GeoTag: Term Project – Part III

REPORT BY JORGE POTOSME

Abstract

GeoTag is an application which allows users to share information based on a geographic location. Information can be "tagged" to specific GPS coordinates and shared with friends who have interested in the tag, its owner, or any searchable relationship between those attributes. Continuing with the previous planning and preparation for development of this application, we have extracted important processes previously designed in specific use cases and laid them out chronologically and hierarchically. We also laid out state diagrams for our user interface, pseudo code for our application and User Interface.

Background

With the development of cell phone applications and the inclusion of GPS technology, we are interested in leveraging the location of a user and providing useful information in response to that. After creating associated use cases, activity diagrams, swim lane diagrams, and class diagrams to demonstrate how different parts of the system would operate together, we are now working on sequence diagrams which flesh the system out further and identify important relationships between the objects and their methods.

Method

From our uses case and sequence diagram that are shown in the appendix we move on to create State cases to show the movement of stats for the peer this allows us do create proper user interface that the user will find appealing and understandable.

With the communication resolved by the sequence diagram we can now start to write the pseudo code for our program. The functions we have in our class diagram for our social manager have been defined using pseudo code. Taking from our pseudo code we can also begin designing our user interface.

Results

The result being state diagrams, pseudo code, and User Interfaces can be found in our appendix.

Discussion

After we have dealt with the design of our pseudo code be can begin proto-typing the project. Proto-typing can be done in several ways on is by using Windows form which is just some thing like the interfaces shown in the Appendix, the other is by using the Android Operating system. The major problem is that this application is a web based application and will require a web service. The Social hub will be required to be a web service. This will then require for a server to be set up and fully functioning in order to do real proto-typing. This will take time to set up and more research to do.

Conclusion

This is a final step in our design step or our system. By design it in this fashion we have fully understands our stake holders and the purpose for our system. From here we must chose how we are going to impalement this, what our language is going to be, what tools will we used.

