NERS PROJECT

SOFTWARE DESIGN DOCUMENT

(In accordance with IEEE 830-1998)

v1.0

Code Whisperers

Mustafa Murat Coşkun - 1881143
Mehmet Gençol -1881226
Oğuz Artıran -1881002
Ahmet Melih Gedikli - 1881200

1. INTRODUCTION

This document is a Software Design Document for the Android Mobile Application named “NERS”.This document is prepared by the following IEEE conventions for software requirement specification.This document includes all the functions and specifications with their explanations to solve related problems as a project of METU CENG department.

1.1 SCOPE

NERS is basically an Android application project. NERS is a social platform based on event sharing, rating, classifying, ordering with respect to distance,type, date and showing them on map. In addition, it allows users to be friend, to message with each other and to socialize with each other through events. The scope of this document is design issues of NERS which are where is NERS deployed, what technologies are used in NERS , data structure of NERS and interaction flow of NERS.

1.2 PURPOSE

Purpose of this software design document is providing complete description of the features which will be implemented in this location-based social platform.
This document aims to describe and visualize the design and architecture of NERS by using different viewpoints and describe the application system which is structured to meet the needs specified in SRS document. This document will be the primary reference for the implementation phase.In addition,this document includes class hierarchies and interactions, data flow and design, algorithmic models, design constraints and restrictions user interface designs and design concerns In short, this document is meant to equip the reader with a solid understanding of the inner workings of NERS application.

1.3 INTENDED AUDIENCE

This document aims to inform:
Developers: With the information in this document, developers can easily understand the software and it helps developers to improve the current project features or add new features to existing system.

Testers: With the information in this document, testers can find some bugs for their testing strategy. Without running application, testers can easily find the design bugs.

Users: By providing this document to end-users, users can easily form opinions about how to use the software. For our mobile application, our expected users are 18-25 age university students, student clubs, small entertainment businesses and young metropolis workers.

1.4 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

SDD	Software Design Document
GCM	Google Cloud Messaging
GUI	Graphical User Interface
OS	Operating Systems
GPS	Global Positioning System
User	Anybody who uses the application NERS
IEEE	Institute of Electrical and Electronic Engineers
External Usage	Android Disk Usage
API	Application Programming Interface
SDK	Software Development Kit

1.5 REFERENCES

IEEE. IEEE Std. 830-1998 IEEE Recommended Practice for Software Design Document. IEEE Computer Society, 1998.

Class Diagram

2. SYSTEM OVERVIEW

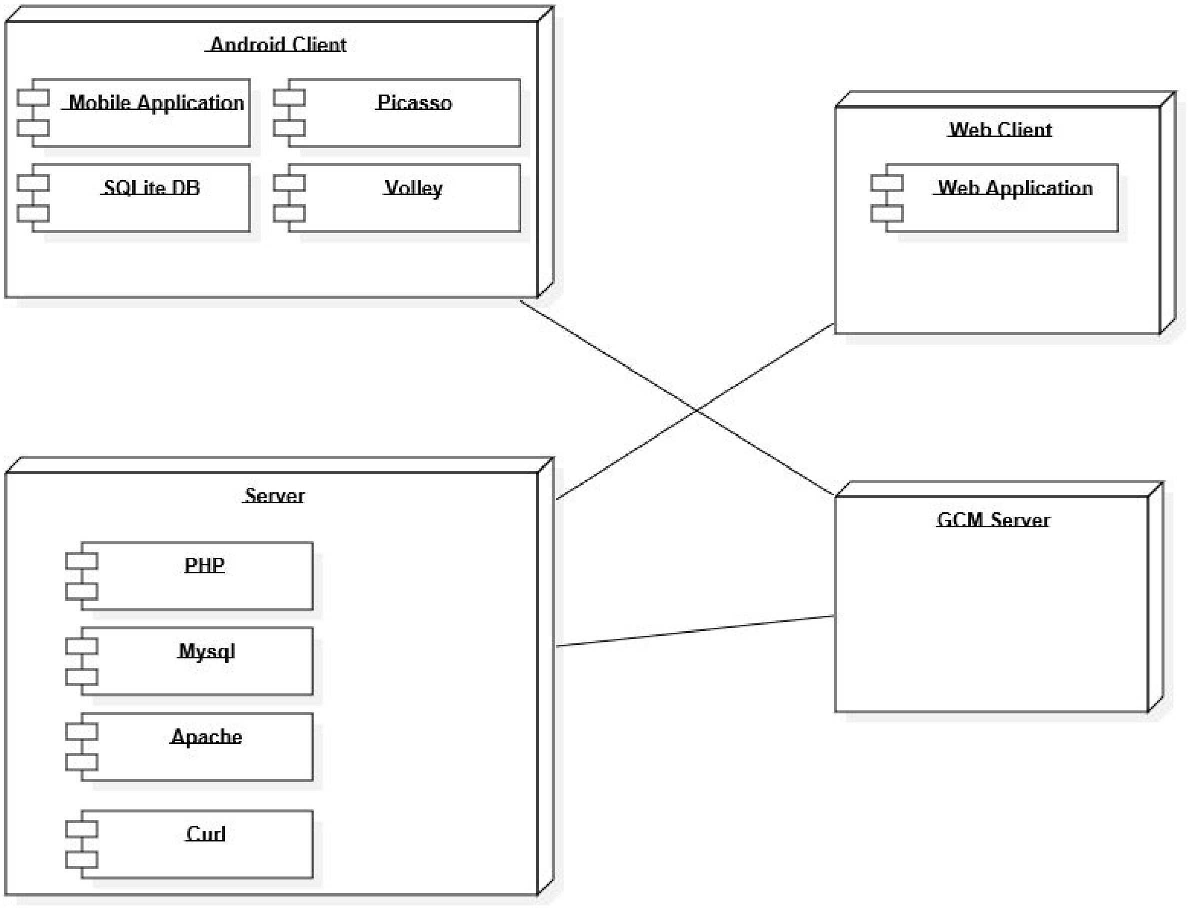


Figure 2.1

This chapter of the SDD is about system overview of the project. The deployment diagram of NERS is provided in **Figure 2.1**.

Android Client: An android mobile phone is the main deployment element in the project. The project is written in Java and the application will be installed into an android mobile phone. Users will use the application in an android mobile phone.

Mobile Application: Mobile application is the project which is this design document for. Application uses **volley** to get data from database and connect to server, **picasso** to get the bitmap of picture and load these pictures into application and **SQLite database** to save the data which are not crucial for the application like user messages and notifications.

Web Client: Web application is for privileged users to add promotions of their businesses. The regular users can't use this client. Privileged events will be added by using this client. PHP,Jquery and javascript are used in developing web application.

Server: Linux apache server is used to respond the client requests. Data requests are handled in server PHP, Mysql Database and PHP curl is used in server side.

GCM server: Google Cloud Messaging server is a world wide server which serves to manage notification sending and handling. It is an unbound server from our project. GCM server will be used in registration and sending notification process. PHP curl will be used to awake the GCM server.

3. SYSTEM ARCHITECTURE

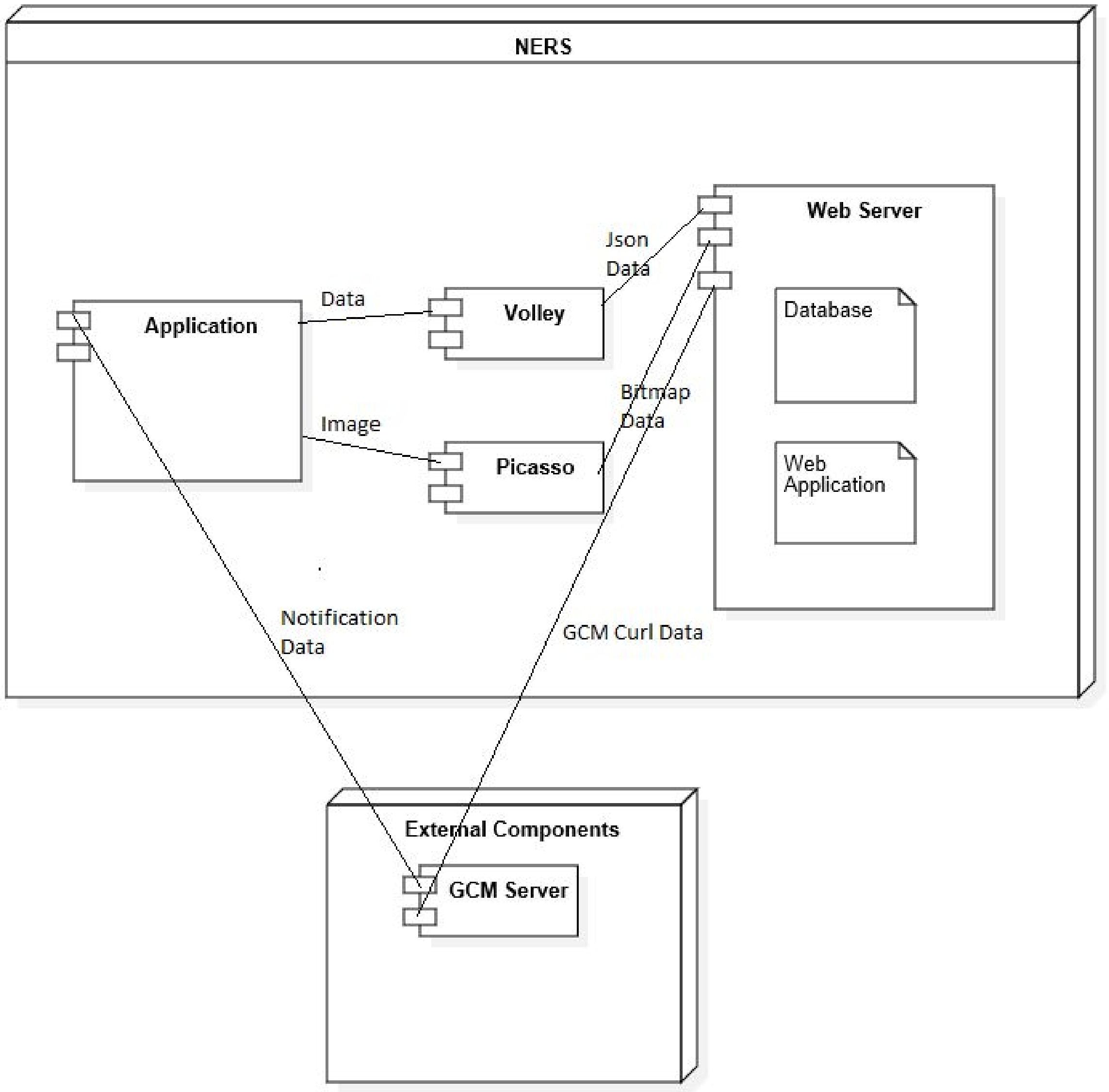


Figure 3.1

This chapter of the SDD is about architectural view of the project. The component diagram of NERS is provided in **Figure 3.1**. These components can also be seen in deployment diagram. Usage of each component in this section will be detailed.
Since, NERS is a social platform it requires internet connection and data flow from internet. Application needs data and sends data continously in short time intervals **volley** library as component handles the server and database connection, specifies the methods(PUT,GET...), and if a data arrival is waited it waits asynchronously without interfering the main activity flow. The data flow is bi-directional in **volley**. Like **volley,**

picasso is also library which can arrange image resolution, image quality, fitting situation. With image url picasso can load the image as bitmap and allow developer to reshape that bitmap. **Picasso** has uni-directional data flow from server to client. GCM server gives an unique key for every application on a mobile phone. Therefore each application on a mobile phone can only have just one unique id. In registration process, GCM assigns a unique id to a phone, this id is recorded in the database. In each time for sending notification, this unique id is used for specifying a user. Application tells server to send a notification to a user. Server uses **Curl** library to send this request to GCM server.

4. DESIGN DESCRIPTION INFORMATION CONTENT

4.1 INTRODUCTION

Up to now general view, deployments of components and reasons of why a specific component is used has been clarified in second and third sections. Wİthout getting into detail, external view and explanation is given. Hereafter, the interior and detailed design will be explained.

4.2. SDD IDENTIFICATION

This document is a first version of System Design Description for this project. This SDD report prepared based on IEEE 1016-2009 standards. Star UML is used for drawing diagrams. Organization of the project team and date of the report are given in cover page of SDD. In the first section an overview of SDD is given. Lastly, for the design viewpoints including context, composition, logical, information, patterns use, interface, interaction, and resource viewpoints refer to the section 5.

4.3 DESIGN STAKEHOLDERS AND THEIR CONCERNS

Design stakeholders are the developer team of this project, their advisors and supervisors . Our stakeholders’ concerns can be listed as, the implementation should be safe and maintainable, the interface shall be easy and charming to read and use. The functionalities should work without any errors. Server can carry whole users at the one and same time. Database should be simple and efficient. Finally implementation should be open to new components and reusable.

4.4 DESIGN VIEWS

Representing the diagrams of view, UML is used. Design views of this SDD are design rational, contextual, composition, interface, logical and interaction views. These design views are governed by design viewpoints that are explained in chapter 5.

4.5 DESIGN VIEWPOINTS

There are different viewpoints of this project which will be explained in the next sections. Each viewpoint will be visualized with diagrams.

4.6 DESIGN ELEMENTS

- NERS uses a Digital Ocean repository for server need.
- Mysql database is used as database.
- Input/output in the project is handled with buttons, swipes and asynctasks.
- The application will be implemented in java and it uses Android JDK.
- Android Studio will be used as IDE.

4.7 DESIGN RATIONALE

In NERS design choices are made according to performance , reliability and self-sustainability. In our design, there will be no platform like Facebook API, Twitter API etc. which makes the project dependent to third party products. All used API’s are Google’s own API’s which are used in android developing.

4.8 DESIGN LANGUAGES

In this project, Unified Modeling Language (UML) is selected as a part of design viewpoint and it will be used for clarifying design viewpoints.

5. DESIGN VIEWPOINTS

In this chapter, the interior viewpoints of the NERS is explained in detail. During this section, UML diagrams will be used to increase understandability.

5.1 CONTEXT VIEWPOINTS

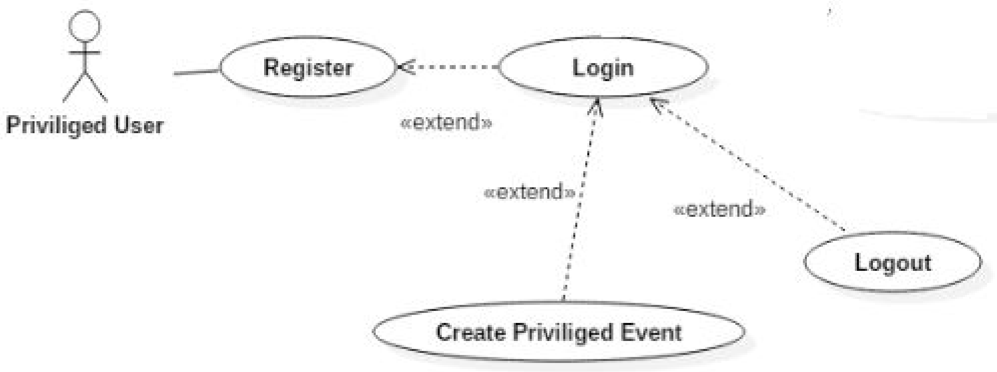


Figure 5.1.1

There are two user types in our system. One is regular android user and the other is privileged user which is in web application. The above use cases are explanatory for the context of these user types.

Use Case	**Description**	**Figure**
Registration	Privileged user should identify them to be privileged and register the system.	Fig-5.1.3
Login	Privileged user can log in to the system in order to use the system.	Fig-5.1.7
Create Privileged Event	Privileged user can create privileged event.	Fig-5.1.11

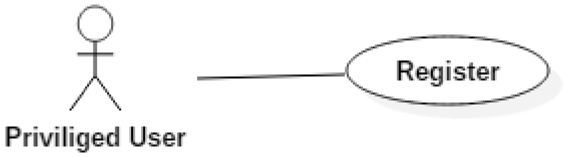


Fig - 5.1.3

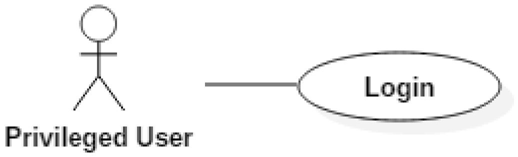


Fig - 5.1.7

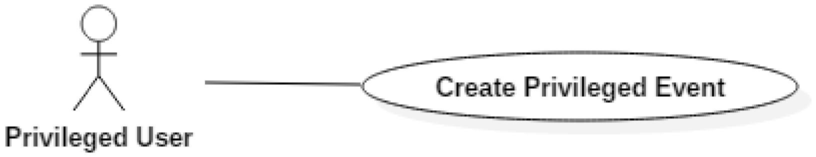


Fig - 5.1.11

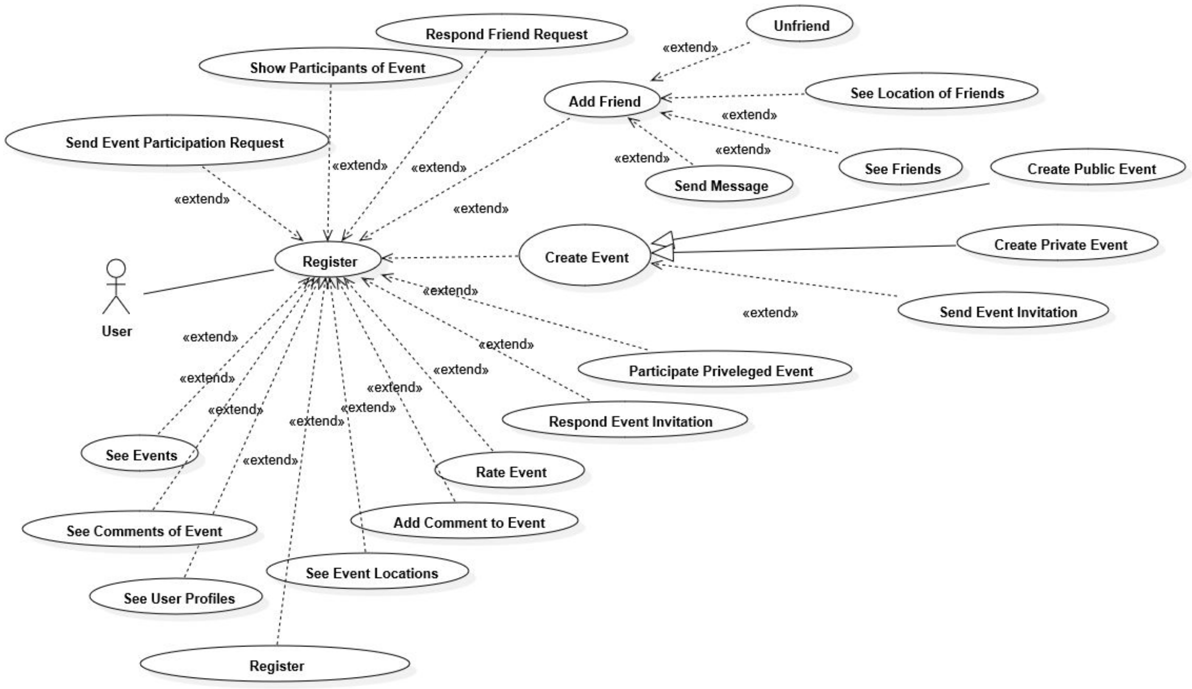


Figure 5.1.2

Use Case	**Description**	**Figure**
Register	User can register to NERS by verifying their phone numbers	**Fig - 5.1.31**
Add Friend	User can add people as friend	**Fig - 5.1.15**
Participate Privileged Event	User can be able to participate all privileged events	**Fig - 5.1.24**
See Event History	User can see the event history(participated/created)	**Fig - 5.1.19**
See User Profile	User can see user profiles	**Fig - 5.1.29**
Unfriend	User can delete people from their friend list	**Fig - 5.1.16**
See Location of Friends	User can see the location of their friends without any permission	**Fig - 5.1.17**
Send Message	Users can send message to their friends	**Fig - 5.1.18**
Create Public Event	Users can create public event both based on a privileged event or not	**Fig - 5.1.21**
Create Private Event	Users can create private event both based on a privileged event or not	**Fig - 5.1.22**
See Events	Users shall be able to see all events with the listview information, that they have permissions on map and in listview.	**Fig - 5.1.32**
Send Event Invitation	Users can send invitation to their friends while they are creating private and public events	**Fig - 5.1.23**
Rate Event	Users can rate events if they are participating events	**Fig - 5.1.26**
Show Participants of Event	Users can see the participations of the events if they are participating	**Fig - 5.1.13**
Send Event Participation Request	Users can send participation request to public events. The creator of the event will decide the participation	**Fig - 5.1.12**
Add Comment to Event	Users can add new comments to events if	**Fig - 5.1.27**

