast server is started and all client devices (client application installed) are able to locate it.

To be able to detect and respond to client events and Wi-Fi direct state events, a broadcast receiver was developed to handle all intents. The broadcast receiver registered in the application handles the following intent actions (refer appendix 8.3 for code)

- WIFI P2P STATE CHANGED ACTION – This enables the application to detect if Wi-Fi Direct mode is enabled on the current device. When a user turns on or off the Wi-Fi Direct mode on the device (android phone), an intent will be broadcasted enabling the application to be informed on the Wi-Fi Direct status. For this application the user will be notified to turn on Wi-Fi direct so as to be able to search and connect to broadcast servers.
- WIFI P2P PEERS CHANGED ACTION – this action enables the application to receive notification on when peers have been discovered. This is triggered following a peer discovery process or broadcast server search
- WIFI P2P CONNECTION CHANGED ACTION – the application must listen to this action in order detect any changes in Wi-Fi connectivity. This action can be triggered when any of the following occurs
    - A connection request is made by the current device and the server device which is the group owner has acknowledged.
    - The connection was lost.
    - A disconnection procedure was successfully accomplished.
    - A device connects to the current device
- WIFI P2P THIS DEVICE CHANGED ACTION – captures change in the device's point-to-point properties, this occurs when the devices status changes from available to connected.

Every time WIFI P2P CONNECTION CHANGED ACTION is triggered, we check if it a disconnection or a connection from the client device, if it is a connection then the server class is executed to fetch the data requested by the client device.

```
if (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION .equals(action)) {
     Server server = new Server(cxt);
     server.execute();
}
```

The server class implements extends AsyncTask to enable us process each client request asynchronously (refer to appendix of this document for the server class code). The server opens a socket which all clients connect through using port 8990

Database Table Fields

Field	Description	Nullable	Data Type	Default Value
ID	Primary Key	No	integer	Auto Increment
AdvertDescc	Advert Description	No	String	XXXXXXXXXXXXX
ParentID	Parent Category ID	No	integer	-1

4.4.2 Client application implementation

The client application is composed of 3 major tasks as depicted by the diagram below

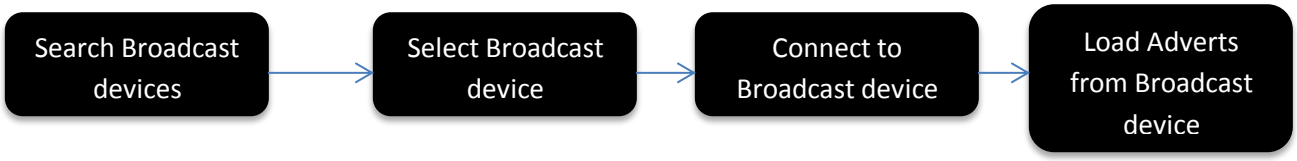


Figure 4.15 Client Application execution sequence

4.4.2.1 Search Broadcast Devices

The code executing this task is meant to return a list of broadcast servers which the user can select a server from.

```
	public void searchPeer() {


		// found devices will be displayed in fragment class
		final DeviceListFragment fragment = (DeviceListFragment)
getFragmentManager()
				.findFragmentById(R.id.devicelist);
		fragment.onInitiateDiscovery();
		fragment.getView().setVisibility(View.VISIBLE);
		//Search for peers
		manager.discoverPeers(channel, new WifiP2pManager.ActionListener() {
			@Override
			public void onSuccess() {

				return;
			}
```

```
                @Override
                public void onFailure(int reasonCode) {
                        Toast.makeText(CONTEXT, "Search Failed: " + reasonCode,
                                        Toast.LENGTH_SHORT).show();
                        return;
                }
        });
                                        }
```

Once Broadcast devices have been detected, DeviceListFragment class which implements PeerListListener Interface responds to the event and display all the detected broadcast devices.

DeviceListFragment class

```
public class DeviceListFragment extends ListFragment implements
                PeerListListener {
        private List<WifiP2pDevice> peerList = new ArrayList<WifiP2pDevice>();
        View myContentView = null;
        ProgressDialog progressDialog = null;

        // callback function when searching is done.
        @Override
        public void onPeersAvailable(WifiP2pDeviceList peers) {
                if (progressDialog != null && progressDialog.isShowing()) {
                        progressDialog.dismiss();
                }
                myContentView.findViewById(R.id.peerlisttext).setVisibility(
                                View.VISIBLE);
                peerList.clear();
                // use a list view to update the result.
                peerList.addAll(peers.getDeviceList());
                ((WiFiPeerListAdapter) getListAdapter()).notifyDataSetChanged();
                if (peerList.size() == 0) {
                        ((MainActivity) getActivity()).showMessage("No device found.");
                }
                return;
        }
}
```