Appendix

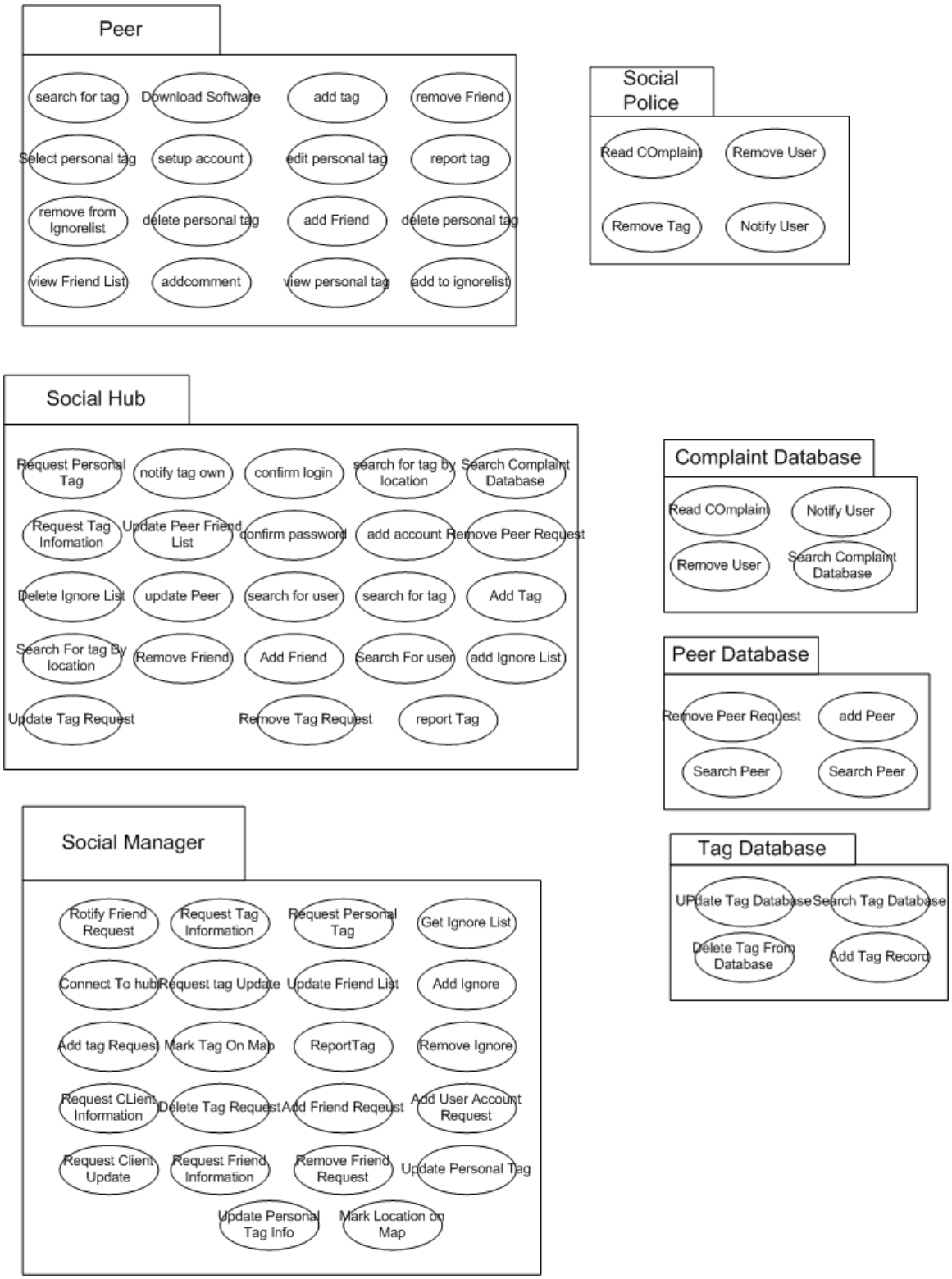

Figure 1: System and Domain Use Cases

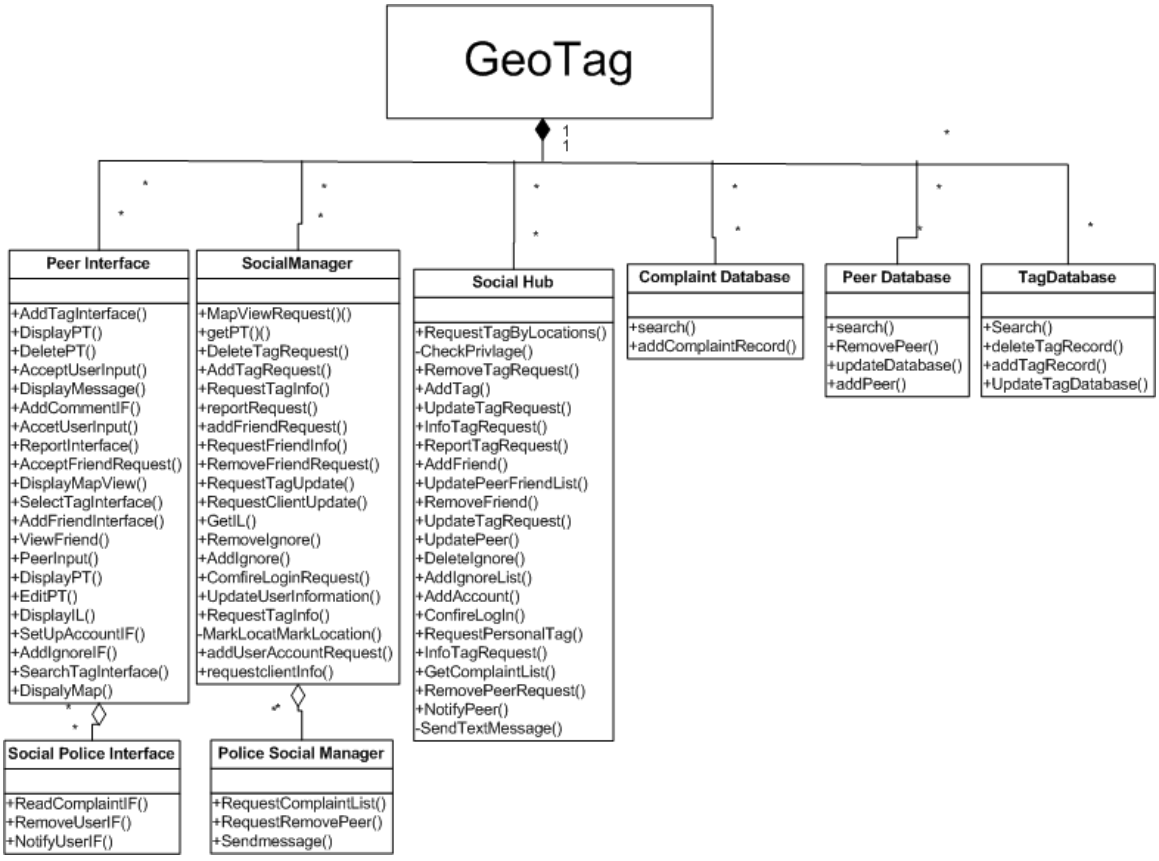

Figure 2: Finalized GeoTag Class Diagram

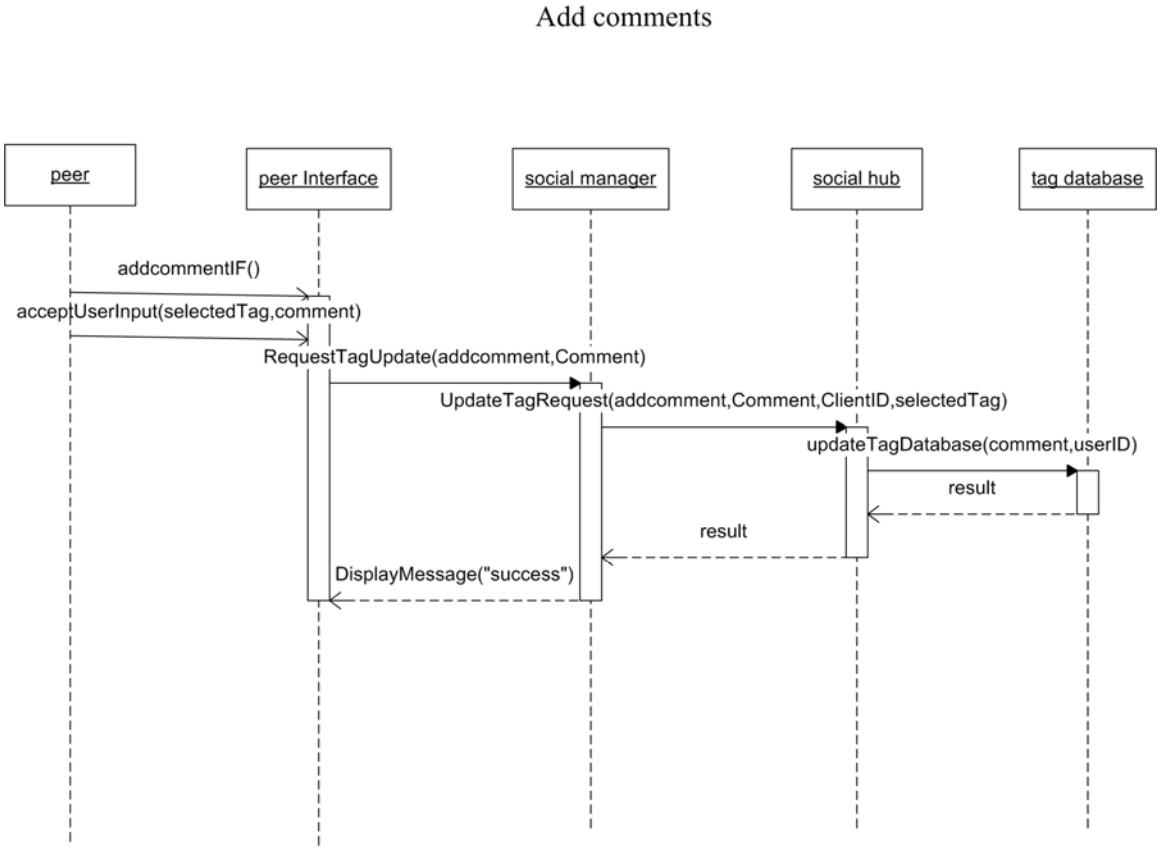

Figure 3: Add Comments Sequence Diagram

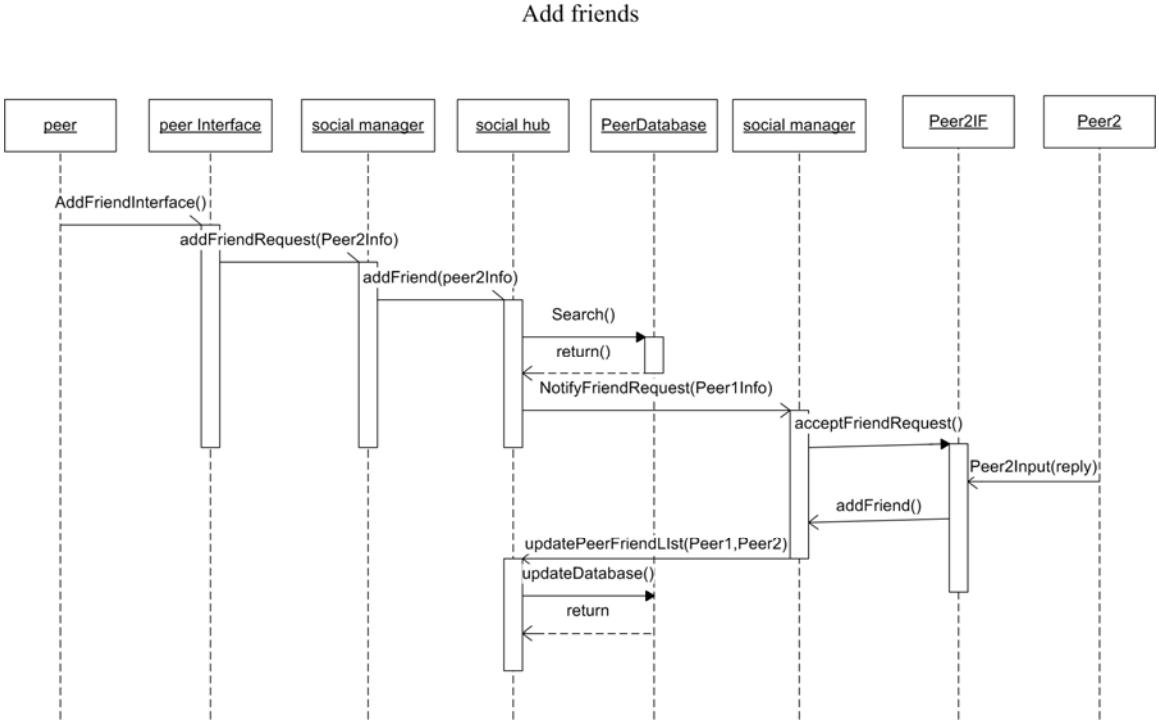

Figure 4: Add Friends Sequence Diagram

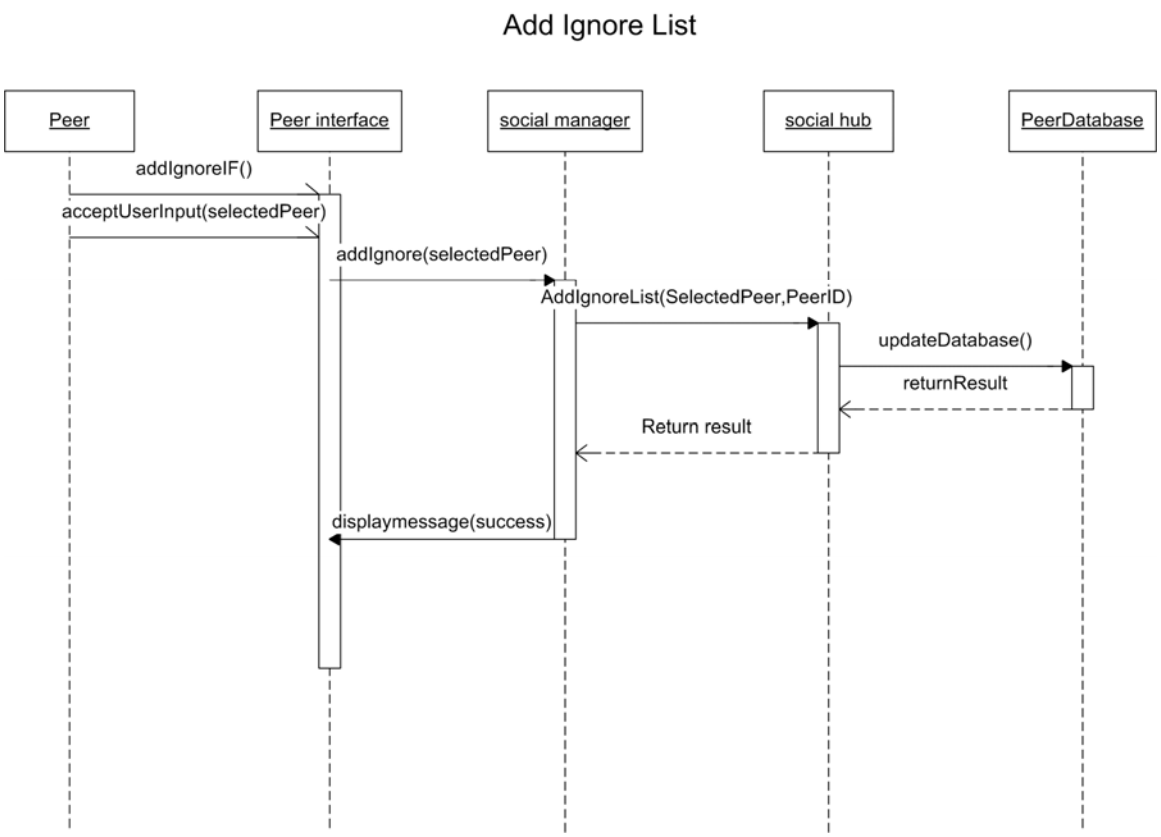

Figure 5: Add Ignore List Sequence Diagram

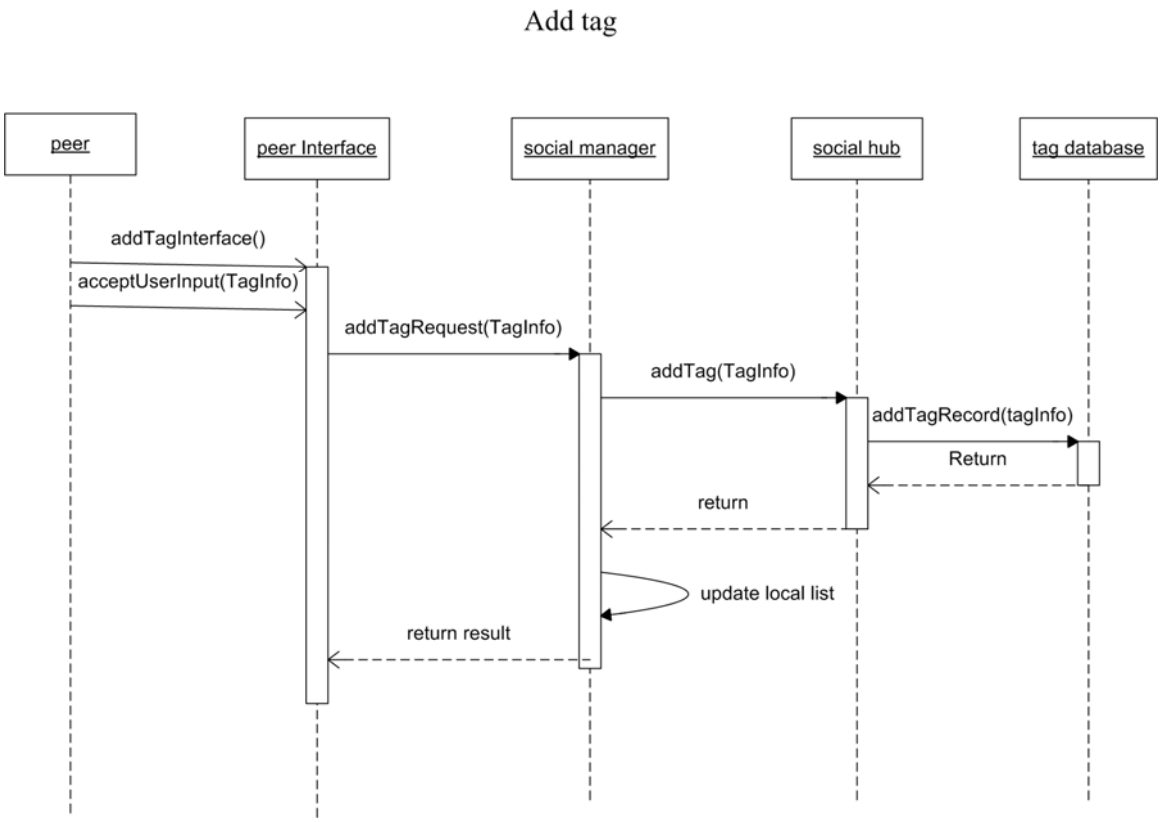

Figure 6: Add Personal Tag Sequence Diagram

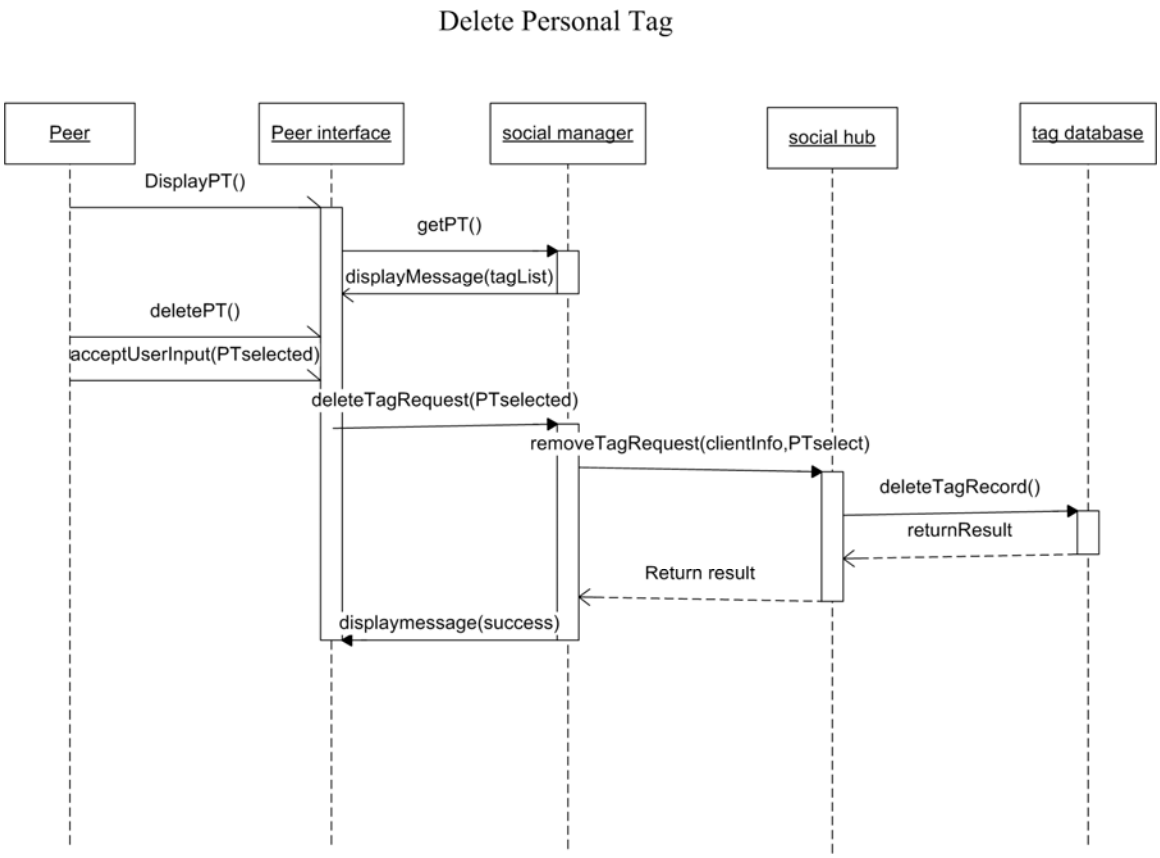

Figure 7: Delete Personal Tag Sequence Diagram

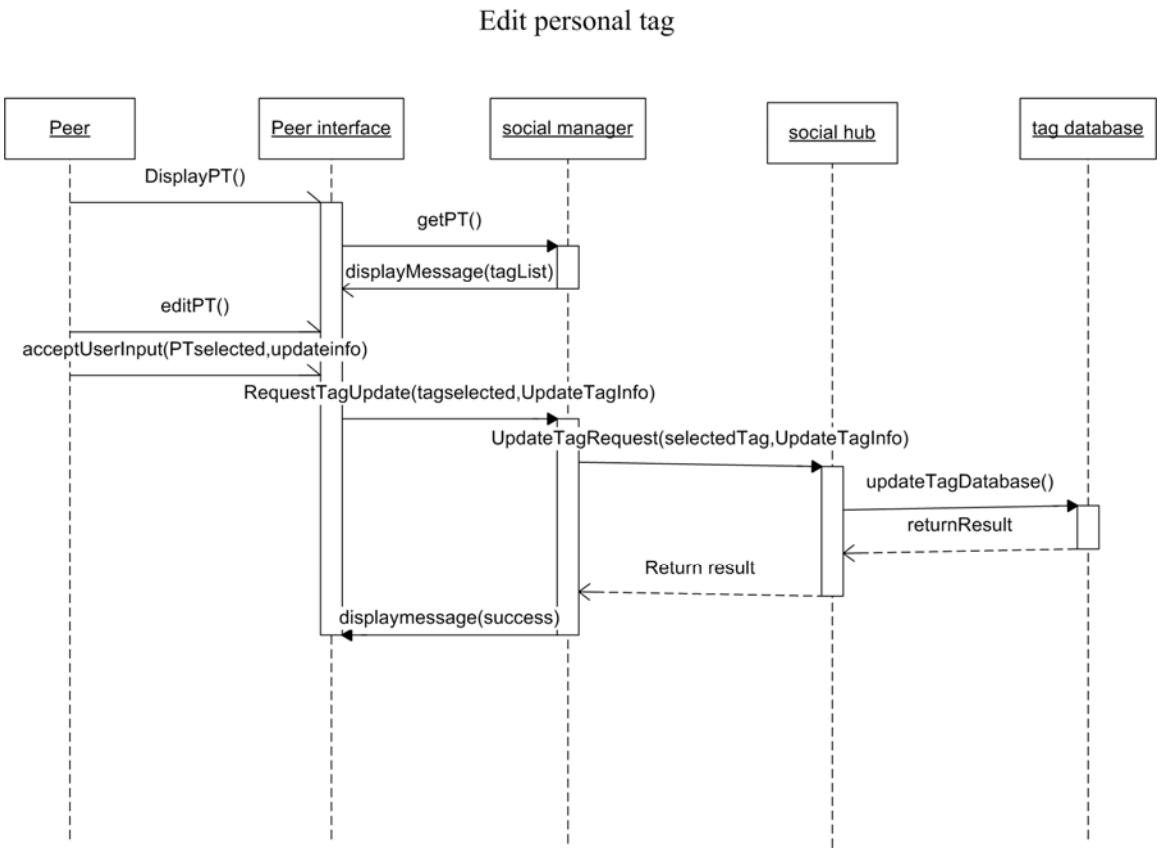

Figure 8: Edit Personal Tag Sequence Diagram

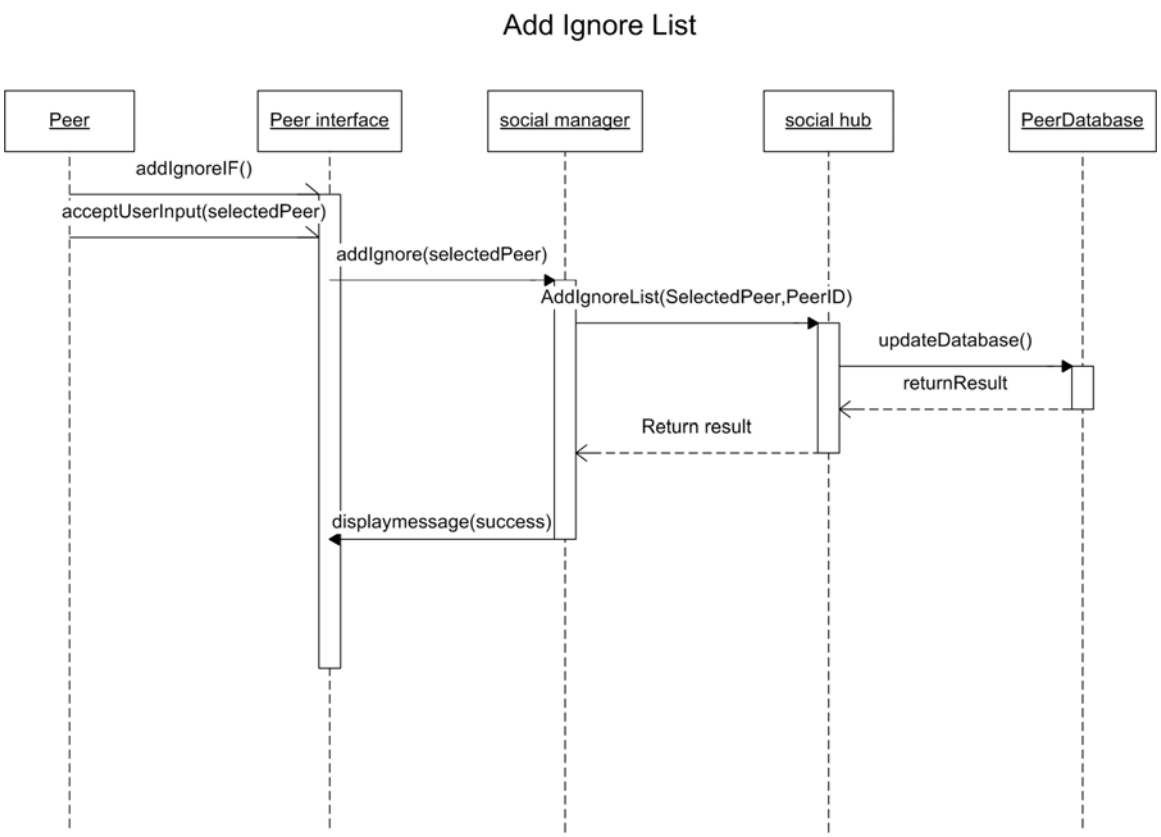

Figure 9: Add Ignore List Sequence Diagram

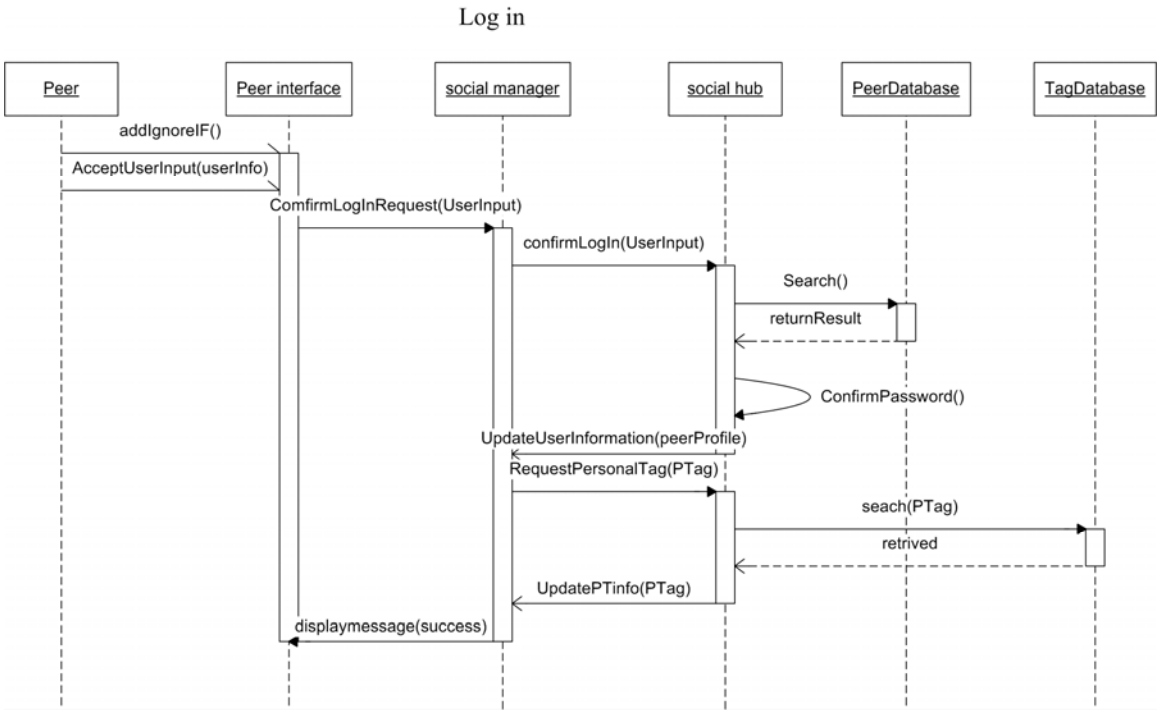

Figure 10: Login Sequence Diagram

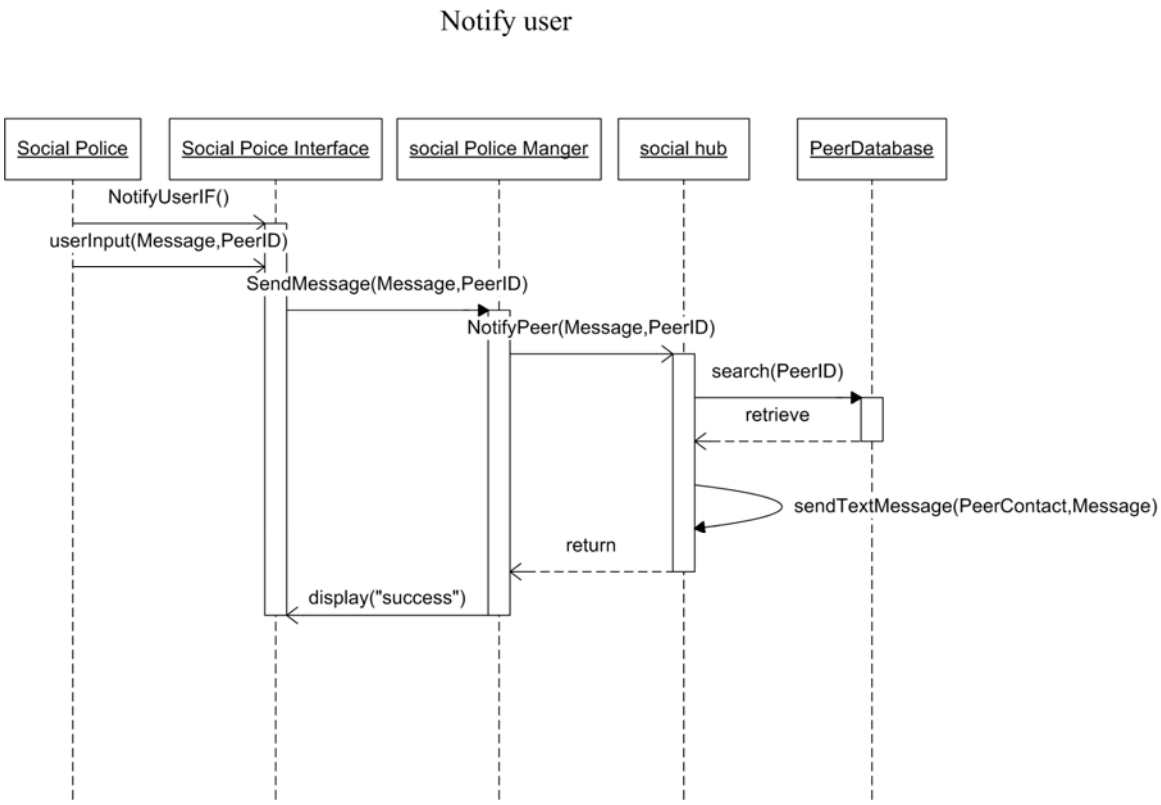

Figure 11: Notify User Sequence Diagram

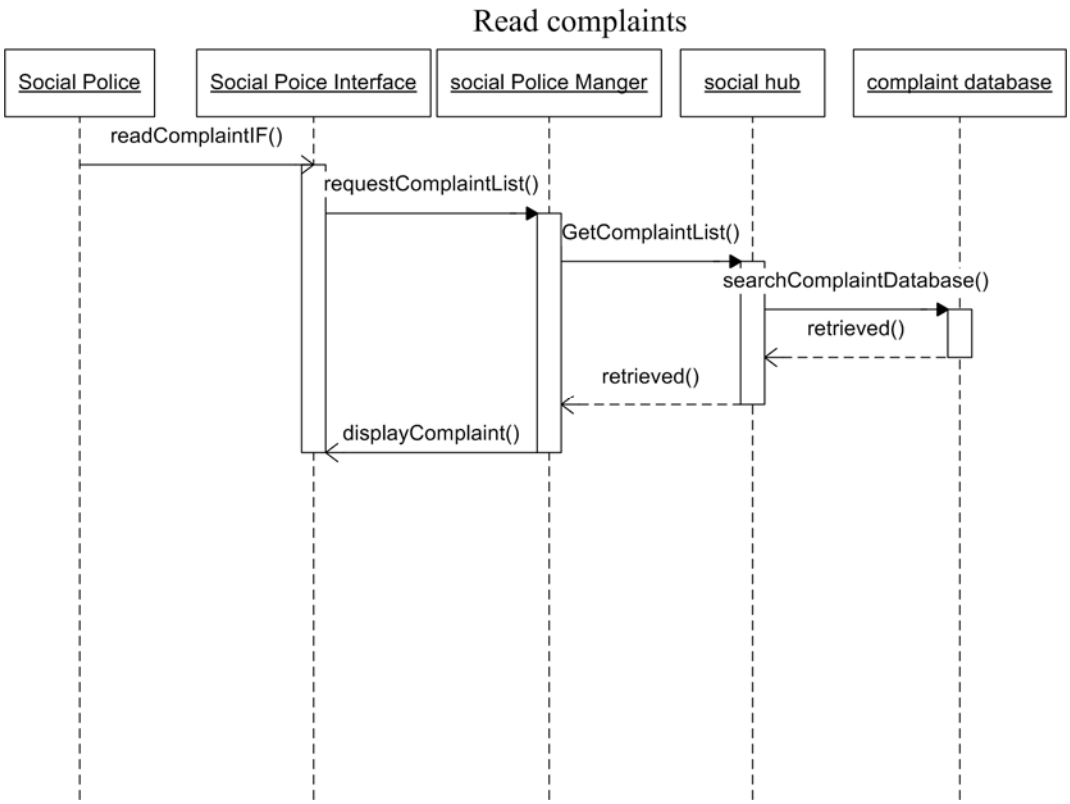

Figure 12: Read Complaints Sequence Diagram

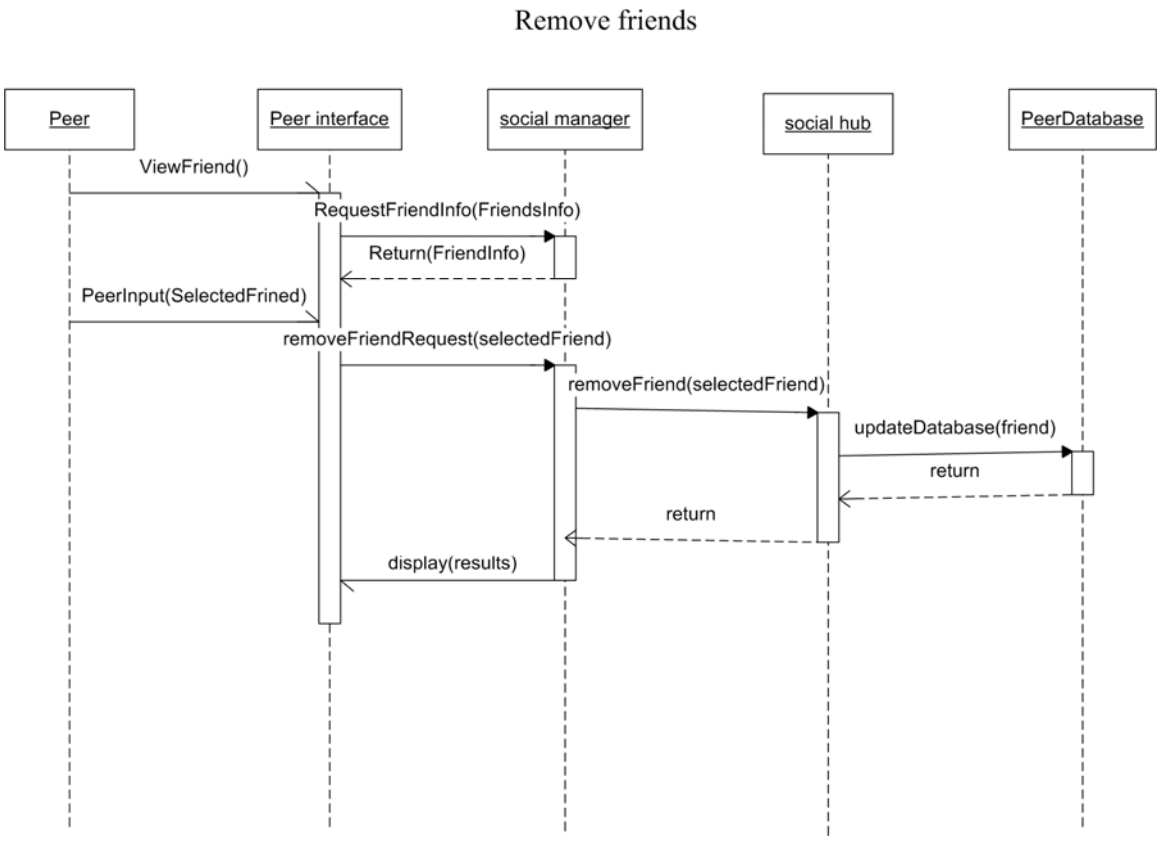

Figure 13: Remove Friends Sequence Diagram

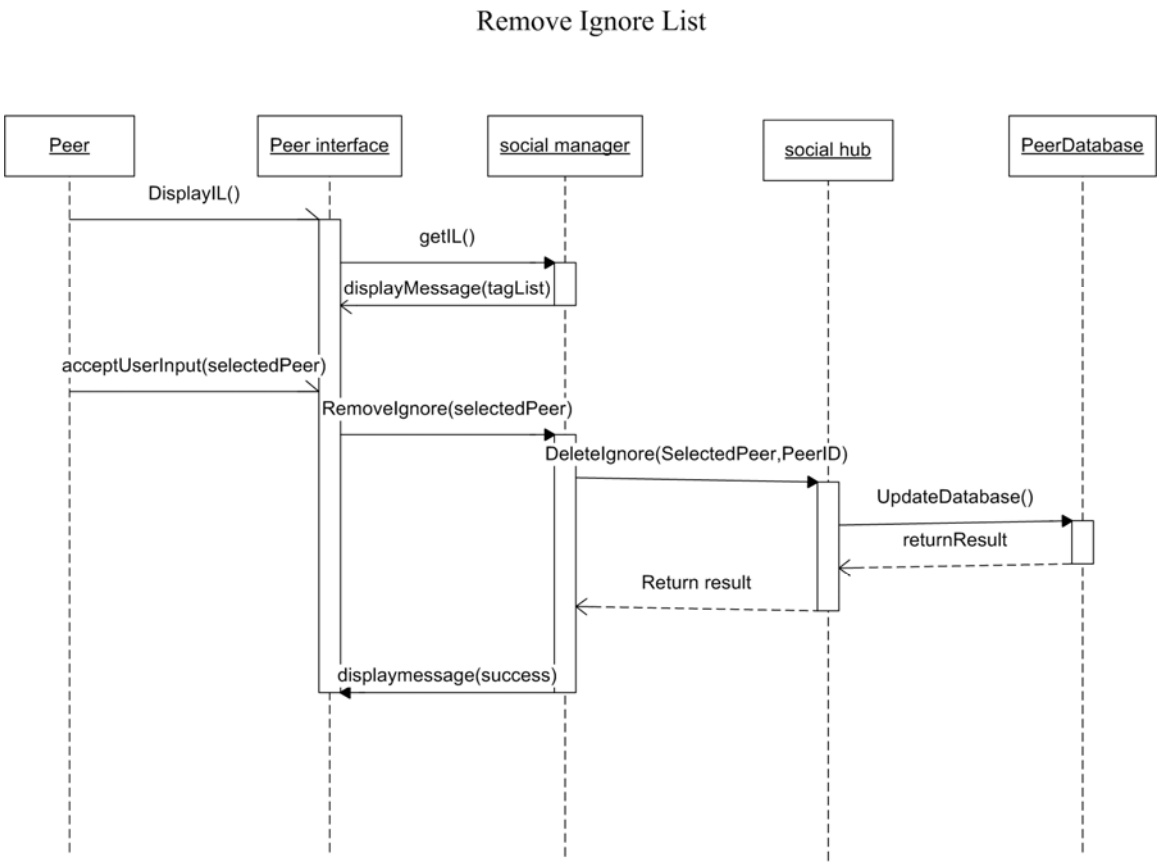

Figure 14: Remove Ignore List Sequence Diagram

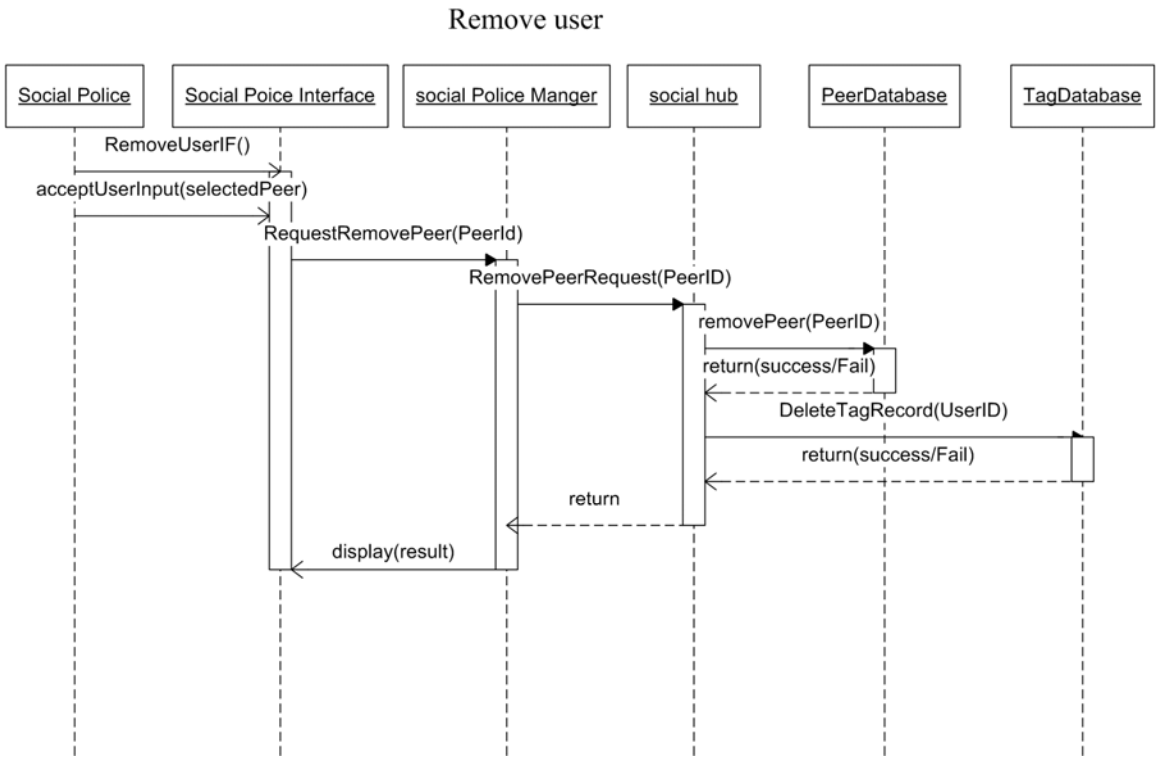

Figure 15: Remove User Sequence Diagram

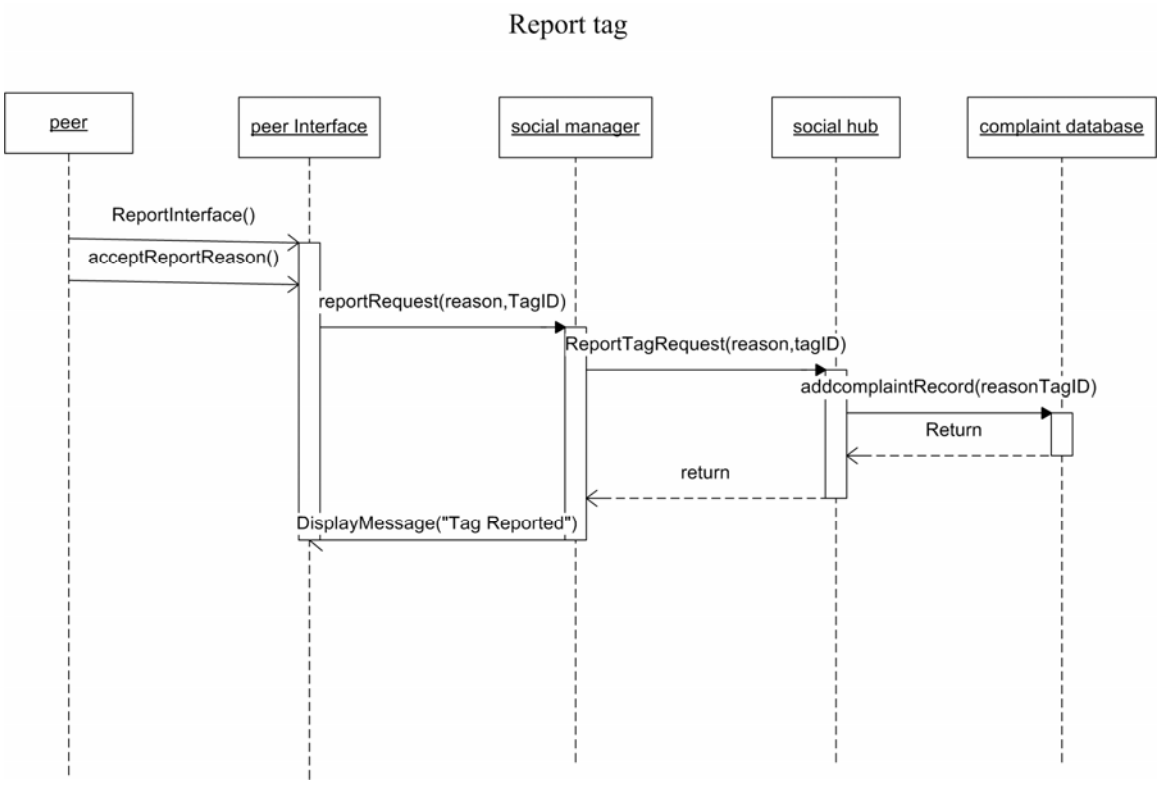

Figure 16: Report Tag Sequence Diagram

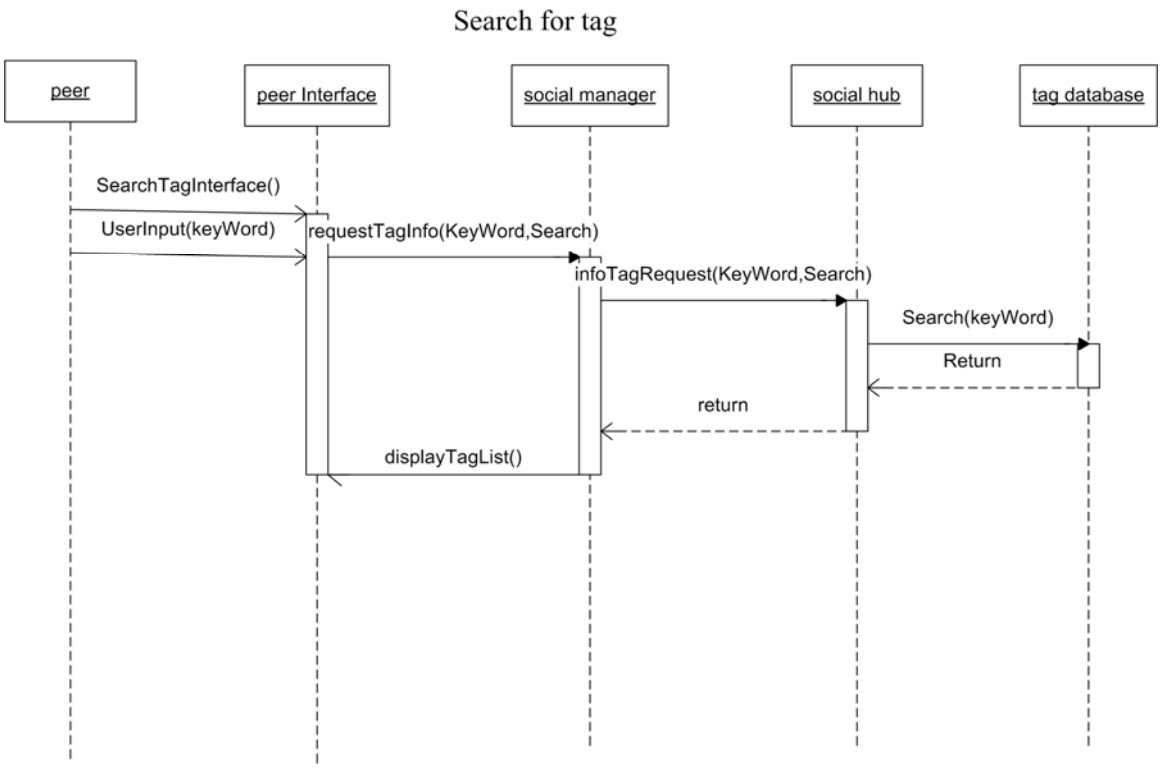

Figure 17: Search for Tag Sequence Diagram

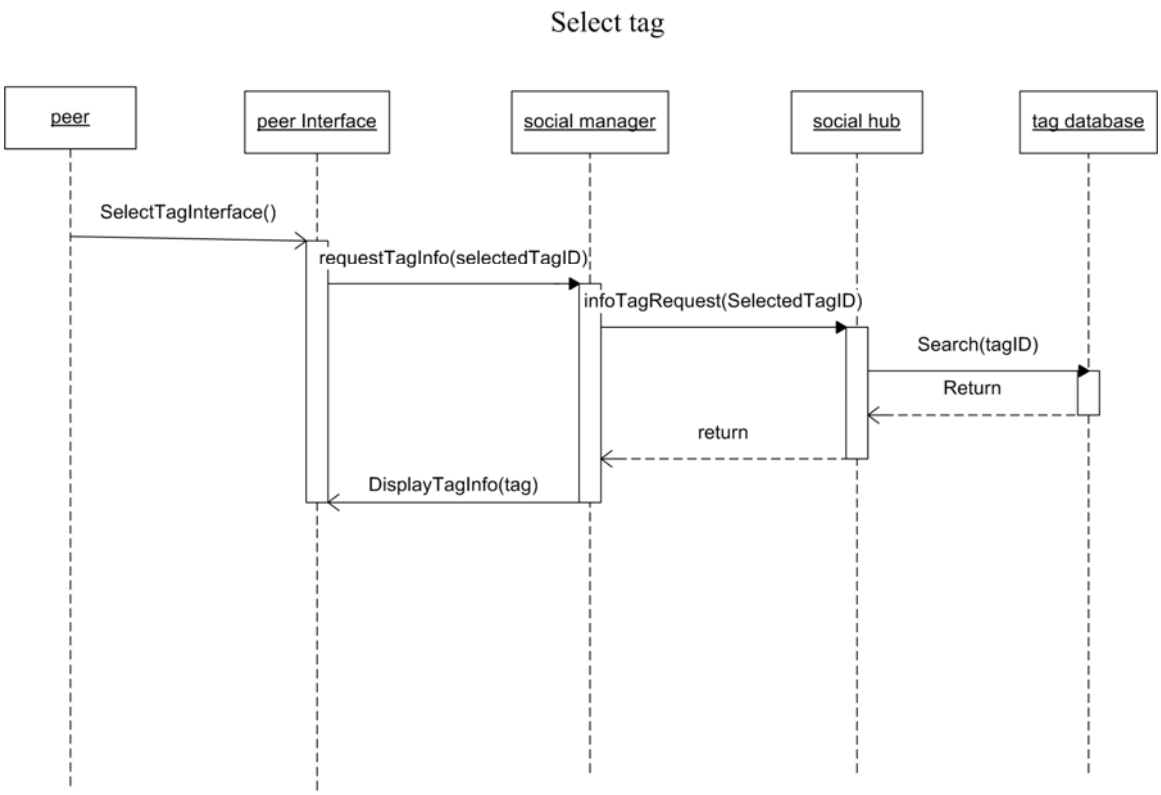

Figure 18: Select Tag Sequence Diagram

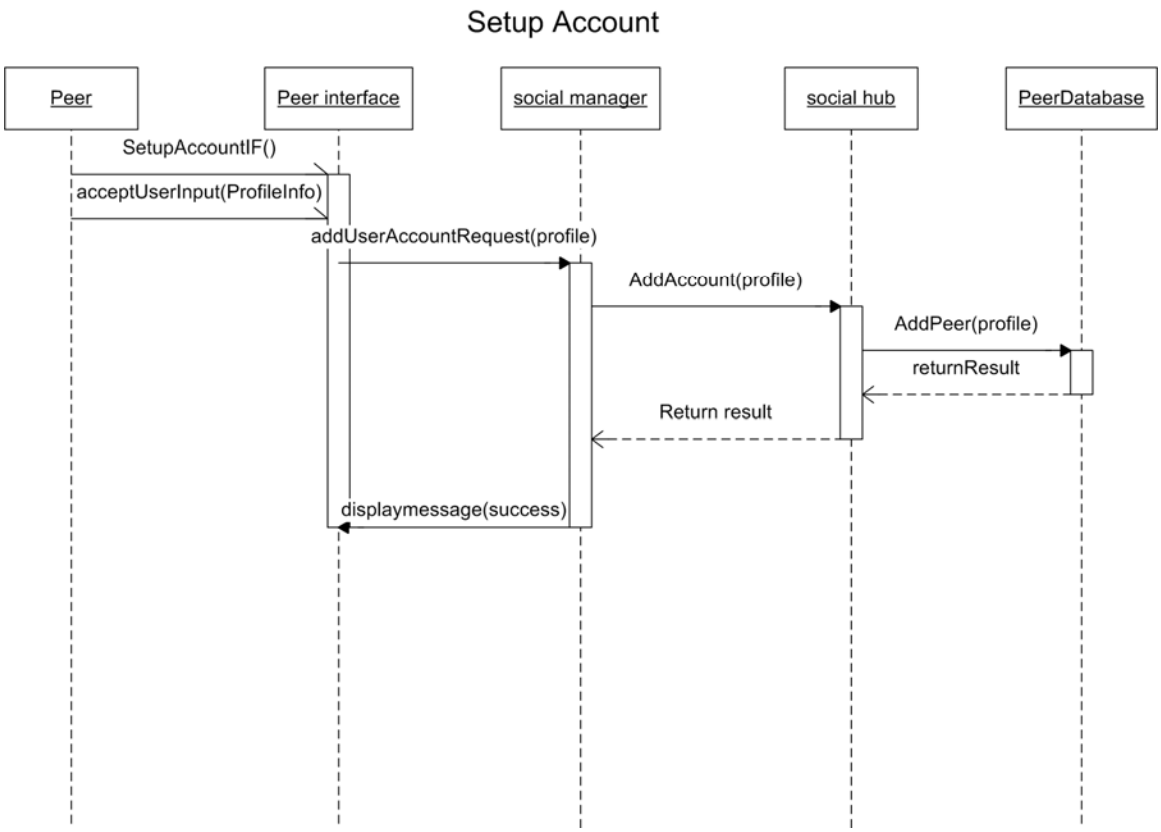

