

PostItApp  
CmpE 436 Term Project Report

Kutay Candan  
2013400003

7 December 2017

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Intro . . . . .	1
1.2	Motivation . . . . .	1
<b>2</b>	<b>Approach</b>	<b>3</b>
<b>3</b>	<b>Experimental Methodology</b>	<b>5</b>
3.1	Database . . . . .	5
3.1.1	Member . . . . .	5
3.1.2	Team . . . . .	6
3.1.3	TeamMember . . . . .	6
3.1.4	PostIt . . . . .	7
3.2	Server Part . . . . .	7
3.2.1	Server Socket . . . . .	7
3.2.2	Socket Threads . . . . .	7
3.3	Android . . . . .	8
3.3.1	Android Interface . . . . .	9
3.3.2	Android Service . . . . .	9
3.3.3	TCP Client . . . . .	10
<b>4</b>	<b>Conclusion, Results and Future Work</b>	<b>11</b>
	<b>References</b>	<b>12</b>

## **Abstract**

In this term project, we need to implement a mobile application that works like a distributed system with using socket programming. In order to perform this demand, I created PostItApp for Android. PostItApp works like a reminder not only for yourself. In PostItApp, you can create a group and put your reminder post here. Also, anyone which is in your group can see and update your post. With using this application, you can share your home needs with your family or you can arrange a meeting with your collaborators.

# Chapter 1

## Introduction

### 1.1 Intro

In CmpE 436 term project, we need to implement a mobile application which multiple users can interact with each other. However, in order to make this requisition, we cannot use normal Java concurrent libraries and normal Internet connection. We need to handle concurrency with using class materials such as monitors or semaphores[1]. Moreover, we need to handle communication with socket programming[2]. Therefore, I add these components while building PostItApp.

In the start of the project , I think the project idea and determine to build a reminder which is for more than one people. For this purpose, I build messaging system in just groups. Also I change messaging type a bit. I use post instead of messages. Users can create,join,delete,exit group, also add,update,delete,read posts. However, there has a key point which is preventing multiple write to posts. Once a post is created, anyone which are member of the group can update the post.

### 1.2 Motivation

I already know that this course has one mobile application term project and I had never studied on mobile application before. In order to learn how mobile application is built(Course's topic is important too.), I take this course. Moreover, my application is inspired by scheduler. While I was thinking the project topic, I had a lot of work to do and I could not handle them if I write them in my phone. However, I could not find proper application in

store. Moreover, I need to share my works to someone at some cases so I need groups too. With using this knowledge I decided to build PostItApp, it my all problem.

## Chapter 2

### Approach

In this term project, I implemented group reminder application and in this application clients talk server with several different requests and use a bunch of different working parts such as Android, Java and MySQL. I will explain working areas and their operation in Experimental Methodology part in detail.

Communication strategy needs to be main approach in this project. Its process is like request response relationship. Firstly, a client sends a connection request to the server and this server opens a new thread for communicating with this client. After client and servers are connected with each other, connection remains and both server and client listen each other until application is deleted or server is stopped. Because I use Service in Android part and once connection is occurred, Service remains open even if application is closed.

After connection is provided, client and server thread opens input and output streams. I use Scanner and PrintWriter for communication. Because when I use DataOutputStream and BufferedReader UTF-8 becomes a problem. I think that Turkish citizen may want to use this application so I handle this problem with Scanner and PrintWriter because they pass data to each other with same bytes encoding types.

Once streams are created both client and server thread starts listening. For this version of application (I really like this project idea if I determine to enhance this application's interface and listening protocol in further versions, I can put it in Store.), they act like this; a client sends a meaningful message, after that server thread reads it, processes this message and if necessary re-

turns a result to a client. Clients get a response message and act on this result. After, they are listening each other again.

Communication is occurred based on some actions. Firstly, member needs to sign up or log in. Secondly, member can create, join or delete group. Then, member can add, update(if no one is updating), delete and read posts in a determined group. For each communication message (from client to server), there is a prefix for determining process, after that parameters are added. For each message fragment I add "-" char in order to split them. Here is all communication types and their meanings.

- sign\_up-username-mail-password (user sign up)
- log\_in-username-password (user log in)
- get\_groups-userID (gets users all groups)
- add\_group-userID-teamname (add group into user)
- join\_group-userID-teamname (join group into user)
- delete\_group-userID-teamname (delete group from user)
- check\_master-userID-teamname (checks whether user is master of a group)
- get\_posts-teamname (gets teams all posts)
- add\_post-teamname-postname-posttext (add post into group)
- update\_post-teamname-postname-posttext(update post in a group)
- delete\_post-teamname-postname (delete post from group)
- read\_post-teamname-postname (read post in a group)
- request\_post\_update-teamname-postname(want to update post)
- release\_post\_update-teamname-postname(finish updating post)

## Chapter 3

# Experimental Methodology

In order to build my application, I split my working area and use a lot of different components.