4.4.2.2 Connect to Broadcast Device

Connecting to a broadcast device involves joining the client device to the Wi-Fi group that the broadcast device has created. Upon joining the group the client application acquires the IP Address of the group owner which is the server broadcast device, this IP address is then used to create socket communication between the client and server.

```
public void connect(WifiP2pConfig config) {
            config.groupOwnerIntent=0;
            manager.connect(channel, config, new ActionListener() {
                  @Override
                  public void onSuccess() {
                        // We will be notified by WPBroadcastReceiver
                  }

                  @Override
                  public void onFailure(int reason) {
                        showMessage("Connect failed: " + reason);
                  }
            });
            return;
      }
```

If the connection was established, we will be notified by WPBroadcastReceiver class which extends BroadcastReceiver, WPBroadcastReceiver will notify DeviceDetailFragment which implements ConnectionInfoListener, at this point we make the adverts load button visible.

4.4.2.3 Load Adverts From Broadcast Device

Once the client is connected, we can query from adverts from the server by creating a network socket using the broadcast device's IP address. On the click event of load data button we execute the client

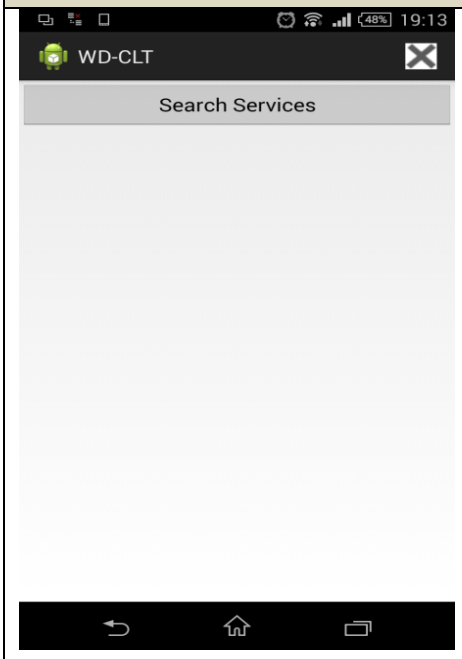
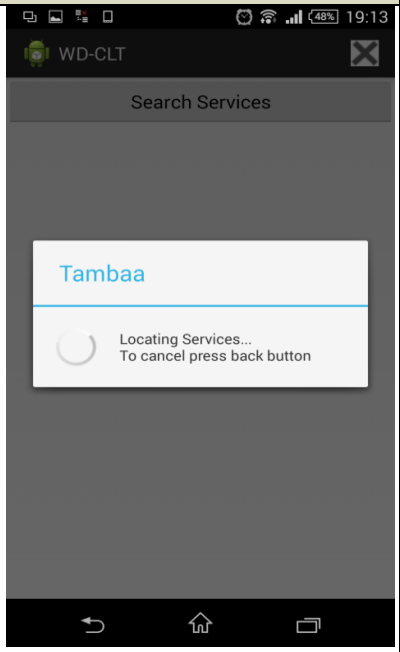
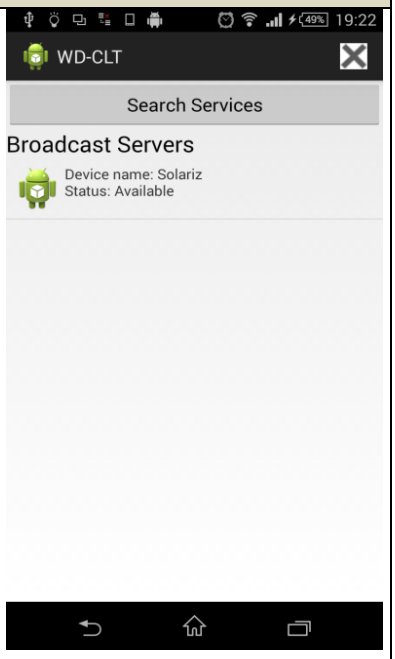
```
myContentView.findViewById(R.id.btnLoadData).setOnClickListener(
                        new View.OnClickListener() {
                              @Override
                              public void onClick(View v) {
                                    // ((MainActivity)
getActivity()).cancelDisconnect();
                                    client = new Client(info.groupOwnerAddress
                                                .getHostAddress(), 8990,
myHandler, -1);
                                    client.execute();

                                    //textView = (TextView)
myContentView.findViewById(R.id.peerdevice);
                                    //textView.setText("Client Started");
                                    Log.d("CLT", "Client Started");
                                    return;
                              }
                        });
```