Figure 19: Setup Account Sequence Diagram

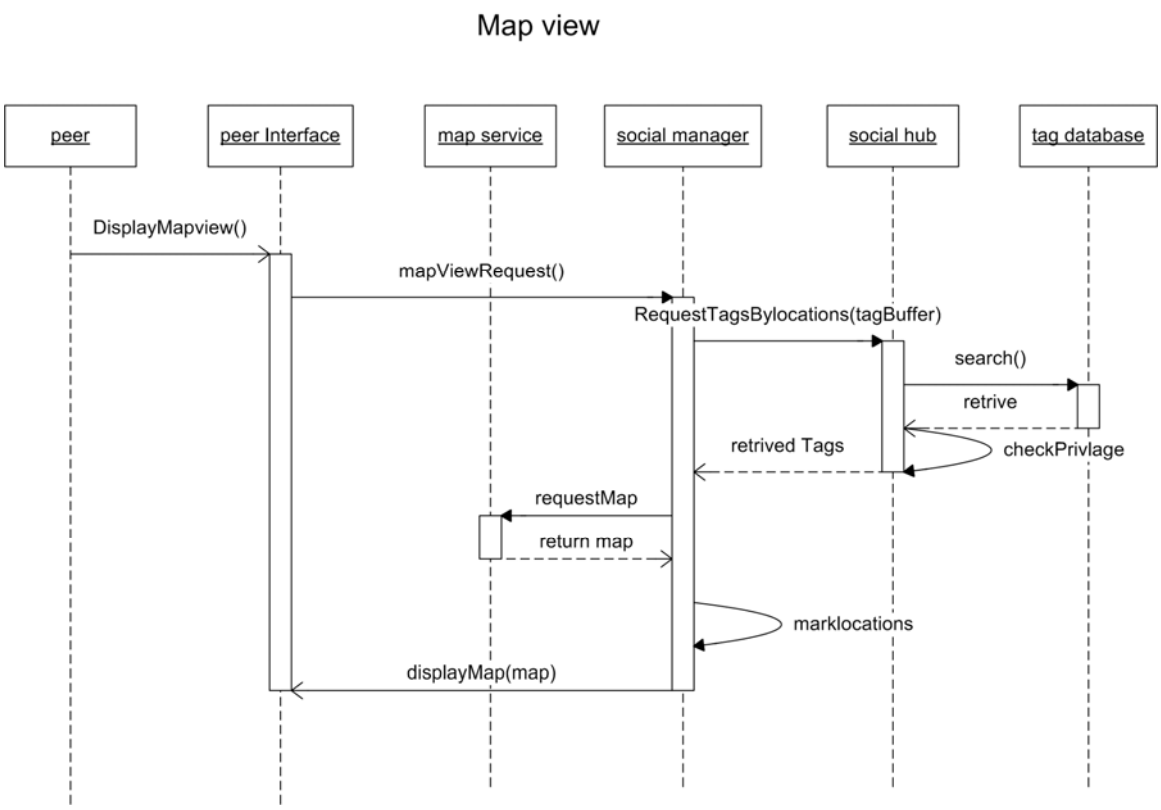

Figure 20: Map View Sequence Diagram

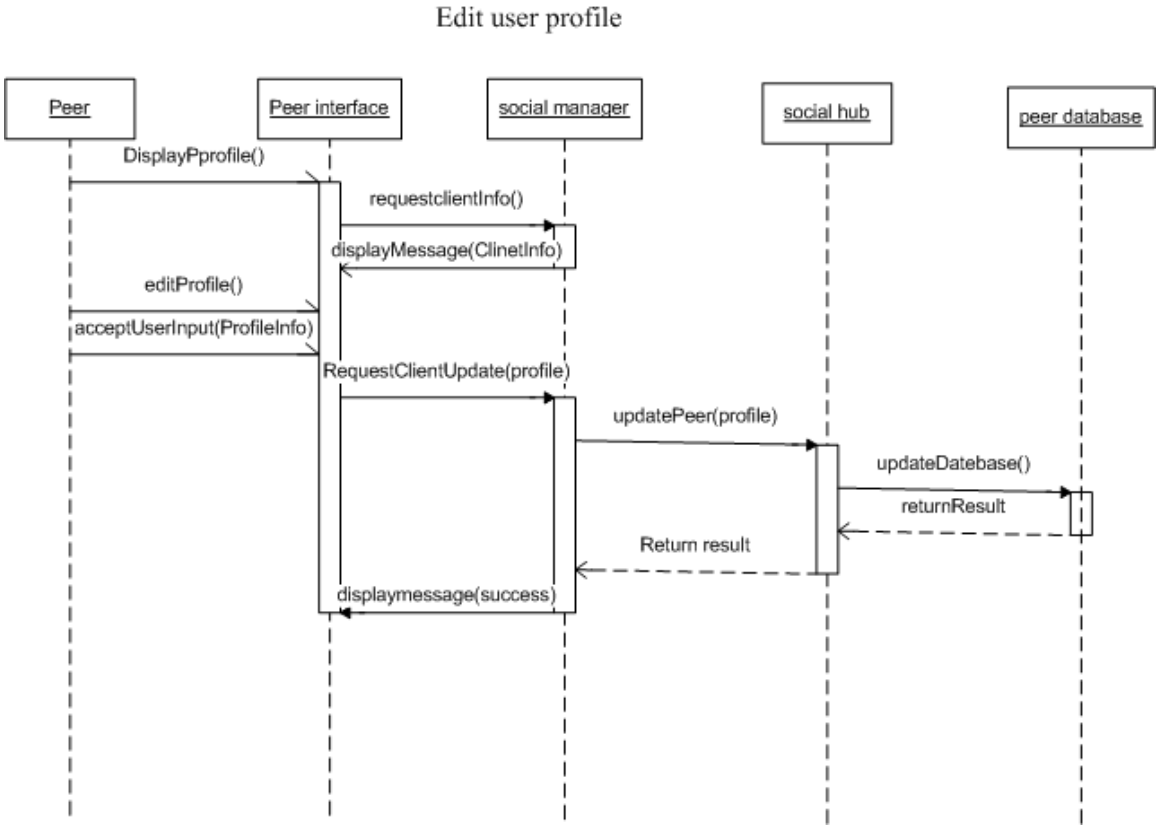

Figure 21: Edit Peer Profile Sequence Diagram

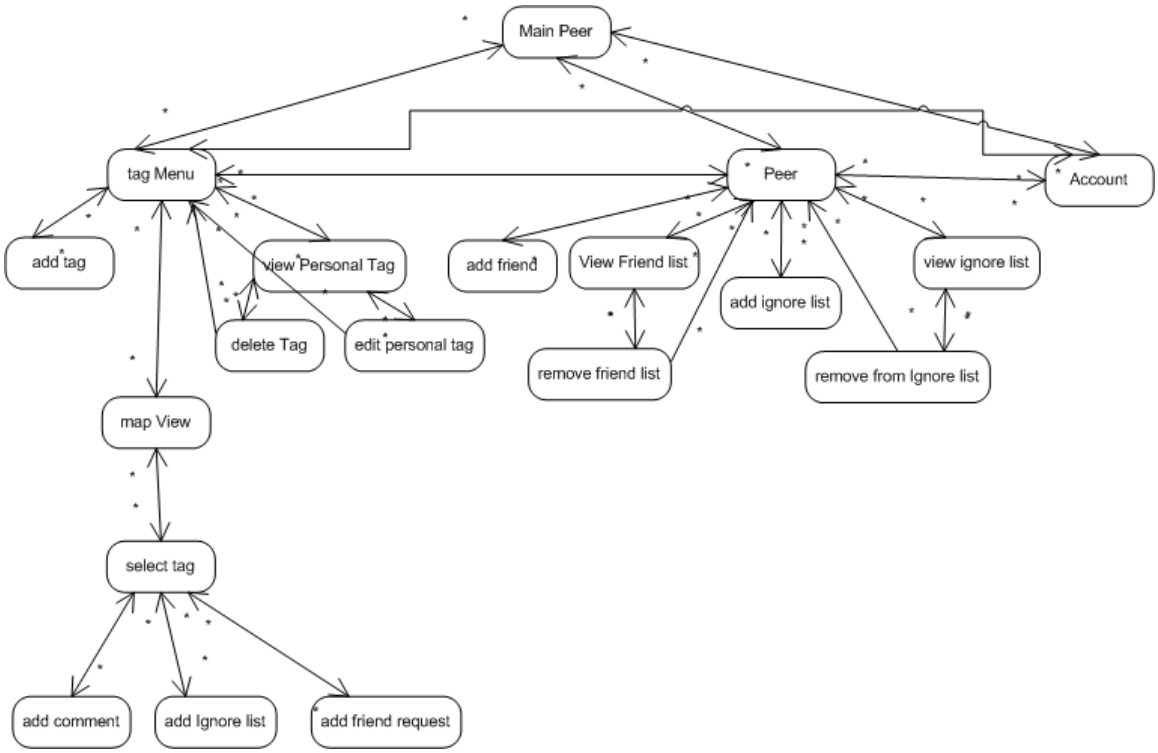

Figure 21: State Diagram for peer

Social Manager

MapViewReqest(interfaceSrcreen & tagInterface, gpsLocation Location)

{

soical_Hub.connect();

result = soical_Hub.RequestTagByLocaions(Location, buffer,peer.ID);

if(result)

{

image = marklocatMarkLocaion(tagbuff);

tagInerface.displayMap(image);

}

else

{

```
                interfaceScreen.Display("Connection Failed")
        |
}
getPT(interfaceSrcreen & tagInterface)
{
        tagInterface.DisplayMessage(peer.PTag);
}
DeleteTagRequest(string PTselected, interfaceSrcreen & tagInterface)
{
        Social_HubConnect();
        bool result = removeTagRequest(peerInfo,PTselected);
        if(result)
        {
                tagInterface.displaymessage("Success");
        }
        else
        {
                tagInterface.displaymessage("fail");
        }
}
AddTagRequest(tag tagInfo, interfaceScreen & tagInterface)
{
        Social_Hub.Connect();
        bool result = social_HUb.addTag(tag);
        if(result)
```

```
        {
                tagInterface.displaymessage("Success");
        }
        else
        {
                tagInterface.displaymessage("fail");
        }
}
RequestTaginfo(string keyWord, string Search,  interfaceScreen & tagInterface)
{
        Social_Hub.connect()
        tagBuff = infoTagRequest(keyWord, Search);
        DisplaytagLIst(TagBuff)
