VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DEVELOPMENT OF MOBILE PEER-TO-PEER APPLICATION FOR CUSTOMER-TO-CUSTOMER E-COMMERCE

by
Pham Le Trung

A thesis submitted to the School of Computer Science and Engineering in partial fulfilment of the requirements for the degree of
Bachelor of Computer Science

Ho Chi Minh City, Vietnam
2018

DEVELOPMENT OF MOBILE PEER-TO-PEER APPLICATION FOR CUSTOMER-TO-CUSTOMER E-COMMERCE

APPROVED BY:

____

Tran Manh Ha, Ph.D., Chair

THESIS COMMITTEE
(Whichever applies)

AKNOWLEDGEMENTS

The thesis I achieve today would not have been possible without the help from many people.

Firstly, I would like to thank my thesis advisor, PhD. Tran Manh Ha of the School of Computer Science and Engineering at International University. He has been a generous motivation for me and for the entire process. His idea and concept guide me to the correct direction whenever I needed it the most.

I also want to thank our department School of Computer Science and Engineering for giving me the fundamental knowledge in Computer Science and the opportunity to work on this topic.

Finally, I must show my gratitude to my family, friends and a special K14 girl from BAFN Department who always help and encourage me whenever I was challenged by the thesis from one time to another. This accomplishment would not have been possible without you. Your wonderful and continuous supports are greater than what I have obtained today. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In early 2009, a Peer-to-Peer car transportation and delivery services so called Uber, has created a breakthrough not only in mobile technology but also in how companies run their business. People invented the term “Uberisation” to prefer economic sharing systems utilized by P2P transactions between clients and providers of services. In short, Uber alone does not possess any cars, but can deliver car transportation services to a vast number of customers by applying P2P. [1]

Followed the P2P inspiration, this thesis attempts to explore further into the capability of P2P and Uberisation in other aspects rather than just transportation services. In fact, we will observe how we can bring P2P to Customer-to-Customer (C2C) E-commerce in Mobile Platform. The final implementation and demonstration can be served as a prototype for applying P2P into C2C E-commerce.

The prototype applies Android Technology, Gnutella Protocol, Web API 2 with Entity Framework. The database for storing information will be SQLite (for mobile device) and Microsoft SQL (for Server Web API).

CHAPTER 1

INTRODUCTION

1.1 Background

Peer-to-peer and *Client-Server* are two major networking architecture that are widely used nowadays. *Peer-to-peer* emphasizes the strong equality between most participants in the whole network. On the other hand, *Client-Server* focuses more on the clear division between what a *Server* and a *Client* can do in the network.

Since the equality nature in Peer-to-peer network, it is highly recommended that a participant joining the network must meet certain requirements to work efficiently as a whole. That is why, when P2P first popularized by Napster, most P2P applications at the time are for Computers and Workstations which can provide high and stable connection with sufficient power to function in the network.

Fortunately, the emergence of *IOS* and *Android OS* in 2007 and 2008 respectively, has significantly push the developments of one type of pocketable device called smartphone. Now, smartphones are much powerful than phones comparing in the last 10 years. And of course, they can operate a decent aspects of P2P network architecture.

Android with the open source and freely customization and sharing systems, provides a vast diversity in mobile devices with reasonable cost. Therefore, it is suitable for applying P2P technology to Android device in this thesis.

1.2 Problem Statement

Amazon, E-bay, Walmart and many others are iconic for online shopping services, “E-Commerce” for short. In such system, buyers look for their favorite goods online and add to their “carts”. After transaction, the good are delivered after a few days.

The problem for such system like Amazon, is that they first contact providers where goods provided as wholesale, some are retails, and then they announce to their buyer customers that they are selling such items. And when the buyers buy any items displayed on the web, they rely heavily on the web’s transportation systems. This is somehow considered as server-client model where everything is relied on server like Amazon and the client is the buyers and the goods’ providers.

Fortunately, there is a solution for this with the advance of P2P technology and smartphone. With this technology, users are now capable of finding and directly contacting the goods’ providers or their buyers by portable devices almost instantly, which significantly reduce the delivery time and system maintenance cost.

1.3 Scopes and Objectives

The project emphasizes majorly on Mobile P2P technology and its application in E-Commerce Industry. Specifically, the main objective is to develop an Android Application, which allows users to buy and sell their books to other users in the system. The core of the application including positioning users are implemented with the help of Google Connection API and Google Map Platform. There is a small server which help rating among users (after all, there should be a standard and validation for every evaluation).

Below is the list of primary objectives of the thesis:

- Study the mobile technologies to develop an e-commerce system where every individual customer can be both a buyer and seller with direct contact to their opposite target (seller and buyer).
- Strengthen analyzing skill in choosing suitable peer-to-peer architecture.
- Have a good understanding about the mobile application development.
- Apply peer-to-peer to mobile (Android) platform and reduce the workload to server.

1.4 Conventions and Limitation

There are some conventions needed to be clarified before the project come to presentation.

The application is supported as an implementation for the thesis topic. It must contain important functions including finding, connecting and transferring data to nearby devices, major data are storing on local database. There is a small server which helps with the rating function. However, the mobile application will not guarantee to have outstanding user interface and animation since it is not the main objective. Moreover, the connectivity and real-time results cannot be completely satisfied in some occasions (relating to Device operate the applications).

1.5 Structures of Thesis

The Document consists of five sections: (1) Introduction, (2) Literature Review, (3) Method and Implementation, (4) Final result and (5) Conclusions and Future Work.

The first section, *Introduction*, demonstrate the research topic and final application. Any background, objectives, and important assumptions are listed in this part. In addition, overall structure of the whole thesis document is also presented.

Secondly, *Literature Review* sections includes the theoretical knowledge behind the applied technology which acquired from Pre-Thesis Process to Thesis, the P2P and Mobile Application. It includes a general idea about Peer-to-Peer, the in-depth comparison between P2P and Server-Client Architecture, the emergence of Smartphone Devices and Mobile Application Development. Such fundamentals provide strong tools for understanding and developing final application.

Method and Implementation section describes the processes of developing the final product, divided into four different stages: (1) Planning for Requirements, (2) Designing Database and Architecture, (3) Set up Mobile Activities, (4) Set up Server Activities. Specifically, the section also provides the frameworks, tool, and techniques related to those stages.

Next, the *Final Results* section displays the current progress through images of product, database and code snippets.

Finally, the *Conclusions and Future Work* genializes and self-evaluating the thesis and considering the future possibility of the work.

CHAPTER 2

LITERATURE REVIEW

2.1 Peer-to-Peer Network

2.1.1 Overview

Peer-to-peer (P2P) network defines as a distributed architecture where participants joining the network share nearly the same tasks, workloads, resources. [2] In the purest definition, there is no server in the system, which makes P2P an architecture with strong decentralized manner. However, when we compare different types of P2P networks and applications, the pure definition may not hold the absolute anymore for Hybrid P2P and Super Peer components. [3]

Since the first appearance of Napster – a P2P music sharing services in 1999, (P2P) has become a controversial IT-related topic. Many experts argue differently based on the undeniable advantages but followed by decent concerns due to the flexible flow of resource within the P2P systems. Many problems related to legalization in music industry and violating intellectual properties. Many others believe that there is not much new to explore in P2P. [2]

However, the unexpected technology have us amazed by the strong development of handheld devices, especially the pocket-size telecommunication device called the smartphone. Such improvement brought the Peer-to-peer out of the clumsy computer to the pocket of our jeans. In 2009, Uber - a P2P car transportation and delivery services on mobile applications, was not only changing the trend of technology but also inventing new terminology of sharing economy, so called “Uberization”, refers economic systems utilized by P2P transactions between clients and providers of services. Resulting in business entrants that are quicker, lower cost and more efficient than existing traditional methods. [4]

2.1.2 Key Factors

There are six major factors often brought down to the table whenever choosing P2P as distributed system, which are: [2]

- *Low Infrastructure Cost* – When system’s maintenance cost becomes too expensive, P2P architecture can help spread the cost over all the peers.
- *Resource aggregation and interoperability* – By combining resource from several nodes, the P2P architecture can perform tough computational functions.
- *Improved Scalability/Reliability* – P2P architecture can be scaled due to the nature of decentralized manner. Additionally, reliability can be met for P2P by using the characteristics of redundancy of resources among peers.
- *Anonymity/Privacy* – By deploying P2P structure in which activities are performed locally, users are having a greater control over their database and resources.
- *Dynamism* – Resources, such as “users – computer” nodes have the privilege of entering and leaving the system continuously.

- *Enabling Ad-hoc communication and collaboration* – P2P systems normally do not depend on established infrastructure – they build on their own, based on the peers involving in the community.

Every Computing System, Architecture happens to support the applications that fulfill the needs of users. It is important to mention, when applying certain P2P architecture, hard to gain every objective without trade-off.

2.1.3 Classification of Peer-to-Peer Network

The operation of any peer-to-peer content distribution system relies on a network of peer computers (nodes), and connections (edges) between them. This network is formed on top of—and independently from—the underlying physical computer (typically IP) network and is thus referred to as an "overlay" network. Overlay networks can be distinguished in terms of their centralization and structure manner by the Table 1. [3]

Network Centralization	
Purely Decentralized Architectures (No Centralization)	All nodes in the network perform the same tasks, and there is no central coordination of their activities and resources.
Partially Centralized Architectures.	The basis is the same as with purely decentralized systems. Some of the nodes, however, are assumed to be more important role, acting as local central indexes for files shared by local peers. These super nodes do not constitute single points of failure for a peer-to-peer network, since they are dynamically assigned and, if they fail, the network will automatically replace them with others.
Hybrid Decentralized Architectures	There is a centralized server coordinates the interaction between peers by storing directories of metadata, describing the shared

	files stored by the peers. Although the end-to-end interaction may occur between two peers, the central servers facilitate this interaction by performing the lookups and identifying the nodes storing the files.
Network Structure	
Unstructured	The placement of content (files) is completely unrelated to the overlay topology. In an unstructured network, content typically needs to be located.
Structured	These have emerged mainly to address the scalability issues that unstructured systems were originally faced with. In structured networks, the overlay topology is tightly controlled and files (or pointers to them) are placed at precisely specified locations.
Loosely Structured	A category of networks that are in between structured and unstructured. Although the location of content is not completely specified, it is affected by routing hints

Table 1 - Classification of Peer-to-Peer Network on Decentralization and Structure

Each type of P2P Network brings different advantages along with its own drawbacks. Table 2 lists some iconic examples for each type of P2P Network. [3]

	Centralization		
	Hybrid	Partial	None
Unstructured	Napster, Publius	Kazaa, Morpheus, Gnutella, Edutella	Gnutella, FreeHaven
Structured			Chord, CAN, Tapestry, Pastry
Loosely Structured			OceanStore, Mnemosyne, Scan, PAST, Kademlia, Tarzan

Table 2 - Example for each type of P2P Network

We will focus more detail on *Gnutella*, a *Pure Decentralize Unstructured* System. Because it clearly demonstrates the characteristics of P2P architecture. In such system, there is no central coordination of the activities in the network and users connect to their peers directly through a software that function both as a client and a server (as show known as “servents”). [3] To locate a file in such unstructured system (in consequences of “no centralize”), nondeterministic searches are the only way since the nodes have no idea where the files needed may lie.