	they can see the events on the listview. The participation status is not checked in that situation.	
See Friends	Users can see the friends of theirs in an android listview.	**Fig - 5.1.20**
Respond Event Invitation	Users can respond the participation requests which they come to them.	**Fig - 5.1.25**
Respond Friend Request	Users can respond the friend requests which they come to them	**Fig - 5.1.14**
See Location of Events	Users can see the locations of events which can be listed to them.	**Fig - 5.1.28**
See Comments of Event	Users can see the comments of events without checking participation status if they can see the event on the list.	**Fig - 5.1.30**

User
Send Event Participation Requests

Fig - 5.1.12

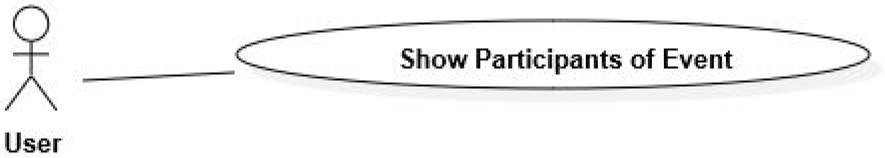


Fig - 5.1.13

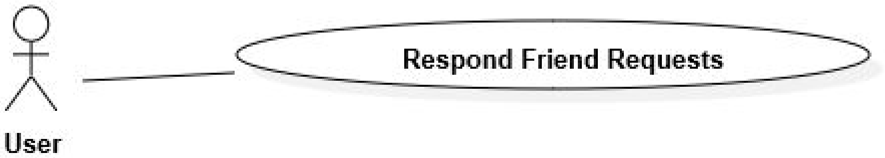


Fig - 5.1.14

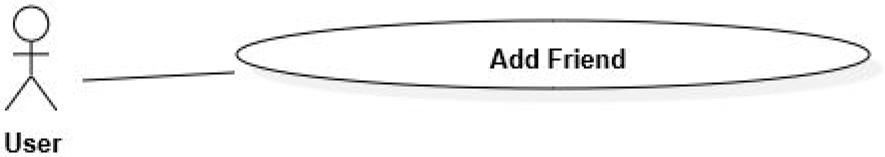


Fig - 5.1.15

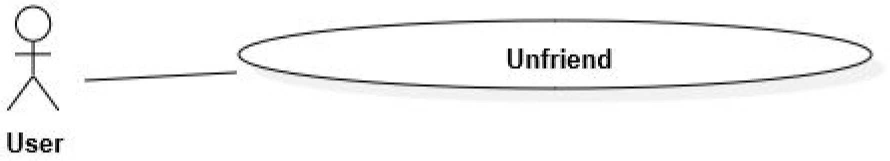


Fig - 5.1.16

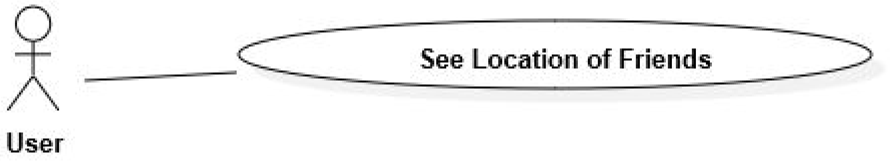


Fig - 5.1.17

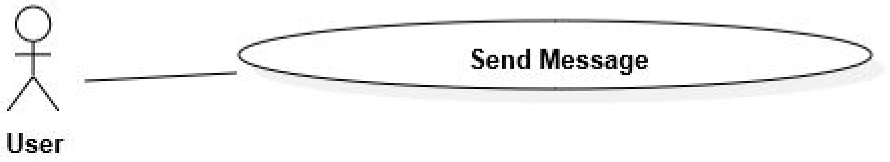


Fig - 5.1.18

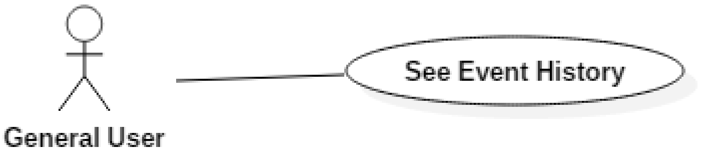


Fig - 5.1.19

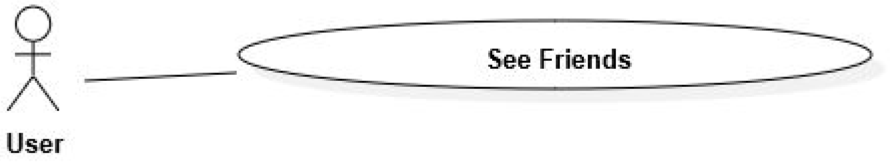


Fig - 5.1.20

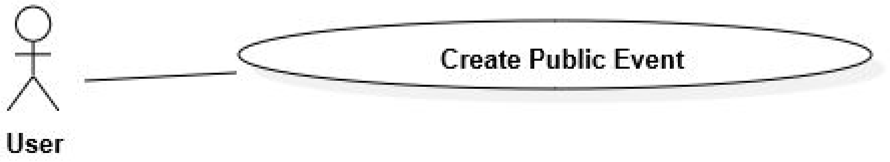


Fig - 5.1.21

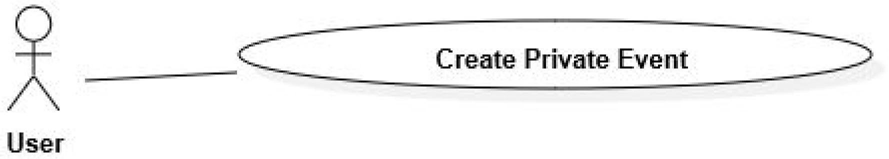


Fig - 5.1.22

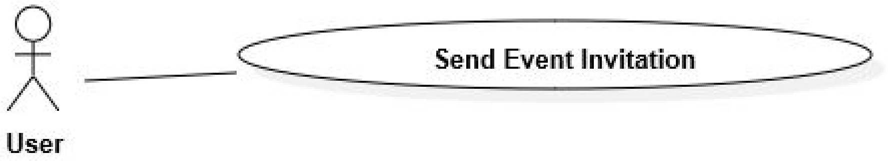


Fig - 5.1.23

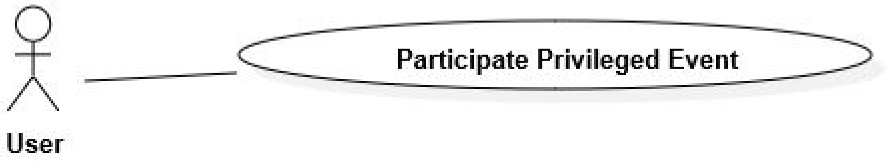


Fig - 5.1.24

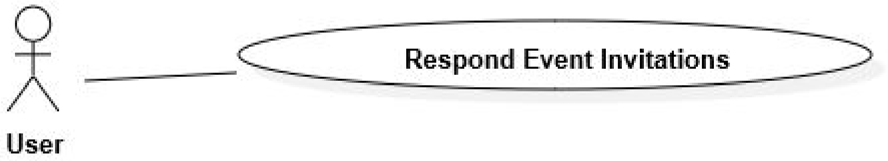


Fig - 5.1.25

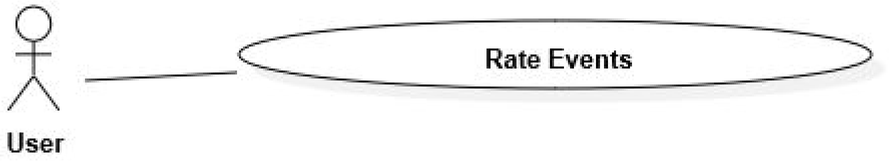


Fig - 5.1.26

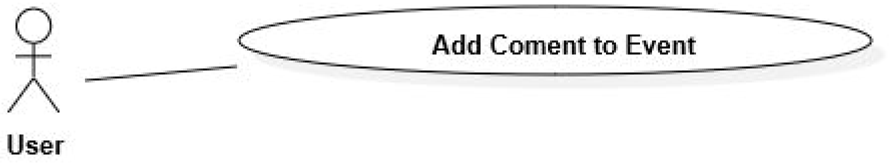


Fig - 5.1.27

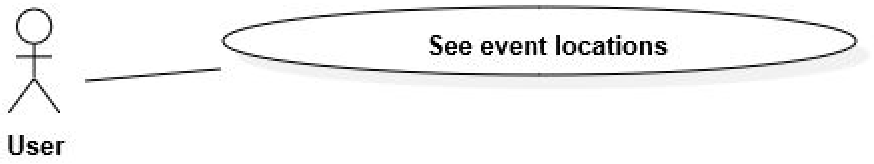


Fig - 5.1.28

User
See User Profiles

Fig - 5.1.29

User
See Comments of Events

Fig - 5.1.30

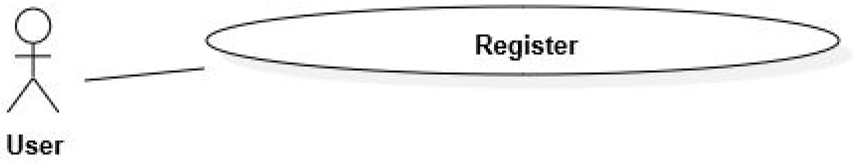


Fig - 5.1.31

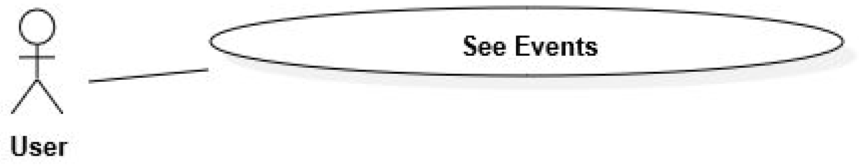


Fig - 5.1.32

5.2 LOGICAL VIEWPOINT

Logical viewpoint will identify all classes and relations between classes in NERS. Aim of logical viewpoint is to clarify and simplify the system design and lead development team. Firstly, a complete class diagram containing all classes and their relations are given. Then, each of the classes and their methods and fields are explained in detail.

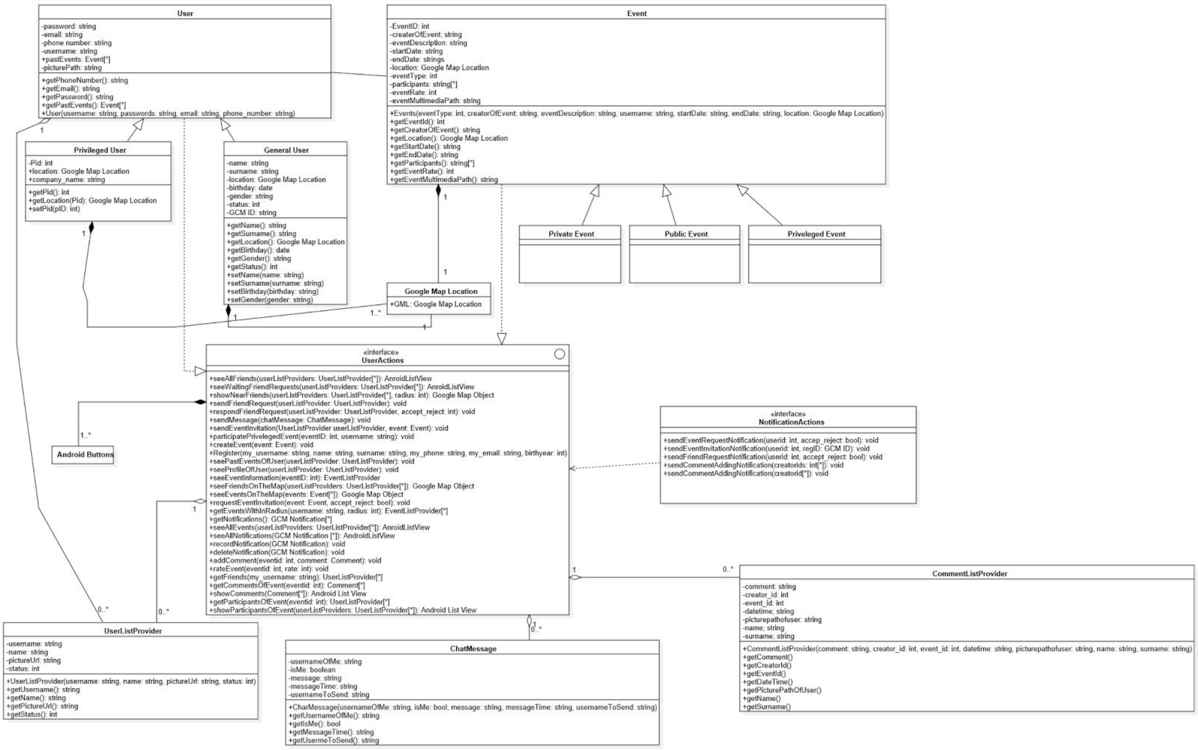

User
-password: string
-email: string
-phone number: string
-username: string
+pastEvents: Event[*]
-picturePath: string
+getPhoneNumber(): string
+getEmail(): string
+getPassword(): string
+getPastEvents(): Event[*]
+User(username: string, passwords: string, email: string, phone_number: string)
1
Privileged User
-Pid: int
+location: Google Map Location
+company_name: string
+getPid(): int
+getLocation(Pid): Google Map Location
+setPid(pID: int)
1
General User
-name: string
-surname: string
-location: Google Map Location
-birthday: date
-gender: string
-status: int
-GCM ID: string
+getName(): string
+getSurname(): string
+getLocation(): Google Map Location
+getBirthday(): date
+getGender(): string
+getStatus(): int
+setName(name: string)
+setSurname(surname: string)
+setBirthday(birthday: string)
+setGender(gender: string)
1
Event
-EventID: int
-createrOfEvent: string
-eventDescription: string
-startDate: string
-endDate: strings
-location: Google Map Location
-eventType: int
-participants: string[*]
-eventRate: int
-eventMultimediaPath: string
+Events(eventType: int, creatorOfEvent: string, eventDescription: string, username: string, startDate: string, endDate: string, location: Google Map Location)
+getEventId(): int
+getCreatorOfEvent(): string
+getLocation(): Google Map Location
+getStartDate(): string
+getEndDate(): string
+getParticipants(): string[*]
+getEventRate(): int
+getEventMultimediaPath(): string
1
1
Private Event
Public Event
Priveleged Event
Google Map Location
+GML: Google Map Location
1..*
1
«interface»
UserActions
+seeAllFriends(userListProviders: UserListProvider[*]): AnroidListView
+seeWaitingFriendRequests(userListProviders: UserListProvider[*]): AnroidListView
+showNearFriends(userListProviders: UserListProvider[*], radius: int): Google Map Object
+sendFriendRequest(userListProvider: UserListProvider): void
+respondFriendRequest(userListProvider: UserListProvider, accept_reject: int): void
+sendMessage(chatMessage: ChatMessage): void
+sendEventInvitation(UserListProvider userListProvider, event: Event): void
+participatePrivelegedEvent(eventID: int, username: string): void
+createEvent(event: Event): void
+Register(my_username: string, name: string, surname: string, my_phone: string, my_email: string, birthyear: int)
+seePastEventsOfUser(userListProvider: UserListProvider): void
+seeProfileOfUser(userListProvider: UserListProvider): void
+seeEventInformation(eventID: int): EventListProvider
+seeFriendsOnTheMap(userListProviders: UserListProvider[*]): Google Map Object
+seeEventsOnTheMap(events: Event[*]): Google Map Object
+requestEventInvitation(event: Event, accept_reject: bool): void
+getEventsWithInRadius(username: string, radius: int): EventListProvider[*]
+getNotifications(): GCM Notification[*]
+seeAllEvents(userListProviders: UserListProvider[*]): AnroidListView
+seeAllNotifications(GCM Notification [*]): AndroidListView
+recordNotification(GCM Notification): void
+deleteNotification(GCM Notification): void
+addComment(eventid: int, comment: Comment): void
+rateEvent(eventid: int, rate: int): void
+getFriends(my_username: string): UserListProvider[*]
+getCommentsOfEvent(eventid: int): Comment[*]
+showComments(Comment[*]): Android List View
+getParticipantsOfEvent(eventid: int): UserListProvider[*]
+showParticipantsOfEvent(userListProviders: UserListProvider[*]): Android List View
1..*
Android Buttons
1
«interface»
NotificationActions
+sendEventRequestNotification(userid: int, accep_reject: bool): void
+sendEventInvitationNotification(userid: int, regID: GCM ID): void
+sendFriendRequestNotification(userid: int, accept_reject: bool): void
+sendCommentAddingNotification(creatorids: int[*]): void
+sendCommentAddingNotification(creatorid[*]): void
1
0..*
CommentListProvider
-comment: string
-creator_id: int
-event_id: int
-datetime; string
-picturepathofuser: string
-name; string
-surname; string
+CommentListProvider(comment: string, creator_id: int, event_id: int, datetime: string, picturepathofuser: string, name: string, surname: string)
+getComment()
+getCreatorId()
+getEventId()
+getDateTime()
+getPicturePathOfUser()
+getName()
+getSurname()
0..*
0..*
UserListProvider
-username: string
-name: string
-pictureUrl: string
-status: int
+UserListProvider(username: string, name: string, pictureUrl: string, status: int)
+getUsername(): string
+getName(): string
+getPictureUrl(): string
+getStatus(): int
1
0..*
ChatMessage
-usernameOfMe: string
-isMe: boolean
-message: string
-messageTime: string
-usernameToSend: string
+CharMessage(usernameOfMe: string, isMe: bool, message: string, messageTime: string, usernameToSend: string)
+getUsernameOfMe(): string
+getIsMe(): bool
+getMessageTime(): string
+getUsermeToSend(): string

Figure 5.2.1 Class Diagram (If the class diagram can not be read there is a class diagram link in 1.5 References section)

5.3 INFORMATION VIEWPOINT

In this section, relations of the different classes are explained. Main purpose of the following ER diagram is to explain class diagram in a different way and give ideas about the database design. One can understand relationships between classes without being confused by attributes easily by this diagram.