}
reportRequest(string Reason, stirng tagID, interfaceScreen & TagInterface)
{
        Social_Hub.connect();
        bool result = Social_Hub.reportTagRequest(string Reason, stirng tagID);
        if(result)
        {
                tagInterface.displaymessage("Success");
        }
        else
        {
                tagInterface.displaymessage("fail");
```

```
        }


}
addFriendRequest(peerInfo friend, interfaceScreen & friendInterface)
{
        Social_Hub.connect();
        bool result = Social_Hub.addFriend(friend);
        if(result)
        {
                friendInterface.displaymessage("Success");
        }
        else
        {
                friendInterface.displaymessage("fail");
        }


}
RequestFriendInfo(interfaceScreen & friendInterface)
{


        friendInterface.displayFriend(peer.friends);


}
//remove in this function is a predefined data type function
RemoveFriendRequest(string friendInfo, interfaceScreen & friendInterface)
```

```
{
        Social_Hub.connect();
        bool result = Social_Hub.removeFriendRequest(friendInfo);
        if(result)
        {
                friendInterface.displaymessage("Success");
                peer.friends.Remove(friendInfo);
        }
        else
        {
                friendInterface.displaymessage("fail");
        }
}
requestTagUpdate(tag TagInfo, interfaceScreen & tagInterface)
{
        Social_Hub.connect();
        bool result = Social_Hub.UpdateTagRequest(TagInfo);
        if(result)
        {
                tagInterface.displaymessage("Success");
        }