The original Gnutella architecture uses a flooding (or broadcast) mechanism to distribute messages. There are four types of messages to consider in such system. [3]

- *Ping* – A request for a certain host to announce itself.
- *Pong* – Reply to a Ping. It contains IP and port of the responding host and number and size of the sharing file.
- *Query* – A search requests. It contains a search string and the minimum speed requirements of the responding host.
- *Query Hit* – Reply to Query message. It contains the IP, port, and speed of the responding host, the number of matching files found, and their indexed result set.

This is a pure P2P model in which no advertisement of shared resources occurs. Instead, each request from a peer is passed directly to connected peers, which themselves flood their peers, until the request is answered, or a maximum number of flooding steps occur (by applying time-to-live (TTL) to the header of the message).

Figure 1 explains how flooding algorithm work in Gnutella System. Node (2) want to query an item to download. It first connects to nodes (1), (5), (3). The message is spread until they found node (4) that keeps the item. After return the result to node (2), it can contact node (4) for downloading item.

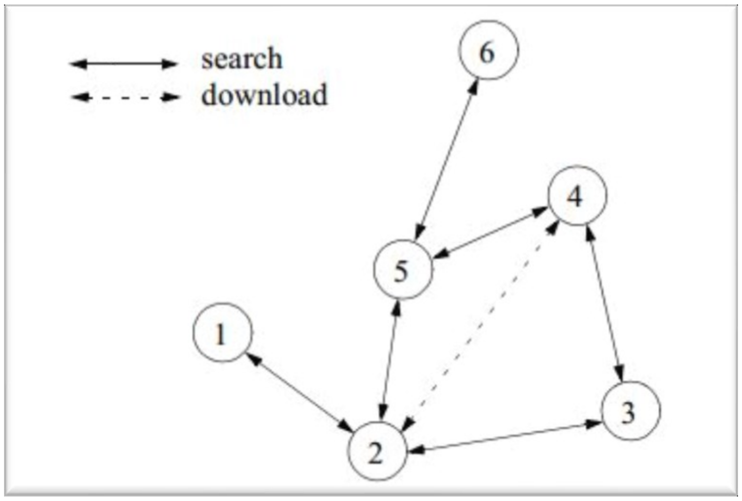


Figure 1. Flooded Requests Algorithm

2.2 Peer-to-peer versus Client-Server

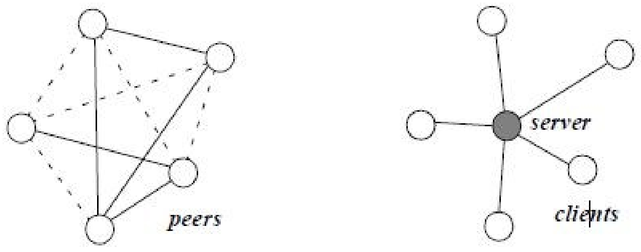


Figure 2. Simplified View of Peer-to-Peer vs Client-Server

2.2.1 Client-Server Architecture

Perhaps this is the most used computing model for today applications. In this architecture, a server computer usually a strong computer that has vast resources and responds to request for resources and data from client computers. Clients cannot see each other in this architecture, they only see the Server and communicate through it. Client computers initiate requests for resources or data from server computer. For example, in Web Browsing, there is always a Web Server on the Internet keeping all the contents associated with a Website. When a page is requested, the Web Server sends the page and its associated files to the requesting clients.

2.2.2 The Comparison

Peer-to-peer networking has the following advantages over client-server networking:

- Content and resources can be shared from both the center and the edge of the network. In client/server networking, content and resources are typically shared from only the center of the network.
- Easily scaled and more reliable than a single server. A single server is subject to a single point of failure or can be a bottleneck in times of high network utilization.
- A network of peers can share its processor, consolidating computing resources for distributed computing tasks, rather than relying on a single computer, such as a supercomputer.

- Allowing peers, more autonomous control over their data and resources.

The biggest advantage of Peer-to-Peer is to lessen or eliminate the dependency on Centralized Elements (contradict to Client-Server which focuses mainly on Centralized Resources). Thus, this architecture raising some problem from its own nature: [5]

- *Spurious content combining with poor connection.* This due to the flexibility nature of P2P; hence, the quality of connection between peers fluctuate unpredictably. Moreover, a resource on different peers are not completely verified or standardized for authenticity purposes.
- *Free-ride problem* has been an interesting topic when discussing P2P, any participant in P2P Community can steal quality links and resource from other peers by replicating the information and offering it elsewhere.
- *Cracker heaven* is believed a security concerns when involving in P2P Community [6]. P2P networks allow users to share their resources publicly. This makes the average users, who do not usually aware of security issues, become vulnerable target for malicious one, who can pirate information from systems, infect them with dangerous viruses.

2.2.3 The Blurred Line

Recognizing some serious problem that pure P2P without centralized left behind, Hybrid Decentralized Architectures as well as Super Peers definition come out with high hope of lessen the disadvantages from the original idea. By providing small administrative elements into the Community, the separate line between Peer-to-peer and Client-Server become less distinct but more complex in many dimensions. [2]

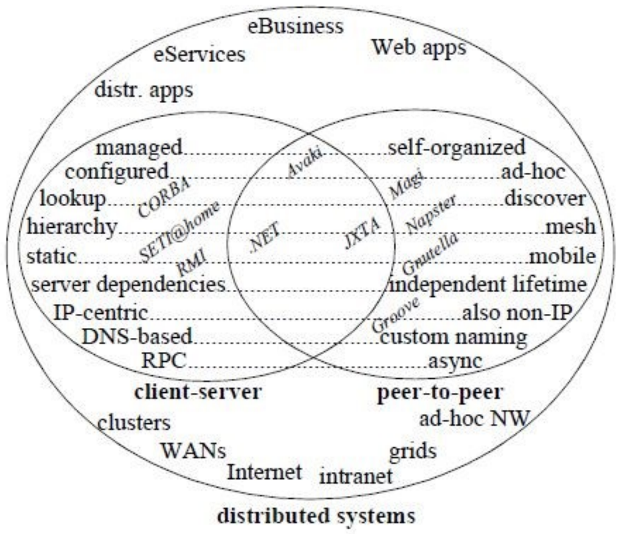


Figure 3. A Complicated View from P2P and Client-Server

Figure 3 indicates the different aspects between client-server and peer-to-peer distributed systems along with the representative applications. The picture shows that there is no clear border between a client-server and a P2P model. Both models can be built on a spectrum of levels of characteristics (e.g., manageability, configurability), functionality (e.g., lookup versus discovery), organizations (e.g., hierarchy versus mesh), components (e.g., DNS), and protocols (e.g., IP), etc. Furthermore, one model can be built on top of the other or parts of the components can be realized in one or the other model. [2]

2.3 Bringing Mobile to the Peer

2.3.1 Why Mobile Technology

With the emergence and rapid growth of mobile and networking technology, customers prefer using smartphone for the Net's services. [6] Easy to notice the outstanding convenience that smartphone bring in this digital era by its enormous physical storage, faster processing unit with a vast range of functionalities that the user can customize on the phone.

Therefore, bringing the Peer-to-Peer to Handheld Devices is no longer an unachievable thing. As mentioned, Uber Company has successfully brought P2P on Mobile Application and gain much recognition. However, there are many obstacles that one should concern when applying P2P to the mobile application.

2.3.2 The difficulties

Firstly, mobile elements are less power relating to stationary elements like Desktop PC or a Workstation Server Computer for a given cost. This difference highlights the drawbacks in computational resources such as processor speed, memory size, storage and information share even though mobiles will improve in many possibilities in the future. [7] Additionally, mobile connectivity is more unpredictably on both performance and reliability since the bandwidth of wireless connection change relating to physical location of the device. And finally, mobile elements rely heavily on a finite battery life. This may affect the routing during the transition of data in the P2P community.

Fortunately, with the remarkable developments in mobile and network technology, new solutions for Mobile Peer-to-Peer (MP2P) technology has been proposed to reduce the negative impacts from handheld devices nature.

2.4 Mobile Application Development.

Combining what we know about networking system, there are four type of application architecture that we can apply to our Mobile Device, including Standalone, Thin Client-Server, Fat/Thich Client-Server and Peer-to-Peer. Each kind of Application has different strengths and weaknesses of their own as compared in Table 3.

	Standalone	**Thin Client-Server**	**Fat/Thick Client-Server**	**Peer-to-Peer**
Overview	Everything is stored on device, there is no need of Internet Connection to run the application.	Mobile Application relies heavily on centralized server. Mobile device get data generated by Server and display on screen.	Mobile Application requires Internet to connect to Server for purchasing or exchange information, but there are some tasks that the app can process offline.	Mobile Application requires Internet to connect to other Mobile for exchanging information. Every important data is stored on the mobiles.
Benefits	Can work anytime without the need of Internet.	Reduce the workload on mobile devices significantly as Server handles all the hard work.	Reduce partially the workload on Server-side as some tasks and resources are stored and processed in mobile locally.	Reduce (or Eliminate) the cost for Server Maintenance.
Drawback	Relies heavily on mobile devices hardware resource. No connectivity for sharing data.	Relies heavily on Server, therefore, raising the cost for server maintenance exchange for light workload at client side.	Mobile Device must spend more resource and energy to conduct and storing tasks on devices exchange for Server workload.	Relies heavily on peers' hardware.

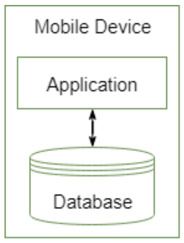
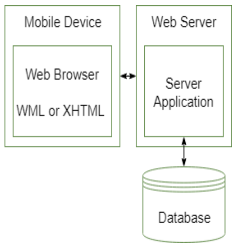
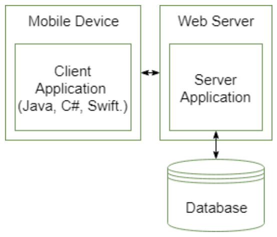
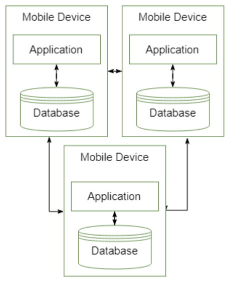
Model		 Mobile Device Web Browser WML or XHTML Web Server Server Application Database	 Mobile Device Client Application (Java, C#, Swift.) Web Server Server Application Database	 Mobile Device Application Database Mobile Device Application Database Mobile Device Application Database
Example	Mobile Calculator, Camera	Newspaper, Weather Forecast on Mobile.	Ticket Booking Application, Facebook Mobile	Uber, Skype.