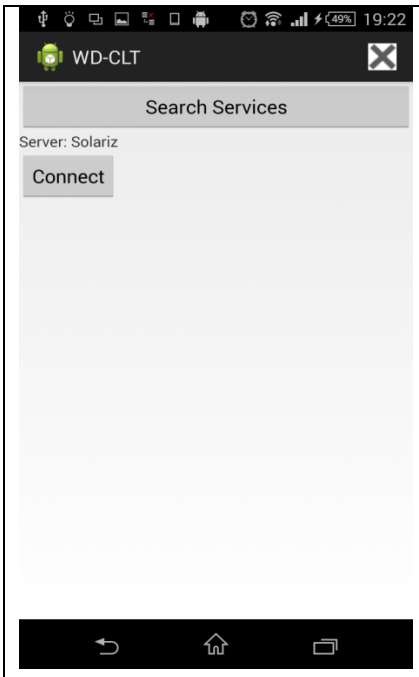
5 CHAPTER 5: TEST, RESULTS AND DISCUSSIONS

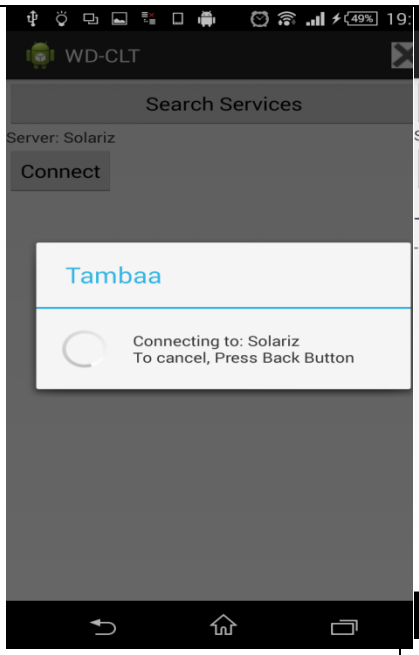
For purposes of this prototype sample advertisement data were loaded into the server device and the output are as follows

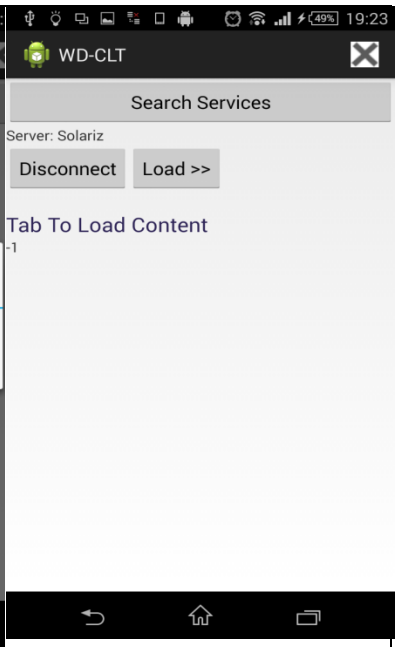
5.1 Client execution sequence

(Sample advertisement data)

Step 1. Open Program	Step 2. Tap on Search Services	Step 3. List of server
		
Step 4. Select Server	**Step 5. Connect (tap Connect)**	**Step 6. Tap load data**