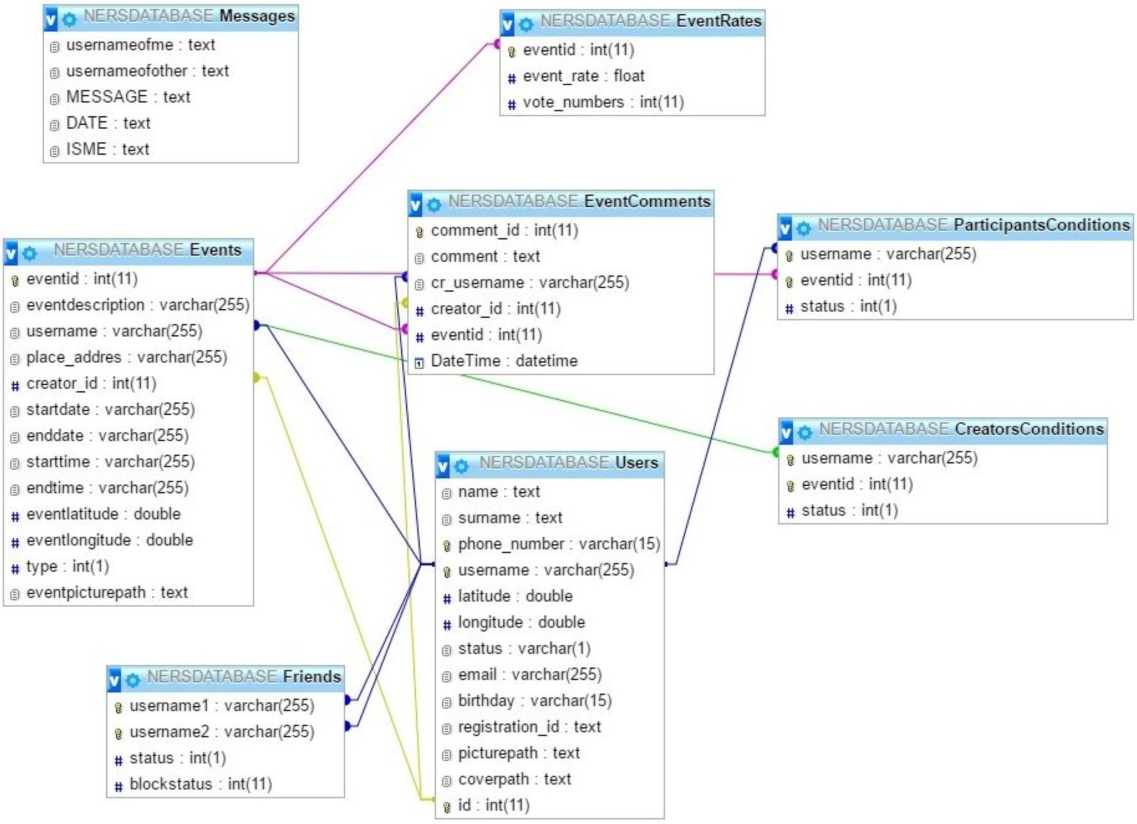


Figure 5.3.1: ER Diagram

5.4 INTERACTION VIEWPOINT

The main concerns of the interaction viewpoint and this viewpoint is important when designing and adopting patterns. Sequence Diagrams are going to be used to show interaction between objects. Since flow of event is shown sequentially in the diagram, order of the events, satisfied and unsatisfied conditions can easily be understood. In this viewpoint we explained the relations of modules and functions to each other and the flow of events. This section will be directly used in implementing functions. This is the essential part of the document.

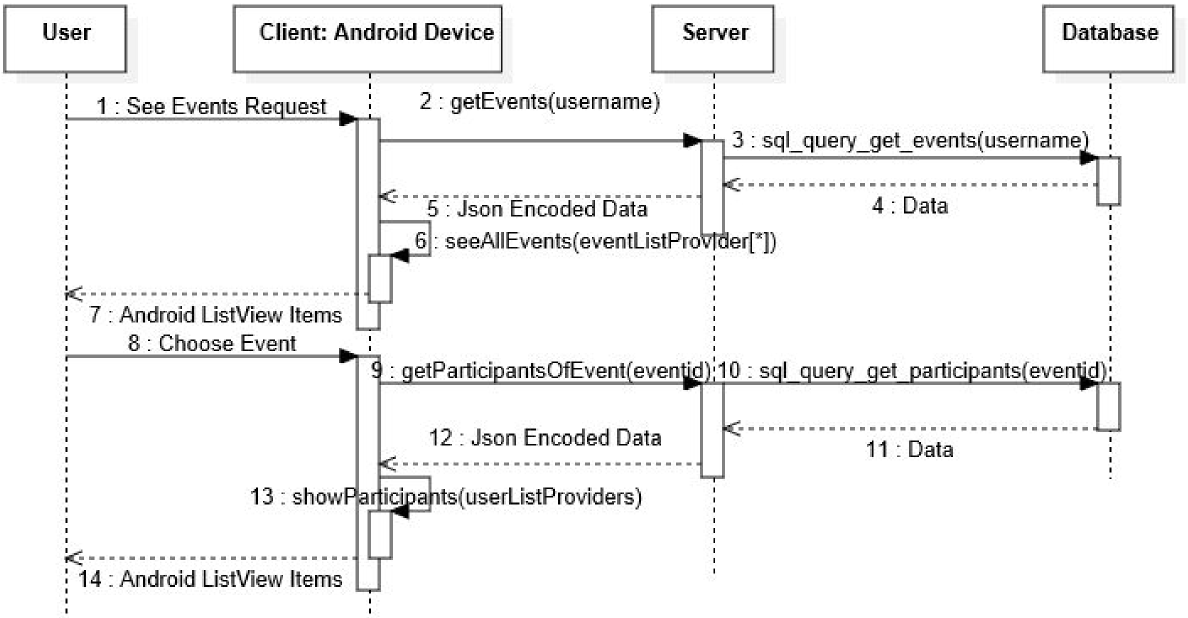

User
Client: Android Device
Server
Database
1 : See Events Request
2 : getEvents(username)
3 : sql_query_get_events(username)
4 : Data
5 : Json Encoded Data
6 : seeAllEvents(eventListProvider[*])
7 : Android ListView Items
8 : Choose Event
9 : getParticipantsOfEvent(eventid)
10 : sql_query_get_participants(eventid)
11 : Data
12 : Json Encoded Data
13 : showParticipants(userListProviders)
14 : Android ListView Items

Figure 5.4.1 showParticipants

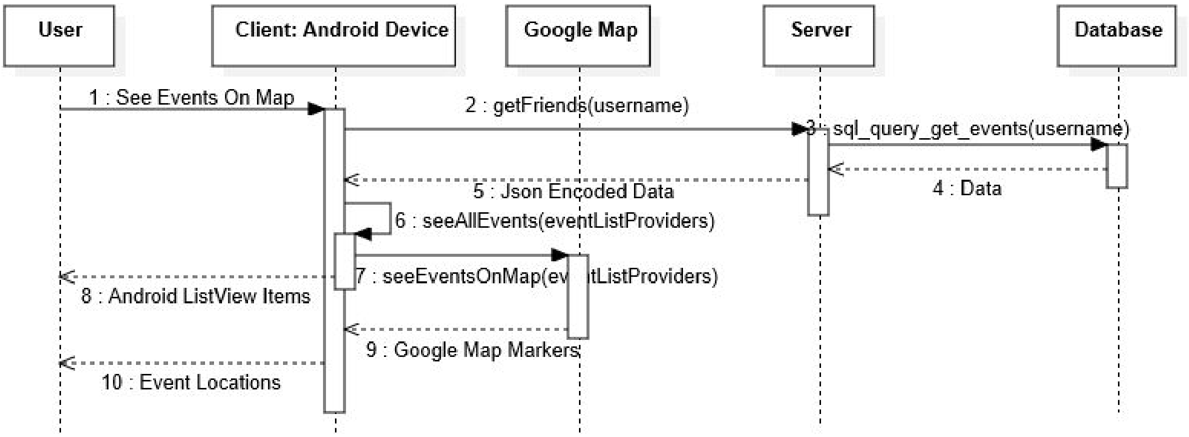


Figure 5.4.2 seeEventsOnTheMap

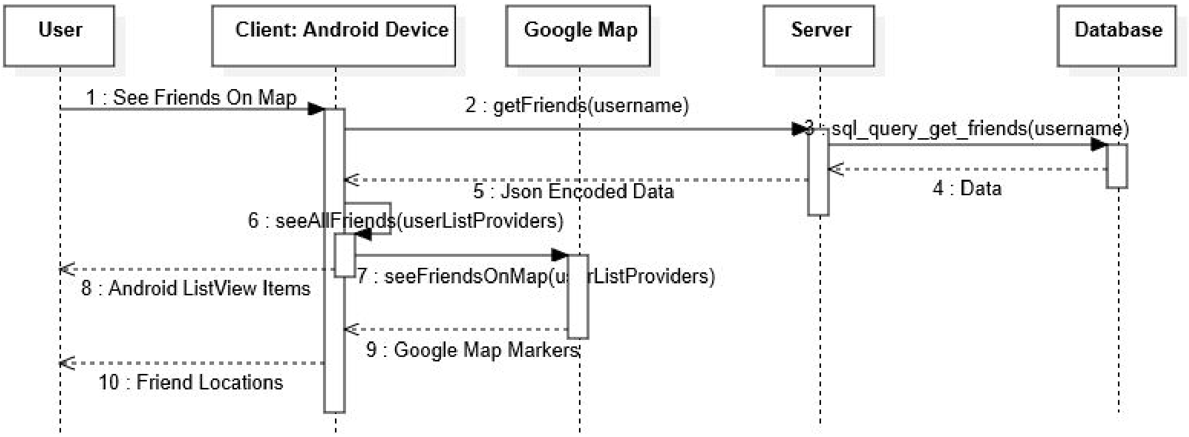


Figure 5.4.3 seeFriendsOnTheMap

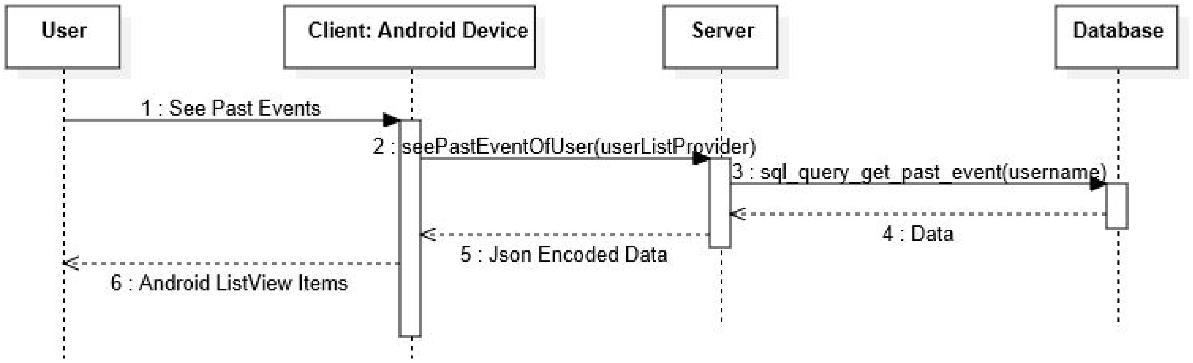

User
Client: Android Device
Server
Database
1 : See Past Events
2 : seePastEventOfUser(userListProvider)
3 : sql_query_get_past_event(username)
4 : Data
5 : Json Encoded Data
6 : Android ListView Items

Figure 5.4.4 seePastEventsOfUser

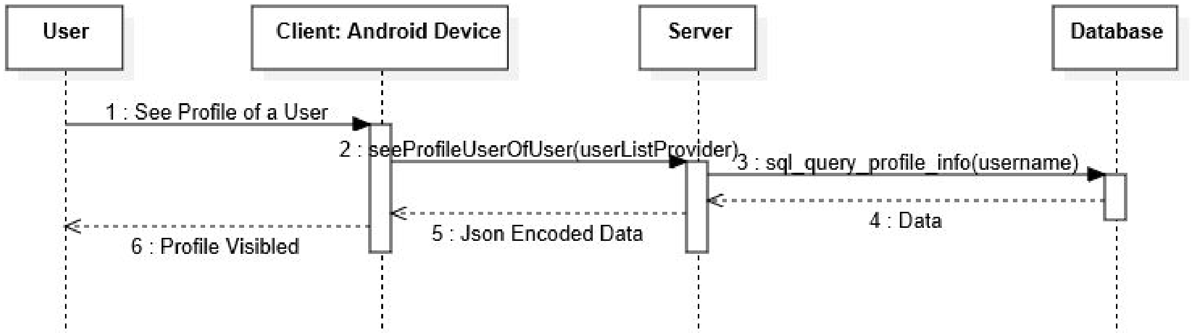

User
Client: Android Device
Server
Database
1 : See Profile of a User
2 : seeProfileUserOfUser(userListProvider)
3 : sql_query_profile_info(username)
4 : Data
5 : Json Encoded Data
6 : Profile Visibled

Figure 5.4.5 seeProfileOfUser

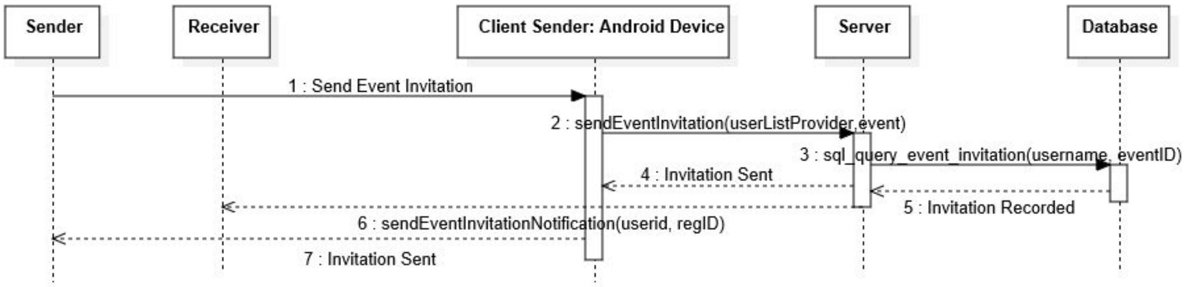

Sender
Receiver
Client Sender: Android Device
Server
Database
1 : Send Event Invitation
2 : sendEventInvitation(userListProvider,event)
3 : sql_query_event_invitation(username, eventID)
4 : Invitation Sent
5 : Invitation Recorded
6 : sendEventInvitationNotification(userid, regID)
7 : Invitation Sent