Table 3 - Comparison of Mobile Application Architecture

2.5 Real-time Data, Computing.

Many peer-to-peer applications relate closely to real time computing, especially in communications. Skype and Uber are two P2P applications that apply real-time processing into the application. It is useful to be aware of real-time computing when developing a P2P application.

2.5.1 Introduction of Real-time.

In computer science, real-time computing (RTC), or reactive computing describes systems that response in a predictable *real-time manner* (short time period, providing near-instantaneous output). Real-time programs must guarantee response within specified time constraints, often referred to as *deadlines* [8]. And the responses are often assumed to be in milliseconds, and sometimes microseconds.

2.5.2 Judging a Real-time system.

There are three major components that comprise a real-time system. Firstly, *time* is the most important factor to manage in real-time systems. This is relating to deadline of the task. Every task must be given and scheduled to be completed before the deadlines. The correctness of the processes from computation depends not only the logical but also on the time at which the results are produces. Second factor is reliability, since the failure of a real-time system could cause an economical lost or in some scenario, could result in loss of human lives (in ATM systems, automatic car-breaking systems, flight systems, etc.). Third and final key factor is the environments of real-time system (PC, mobile, or a chip on a car). [4]

2.5.3 Classification of a Real-time system.

As mentioned, a system is said to be real-time if the correctness of an operation depends not only its logical output, but also the time in which it is performed. [9] Real-time systems, as well as their deadlines, are classified by the consequence of missing a deadline:

- *Hard* - missing a deadline cause a catastrophic system failure.
- *Firm* - missing a deadline occurs in a small frequent but degrade the system hardly. The usefulness of a result is zero after missing its deadline.
- *Soft* - the usefulness of the output degrades after its deadline, therefore lower the system's quality of service

2.6 Peer-to-Peer changing the Ecommerce Picture

2.6.1 The inspiration from Taxi Service

Taxi service is highly demands almost anywhere. Especially when the population keeps growing and there is less space for parking lots, using taxi is far more convenient than any other means of transportation.

Figure 4 illustrates the picture of how traditional taxi service operates. In a traditional fashion of taxi service, a taxi company includes a call centers with many telephone lines, a very large number of taxis in different types. A user contacts the call center through a hotline to book a vehicle, then the call center requests a taxi driver come to user's location. [1] After lifting to the destination by the taxi driver, user pay the bill to taxi vehicle. Such model highlights the centralized manner of the call center as a bridge between taxi driver and user who use the service. This contains many disadvantages, for instance, the users have no clue about the waiting time for the demanding vehicle to come to their spot (They may ask the call center, but the precision is poor); Central point of failure is another vital drawback in such model. [1] When the call center is accidentally shutdown, there will be no contact bridge between user and taxi driver, lost in business is unavoidable.

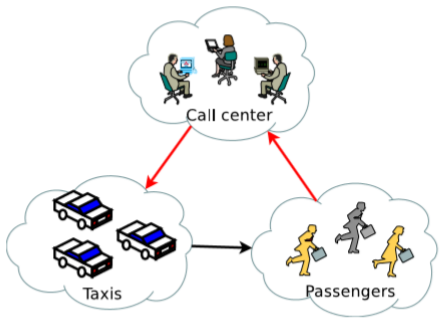


Figure 4. Traditional Taxi Model

Realizing the difficulties from traditional taxi model, Uber Company has successfully taken advantage of technology and sharing economy model. [1] It provides a digital world to connect users for requesting taxi service. Uber only hire drivers; the company possesses no taxi vehicles. In this model, as shown in Figure 5, a user requests a vehicle through an application provided by Uber. The system then seeks the most suitable offer for user from collections of real time data relating to vehicles and their locations. [1] A driver also receives a request from the system and will directly contact his customer.

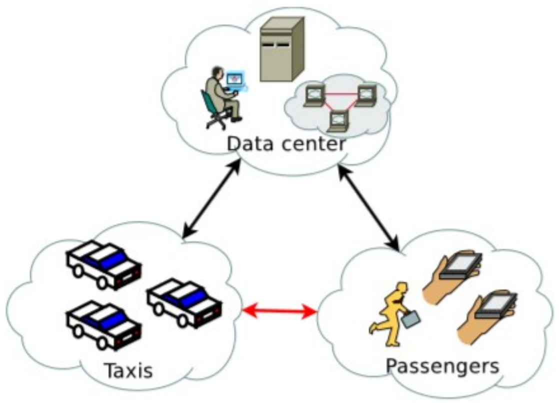


Figure 5. Uber taxi service model

The biggest advantage of this model is the transparency of detailed information shown to both users and drivers. They both know the locations, price and route. Moreover, the user can also rate the service by leaving comment and giving score to the driver. The Uber's model is a kind of P2P architecture model that has users and vehicles play a role of peers and the data centers only for registration and management purposes. [1]

2.6.2 C2C E-Commerce with P2P

C2C, or customer-to-customer, is a e commerce business model that operates the transaction of products or services between customers. It is one of four categories of e-commerce, along with B2B (business to business), C2B (customer to business) and B2C (business to customer) [10]. An example in this type of E-Commerce is auction like E-bay where millions of users publish their items for selling, while other customers auction for the highest bid to get the item. Generally, the model highlights the strong relationship between individual Customer to other Customers rather than any Enterprise Business.

In C2C E-commerce, there is a web platform serving as a marketplace for the buyers and suppliers. The suppliers will post their currently selling item to the E-market, while the buyers access the marketplace, looking for the best item they want to buy and confirm the transaction. In such system, the server e-market acts an important role as a connecting bridge between the buyers and suppliers. Therefore, the server must be efficient in both bandwidth and capacities to keep the service running. Storing and Delivering method is another factor that effects this kind of business model. For the companies that possess warehouse storage, it is important to update the warehouse storage to meet the customers' orders based on suppliers' items. On the other hand, some E-commerce companies do not own any warehouse, will use third party shipping service to bring the item from the suppliers to the consumers, therefore depends heavily on the shipping services. These models, as Figure 6 illustrates, represent server-client architecture as the e-commerce's webpage, warehouse is the server, the consumers and suppliers are clients in the system. Such system contains some serious disadvantages similarly to the traditional taxi model discussed in Section 2.6.1, including central point of failure at the server (the online marketplace). Moreover, systems upgrade and maintenance cost, combined with shipping fee is sometimes too high for the company.

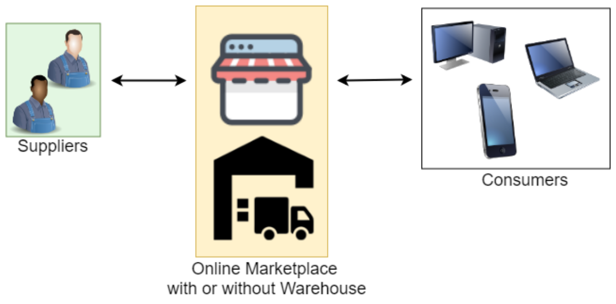


Figure 6. Traditional C2C E-Commerce Model

In the same fashion as Uber, bringing some peers' manners to the C2C E-Commerce system could lessen the disadvantage from the server-client architecture. When buyers and suppliers become peers in the system, they could directly contact and exchange resource to each other and not through any intermediate layers. By this advantage, the cost for the shipping method is significantly reduced since our system does not interfere the delivering process as shown in Figure 7.

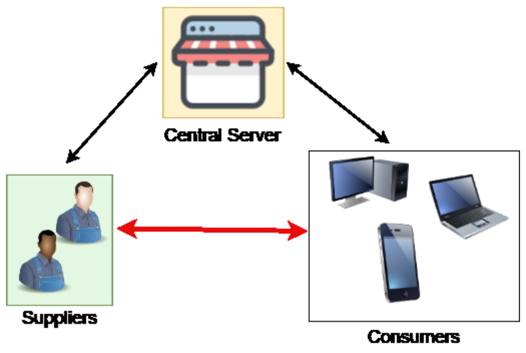


Figure 7. Proposing P2P Model for C2C E-Commerce

However, it is the application's responsibility to help Consumers decide the most benefit Suppliers and vice versa. For instance, a Consumer may use the application to scan for the nearby Suppliers who public their items for selling, rating and pricing could be some interesting criteria when looking for peers. Additionally, the central server must also provide an alternate solution for the users when they are not able to scan for nearby suppliers or buyers by themselves.

CHAPTER 3

METHOD AND IMPLEMENTATION

The final product is an Android Application that helps users exchanges items (books in specific) to others with as less interact with servers as possible. Google Connection API and Gnutella Protocol will be the core of the application which helps locating and transferring information between devices. The reason while Android Application is chosen to be the prototype for this project, is because the big guy Android lied under a wide range of mobile devices. On the other hand, Google and Android Documents are very well supported by its big community of open-source.

There are four major processes that need to be achieved for the application to be complete: (1) Initialize Requirements, (2) Designing Database and Architecture, (3) Setting up Mobile Activities and (4) Setting up Server Activities.

3.1 Planning for Requirements

3.1.1 Use Case Diagram

Use Case Diagram specifies the relationship between Actors (Users of the System) and System's Function. This diagram is generated based on initial business requirements, functional requirements and non-functional requirements. On business requirements, we have:

- This application allows uses to trade books directly with other users using a P2P architecture.

- This application only uses a server for rating functions.
- Users joining the network don’t need to register through a server.
- The application provides functionality for users to join the network/advertise and retrieve books/communicate with the book's owner for trading.
- There are two types of users: customers and operators (we assume operator as admin)
- (Optional) Serving several geographically dispersed users.
- (Optional) Different technologies/platforms used by different users.

The business requirements give us an abstract view of how the implementation system works. However, to make it more “programmable”, we still need a list of detailed functional requirements:

- As a User, I should be able to initialize my personal information when first used the application (My Name and My Phone Number).
- As a User, I should be able to view my personal information (My Name and My Phone Number).
- As a User, I should be able to change my personal information (My Name and My Phone Number).
- As a User, I should be able to view my Trading History.
- As a Buyer User, I should be able to create a Wishlist of Books (Books that I want to Buy).

- As a Buyer User, I should be able to notice if anyone near me sells the book that I declared in my Wishlist.
- As a Buyer User, I should be able to get personal information (including rating) of the Seller User who is selling the book that I declared in my Wishlist.
- As a Buyer User, I should be able to message my Seller User when I found him.
- As a Buyer User, I should be able to rate the Seller User who sold me the book.
- As a Seller User, I should be able to create an Inventory of my Selling Books (Books that I want to Sell).
- As a Seller User, I should be able to notice if anyone near me wishes to buy the book that I declared in my Inventory.
- As a Seller User, I should be able to get personal information (including rating) of the Buyer User who wants to buy my book.
- As a Seller User, I should be able to message my Buyer User after I found him.
- As a Seller User, I should be able to rate the Buyer User who bought my book.