        else
        {
                tagInterface.displaymessage("fail");
        }
```

```
}
GetIL(interfaceScreen & friendInterface)
{
	FriendInterface.DisplayIgnoreList(peer.IgnoreList);
}
RemoveIgnore(string ingoreInfo, interfaceScreen & friendInterface)
{
	Social_Hub.connect();
	bool result = Social_Hub.DeleteIgnore(peer.PeerID, ignoreInfo);
	if(result)
	{
		friendInterface.displaymessage("Success");
		peer.Ignore.Remove(ignoreInfo);
	}
	else
	{
		friendInterface.displaymessage("fail");
	}
}
// type can bee ID
AddIgnore(string PeerInfo, interfaceScreen & friendInterface)
{
	Social_Hub.connect();
	bool result = Social_Hub.AddIgnoreList(PeerInfo ,peer.PeerID);
	if(result)
```

```
        {
                friendInterface.displaymessage("Success");
                peer.Ignore.Add(ignoreInfo);
        }
        else
        {
                friendInterface.displaymessage("fail");
        }
}
bool ComfireLoginRequest(string PeerPhone, string password)
{
        peer.Phnumb = PeerPhone;
        peer.Pword = password;
        SocialManager.Social_Hub = new social_Hub(peer.phnumb, peer.Pword);
        bool result = Social_Hub.connect();
        return result;
}
updateUserInformaion(peerInfo, interfaceScreen & profileInterface)
{
        Social_Hub.connect();
        bool result = social_Hub.updatePeer(peerInfo);
        if(result)
        {
                profileInterface.displaymessage("Success");
```

```
        }
        else
        {
                profileInterface.displaymessage("fail");
        }

}

tag RequestTagInfo(string tagID)
{