Figure 5.4.6 sendEventInvtation

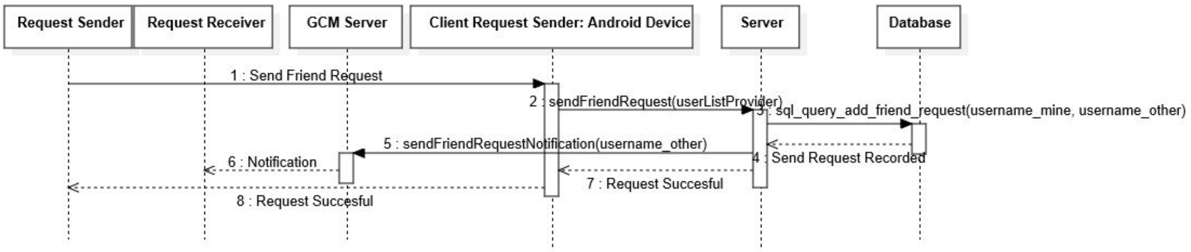
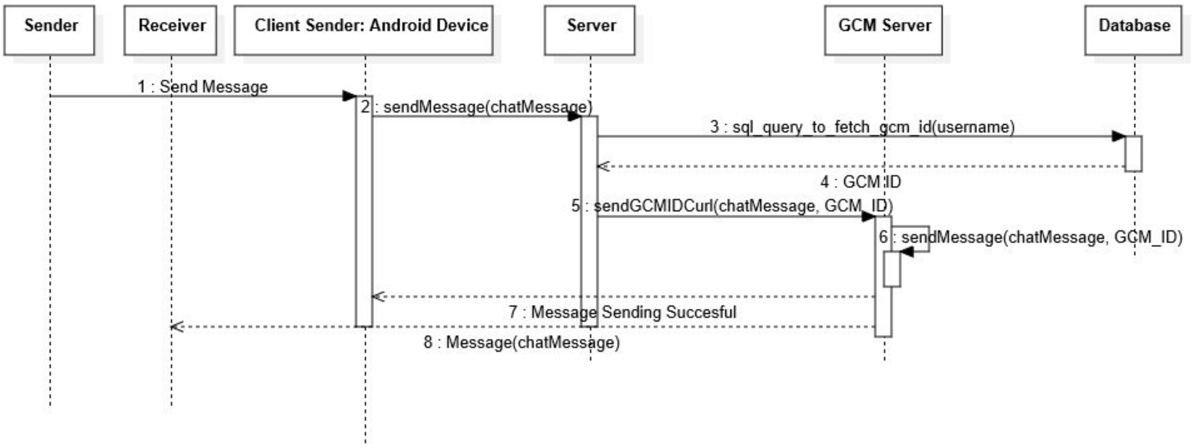
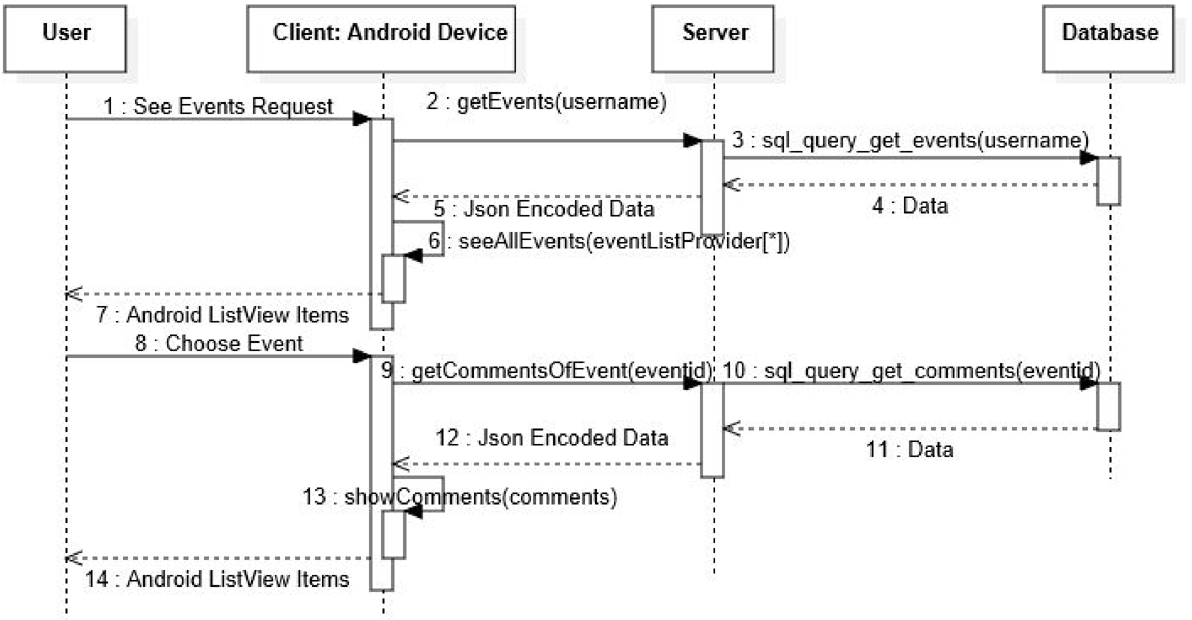

Request Sender
Request Receiver
GCM Server
Client Request Sender: Android Device
Server
Database
1 : Send Friend Request
2 : sendFriendRequest(userListProvider)
3 : sql_query_add_friend_request(username_mine, username_other)
4 : Send Request Recorded
5 : sendFriendRequestNotification(username_other)
6 : Notification
7 : Request Succesful
8 : Request Succesful

Figure 5.4.7 sendFriendRequest



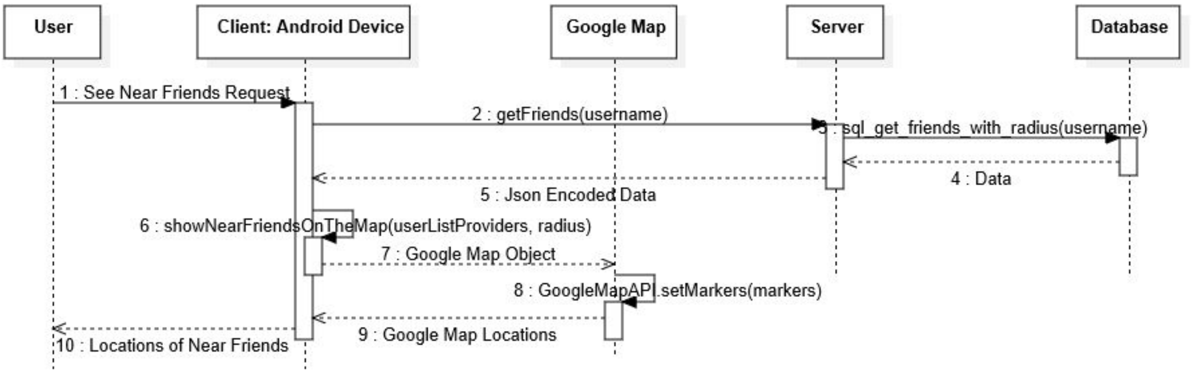
Sender
Receiver
Client Sender: Android Device
Server
GCM Server
Database
1 : Send Message
2 : sendMessage(chatMessage)
3 : sql_query_to_fetch_gcm_id(username)
4 : GCM ID
5 : sendGCMIDCurl(chatMessage, GCM_ID)
6 : sendMessage(chatMessage, GCM_ID)
7 : Message Sending Succesful
8 : Message(chatMessage)

Figure 5.4.8 sendMessage



User
Client: Android Device
Server
Database
1 : See Events Request
2 : getEvents(username)
3 : sql_query_get_events(username)
4 : Data
5 : Json Encoded Data
6 : seeAllEvents(eventListProvider[*])
7 : Android ListView Items
8 : Choose Event
9 : getCommentsOfEvent(eventid)
10 : sql_query_get_comments(eventid)
11 : Data
12 : Json Encoded Data
13 : showComments(comments)
14 : Android ListView Items

Figure 5.4.9 showComments



User
Client: Android Device
Google Map
Server
Database
1 : See Near Friends Request
2 : getFriends(username)
3 : sql_get_friends_with_radius(username)
4 : Data
5 : Json Encoded Data
6 : showNearFriendsOnTheMap(userListProviders, radius)
7 : Google Map Object
8 : GoogleMapAPI.setMarkers(markers)
9 : Google Map Locations
10 : Locations of Near Friends

Figure 5.4.10 showNearFriends

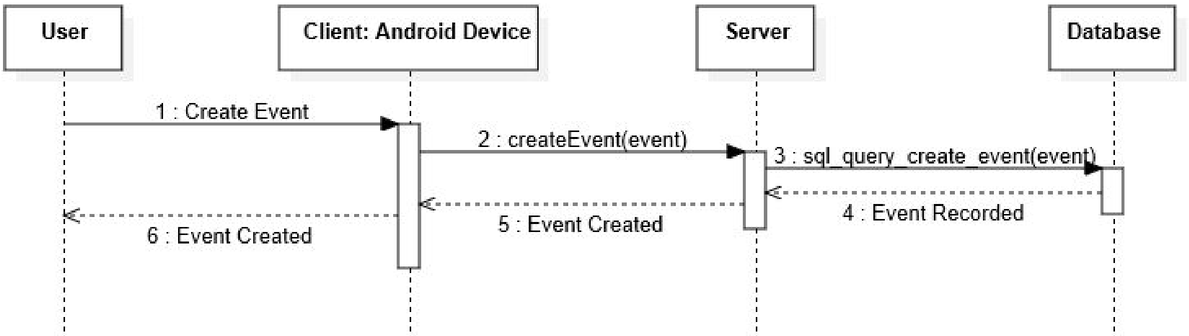


Figure 5.4.11 createFriend

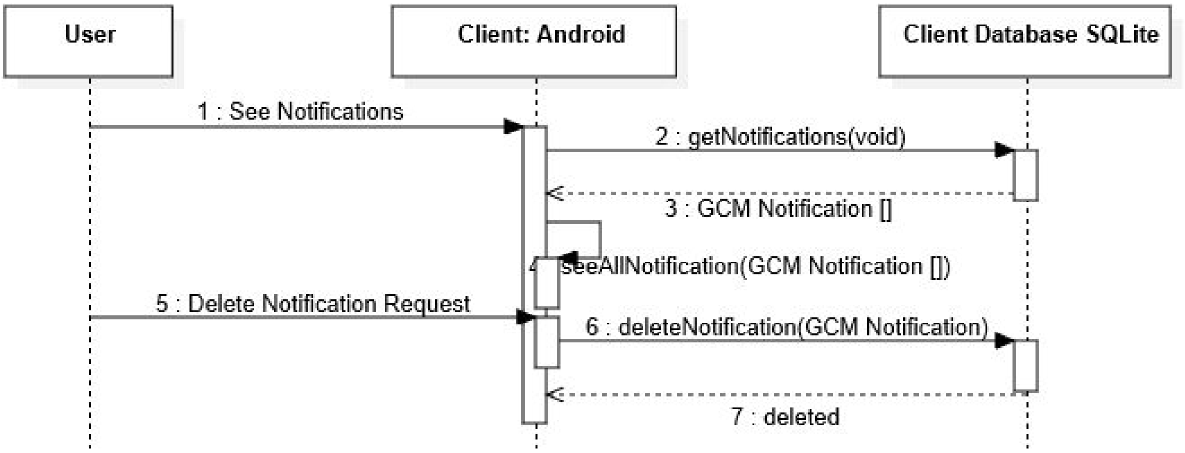

User
Client: Android
Client Database SQLite
1 : See Notifications
2 : getNotifications(void)
3 : GCM Notification []
4 : seeAllNotification(GCM Notification [])
5 : Delete Notification Request
6 : deleteNotification(GCM Notification)
7 : deleted

Figure 5.4.12 deleteNotification

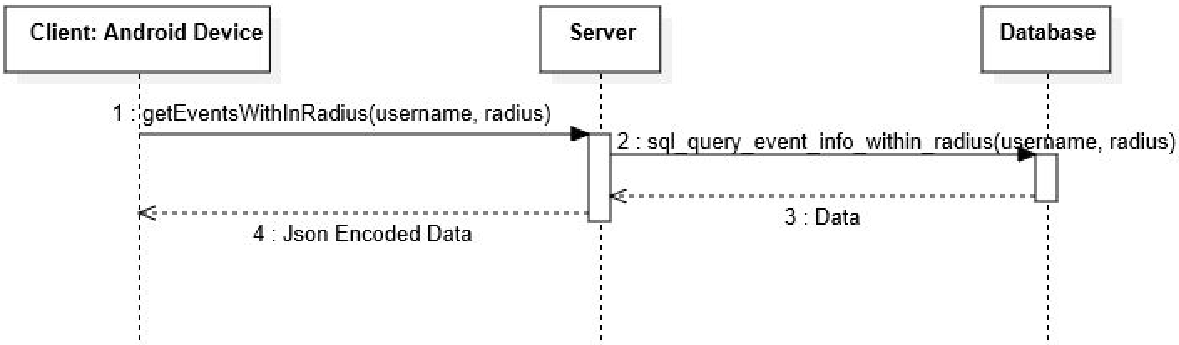

Client: Android Device
Server
Database
1 : getEventsWithInRadius(username, radius)
2 : sql_query_event_info_within_radius(username, radius)
3 : Data
4 : Json Encoded Data

Figure 5.4.13 getEventsWithInRadius

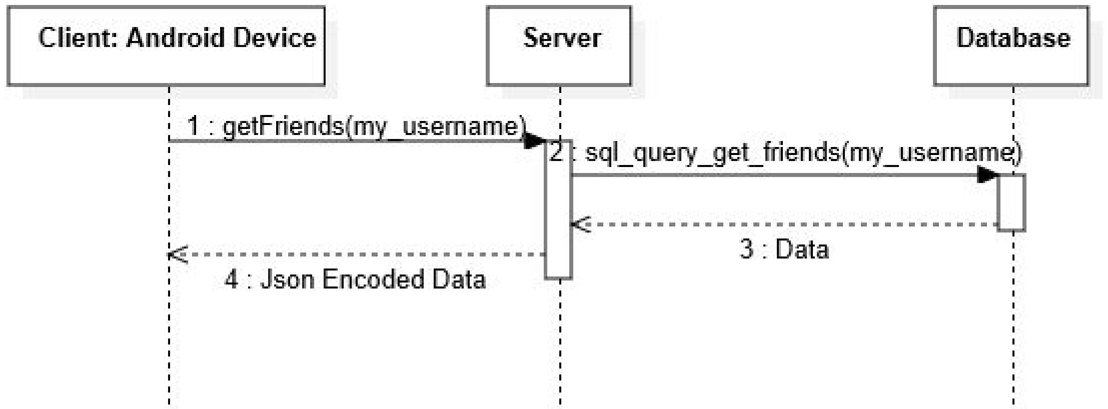

Client: Android Device
Server
Database
1 : getFriends(my_username)
2 : sql_query_get_friends(my_username)
3 : Data
4 : Json Encoded Data

Figure 5.4.14 getFriends

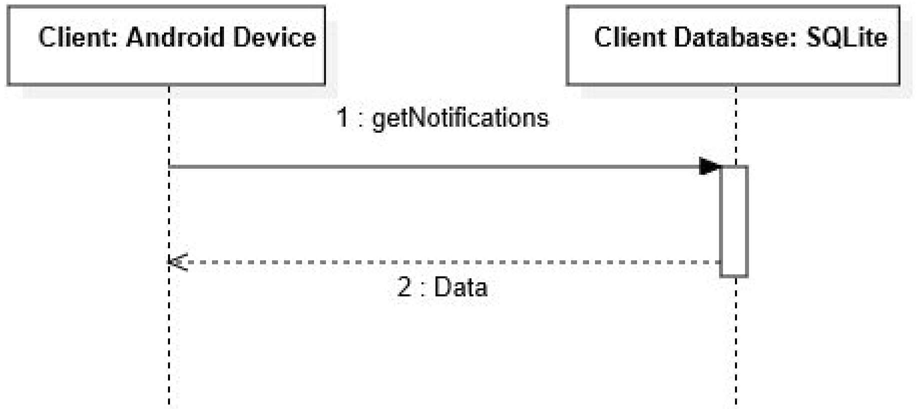


Figure 5.4.15 getNotifications

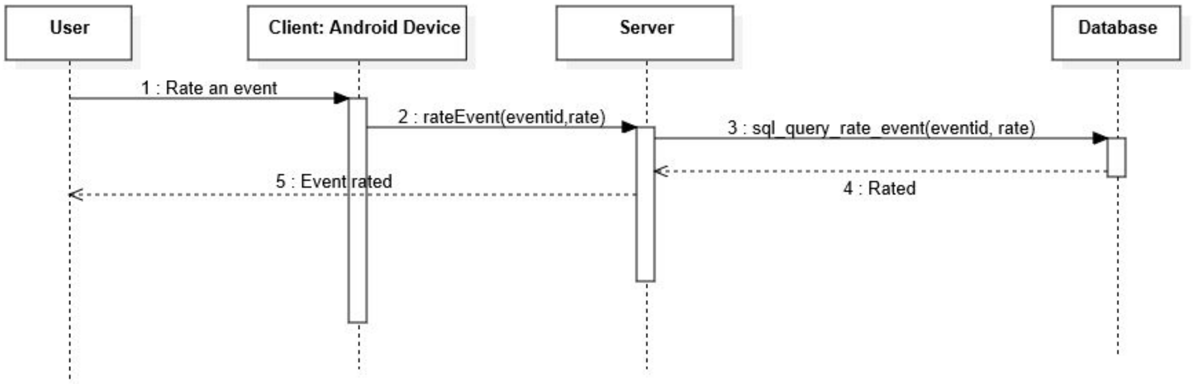


Figure 5.4.16 rateEvent

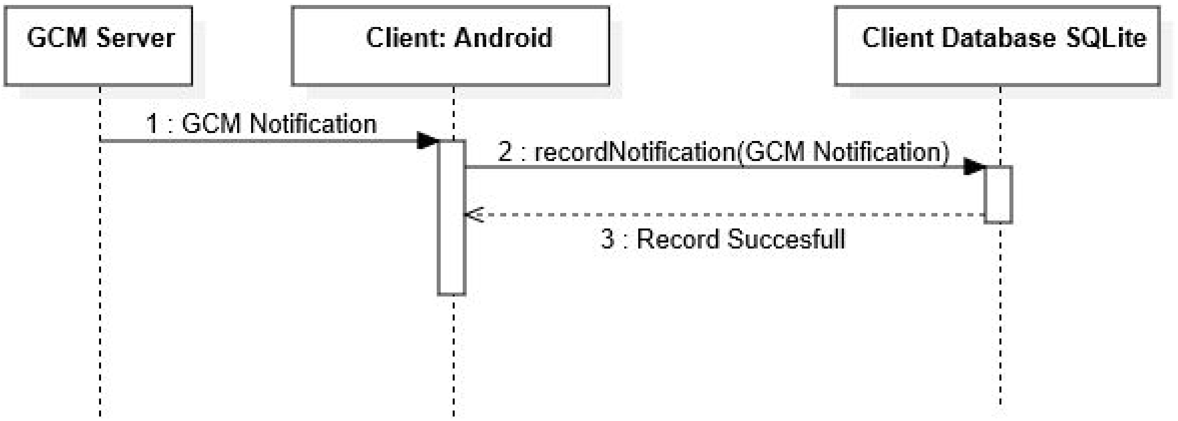


Figure 5.4.17 recordNotification

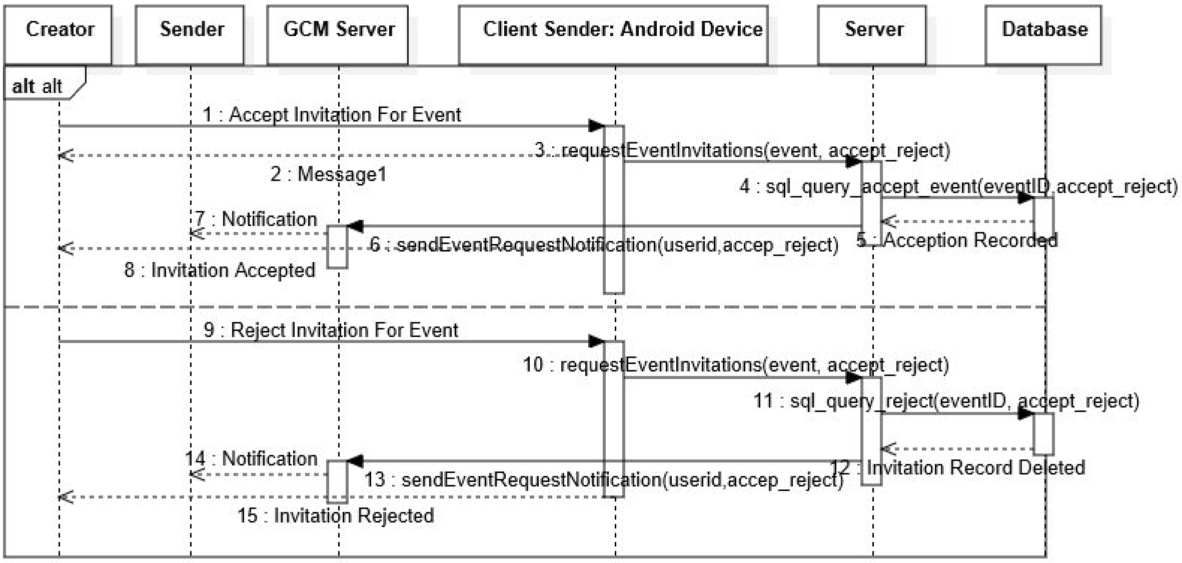

Creator
Sender
GCM Server
Client Sender: Android Device
Server
Database
alt alt
1 : Accept Invitation For Event
3 : requestEventInvitations(event, accept_reject)
2 : Message1
4 : sql_query_accept_event(eventID,accept_reject)
7 : Notification
6 : sendEventRequestNotification(userid,accep_reject)
5 : Acception Recorded
8 : Invitation Accepted
9 : Reject Invitation For Event
10 : requestEventInvitations(event, accept_reject)
11 : sql_query_reject(eventID, accept_reject)
14 : Notification
13 : sendEventRequestNotification(userid,accep_reject)
12 : Invitation Record Deleted
15 : Invitation Rejected