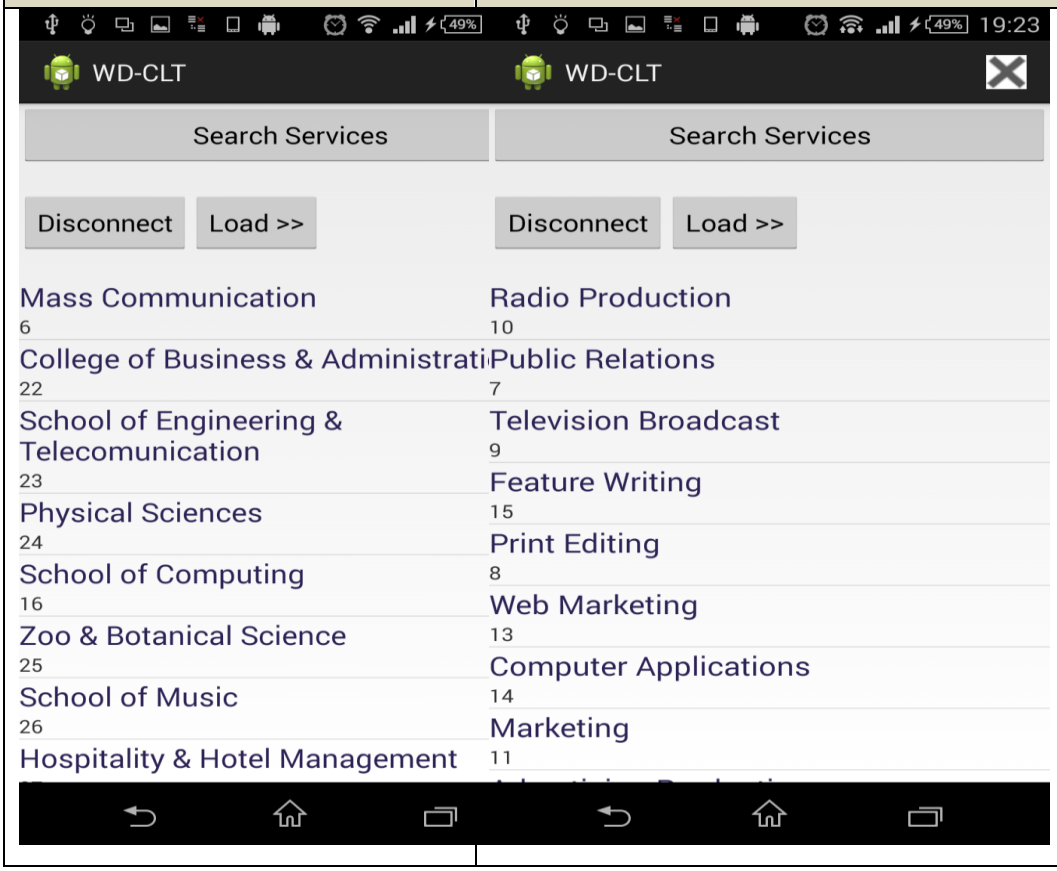




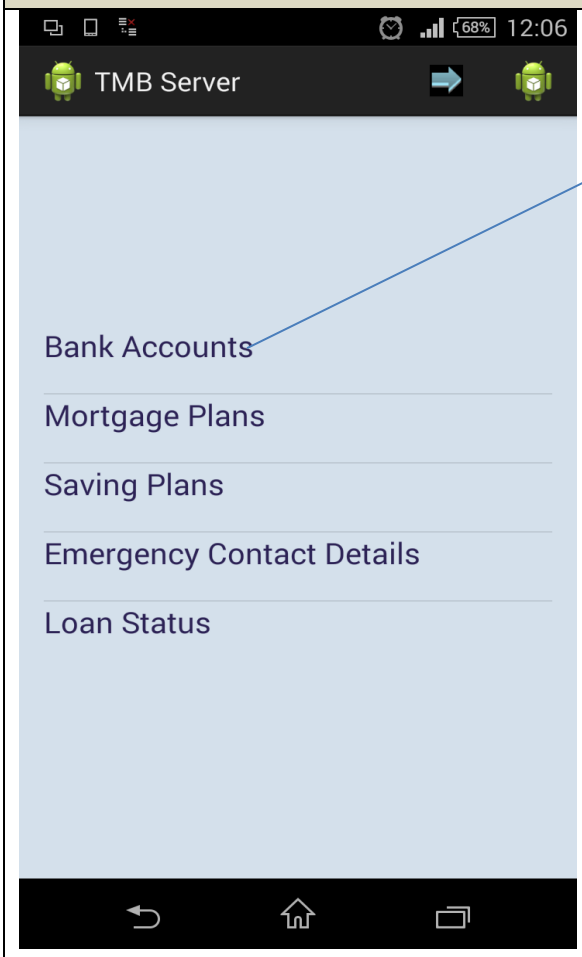
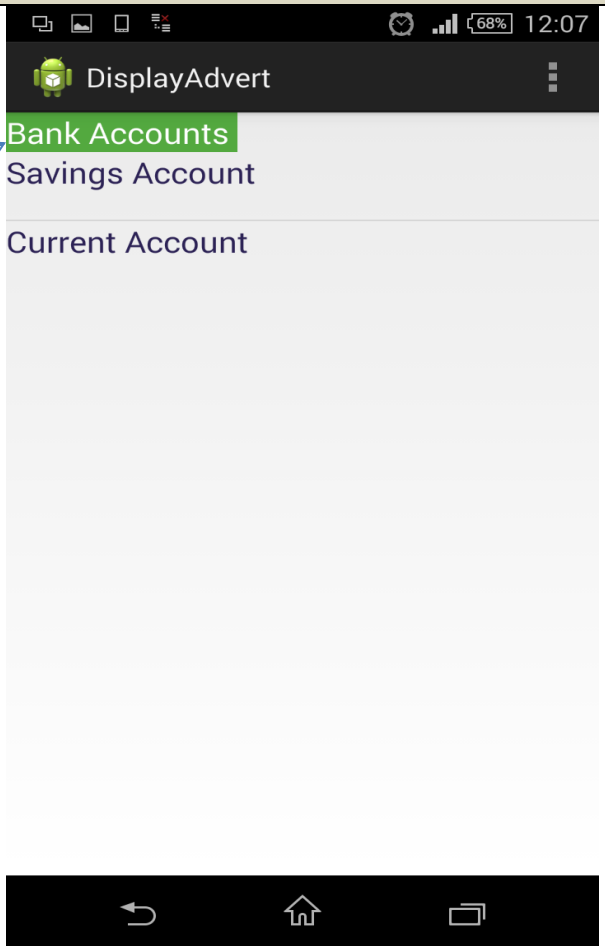