Along with the functional requirements is the non-functional requirements, such requirements help us evaluate the operation performance of the application:

- The Server should be lightweight with as less logical operations as possible.
- The Server should be able to compromise multiple queries from different users at a time.
- The Server should be available/online for most of the time.
- The Application should work without the need of internet connection (except for Rating, Finding other Users).
- The Application should be able to locate at most 4 different users that meet the requirement.
- The Application should be able to update location of any one specific user in 5 seconds interval.
- The Application should have enjoyable User Interface.
- The Application should have self-explanatory User Interface.
- The cost for system maintenance should be low.
- The downtime for System Maintenance should be quick in approximately an hour.

From business, functional and non-functional requirements, we can design a use case diagram demonstrating different ways of how users interact with the system (Figure 8). A use case diagram consists of types of users, system's components and use cases. System's components including the application that installed directly on customers' device where core database is stored, and a server to keep general users' rating.

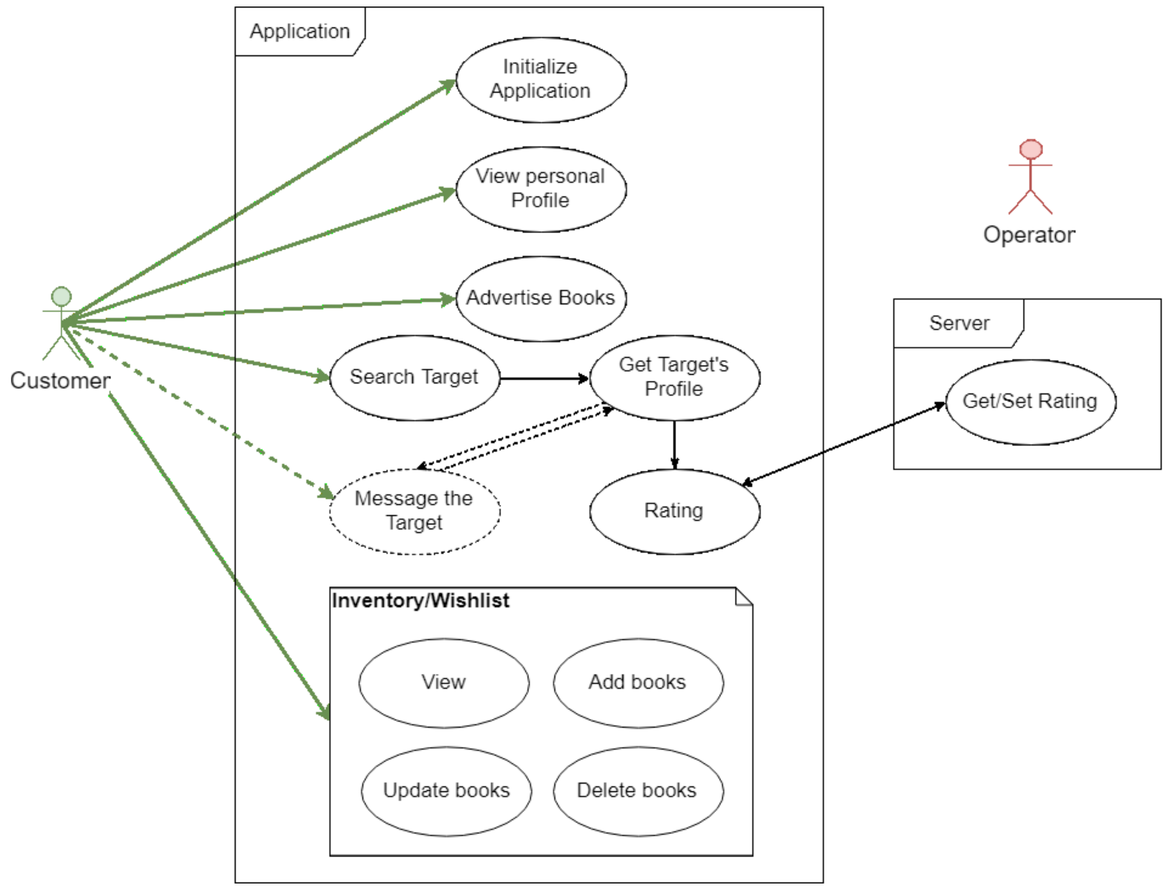


Figure 8. Use Case Diagram

3.1.2 Functional Description (from Use Case Diagram)

This section provides details functional description from Use Case Diagram:

- *Initialize application:* User should be able to initialize personal information from introduction page.
- *View personal profile:* User should be able to view their personal information stored on phone.
- *View (Inventory):* User should be able to view their inventory storing their want to buy/sell books.

- *Update books:* Users should be able to update the list of books that they want to sell/buy.
- *Delete books:* Users should be able to remove books from the inventory that they want to sell/buy.
- *Advertise books:* Users should be able to public the books that they demands for selling.
- *Search for sellers/buyers:* User should be able to search for sellers/buyers that meet his need.
- *Get buyer/sellers' information:* After a user found a buyer/seller that meets his needs, he should be able to find the buyer/seller 's information, including their location.
- *Message the Target:* Both buyer and seller should be able to chat to each other after discovering. (Optional – By the time the Report was conducted, the Message function is not yet developed)
- *Rating:* User should be able to rate a buyer/seller if they want to (based on five-star-benchmark).

3.2 System Architecture

System Architecture Diagram is a formal way to describe how components inside the systems interact with each other as well as the structure of the system.

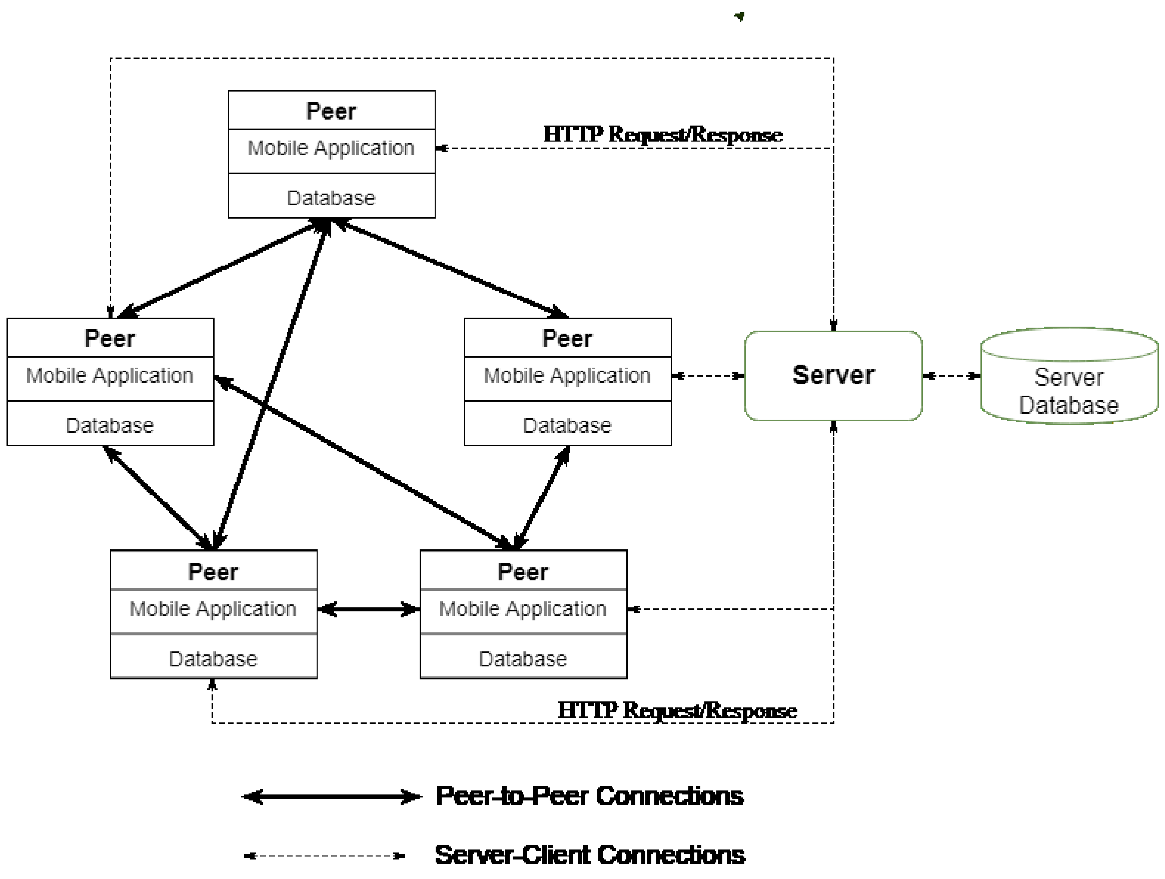


Figure 9. System Architecture

Figure 9 shows that our system is compound of different Mobile Peers connecting and transfer data in an unstructured manner. As mentioned, the system applies pure peer-to-peer technology, therefor most functions will occupy between peers, in this case, mobile devices. Before the ping pong and query resource happen, one peer must be able to implement Nearby Connection API to scan and collect surrounding devices' IP. Then there comes a time when peer need to query for certain resources, it will use collection of detected IP combining with Gnutella Protocol to obtain the needed information. By this methodology, almost no server is required for peers' connectivity. However, under circumstances, when mobile device connects to a WI-FI router, the device's IP will be a private one, which cannot be seen for other devices that do not connect to the same network. For such cases, a web service is a must to retrieve the device public IP. We will use an API from ipify.org server to obtain the public IP if needed.

Additionally, we still need one server to guarantee the objectives of some types of information. In this scenario, ratings of peers should be stored on a central database and there must be a server to query those ratings. When a peer wants to know the target's rating, it will send a HTTP GET request to a server with the target's phone, returning from the server is a message with the rating of the target. After the transaction process complete, happily or not, peers will also want to submit some rating. In this case, peer will send a HTTP POST request including target's phone and rating. The server after receiving the HTTP POST request, will update the database and return a successful HTTP message as acknowledgement.

However, for each peer, there is both a client manner and server manner (that is why sometimes peers are called "servants"). As a client, peer can send ping and pong message and view results set. On the other hand, peer also has the abilities to receive queries, looking in the local database and respond with suitable results.

3.3 Designing Database

3.3.1 Database

Database is where we keep the all the information of the books, user information and the transaction history when users use the application. Entity Relational Diagram

(ERD) presents how data is organized inside certain databases with tables, fields and their relationships.