Figure 5.4.18 requestEventInvitations

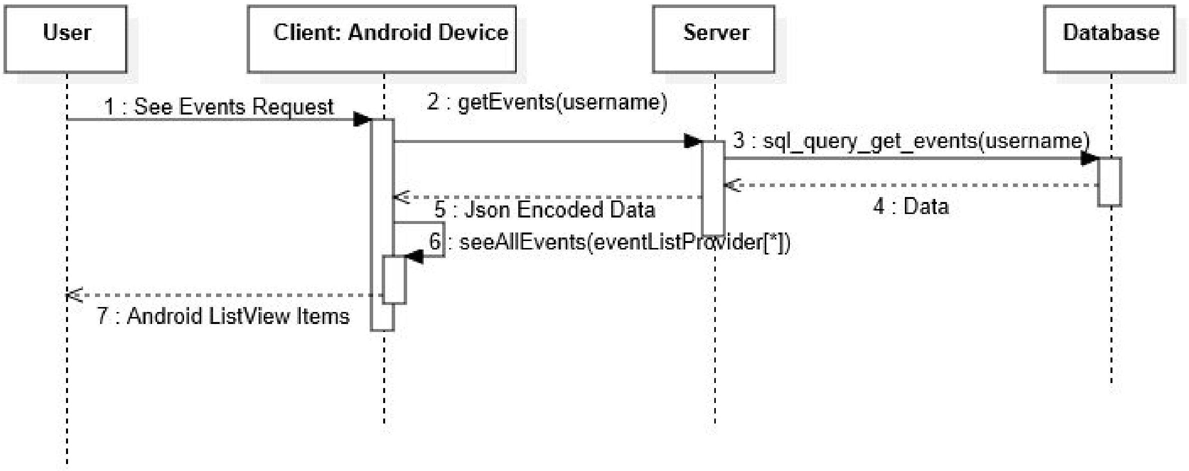


Figure 5.4.19 seeAllEvents

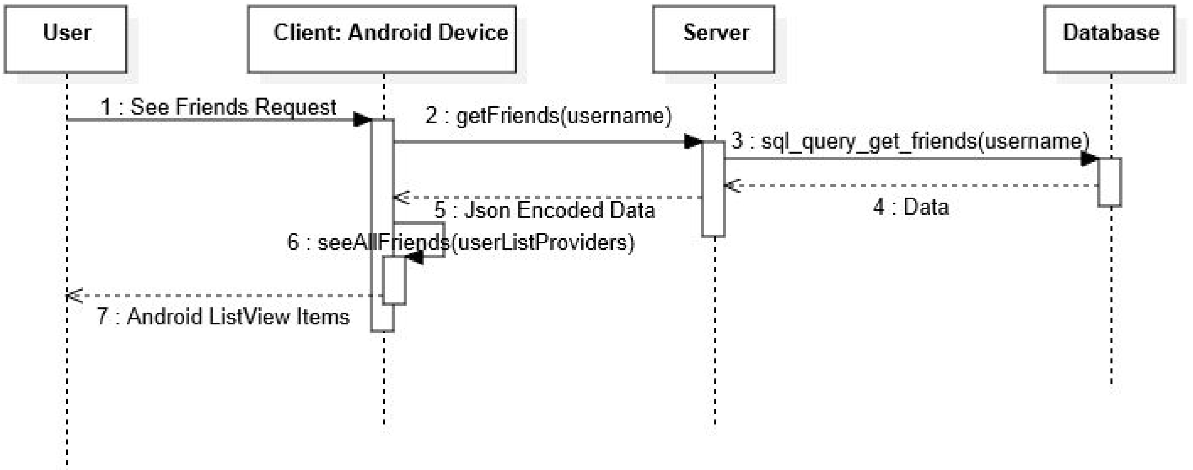


Figure 5.4.20 seeAllFriends

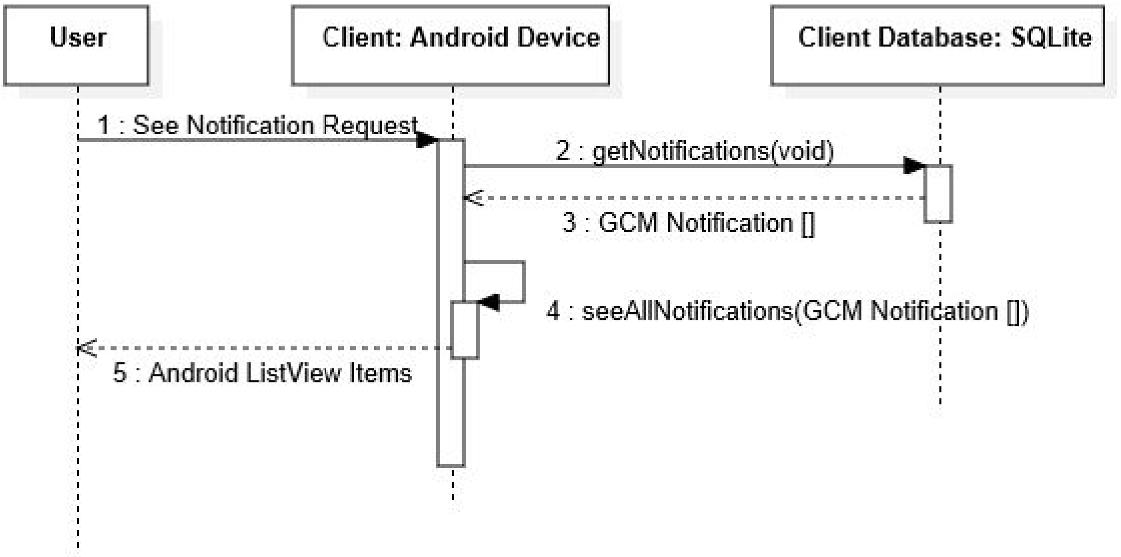

User
Client: Android Device
Client Database: SQLite
1 : See Notification Request
2 : getNotifications(void)
3 : GCM Notification []
4 : seeAllNotifications(GCM Notification [])
5 : Android ListView Items

Figure 5.4.21 seeAllNotifications

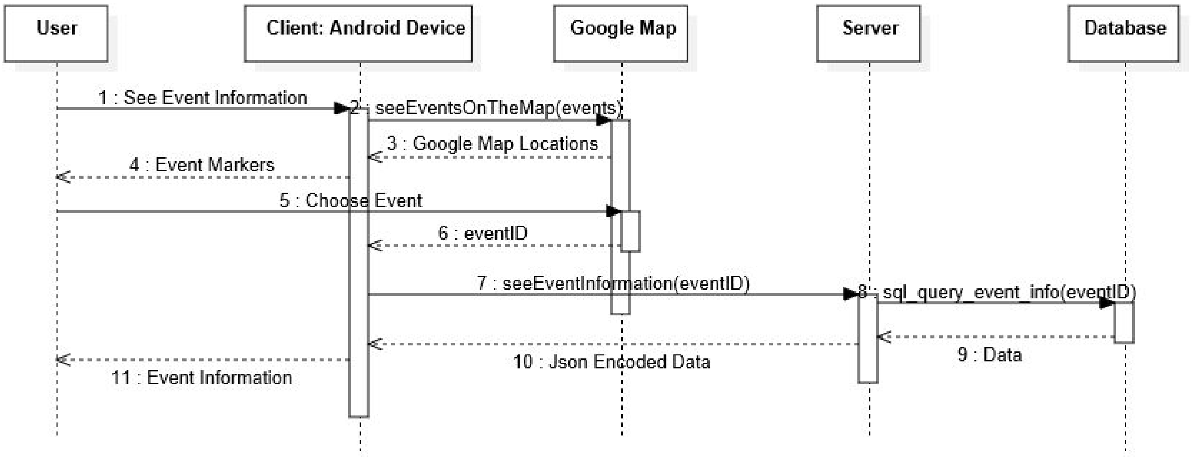


Figure 5.4.22 seeEventInformation

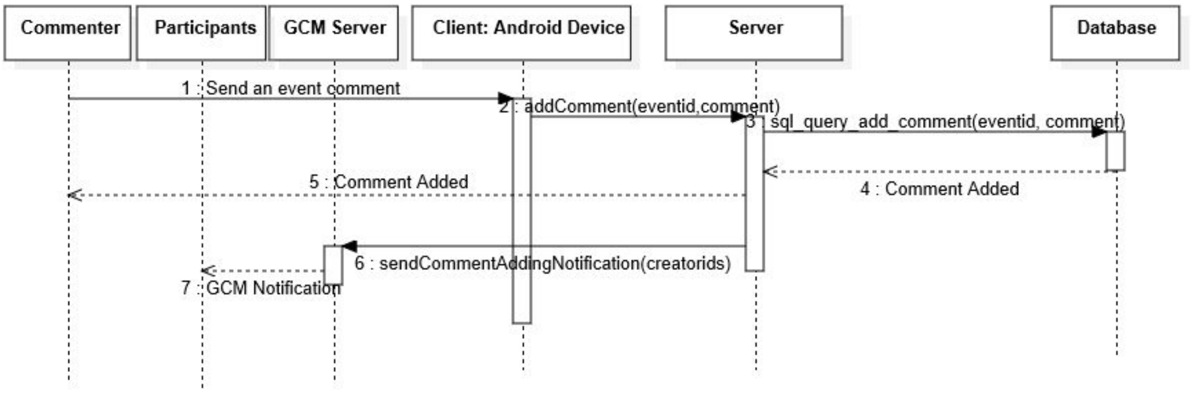

Commenter
Participants
GCM Server
Client: Android Device
Server
Database
1 : Send an event comment
2 : addComment(eventid,comment)
3 : sql_query_add_comment(eventid, comment)
4 : Comment Added
5 : Comment Added
6 : sendCommentAddingNotification(creatorids)
7 : GCM Notification

Figure 5.4.23 addComment

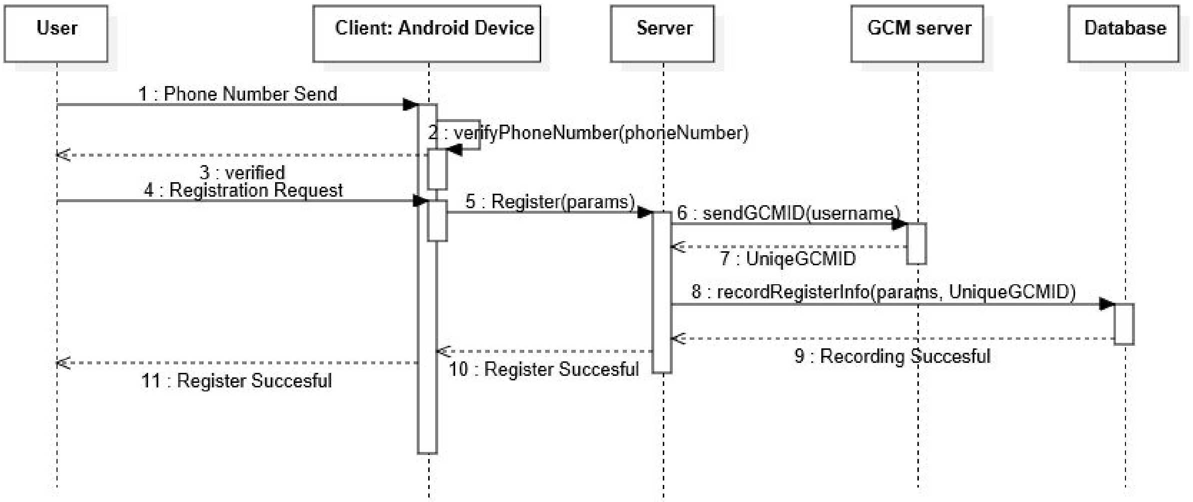


Figure 5.4.24 Register

5.5 STATE DYNAMICS VIEWPOINT

In this section, the activity diagram of the project is shown which helps the developers to understand the states of the application.

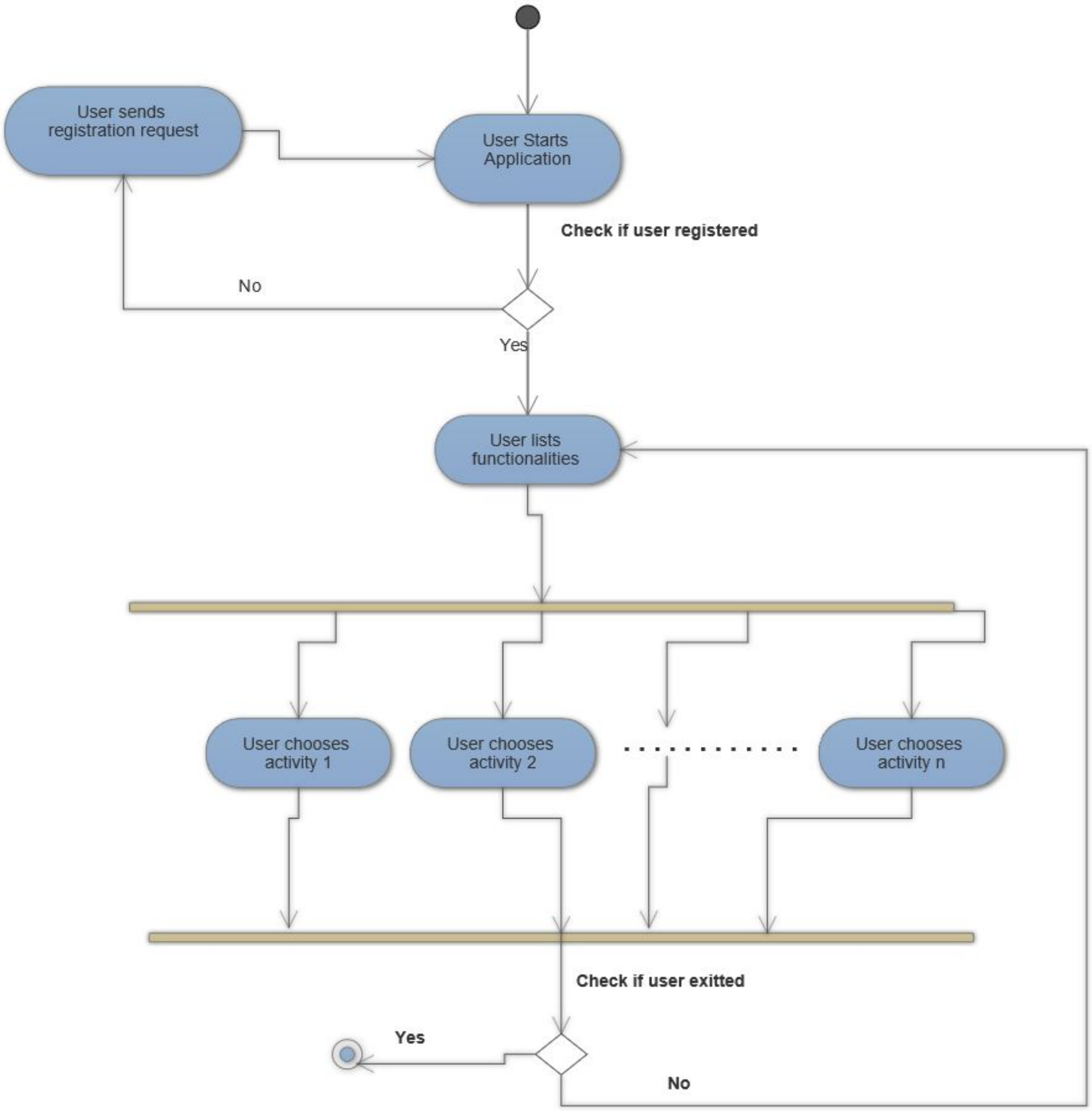
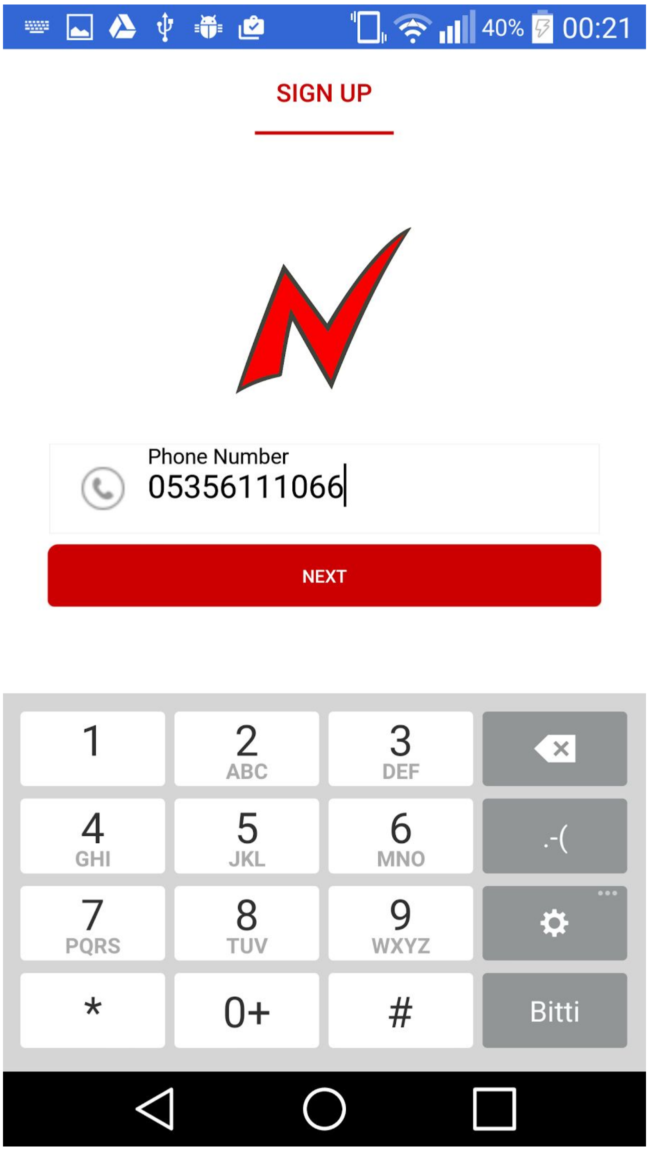


Figure 5.4.1 Activity Diagram

5.6 INTERFACE VIEWPOINT

5.6.1 Register Interface



Phone Number
05356111066

Figure 5.6.1.1

In figure 5.6.1.1 the phone number registration interface can be seen. User is expected to give his/her phone number to edittext.