### 3.1 Database

For this term project, I need to keep users' information in a server and once they request their information I need to serve them. For corresponding their need, I use database in my server side. Because I am familiar to work with MySQL, I use MySQL database and SequelPro for visuality[3]. I just need 4 different tables to keep all information related to this application.

#### 3.1.1 Member

In this database table, I keep users' personal information. I also add more database components than I use in this version of application, because enhancing a database is a difficult situation for not only changing all parts of database connection but also can lose some previous data while adding a component. I also add `isDeleted` boolean for all tables, because it is important to keep data even if user wants to delete it at some cases but I am not using it too. For this version I only use username, password and mail as log in and sign up. This list is my all components of database columns for Member table.

- id
- username
- password
- mail



- firstName
- lastName
- creationDate
- updateDate
- isDeleted

### 3.1.2 Team

In this database table, I keep users groups which contain users posts. I just keep group names for this table as an important information. Users can create group if only group name is not taken yet. I use team instead of group because group has other meaning for database and it may causes problem while communicating database. This list is my all components of database column for Team table.

- id
- teamname
- creationDate
- updateDate
- isDeleted

### 3.1.3 TeamMember

In this database table, I keep users and groups relation. If a user creates or joins a group I write related user and group ID here. Moreover, another important component is isMaster boolean when a user is created a group he automatically become the master of this group. Its only difference is once he delete group, he delete kicks all users from group, and deletes all posts. If he is not the master of a group, when he deletes group, only he will exit on group. Here is my all components of database column for TeamMember table.

- id
- teamID
- memberID
- isMaster
- creationDate
- updateDate
- isDeleted

### 3.1.4 PostIt


The last table of my database table keeps users all posts. While keeping these posts, it takes teamID for determining which posts are posted in which teams. Furthermore, this table has the vital component which is isUpdatedNow boolean. If a user updates a post, this column is automatically triggered and prevent other users to update this post at the same time, in order to avert a data loss. Here is my all components of database column for PostIt table.

- id
- postname
- text
- teamID
- isUpdatedNow
- creationDate
- updateDate
- isDeleted

## 3.2 Server Part

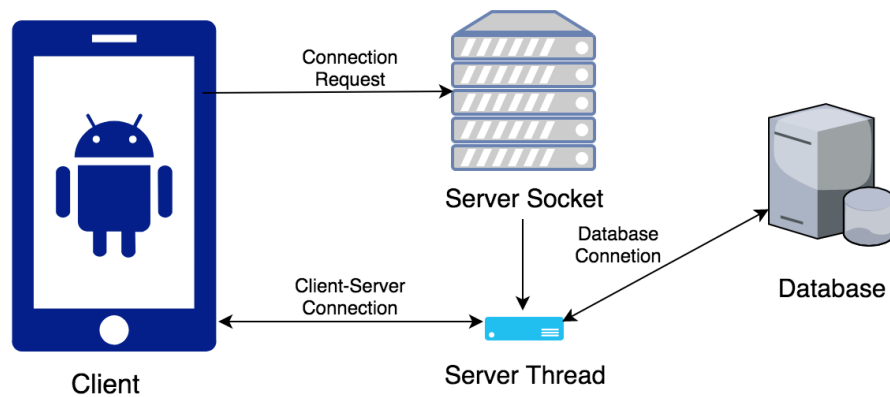
For this term project, I need to build a server that it has to listen all clients at all time and give response to clients. In order to satisfy this property, I use socket programming in Java for building my server side[4]. I split works as a ServerSocket and Socket threads.

### 3.2.1 Server Socket

In order to connect client and server, we need a system. Java ServerSocket class was made for this purpose. It provides TCP connection. At anytime, a client wants to communicate with server, ServerSocket creates a new Socket thread, connect client to it and continue to listen other connection requests. I also draw diagram to ease understanding.

### 3.2.2 Socket Threads

After ServerSocket creates Socket Thread, it gives 2 parameter. First of them is BinarySemaphore. Clients may send messages in order to change databases and they communicate via different server thread. Hence, these threads must wait each other while they are doing operation with usings database. If I do not put same BinarySemaphore to all thread some messages may conflict and



**Figure 3.1:** Basic Schematic of Connections[5]

problems may be occurred.

Second parameter is socket which holds connection between server and client. Using this socket, a server thread listens the particular client and when message comes, it tries to understand what client commands. Messaging is mentioned in Chapter 2.

Server thread also talks other classes which are for providing connection with database. For each message, a server thread talks different class. It firstly sets all parameters which come from client and runs particular method which is also taken from client as a prefix.

### 3.3 Android

In client side, we need to implement a mobile application. For this demand, I choose Android for building TCP Client and user interface[6]. I split my works as a TCP Client for connection, Android Service for keeping this connection without using main thread in Android and Android activities and layouts for user interface.