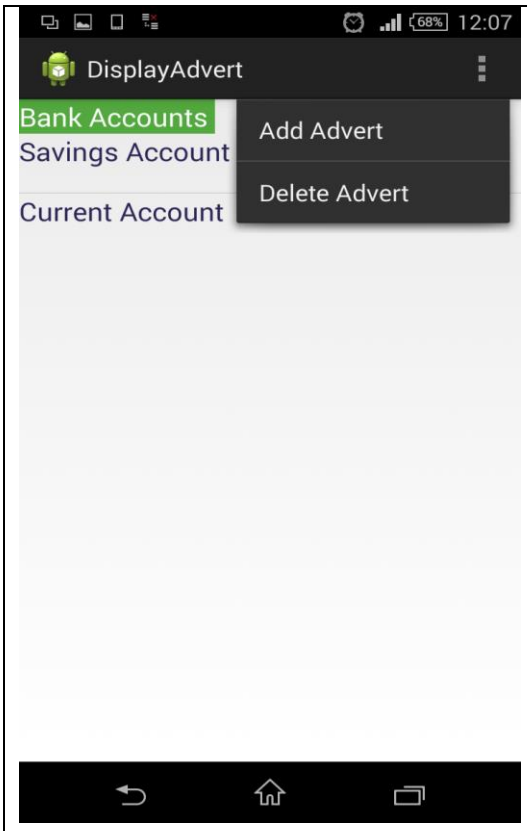
Step 7. Tap load data

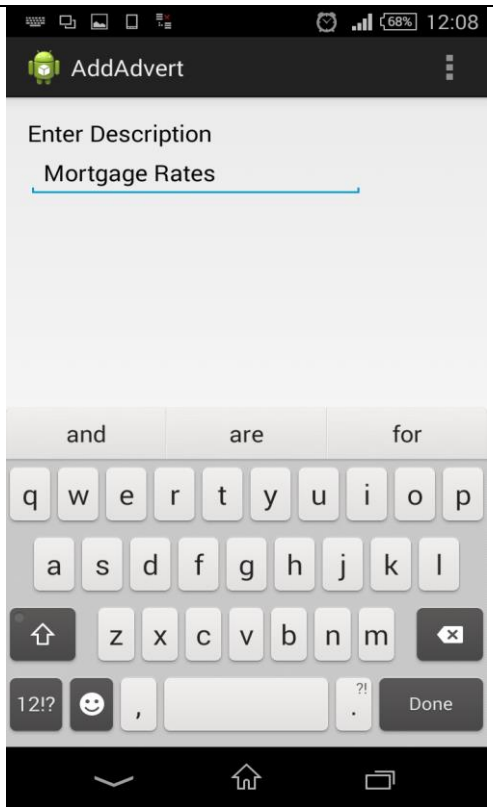
Step 8. Drill through by selecting an item from the list



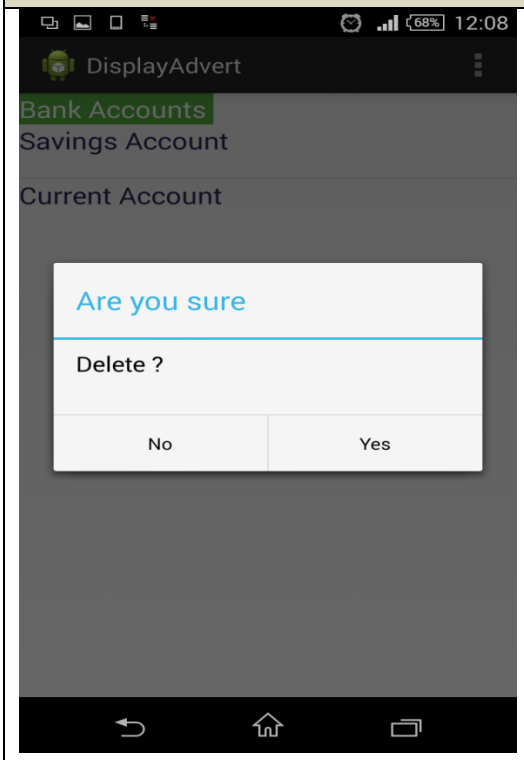
5.2 Server execution sequence

Existing Adverts	Sub adverts
	
Actions	**Add New Choosen**





Delete Choosen



The server application was successfully deployed and was able to receive and serve clients requests as per the design. Similarly the client application was successfully deployed on an android client phone and it was able to search, connect to server device and load advert for the user to view.