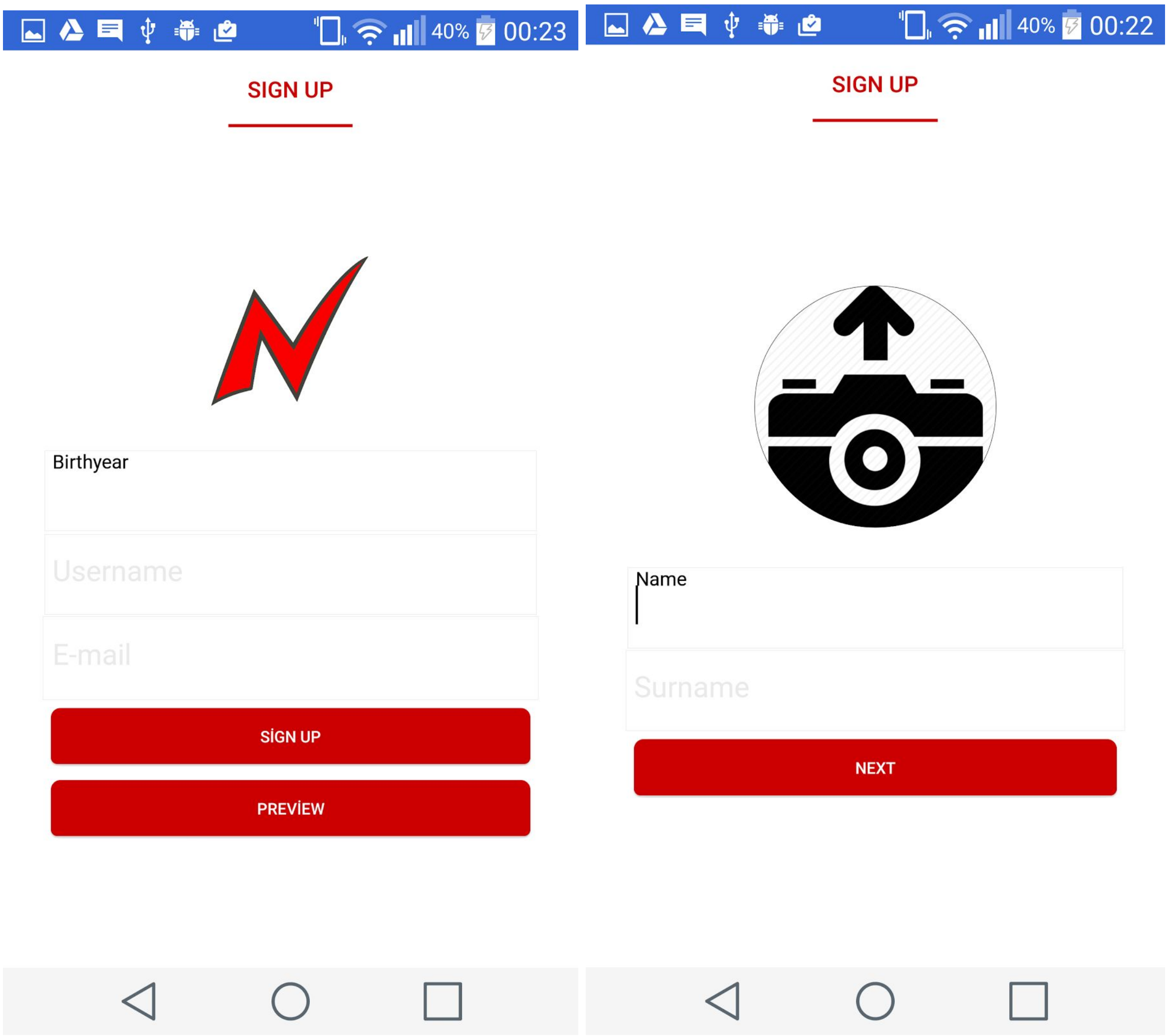


Figure 5.6.1.2

In figure 5.6.1.2 the user information input and its interfaces can be shown. Name, Surname, Username, Birthyear, E-mail and Photo will be user input in the interface. Sign Up button ends the register process so register interface.

5.6.2 Application Opening Interface

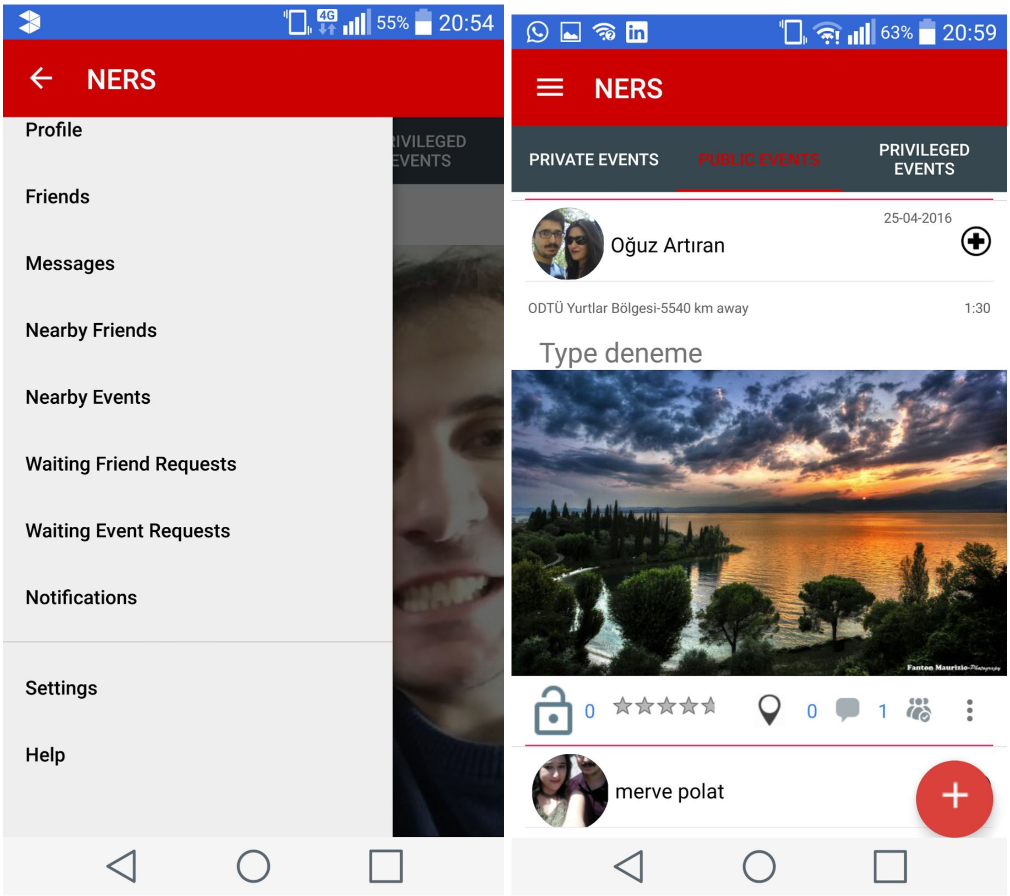
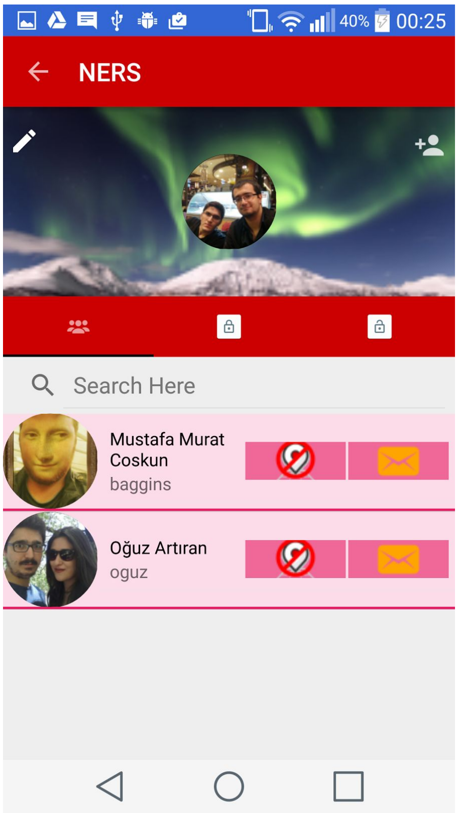


Figure 5.6.2.1

The start screen of the application can be seen in figure 5.6.2.1. There will be three tabs which shows private,public and privileged events to users with some information about them like comment count, rate etc in start screen. In addition there will be a navigation bar in the left and up side, which shows the user activities. In event list interface users will rate, comment and show participants of events on this start screen. In addition the plus sign in the right and down side will be used to create new events.

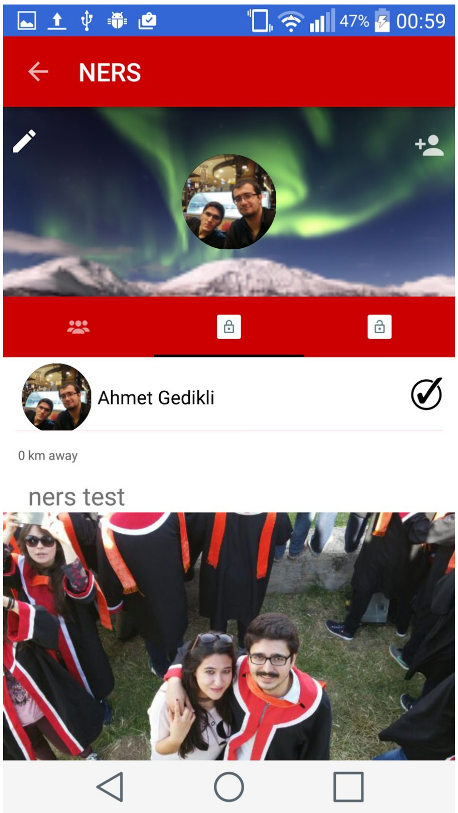
5.6.3 User Profile Interface

In user profile interface, users will be able to see their friends, add new friends, show profile photo, see the past public and private events.



Mustafa Murat Coskun
baggins

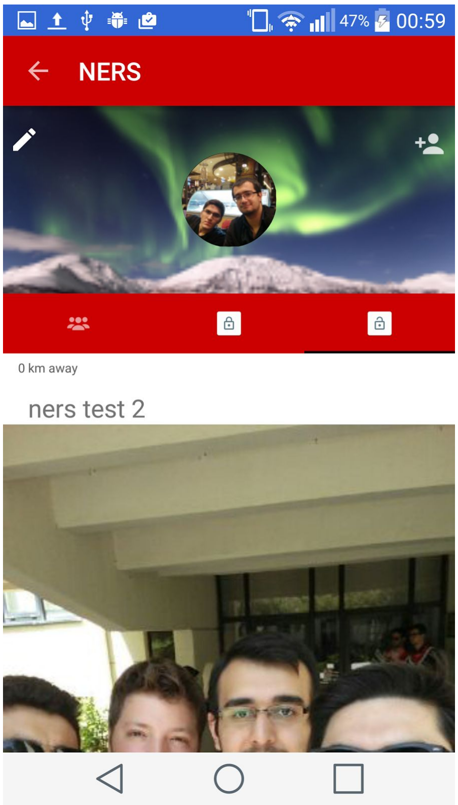
Oğuz Artıran
oguz



Ahmet Gedikli

0 km away

ners test

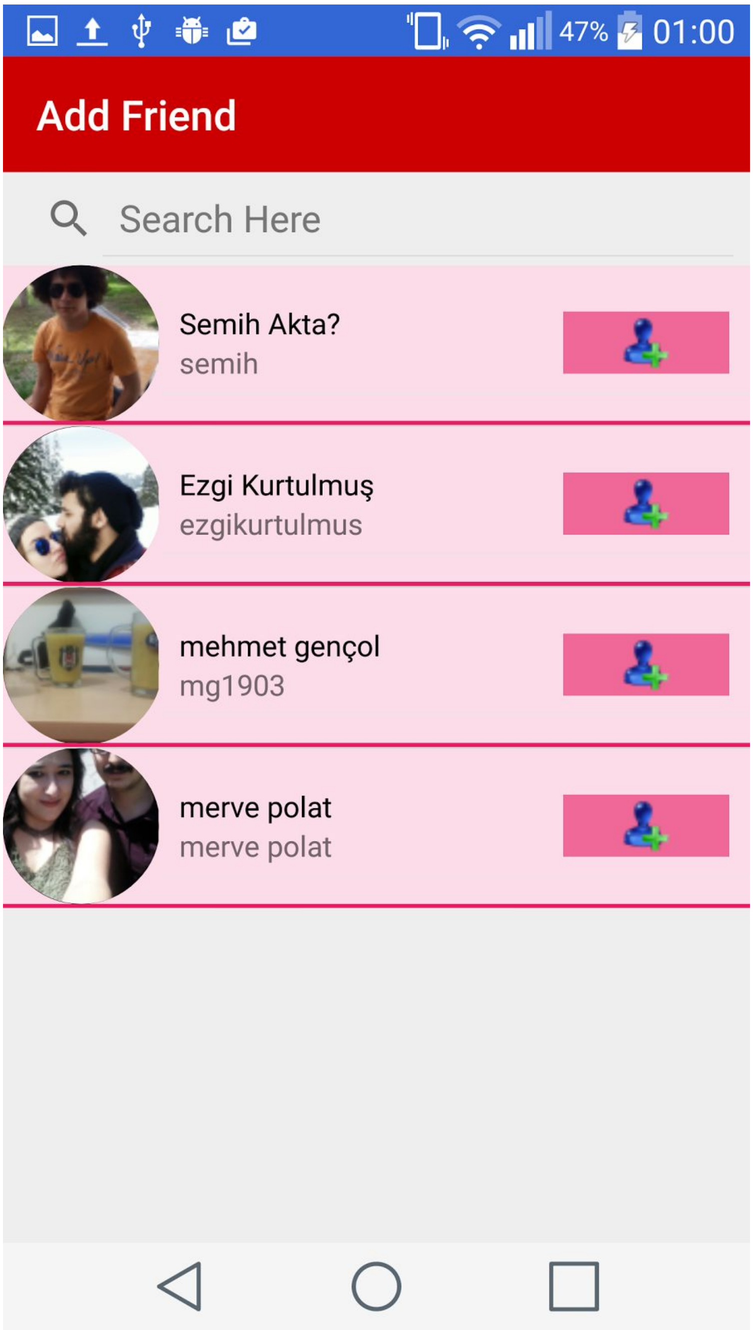


0 km away

ners test 2

5.6.4 Add Friend Interface

User will be able to add friends using this interface. An example interface can be seen in figure 5.6.4.1.