        Social_Hub.connect();
        taginfo = social_Hub.InfoTagRequest(tagDI);
        return tagInfo;
}
bool addUserAccountRequest(string PhoneNumber, String Password)
{
        Social_Hub.connect();
        bool result = Social_Hub.addAccount(PhoneNumber, Password);
        return result;

}
```

Figure 22: pseudo-code for Social Manager

```
bool requestTagByLocaion(buffer, location,peerID)
{
        tagDB.Connect();
```

```
        HighX = location.x +5;

        LowX  = location.x -5;

        HightY = location.y +5;

        LowY = location.y - 5;

        string query1 = "select from tag where Geotagx > LowX and Geotagx < HightX and
Geotagy < HighY and Geotagy > LowY";

        result = tagDB.search(Query1);

        CheckPrivlage(tagList, PeerID);

        buffer = tagList;



}

CheckPrivlage(tagList, PeerID)

{

        peerDB.Connect();

        string query1 = "select * from peer" + PeerID;

        list<peer> FriendList = peerDB(query1);

        list<tag> FriendTag;

        for(int x =0; x <FriendList.count; x++)

        {

                for(int y = 0; y < tagList.count; y++)

                {

                        if(tagList[y].Owner == FriendList[x] && tagList[y].Privlage == Friend)

                        {

                                FriendTag.add(tagList[y]);

                        }

                }
```

```
        }
        for(int x =0; x < tagList.count;x++)
        {
                if(tagList[x].privlage==private && tagList[x].owner != PeerID)
                {
                        tagList[x].remove();
                }