The client phone user is able to drill down on the adverts, disconnect from the server device or terminate the client application altogether.

5.3 Performance Testing

5.3.1 Deployment device specification

The server and client applications were deployed to different physical devices as indicated in the table 5.1 below.

Device	Model	Android version	Processor	Memmory (RAM)
Server Device	Techno S3	4.2.2 Jelly Bean	1.0 GHz dual-core Mediatek MT6572M CPU	512 MB
Client Device	Sony Xperia T3	4.4.2 KitKat	1.4GHz quad-core Cortex-A7 CPU, Qualcomm Snapdragon 400 chipset, Adreno 305 GPU	1GB

Table 5-1 Testing devices

5.3.2 Test Results

The system was tested for performance by measuring the response time against distance of the client device from the server device, the testing was conducted at Chiromo Campus grounds and the results are as tabulated below.

Distance Between Server & Client (Meters)	Response Time (Milliseconds)		
	Test 1	Test 2	Average
10	22	28	25
20	20	20	20
30	80	60	70
40	35	19	27
50	21	48	34.5
60	104	22	63
70	70	76	73

80	32	29	30.5
90	75	99	87
100	40	30	35

Table 5-2 Test Results

From the results above, it is clear that once a connection has been established between the client device and the server device, the time taken for the server to respond to a client request will not be determined by the distance between the server device and the client device.

5.4 System limitations

Wi-Fi Direct devices operate in the 2.4 GHz frequency range and can reach up to a distance of between100 and 200 meters (Wi Fi Alliance, 2010), clients trying to connect beyond this distance will not be able to locate the broadcast device and therefore unable to load and view the broadcast content. The other challenge is if the server device is mounted behind thick walls that will block line-of-sight connectivity between the server and client device.

6 CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

The objectives of this research project were laid down in section 1.3 of this document. The server application was designed, developed, deployed and tested, the system was able to accept new adverts and broadcast them to in coming client requests. Similarly the client application was designed, developed, deployed and tested, the application was able to locate Wi-Fi direct broadcast servers within a radius of 100 meters, connect and query data (adverts) from the server device. As depicted in figure 4.2.2, the communication between the server application and the client application is achieved through networking sockets. This system is applicable to any business that needs to wirelessly reach clients via their mobile phones; it can also be used to provide any other type of information product purchase offers.

Wi-Fi Direct devices operate in the 2.4 GHz frequency band with capabilities of connecting to both 802.11g and some 802.11n devices, this research has demonstrated the use of Wi-Fi direct to distribute adverts (saved in SQL Lite database) to clients connecting to a central device from a distance of up to 100 meters, apart from running advertisements in the location based system developed, it can also be used to distribute any other type of information like offers and promotions to connecting clients.

6.2 Recommendation

In order to achieve better performance, the server application needs to be installed in an android device with high computing resources and this will depend on the number of envisaged concurrent connections. As depicted by figure 4.2.8, the server application is multi-threaded and there will require more computing resources (processor speed, memory- RAM) as the number of connections increase, therefore a Wi-Fi direct enabled desktop would be appropriate for the server device in which case a practical environment setup would be as per figure 6.1 Next.