47% 01:00

Add Friend

Search Here

Semih Akta?
semih

Ezgi Kurtulmuş
ezgikurtulmus

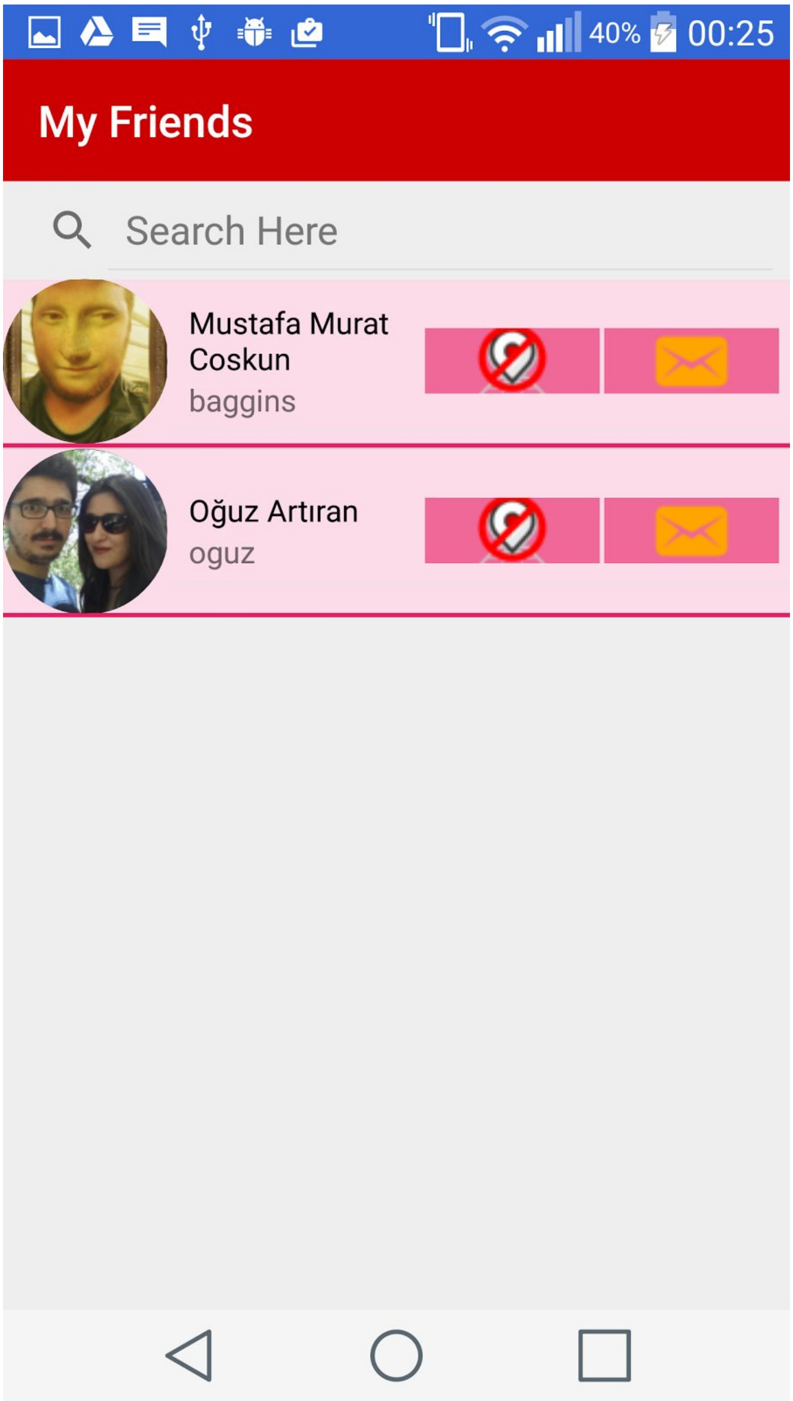
mehmet gençol
mg1903

merve polat
merve polat

Figure 5.6.4.1

5.6.5 See Friends Interfaces

There are two ways of showing friends one is showing them on a listview which users will be able to message them and block them on the map and the second is users will be able to see their friends on the map in real time if they are online. In figure 5.6.5.1 the listview can be seen. The map interface for friends can be seen in figure 5.6.5.2



40% 00:25

My Friends

Search Here

Mustafa Murat Coskun
baggins

Oğuz Artıran
oguz

Figure 5.6.5.1

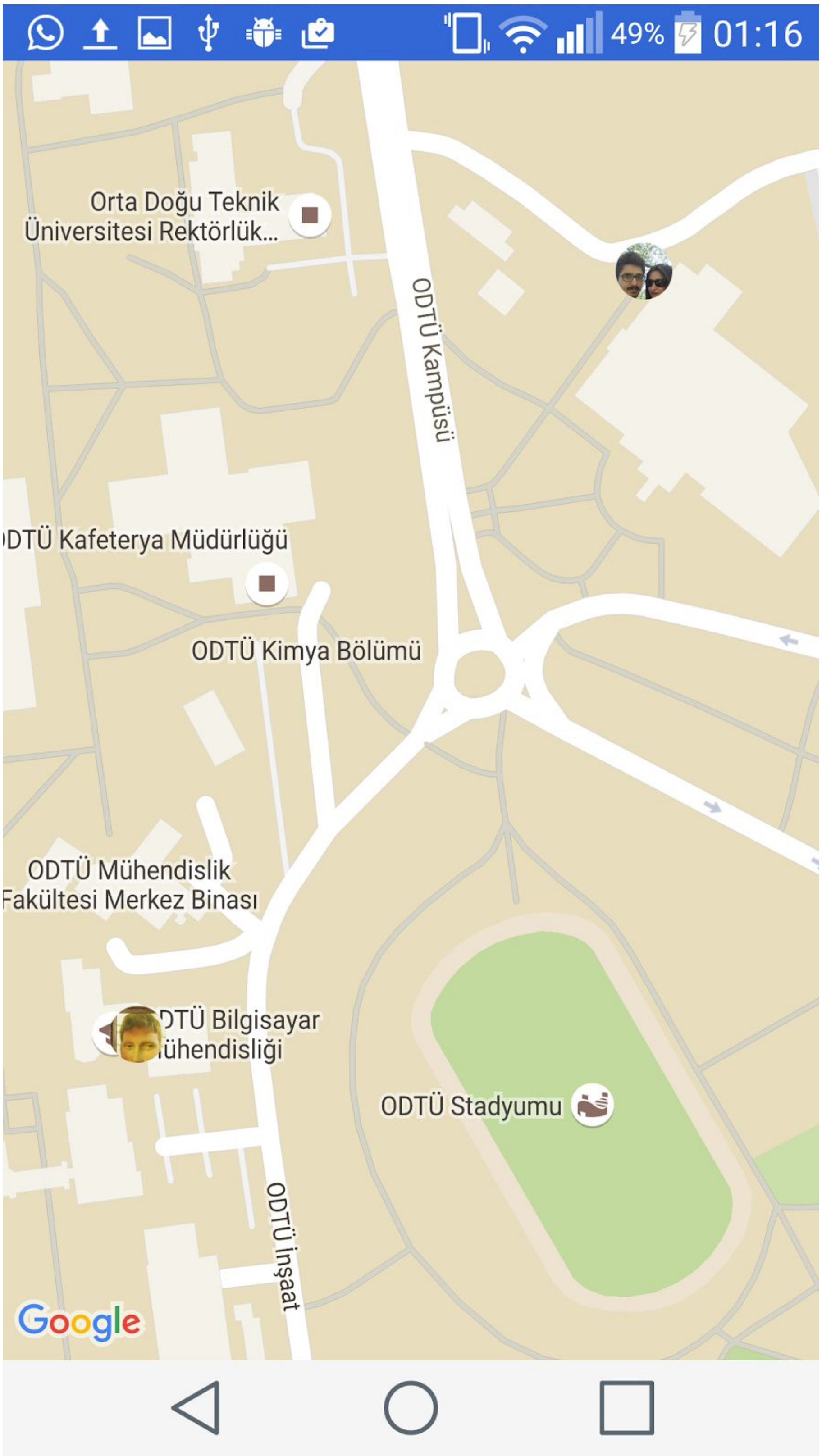
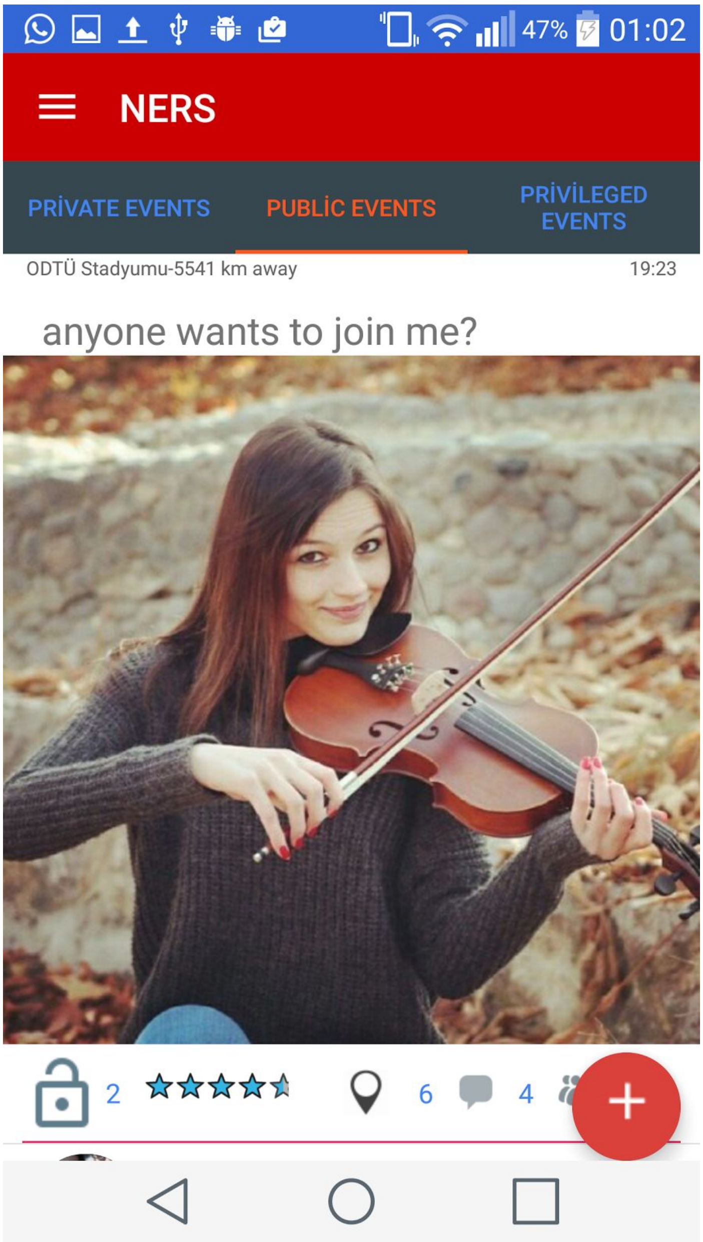

49%
01:16
Orta Doğu Teknik
Üniversitesi Rektörlük...
ODTÜ Kampüsü
DTÜ Kafeterya Müdürlüğü
ODTÜ Kimya Bölümü
ODTÜ Mühendislik
Fakültesi Merkez Binası
DTÜ Bilgisayar
ühendisliği
ODTÜ Stadyumu
ODTÜ İnşaat
Google

Figure 5.6.5.2

5.6.6 See Events Interfaces

Events again can be seen in two user interfaces and one is on listview and the other is on map like see friends interfaces. In figure 5.6.6.1 the listview interface can be seen. Rating event, participating to event, show participants, and add comment buttons are in this interface. In second interface the events will be shown on the map to users and their identifiers will be the descriptions of the event. The types of events affects the marker color.



ODTÜ Stadyumu-5541 km away 19:23

anyone wants to join me?

Figure 5.6.6.1

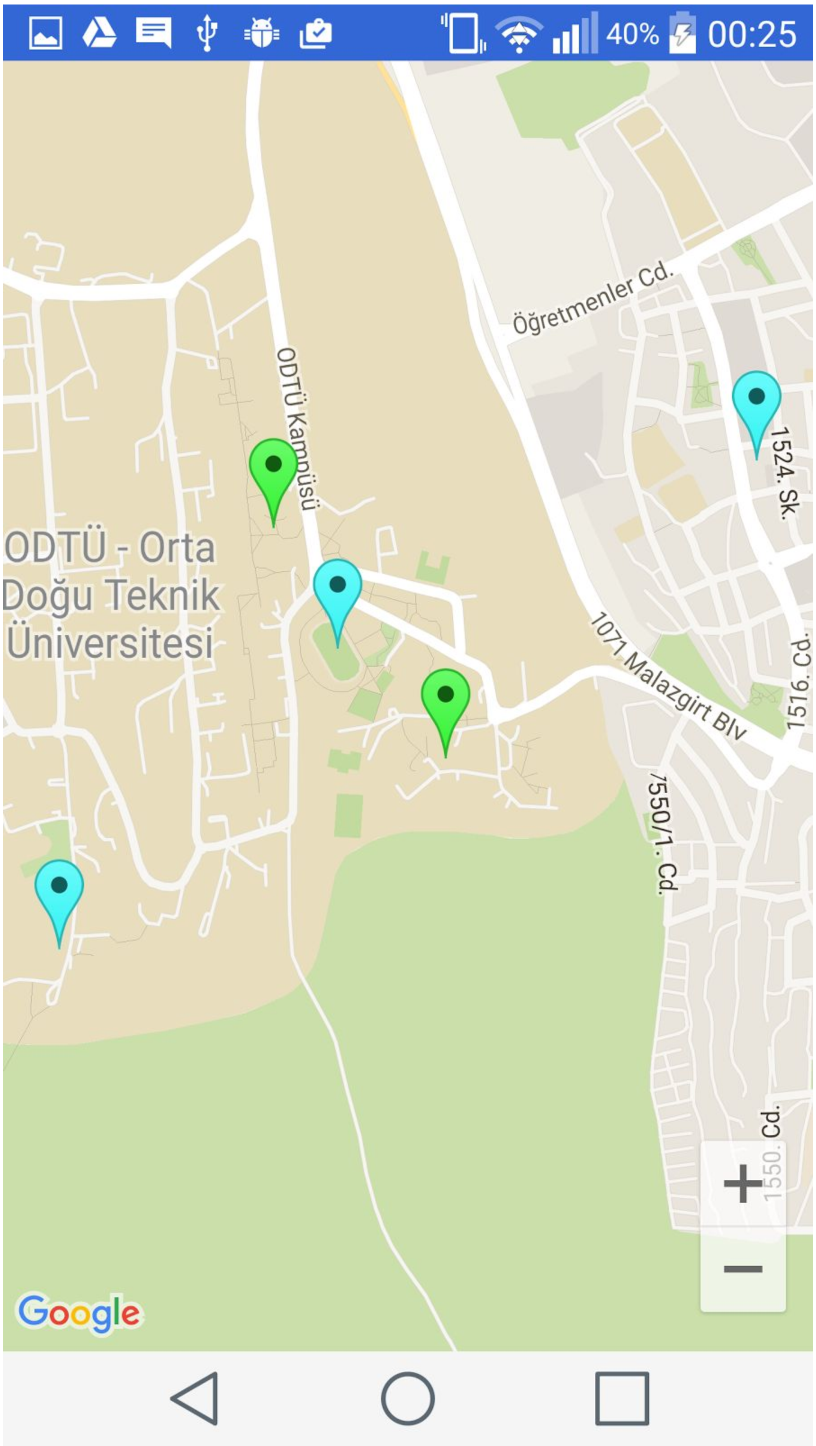

40%
00:25
Öğretmenler Cd.
ODTÜ Kampüsü
1524. Sk.
ODTÜ - Orta Doğu Teknik Üniversitesi
1071 Malazgirt Blv
1516. Cd.
1550/1. Cd.
1550. Cd.
Google

Figure 5.6.6.2

5.6.7 Waiting Requests Interfaces

Waiting requests interfaces consists of two different waiting requests. One is waiting friend request and the other and event requests. In waiting friend requests users will be able to accept or reject friend request. An example interface for waiting friend request can be seen in figure 5.6.7.1. The second interface is for waiting event requests, there are two tabs in that interface which are ‘Event Invitations’ and ‘Participation Requests’. In ‘Event Invitations’ users will be able to reply the event invitations which are sended from the creator. In ‘Participation Requests’ users will be able to send participation requests to creator and creator accept or reject the request in that interface.These process can be seen in figure 5.6.7.2.

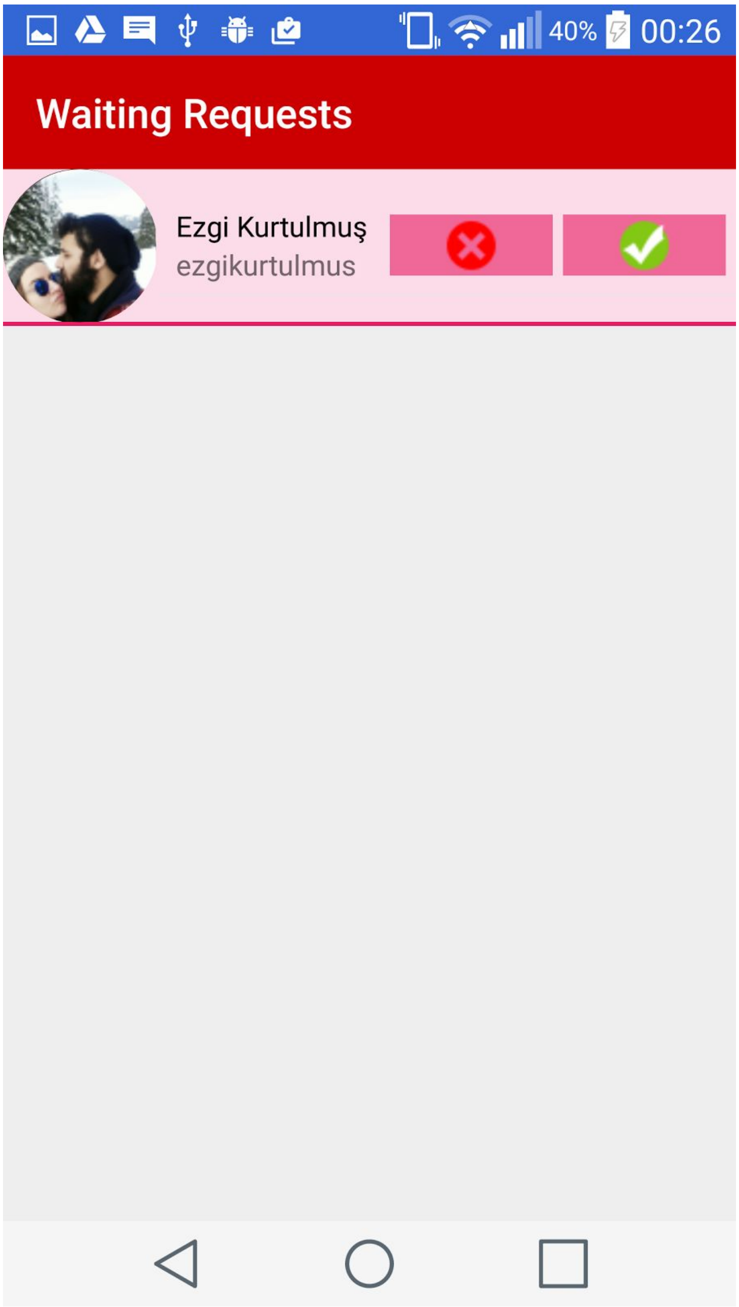
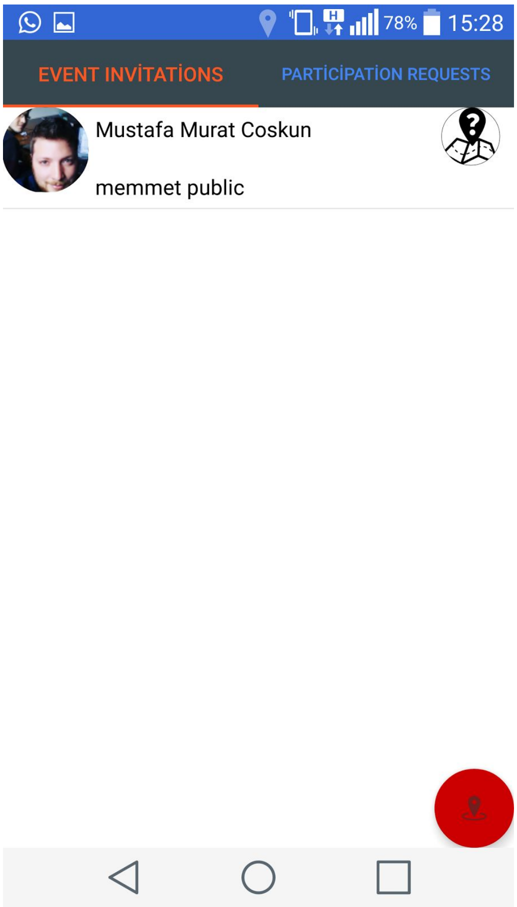


Figure 5.6.7.1



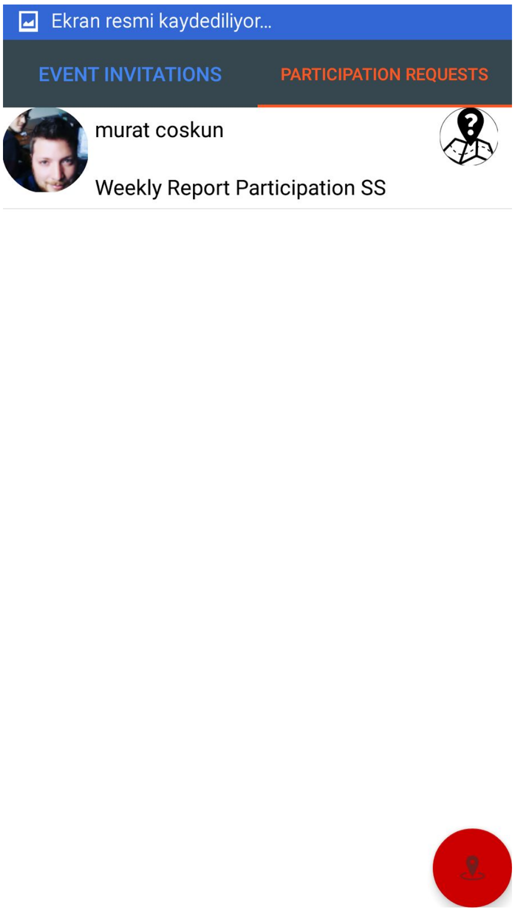
78% 15:28

EVENT INVİTATİONS

PARTİCİPATİON REQUESTS

Mustafa Murat Coskun

memmet public



Ekran resmi kaydediliyor...