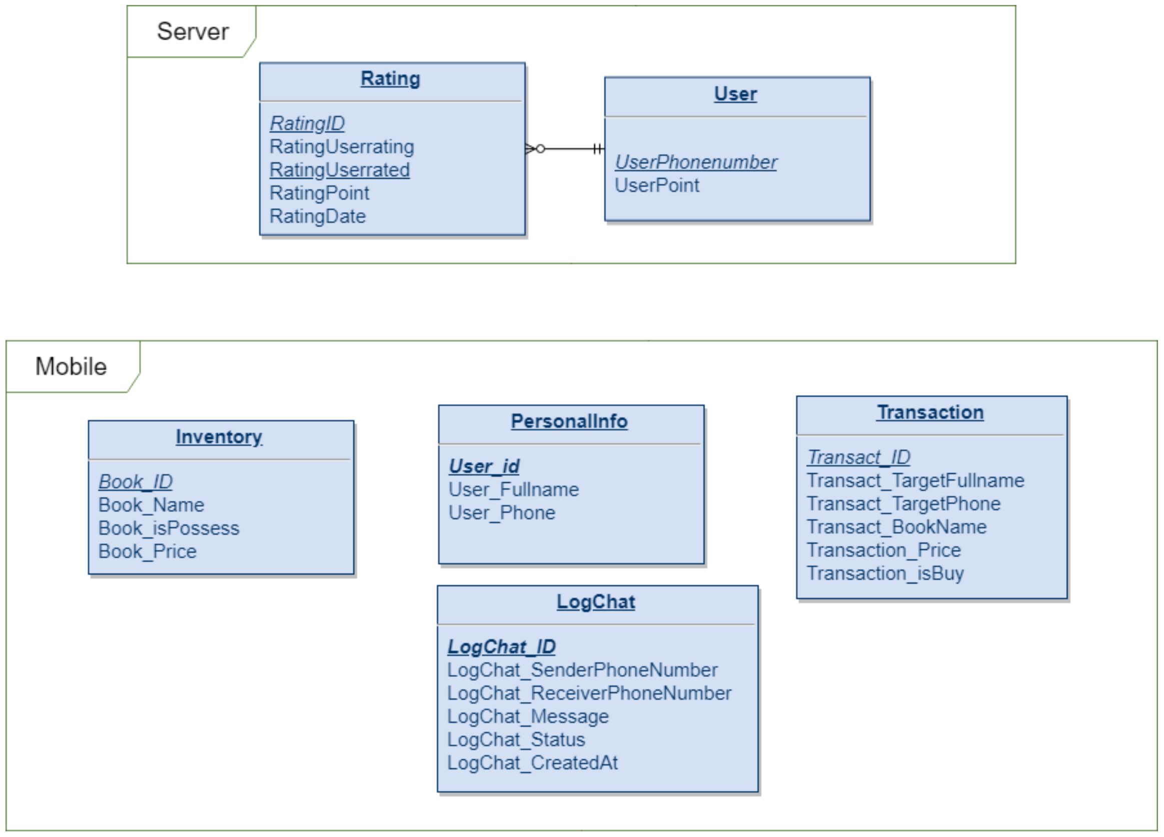


Figure 10. Entity Relational Database

Figures 10 implies there are two databases for two different types of components in our systems, the Peers (the Mobile) and the Server. The system mainly uses P2P Technology, therefore most important and core data will be stored directly on users' mobile devices. In this case, Personal Information, Transaction History, Log Chat and Possessing Item and Nearby Peers are such information that is stored on the Peer. User will have full privilege on their database and information.

We mentioned earlier in section 3.2 is that, despite applying the P2P Technology, the system still needs a Server for objectives and centralized nature of some type of data, specifically, the rating of peers in business manner. One user cannot rate himself and announce that to his business partner. Using a Centralized Server, we can guarantee that rating points is true and evaluated from different users using the application.

3.4 Setting up Android Application

3.4.1 Activities Flow Diagram

The Activities Flow Diagram (Figure 11) shows how mobile device displays application layouts in sequence back and forth when users interact with the application.

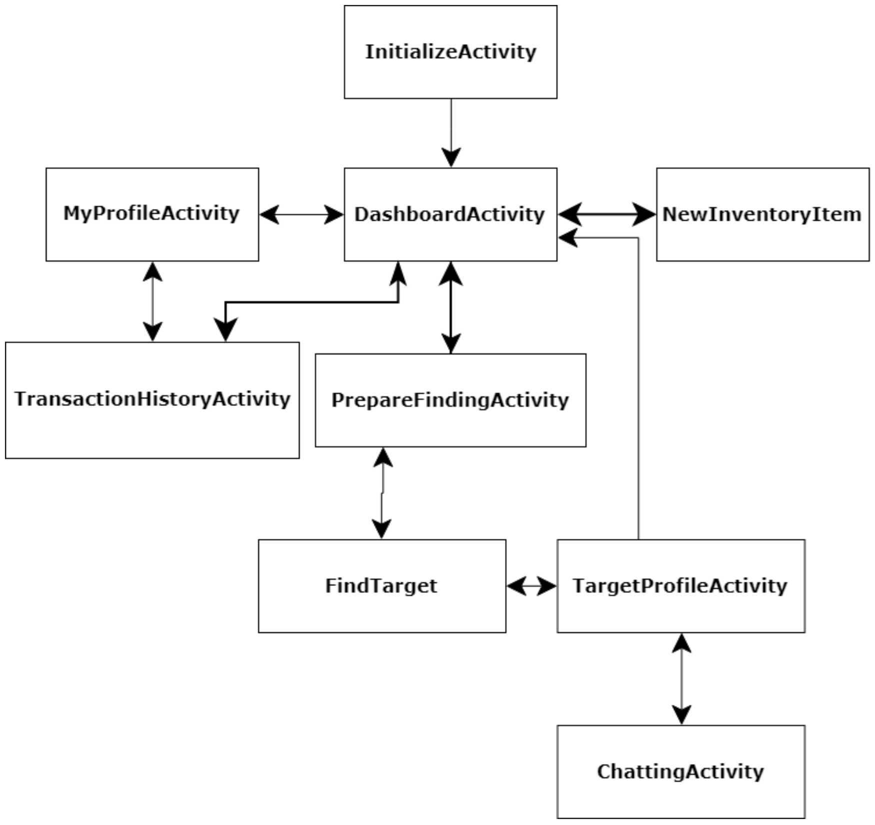


Figure 11. Activity Flow Diagram

- *InitializeActivity:* Users are first directed to this activity when the application is opened without initial Personal Information. In this activity, users must fill in their name and phone number to access the application. Reminding that this activity only appears when there is Personal Information missing.
- *DashboardActivity:* The main screen of the application where all your books and your wish list are displayed. Users can delete books in their list, add more books to the list and more importantly, a user can search for a specific book in the network that he wants to buy.
- *NewInventoryItemActivity:* If there is a book that a user want to public for sell, he can go to NewInventoryItemActivity to add new books the the public inventory list.
- *MyProfileActivity:* Personal Information, Summary of Transactions are displayed in this activity. From here, a user can delete his Personal Information and Initialize the Application again.
- *TransactionHistoryActivity:* When a transaction is completed between two users, it will be recorded to local database of the device. This activity is responsible for displaying all transaction records.
- *PrepareFindingActivity:* When a user wants to look for a specific resource (book) that other users might keep, he must fill in the keyword of book for the Gnutella Protocol Flooding in the network of peers. This activity will

pass the user searching keyword to form a Query Message and send to other users.

- *TargetProfileActivity:* After finding a resource in the network, as well as the owner who is keeping it, user will be directed to TargetProfileActivity to view the profile of the book's owner. From this activity, user can contact the owner of the book.
- *ChattingActivity:* If the user wants to have a private chat channel with the supplier, he can access ChattingActivity after having the owner's phone number.

3.4.2 Entity Class Diagram in Mobile

This Mobile Application's Class Diagram (Figure 12) is used for illustrating different types of objects that are passed to different functions of the application in the most compactable manner. Including three classes: PersonalInfo, Book and Transaction class.

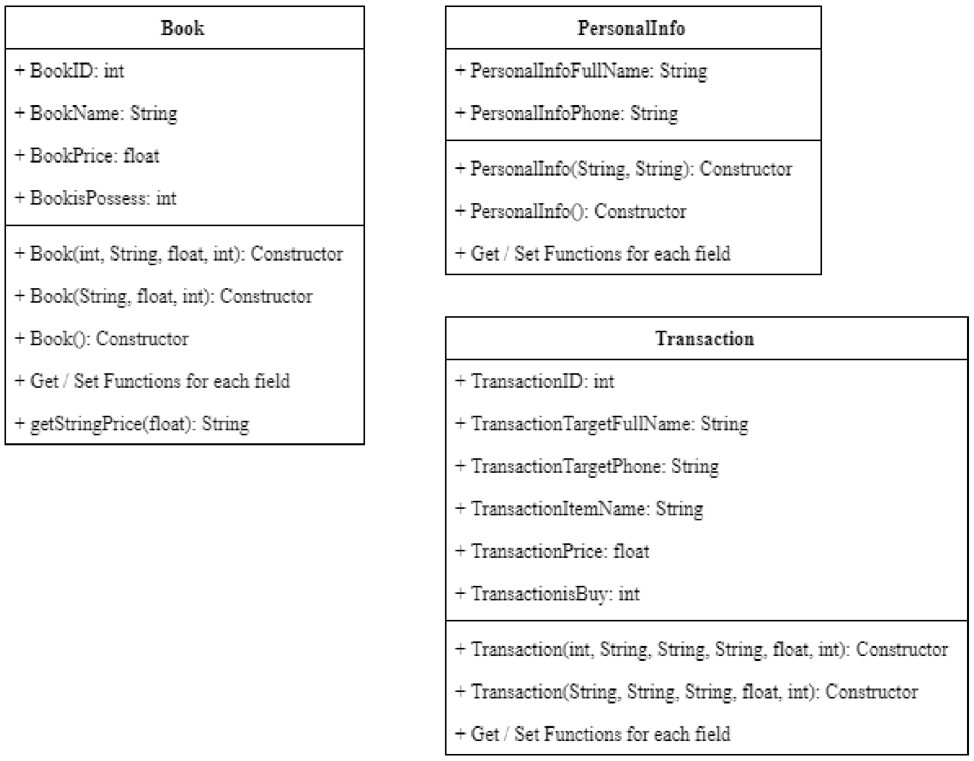

Book
+ BookID: int
+ BookName: String
+ BookPrice: float
+ BookisPossess: int
+ Book(int, String, float, int): Constructor
+ Book(String, float, int): Constructor
+ Book(): Constructor
+ Get / Set Functions for each field
+ getStringPrice(float): String
PersonalInfo
+ PersonalInfoFullName: String
+ PersonalInfoPhone: String
+ PersonalInfo(String, String): Constructor
+ PersonalInfo(): Constructor
+ Get / Set Functions for each field
Transaction
+ TransactionID: int
+ TransactionTargetFullName: String
+ TransactionTargetPhone: String
+ TransactionItemName: String
+ TransactionPrice: float
+ TransactionisBuy: int
+ Transaction(int, String, String, String, float, int): Constructor
+ Transaction(String, String, String, float, int): Constructor
+ Get / Set Functions for each field

Figure 12. Class Diagram for Android Application

Table 4 presents detailed description for each class including their serving purposes in the application and their categories.