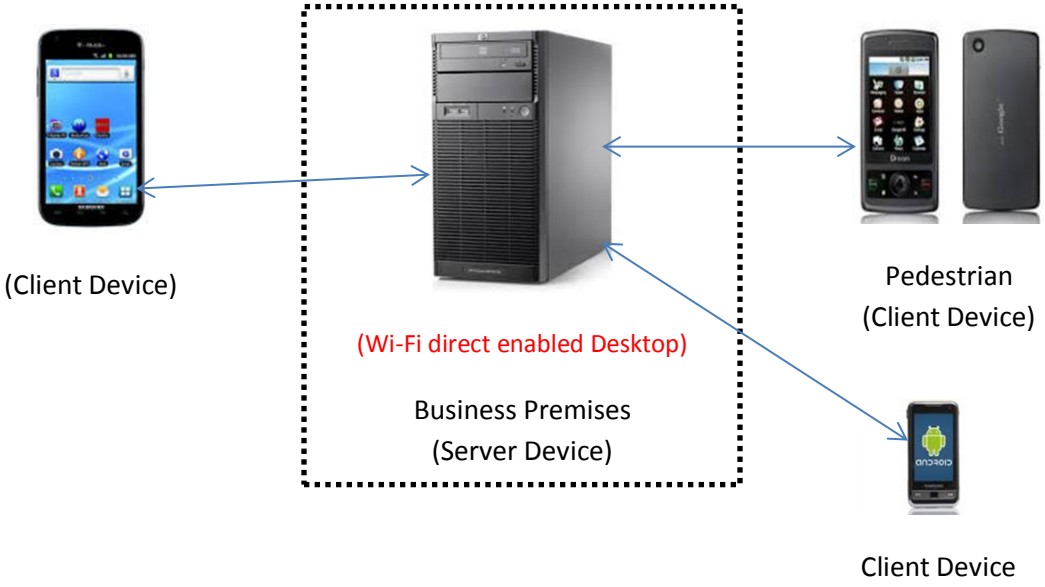


Figure 6.1 Practical environment setup design

7 References

1. Chris McTiernan. (2014) *B2B Advertising: Challenges, Opportunities*. [ONLINE] Available at:http://www.clickz.com/clickz/column/1693935/b2b-advertising-challenges-opportunities. [Accessed 01 July 14].
2. Wi-Fi Alliance. (2010) *Wi-Fi CERTIFIED Wi-Fi Direct™*. [online] October 2010. Available from: http://blog.broadcom.com/wp-content/uploads/2013/10/Wi-Fi-Direct-White-Paper.pdf . . [Accessed 1st July 2014].
3. Wikipedia. (2007) *Information Society*. [Online] Available from: http://en.wikipedia.org/wiki/Information_society . [Accessed 1st July 2014].
4. Mike, M. (2012) *GM developing Wi-Fi Direct driver assistance systems for wireless pedestrian and bicyclist detection*. [online] Available from: http://www.greencarcongress.com/2012/07/gmwifid-20120726.html. [Accessed 2nd July 2014].
5. Android developers. (2013) *Application Fundamentals*. [online] Available from: http://developer.android.com/guide/components/fundamentals.html . [Accessed 2nd July 2014].
6. Waterfall model. (2014) *The Waterfall Model*. [online] Available from: http://www.waterfall-model.com/ . [Accessed 7th December 2014].
7. SHELLY, G. ROSENBLATT, H. (2010) *System Analysis and Design*: GEX Publishing Services.

8 APPENDIX

8.1 Client Code

Client class code

```
public class Client extends AsyncTask<Void, Void, String> {
	private OutputStream outToServer = null;
	private InputStream inFromServer = null;
	private Socket clientSocket = null;
	private int tries = 0;
	private String serverIP = null;
	private int serverPort = 0;
	Handler myHandler = null;
	int nextParent;

public Client(String serverIP, int serverPort, Handler myHandler, int pid) {
		this.serverIP = serverIP;
		this.serverPort = serverPort;
		this.myHandler = myHandler;
		this.nextParent = pid;
	}
@Override
	protected String doInBackground(Void... arg0) {
		try {
			Log.d("CLT", "********* Here ******* Tos *********");
			Socket sock = new Socket(serverIP, serverPort);
			Log.d("CLT", "********* Here ******************");
			ObjectOutputStream oos = new ObjectOutputStream(
					sock.getOutputStream());

			Log.d("CLT", "* Starting Client *");
			oos.writeObject(String.valueOf(this.nextParent));
			ObjectInputStream ois = new
ObjectInputStream(sock.getInputStream());
			Log.d("CLT", "Input Stream Created");
			HashMap<String, String> inData = (HashMap) ois.readObject();
			Log.d("CLT", "Server Data Received");
			String[] columns = new String[] { "Descc", "_id" };

			MatrixCursor matrixCursor = new MatrixCursor(columns);
			// startManagingCursor(matrixCursor);
			for (HashMap.Entry<String, String> entry : inData.entrySet()) {
				System.out.println(entry.getKey() + " :: " +
entry.getValue());
				matrixCursor.addRow(new Object[] { entry.getValue(),
						entry.getKey() });
			}

			Log.d("CLT", "Receive Complete, rows: " +
matrixCursor.getCount());
			showMessage("", matrixCursor);
```

```
                        // close resources
                        ois.close();
                        oos.close();
                        sock.close();

                } catch (Exception e) {
                        Log.d("CLT", "Client not starting: " + e.getMessage());

                }

        }
}
```