### 3.3.1 Android Interface

For providing visuality I use a lot of different components. They are Android layouts, Android widgets, icons, and activities for changable informations.

For design issue, I firstly open a blank activity and it has one layout which includes simple interface which is restricted and does not change with information. I also look color harmony in internet and select black and grey tones for application background, text background and text. Moreover, I use pink tone for button. I also add 2 list item layout for group and posts item which do not has activity.

In PostItApp application, interfaces changes with users database information. In this process, some layout items should be set in activity. I do not use fragment for the sake of simplicity. I change activity in order to go different layout. Here is all activity which I use in this application.

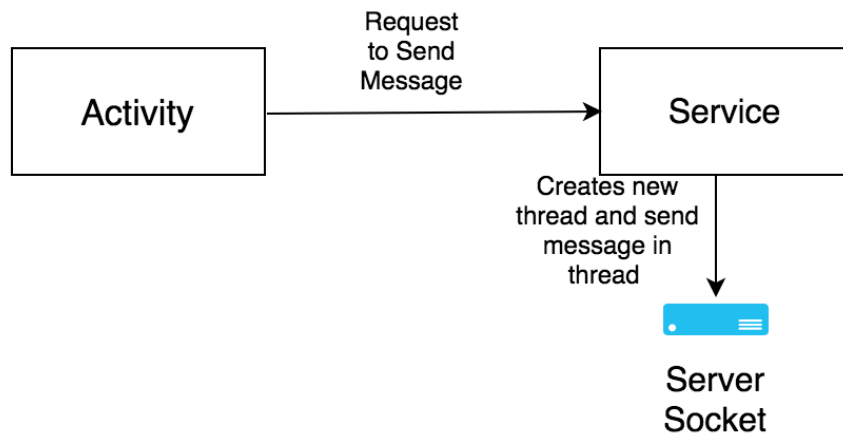
- LoginActivity
- SignUpActivity
- GroupsActivity
- AddGroupActivity
- JoinGroupActivity
- DeleteGroupActivity
- PostsActivity
- AddPostActivity
- UpdatePostActivity
- DeletePostActivity
- ReadPostActivity

I design 5 icons for visuality. Which are add group, join group, delete(both group and post), add post, update post. I use icon maker tool for design.[7]

### 3.3.2 Android Service

This is most important part of Android component. In Android, if you use background process, you need to extend selected classes and write your functionality in here. Because the main thread of Android just serves interface. In order to provide this utility, I use Android Service. It works like this. At the start of the application service requires TCP connection and once connection is occurred it keeps even if application is closed(Of course that is server

is down connection is lost.). Then, service waits activities for sending and receiving messages. It has functions for sending correct message to server.



**Figure 3.2:** Basic Demonstration of Service Work [5]

### 3.3.3 TCP Client

TCP Client has two key work. First of them is creating connection with server. Second is sending and receiving messages to/from server. This class is called in Android Service. I do not add detailed messaging and creating connection because I already mention them in previous sections and chapters.

## Chapter 4

# Conclusion, Results and Future Work

At conclusion, I want to mention some key points again. The important parts of this project are preventing update post conflict, working on multiple clients, interface bugs and database errors. I tested them all in emulator and other Android phones and I do not observe any problem.

I really want to enhance this project and upload in a store. Because it satisfies specific demand and I could not find application which does this. However, I need to enhance interface a lot. Moreover, now user can join a group just only knowing its name, users should send a request to another user for joining group. If these improvement have provided, PostItApp will be great application.

## References

- [1] . URL: <https://piazza-resources.s3.amazonaws.com/j84c7qq8zkd5xj/j8yf8rgik43ni/lecture3semaphoremonitor.pdf?AWSAccessKeyId=AKIAIEDNRLJ4AZKBW6HA&Expires=1512654299&Signature=UmQdm7HKWigU3NSBFGLd9mJHsbQ%5C%3D> (cit. on p. 1).
- [2] . URL: <https://piazza-resources.s3.amazonaws.com/j84c7qq8zkd5xj/j96vmden8zp2c3/lecture42.pdf?AWSAccessKeyId=AKIAIEDNRLJ4AZKBW6HA&Expires=1512654196&Signature=7kx5p1J%5C%2F8YJ6eFsrOp3PZn9DS%5C%2FM%5C%3D> (cit. on p. 1).
- [3] . URL: <https://www.mysql.com/> (cit. on p. 5).
- [4] . URL: <https://www.oracle.com/java/index.html> (visited on 10/29/2017) (cit. on p. 7).
- [5] . URL: <https://www.draw.io> (cit. on pp. 8, 10).
- [6] . URL: <https://developer.android.com/studio/index.html> (visited on 10/29/2017) (cit. on p. 8).
- [7] . URL: <http://romannurik.github.io/AndroidAssetStudio/> (cit. on p. 9).