Class Name	Description
PersonalInfo	PersonalInfo class keeps only one record, which is the Initial Personal Information. This record will be used as identical for the user when connects to other peers.
Book	Book class keeps all the books that users want to add to the application. There will be two types of Books in the database. • Possessing Books: Books that are possessed by the application's user and may be public for selling, trading purposes. • Wish list Books: Books that are not possessed by the application's user. Declare only for remembering an ought-to-buy books.

Transaction	Transaction class keeps all transactions records after a user rate another user after the trading processes. There will be two types of transactions storing in the local database. • Buying Transaction: Transaction that application's user acts as searcher, buyer. • Selling Transaction: Transactions that application's user acts as supplier, a resource holder.

Table 4 - Mobile Application Classes' Description

3.4.3 Google Nearby Connection API

Nearby Connections is a peer-to-peer networking API that allows application to easily discover, connect to, and exchange data with nearby devices in real-time, regardless of network connectivity. The connection is stated to implement a combination of Bluetooth, BLE, and Wi-Fi hotspots technology. [11]

In Nearby Connection API, there are two main strategies depending on the topology that the system required: [11]

- *P2P_Cluster:* is a peer-to-peer strategy that supports an M-to-N, or cluster-shaped, connection topology. Each device can both initiate outgoing connections to M other devices and accept incoming connections from N other devices.
- *P2P_Star:* is a peer-to-peer strategy that supports a 1-to-N, or star-shaped, connection topology. Each device can, at any given time, play the role of either a hub (where it can accept incoming connections from N other devices), or a spoke (where it can initiate an outgoing connection to a single hub), but not both.

For our system, a user not only scans for IPs of nearby peers but also advertise his IP for others to retrieve. Therefore, choosing the P2P_Cluster strategy seems to be more reasonable than the other. After choosing the strategy, we must let the application have the appropriate permission to the android device. This including: [11]

- BLUETOOTH: Allows application to connect to paired Bluetooth devices.
- BLUETOOTH_ADMIN: Allows application to discover and pair Bluetooth devices.
- ACCESS_WIFI_STATE: Allows application to access information about WI-FI networks.
- CHANGE_WIFI_STATE: Allows application to change WI-FI connectivity state.

The Connection API appears two separate phases: pre-condition, and post-condition. In the pre-connection phase, Advertisers advertise themselves, while Discoverers discover nearby Advertisers and send connection requests. After a connection request is accepted by both sides, the connection is considered to be established and the devices enter the post-connection phase, during which both sides can exchange data. [11]

The prototype takes advantage of Nearby Connection API to scan for surrounding devices that also installed the application. After discovered another device, the application performs the exchange information process to get other device's IP.

3.4.4 Gnutella Protocol

After receiving a list of nearby devices' IPs from Nearby Connection API, the application should able to use those IPs as initialized hosts and start joining the Gnutella Network and Protocol as Figure 13.

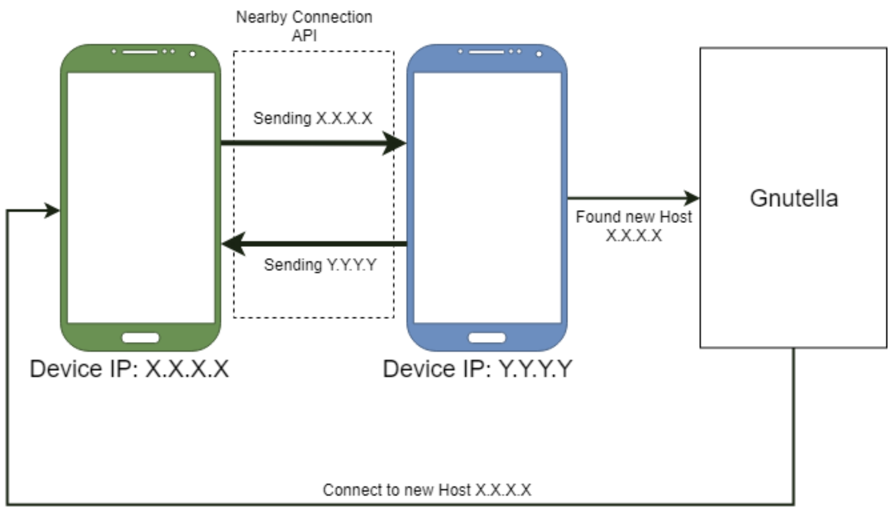


Figure 13. Getting Nearby Host IP by Nearby Connection API

Then, when a user wants to public his books, the system will scan through the Inventory Table in Local Database, create xml files naming books resource, inside is the phone number, name, and IP address to connect of the owner. The number of created files matches number of books the user wants to public. In this way it will be easier for system to flood a request through networks of peers. If inside the Shared Directory, there is a file matching the Query, the peer who keep the resource will return QueryHit message to the peer who sent the Query, therefore the owner of the Query can download the file including the phone of QueryHit's Owner. On the other hand, if there is no resource stored in directory matching the Query, it is directory's owner responsibility to create new Query and send it to other Peers. Figure 14 gives us the general idea of how resource is advertised, how a peer query for a resource, get QueryHit from the resource owner and download the resource including the phone number of QueryHit's Peer.

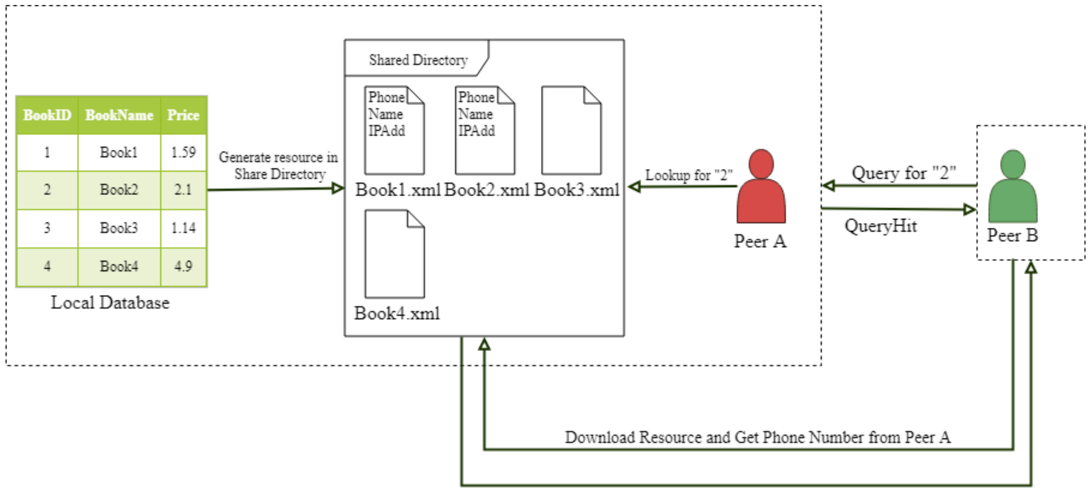


Figure 14. Searching for Resource Process between Peers

In the prototype application, there is a package holding all necessary components for the mobile peer to join and look up for resources by using Gnutella Protocol. Such components include three types of functionalities based on client, server and networking side. [12]

- *Client:*
    - Read user preference from a text file called "preference.txt"
    - Start a "Pinger" to continuously refresh the information about live neighbors.
    - Start a "Connector" to connects to saved host records and new hosts.

    - Link with layout resource and activities to display status of query and download results through “GnutellaCoordinator”.
- *Server:*
    - Starts a listener to catch incoming connections.
    - Starts a downloading service which searches in the upload directory when a query is received and create query hits for peers to download resource.
    - Handle different package type.
- *Networking:*
    - There is a Network Manager class that keeps track of live neighbors.
    - Each packet sent among peers contains header with following information:
        - payload descriptor (packet type)
        - payload length
        - TTL (Time to Live)
        - Hops
        - Message ID (to create a suitable QueryHit for a Query)
    - Each packet has corresponding Handler for routing purposes, capabilities as well as requirements for each packet types.

Figure 15 illustrates the UML diagram of GnutellaProtocol Package included in the Implement Application. The application will interact with GnutellaProtocol through

GnutellaCoordinator. The design of Gnutella is inspired from a Gnutella work for Desktop Peers [12]. However, there must be adjustments to fit the Mobile Development and Initial Functional Requirements.

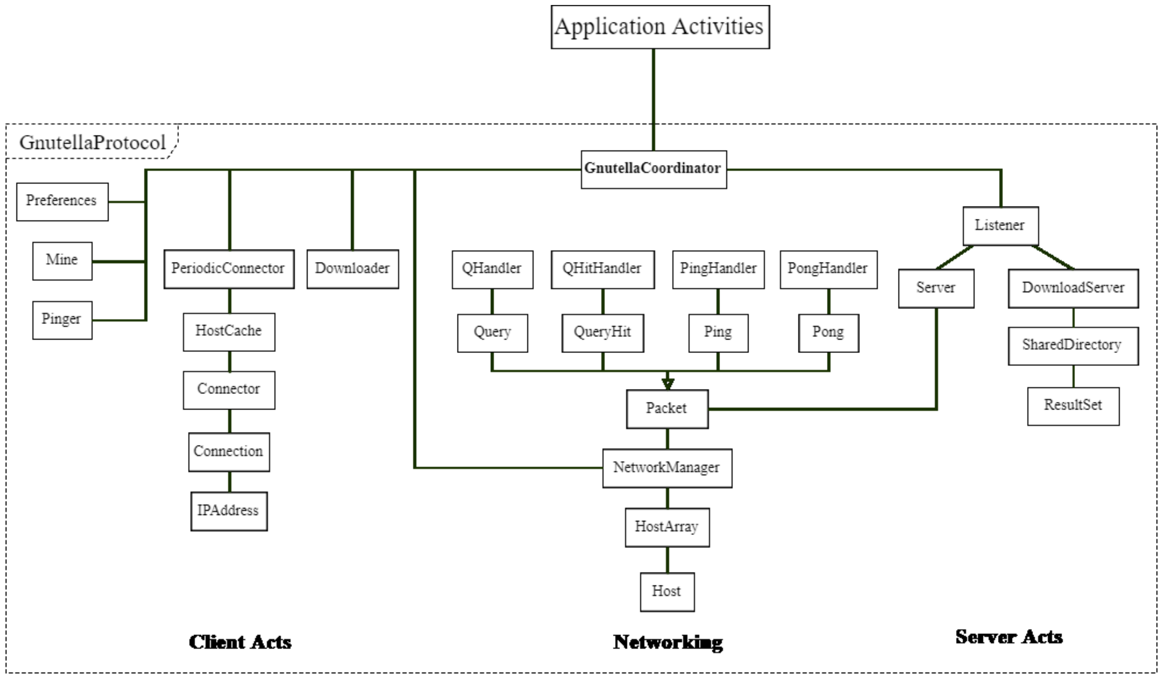


Figure 15. UML diagram for Gnutella Peers

From the Application views, there will be three functions that requires the implementation from GnutellaProtocol. Table 5 summarizes how GnutellaProtocol supports each function of the prototype by Gnutella mechanism.