8.2 Server Code

```
package com.example.serverd;

public class Server extends AsyncTask<Void, Void, String> {
        private ServerSocket serverSocket = null;
        private int serverPort = 8990;
        private Socket clientSocket = null;
        private OutputStream outToClient = null;
        private InputStream inFromClient = null;
@Override
        protected String doInBackground(Void... arg0) {
                // create serversocket
                try {
                        serverSocket = new ServerSocket(serverPort);
                        Log.d("SVR", "Server: Socket Created.");

                } catch (Exception e) {
                        Log.d("SVR", "Unable to create server socket: " +
e.getMessage());
                        return null;
                }
                while (true) {

                        try {
                                clientSocket = serverSocket.accept();
                                Log.d("SVR", "Server: Client Accepted");

                                ObjectInputStream in = new ObjectInputStream(
                                                clientSocket.getInputStream());
                                // convert ObjectInputStream object to String
                                String message = (String) in.readObject();
                                Log.d("SVR", String.valueOf(message));

                                DBHelper mydb = new DBHelper(this.cxt);

                                Log.d("SVR", "********** List Init *****");

                                // Cursor list = mydb.getAllAdvertss(-1);
```

```
                        HashMap<String, String> list =
mydb.loadAllAdvertss(message);

                        Log.d("SVR", "********** List Created *****");

                        ObjectOutputStream out = new ObjectOutputStream(
                                    clientSocket.getOutputStream());
                        Log.d("SVR", "# of rows: " + list.size());

                        out.writeObject(list);// "Server Reply: No Spicy"
                        Log.d("SVR", "Response Sent #################");

                        // close resources
                        in.close();
                        out.close();
                        clientSocket.close();

                  } catch (Exception e) {
                        Log.d("SVR", "Server Error: Accept failed. " +
e.getMessage());
                        return null;
                  }

      }

      protected void onPostExecute(String result) {
            Log.d("SVR", "Server: Finished.");
      }

      protected void onPreExecute() {
            Log.d("SVR", "Server: Started.");
      }
}
```

8.3 Broadcast receiver

```
package com.example.serverd;

import android.net.NetworkInfo;

import android.net.wifi.p2p.WifiP2pDevice;

import android.net.wifi.p2p.WifiP2pGroup;

import android.net.wifi.p2p.WifiP2pManager;

import android.net.wifi.p2p.WifiP2pManager.ActionListener;

import android.net.wifi.p2p.WifiP2pManager.Channel;

import android.net.wifi.p2p.WifiP2pManager.PeerListListener;

public class WDBroadCastReceiver extends BroadcastReceiver {
```

```
private WifiP2pManager manager;
private Channel channel;
private WiFiActivity activity;
Context cxt;
private boolean isWifiP2pEnabled = false;
public void setIsWifiP2pEnabled(boolean isWifiP2pEnabled) {
        this.isWifiP2pEnabled = isWifiP2pEnabled;
        return;
}
public WDBroadCastReceiver(WifiP2pManager manager, Channel channel,
                WiFiActivity activity) {

        super();
        this.manager = manager;
        this.channel = channel;
        this.activity = activity;
        this.cxt = this.activity;
        // this.manager.di
}
public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {
                // Verify  if Wi-Fi is enabled and generate appropriate notification
                int state = intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, -1);
                if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {
                        // Wifi Direct is enabled
                        activity.setIsWifiP2pEnabled(true);
                } else {
                        // Wi-Fi Direct is not enabled
                        activity.setIsWifiP2pEnabled(false);
                }
                return;
```

```
        } else if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {
                // Call WifiP2pManager.requestPeers() to get a list of current peers
                if (manager != null) {
                        manager.requestPeers(channel, (PeerListListener) activity

.getFragmentManager().findFragmentById(R.id.devicelist));
                        return;
                }
        } else if (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION
                        .equals(action)) {
                // Respond to new connection or disconnections
                if (manager == null) {
                        return;
                }
                NetworkInfo networkInfo = (NetworkInfo) intent
                                .getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);
                if (networkInfo.isConnected()) { //connection succeeded
                        if (!MainActivity.serverRunning) {
                                // start server
                                Server server = new Server(cxt);
                                server.execute();
                                MainActivity.serverRunning = true;
                        } else {  }
                } else {  //we have hit disconnect           }
                return;
        } else if (WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION
                        .equals(action)) {
                // Respond to this device's wifi state changing
                activity.updateThisDevice((WifiP2pDevice) intent
.getParcelableExtra(WifiP2pManager.EXTRA_WIFI_P2P_DEVICE));
                return;
```

```
                }
        }
}
```