        }
        for(int x=0; x<FriendTag.count;x++)
        {
                tagList.add(FriendTagp[x]);
        }
}
RemoveTagRequest(tagID)
{
        tagDB.Connect();
        string Query1 = "delete from tag where tagID ="+tagID;
        bool result = tagDB.deleteTagRecord(Query1);
        return result;
}
AddTag(tagInfo)
{
        tagDB.connect();
        string Query1 = "insert into tag Geotagx  = " +tagInfo.Geotagx + ", Geotagy = "
+tagInfo.Geotagy", tagID = "tagInfo.TagID", Privlage = "+ tagInfo.Privlage;
        bool result= tagDB.addTagRecord(Query1);
```

```
        return result;
}
UpdateTagRequest(tagInfo)
{
        tagDB.connect();
        string Query1 = "update tag tag Geotagx  = " +tagInfo.Geotagx + ", Geotagy = "
+tagInfo.Geotagy", Privlage = "+ tagInfo.Privlage " where  tagID = "tagInfo.TagID;
        bool result = tagDB.updateTagDatabase(Query1);
        return result;
}
InfoTagRequest(tagID)
{
        tagDB.connect();
        string query1 = "select from tag where tagID = " +tagID;
        tag = tagDB.serach(query1);
        return tag;
}
ReporttagReqeust(tagID, Reason)
{
        complainDB.connect()
        string query1 = "insert into complain tagID = " + tagID +" reason = " +Reason;
        bool result;= complainDB(query1);
        return result;
}
AddFriend()
{
```

```
}

UpdatePeerFriendList()

RemoveFriend()

UpdateTagRequest()

UpdatePeer()

DeleteIgnore()

AddIgnoreList()

addAccount()

ConfirmLogin()

RequestPersonalTag()

InfoTagReqeust()

GetComplaintList()

RemovePeerRequest()

NotifyPeer()

SendTextMessage()
```