Function	Gnutella Support
Advertise Books	Upload book files to SharedDirectory from local database. Start “Listener” for download server and a lookup function in SharedDirectory for incoming Query messages

Search Target	After getting list of live neighbors, create a Query message from the search keywords of user and send Query to those neighbors. Each nearby neighbor, after receiving the Query message, will search in the SharedDirectory. If there is a resource that matches the query, return a QueryHit, else, create new Query to other peers except the initial one (the one that send the Query message).
Message Target	Modify the Query and QueryHit to adapt the required function (become Chat and ChatHit message), which perform similarly to Query and QueryHit mechanism, except no look up and forward Query message for resource.

Table 5. Support of Gnutella in Prototype Application

3.5 Setting up Server Application

3.5.1 Framework and Technology

For server implementation, I decided to use Microsoft-relating Technology since the system is developed in Window Environment, it is considered more convenient when I build and deploy server using Microsoft Products and Technology. The compatibility of Server and Environment will be easier to handle.

- *ASP.NET WEB API 2.0:* a framework for building Web APIs on top of the .NET Framework. [13] Pre-defined Verb-Oriented Functions like “Get”, “Post”, “Put”, “Delete” help us shorten developing time of a central server for other device to retrieve ratings and post new ratings.

- *Entity Framework 6.0 :* Microsoft's recommended data access technology, an object-relational mapper that enables .NET developers to work with relational data, it reduces most of the database-access code that developer need to write for accessing the database. [14]
- *Microsoft SQL Server 2017 Express Edition:* Free Edition of SQL Server, ideal for development and production of database for desktop, web, and small server applications. Suitable for storing small data structure and data records.

3.5.2 Controller, Model and Database on Server

The Web API will follow the MVC (Model-View-Controller) Pattern which is widely used in web application. However, the View component is reduced in this prototype system since the Web API is only used for retrieving and sending data from device to the server database.

Figure 16 shows the general architecture of Web API server. Inside the server, there are three main components:

- *Controller:* Receiving request from client side, routing request to corresponding functions to interact with Model classes. After the Model components send the result sets relating to the function, it is controller duty to send the result back to client in suitable format.

- *Model:* Acts as an important connection between Controller and Database. Map the data sent to Controller from Client into suitable data that can Interact with the Database.
- *Database:* Stores, Creates, Updates and Retrieves Data based on User Requests.

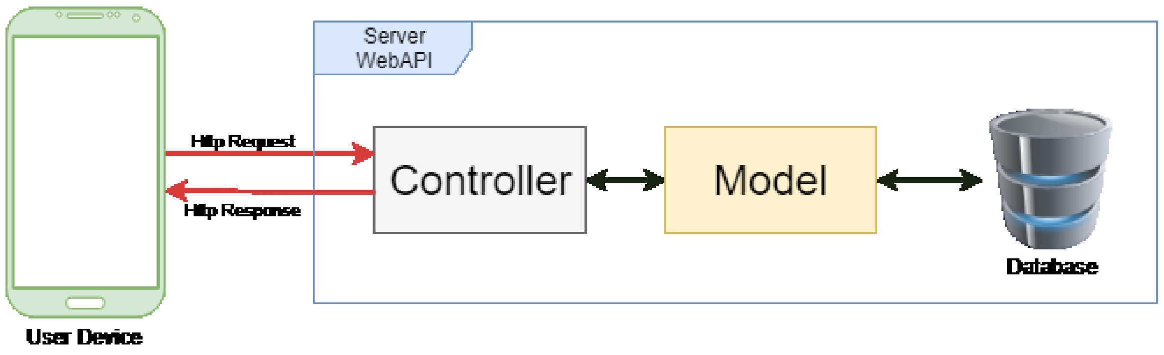


Figure 16. Web API Server Architecture

3.5.3 Class Diagram in Server

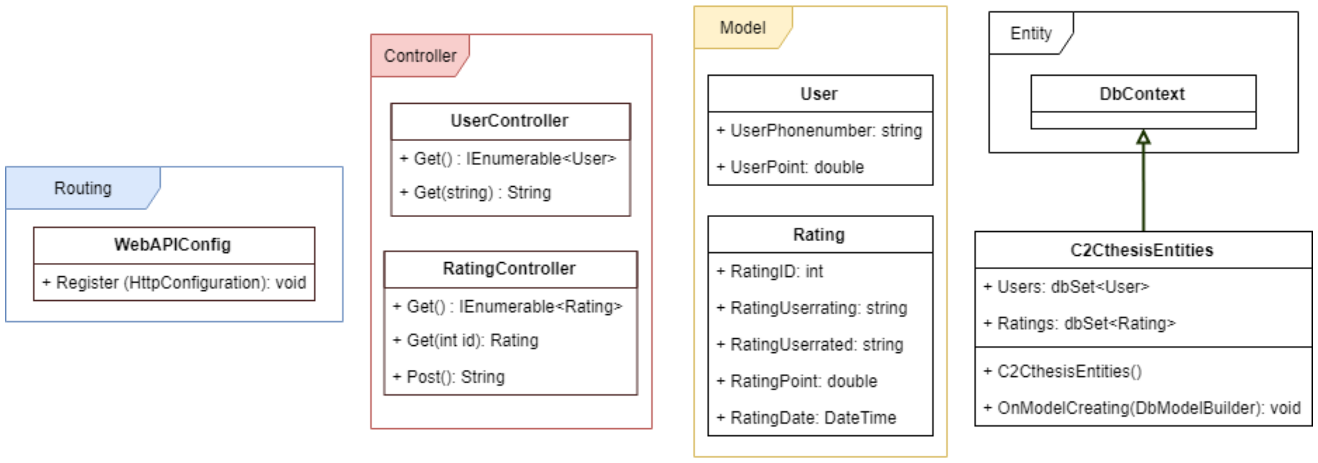


Figure 17. Class Diagram at Server Side

Figure 17 sketches the most abstract but still can cover key classes that forming the Web API. From the Class diagram, beside the class WebAPIApplication which called when the WebAPI Start, there are three types of classes in the server. Table 6 provides a more detail descriptions of purposes of each key class serving the Web API.

Class Name	Description
Routing	
WebAPIConfig	This class is used for configurating the route for Web API. In general, it guides the API which controllers and functions will execute when the API receive the API request from client.
Controller	
UserController	UserController class includes two functions relating to User Table in the Database. • Get all User records stored in database. (this function is used for testing purposes only) • Get a users' average point from his phone number.
RatingController	RatingController class includes three functions relating to Rating Table in the Database. • Get all Rating records stored in the database. (this function is used for testing purposes only) • Get a specific Rating based on the RatingId. • Post new Rating to the Database.
Model	
User	User class map a User Object in the Controller to a User Record in the Database. Assuming the primary key in this scenario is phone number.
Rating	Rating class map a Rating Object in the Controller to a Rating Record in the Database. It has it own Id as primary key and assuming that RatingUserrating and RatingUserrated are foreign keys from the User Table
Entity	
C2CThesisEntities	C2CThesisEntities class inherits from Entity Framework Class DBContext. Play an important role in retrieve data from data base and map to corresponding Model Class.

Table 6 - Server Classes' Description

CHAPTER 4

FINAL RESULTS

This Section finalizes what the prototype has achieved considering from initial requirements. Then, discussing in what aspect the prototype can prove that by applying P2P to E-Commerce, the business would be significantly improved.

4.1 Mobile Application with Gnutella Protocol.

The core of the prototype lies in the design of the mobile device. In concern to the front-end design and the displacement of data based on user interaction, this mobile application follows strictly the design flow that we initialize in Methodology Section. From Figure 18 to Figure 24 display the GUI of the application when user uses our application.

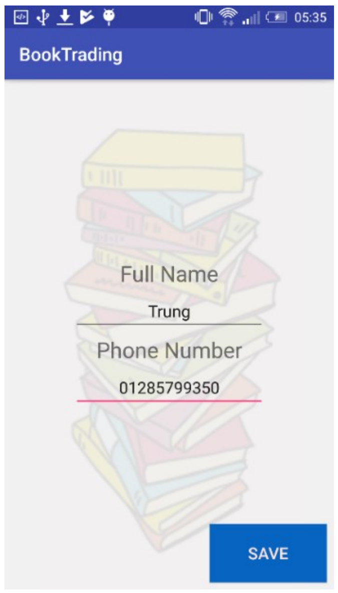


Figure 18. InitializeActivity

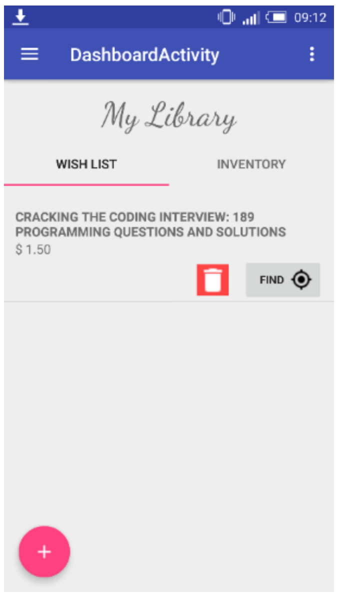
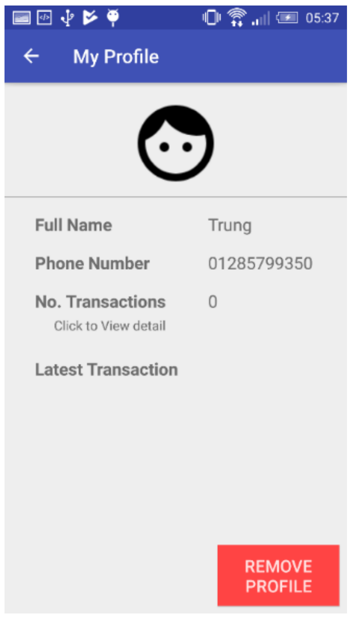

09:12
DashboardActivity
My Library
WISH LIST
INVENTORY
CRACKING THE CODING INTERVIEW: 189 PROGRAMMING QUESTIONS AND SOLUTIONS
$ 1.50
FIND
+