Figure 23: pseudo-code for Social Hub

Figure 21: Login Interface

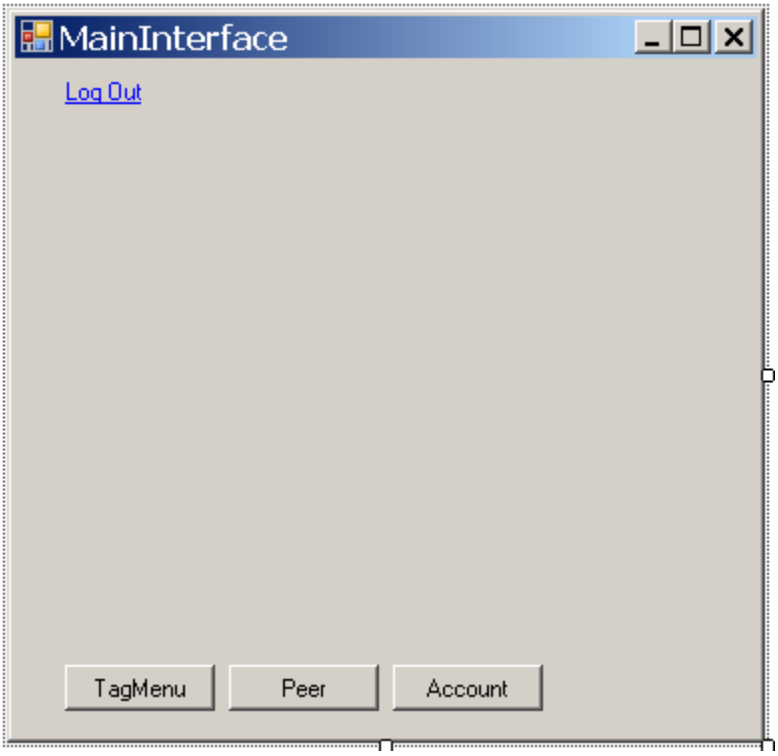

Figure 21: Main Menu Interface

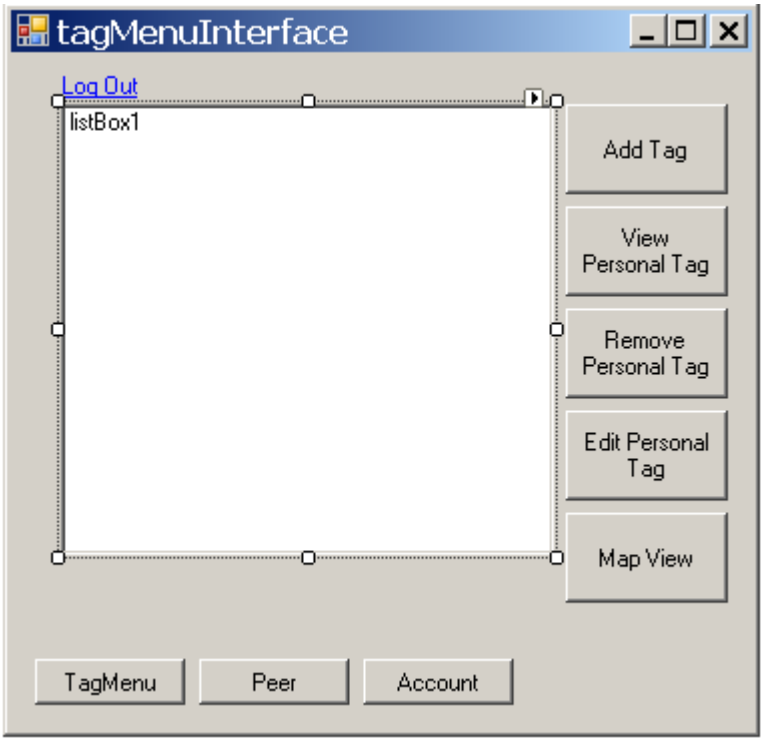

Figure 21:Tag Menu Interface

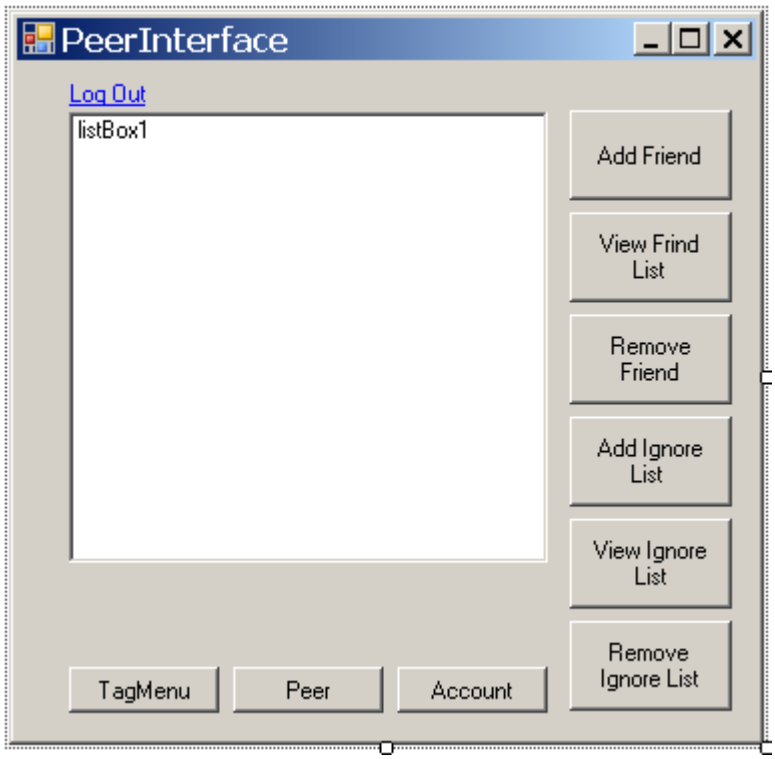

Figure 21: Peer Menu Interface

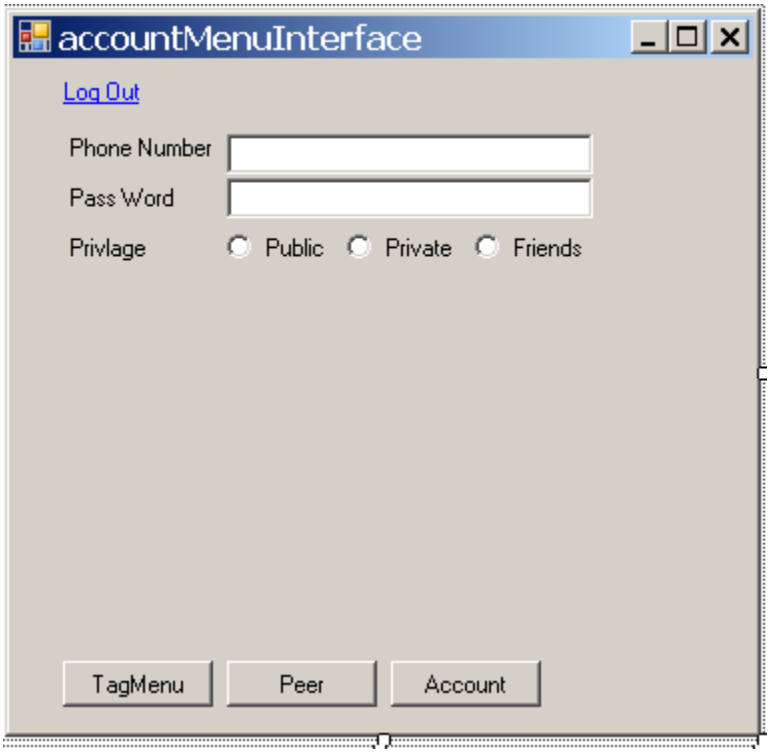

Figure 21: account Menu Interface

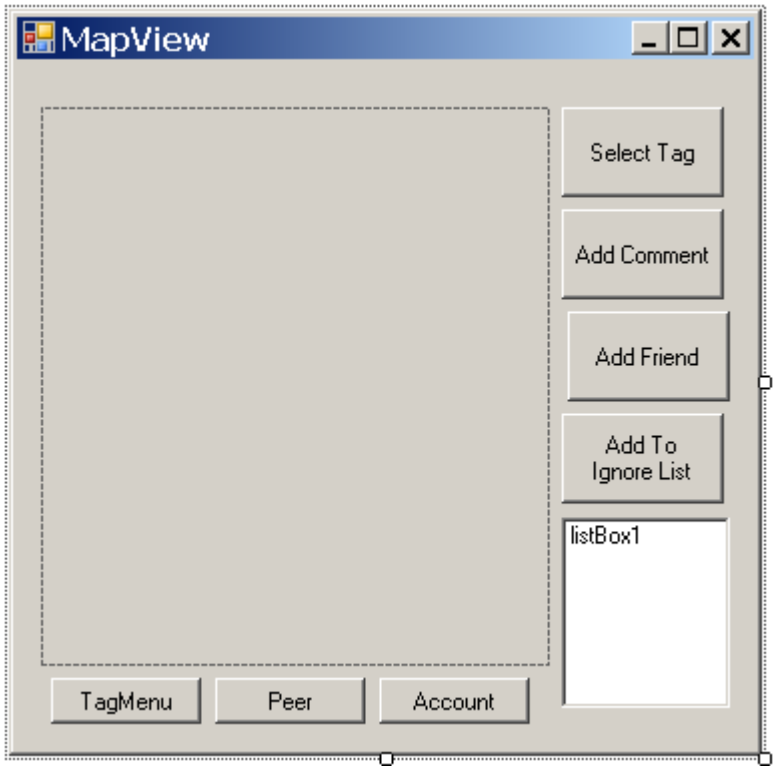

Figure 21: Map Menu Interface