Figure 19. DashBoardActivity

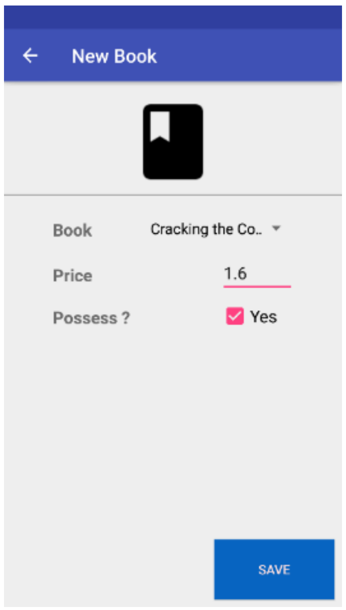


Full Name	Trung
Phone Number	01285799350
No. Transactions	0

Click to View detail

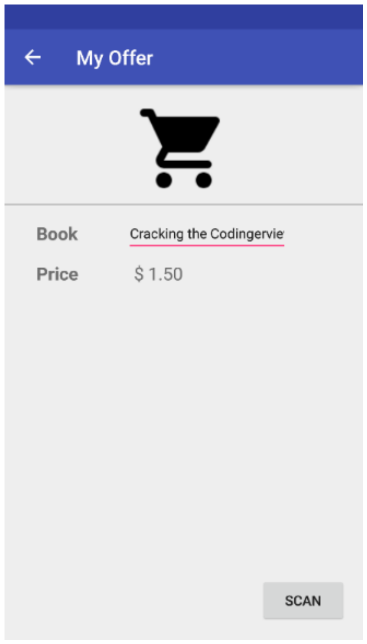
Latest Transaction

Figure 20. PersonalInfoActivity



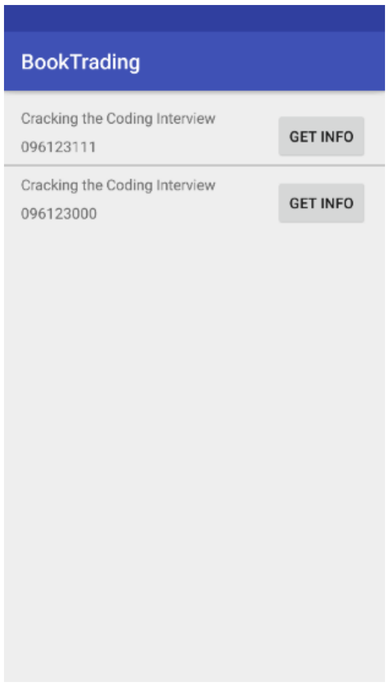
Book	Cracking the Co..
Price	1.6
Possess ?	✓ Yes

Figure 22. NewInventoryActivity



Book	Cracking the Codingervie
Price	$ 1.50

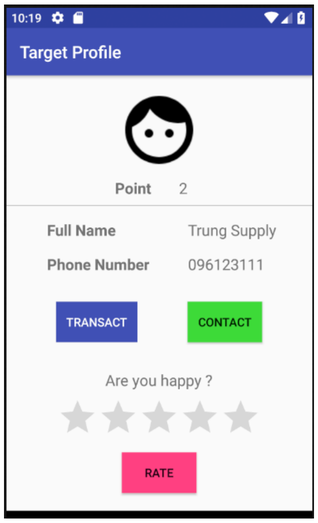
Figure 21. PrepareFindingActivity



Cracking the Coding Interview
096123111

Cracking the Coding Interview
096123000

Figure 23. FindTargetActivity



Full Name: Trung Supply
Phone Number: 096123111

Figure 24. TargetProfileActivity

The connectivity between is something worth mentioning in this scenario. Firstly, about the Nearby Connection API, to implement Connection API, both WI-FI and Bluetooth on peers must be activated, this somehow drains large amount of battery. When test on physical Android devices, the unexpected heat up may cause discomforts in users' experience.

Then, after getting IP of nearby device, the application starts connecting to those IP to established relation to neighbors and periodically ping those hosts to get their status. When user input the keywords for finding resource, is when the Gnutella Protocol on the device generates a Query message with pre-defined hops, sending across all hosts in the networks. In general, due to the small scale of our virtual Gnutella Network, the time between Query send and QueryHits response is very acceptable. Not too long after a Query sent, the Application receive the QueryHits and get the Resource.

4.2 Web API Server and Database

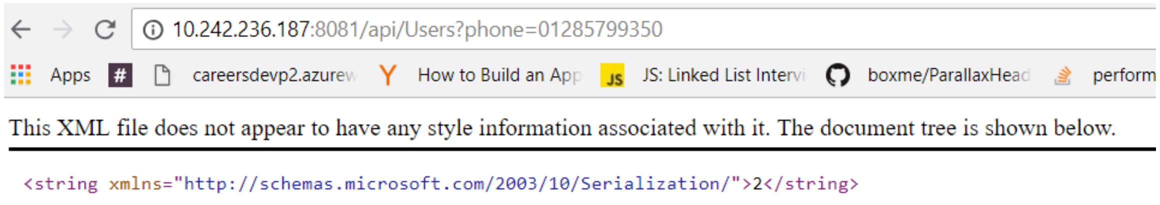
Web API plays a small role of storing and retrieving all users' rating in our system. From System Perspective, there are only two functions that allows clients (mobile devices) to contact with the server (Web API). Including:

- Retrieving a User Point from User's Phone Number from HTTP GET request through URL: *"[ServerHostIp]:[Port]/api/Users?phone=[User Phone Number]"*.

- Posting new Rating to Server through HTTP POST request to URL: *"[ServerHostIp]:[Port]/api/Ratings/"*.

The rest functionalities of Web API introduced in the Methodology Section only serve for testing purpose and support the primary functions.

About the retrieving rating function, by using GET Request, with simple HTTP Response Message, this application seems to function very well under multiple query requests (we try to query to database with 4 physical devices under the same WI-FI router to connect to internal server). The result in detailed is showed in Figure 25 when retrieving Rating of a User who phone number is 01285799350.



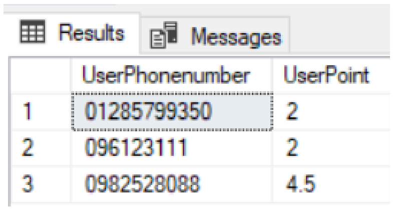
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">2</string>
```

Figure 25. Return XML from calling Get User Rating from Web API

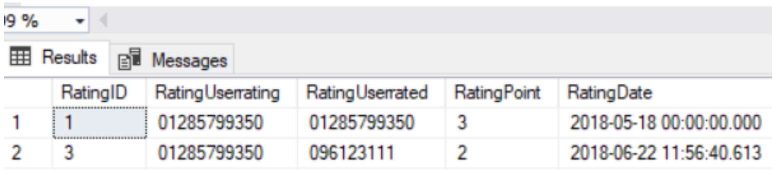
Posting new rating requires a more complex method. In order to post new rating to the database, there is at least three fields that needs submit from the user. Including User's Phone Number who rates, User's Phone Number who is rated, and the Rating Point. It is suitable to use HTTP POST request to hide these parameter, such sensitive data will be used to update database. On the other hands, at server side, there should be a checking function to confirm whether the users involved in rating process existed in the database before or not. If not, then it is compulsory to add new User to the User table first then Update the Rating table later.

Due to many conditional layers within the function, latency is unavoidable when users send POST Rating Request. The more number sending the POST request, the longer it take for the response time. Figure 26 shows the User table after adding new Rating from user 01285799350, who rates user 096123111, who also, has not been in the User Table before. And Figure 27 illustrates the Rating table after add Rating to the Rating table.



	UserPhonenumber	UserPoint
1	01285799350	2
2	096123111	2
3	0982528088	4.5

Figure 26. User Database after sending a Post Rating with new User



	RatingID	RatingUserrating	RatingUserrated	RatingPoint	RatingDate
1	1	01285799350	01285799350	3	2018-05-18 00:00:00.000
2	3	01285799350	096123111	2	2018-06-22 11:56:40.613

Figure 27. Rating Database after Post Rating Method

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

Peer-to-peer technology seems to be very potential in digital era when mobile devices tremendously develop from time to time. As Grab and Uber successfully exploited in transportation services, Airbnb applied P2P to hospitality services and so on. These names change the business by letting the users who request the service, directly contact with the suppliers of the services with less interference of centralized components as possible.

This thesis along, shows that peer-to-peer for mobile could also be applied in C2C e-commerce aspect. Despite the fact that the demonstration prototype contains many errors and missing functionalities, unclear and clumsy in application design; the final application briefly sketches the C2C e-commerce picture under the hood of P2P Technology.

Thanks to the P2P, the E-Commerce Company can significantly reduce the system maintenance and upgrading cost; improving users' experience by letting the contact directly their target and many more if we can fully take advantage of P2P technology.

5.2 Future Work

Many aspects and corners are left when this thesis discusses the Methodology due to the lack of time. There are more problems arising from this work that worth perusing and more researching.

Future work will concern about deeper problem in querying method between peers joining in the network. If the application adapts query with more criteria like prices, location, it is more likely to enhance user's experience, but will it slower the scanning speed of flooding algorithm. On the other hands, if one user turn off the application, meaning leaving the Gnutella network, how will it effects the whole network.

The Server Web API in the demonstration acts as a fair referee amongst peers for their rating. Is it possible to improve the reliability of the ratings? For example, when to allow a user update rating is very important. Currently, the application let the user freely vote his trader if he can find the trader. If we can add more validation and barriers in the voting process, perhaps a virtual contract between two users that only submit the rating when both peers finished their trading.

Finally, we now storing all users ratings on a centralized server. It would be a relief if there is a way to make peers self-sustain a ratings table to query from without the need of sending request to central server. Of course, the system must always guarantee that users cannot change their ratings personally while they join the system. If such task is accomplished, then our system would become pure Peer-to-Peer architecture in both Technical and Business manner.

REFERENCES

[1] H. M. Tran, S. V. Nguyen , T. T. Tran and L. Q. S. Pham , *A Study of Uber-based Applications,* Nha Trang City, 2017.

[2] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins and Z. Xu , "Peer-to-Peer Computing," 3 July 2003.

[3] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies," *ACM Computing Surveys,* vol. 36, pp. 335-371, December 2004.

[4] K. N. M. Lazo, *Execs wary 'disruptive tech' to heighten biz competition – IBM,* The Manila Times, 2016.

[5] M. Parameswaran, A. Susarla and A. B. Whinston, *P2P Networking: An Information-Sharing Alternative,* vol. 34, IEEE, 2001, pp. 31-38.

[6] H. M. Tran, K. V. Huynh, K. D. Vo and S. T. Le, *Mobile peer-to-peer approach for social computing services in distributed environment,* Danang, 2013 , pp. 227-233.

[7] M. Satyanarayanan, *Fundamental challenges in mobile computing,* Philadelphia, Pennsylvania, 1996, pp. 1-7.

[8] M. Ben-Ari, Principles of Concurrent and Distributed Programming, 1st ed., Prentice Hall, 1990, p. 164.

[9] K. G. Shin and P. Ramanathan, *Real-time computing: a new discipline of computer science and engineering,* vol. 82, IEEE, 1994, pp. 6-24.

[10] l. J. Hom, "What is C2C?," Business News Daily, 2013.

[11] Google, "Nearby Connection API Overview," 7 June 2017. [Online]. Available: https://developers.google.com/nearby/connections/overview. [Accessed 22 June 2018].

[12] I. Varsandan, "A Peer to Peer Network for Distributed Case Based Reasoning," 2007.

[13] M. Wasson, "Get Started with ASP.NET Web API 2 (C#)," Microsoft, 28 November 2017. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api. [Accessed 22 06 2018].

[14] Microsoft, "Introduction to Entity Framework," 23 October 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx. [Accessed 22 June 2018].