EVENT INVITATIONS

PARTICIPATION REQUESTS

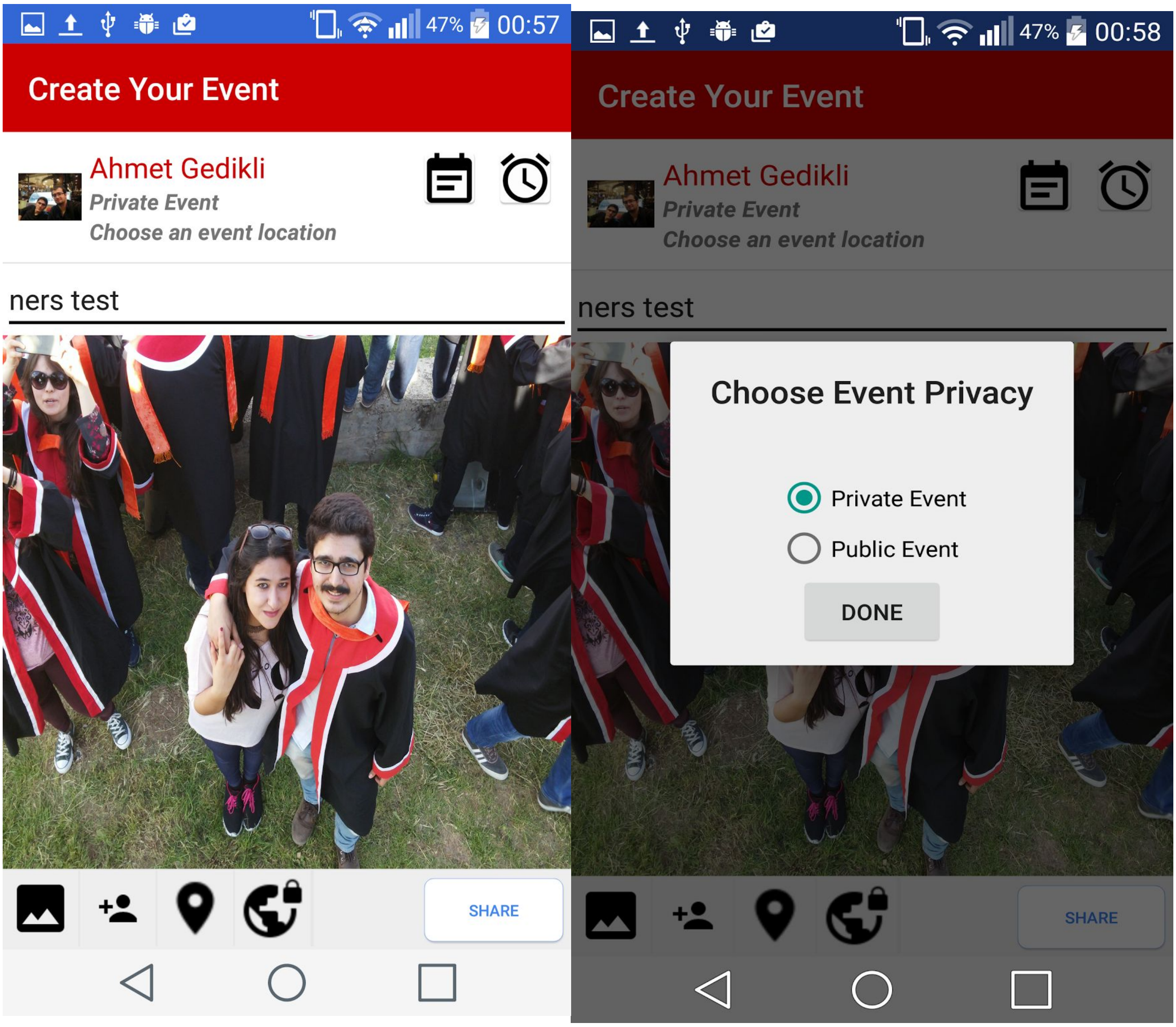
murat coskun

Weekly Report Participation SS

Figure 5.6.7.2

5.6.8 Create Event Interface

In create event interface users will be able to create an event with any feature which they want like choosing private or public event,they will be able to choose event types, event date, event photo, invitees of event, the location of event and the description of the event. These interfaces and the user interactions with interface can be seen in figure 5.6.8.1, figure 5.6.8.2 and figure 5.6.8.3.



Create Your Event

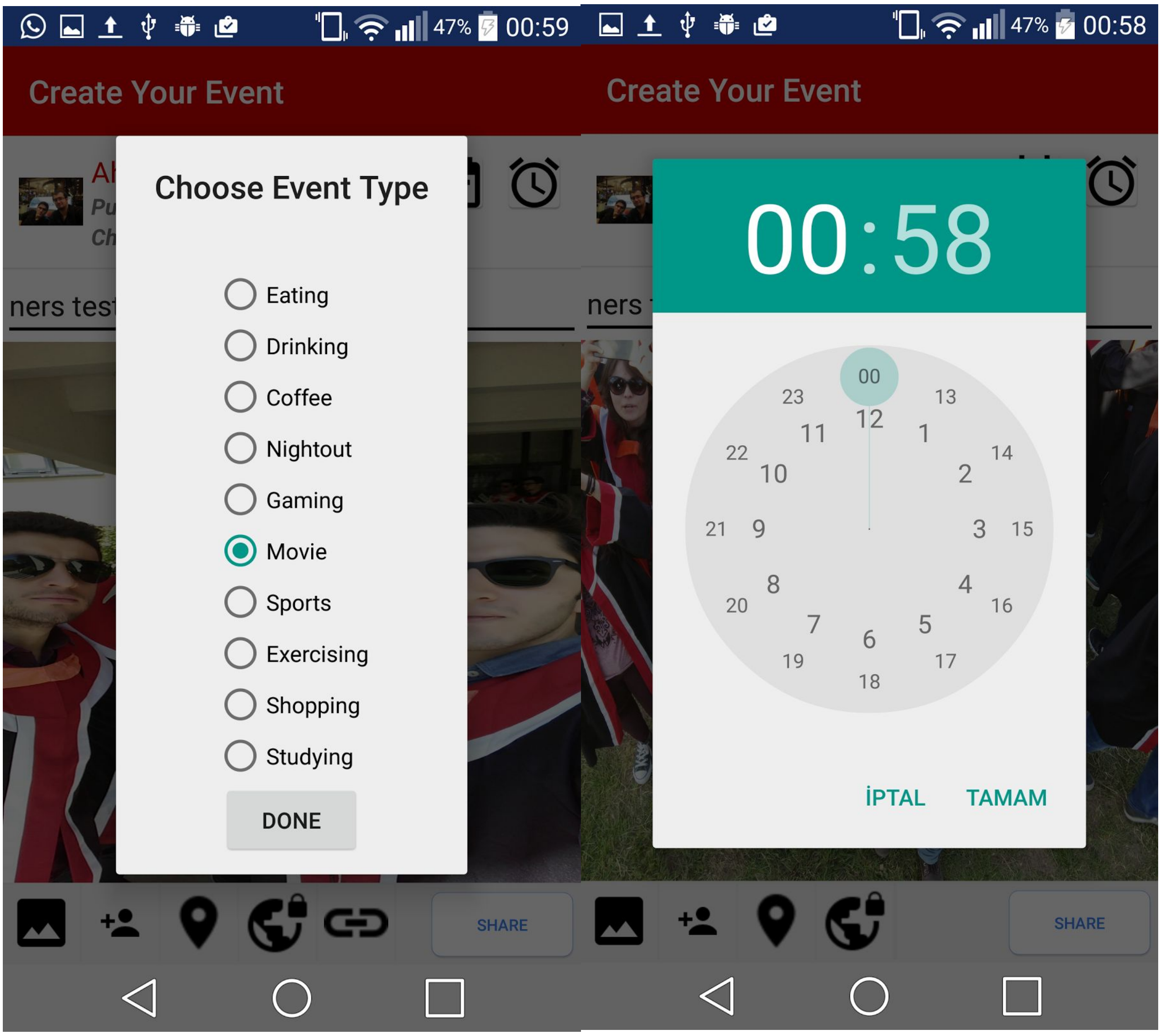
Ahmet Gedikli
Private Event
Choose an event location

ners test

Choose Event Privacy

● Private Event
○ Public Event

Figure 5.6.8.1



Create Your Event

Choose Event Type

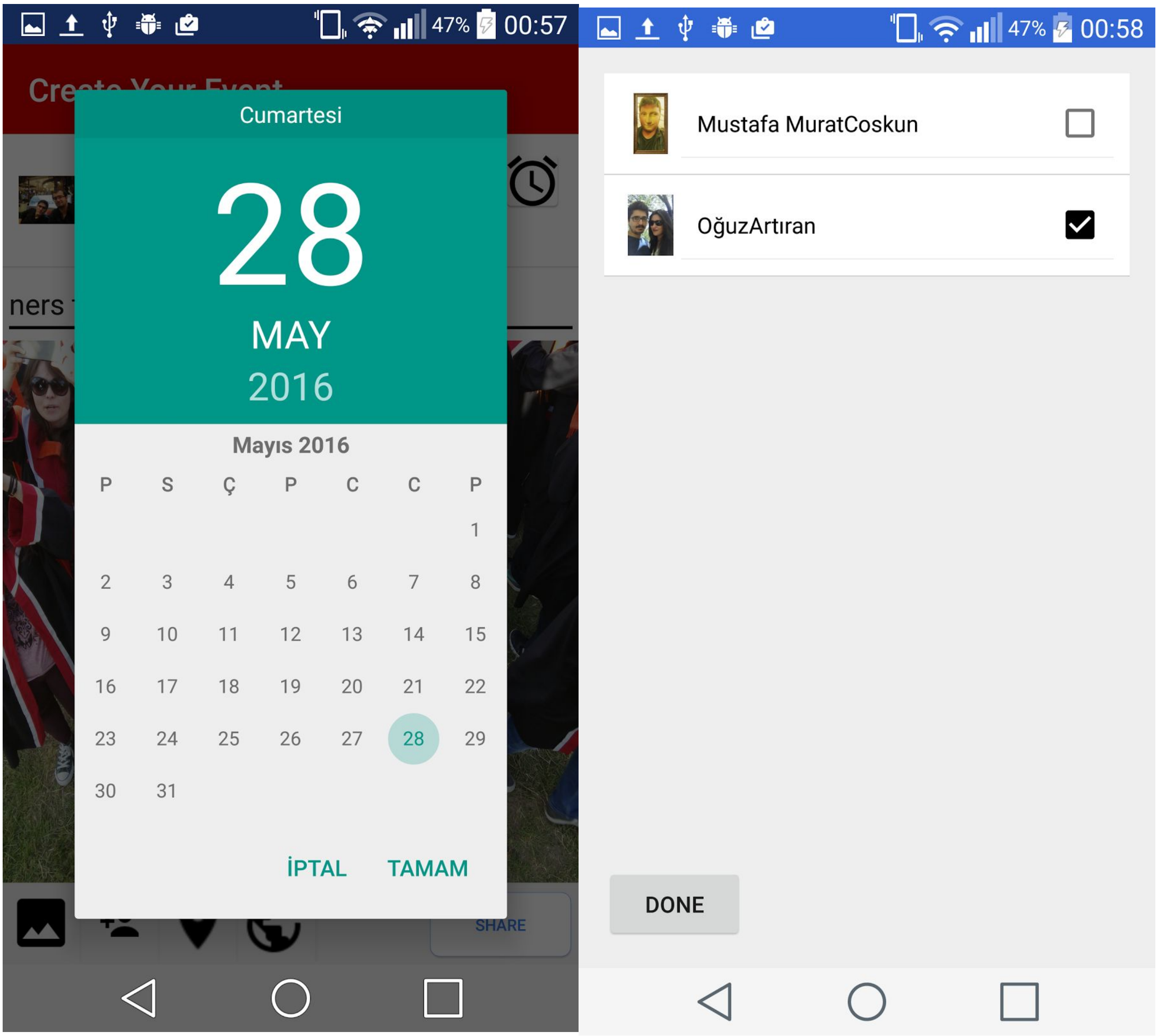
○ Eating
○ Drinking
○ Coffee
○ Nightout
○ Gaming
● Movie
○ Sports
○ Exercising
○ Shopping
○ Studying

DONE

Create Your Event

00:58

Figure 5.6.8.2



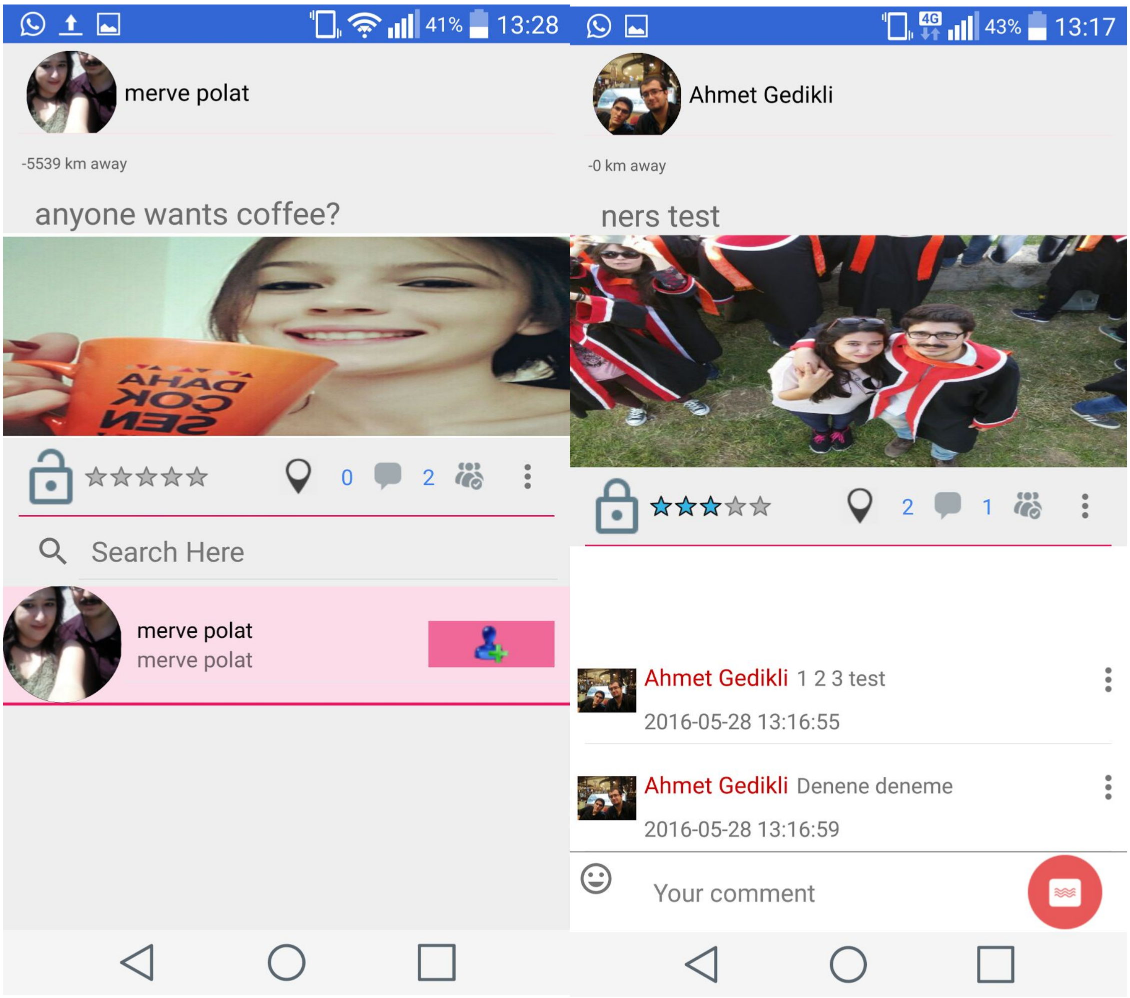
Mustafa MuratCoskun □

OğuzArtıran ☑

Figure 5.6.8.3

5.6.9 Event Activities Interface

Users will be able to rate events, see participants of the events, see the comment counts of events , see rate count of events, see the rate of the events, see the participation counts of events and see the comments of the events. These processes and their relevant interfaces can be seen in figure 5.6.9.1 and figure 5.6.9.2.



merve polat

-5539 km away

anyone wants coffee?

Search Here

merve polat
merve polat

Ahmet Gedikli

-0 km away

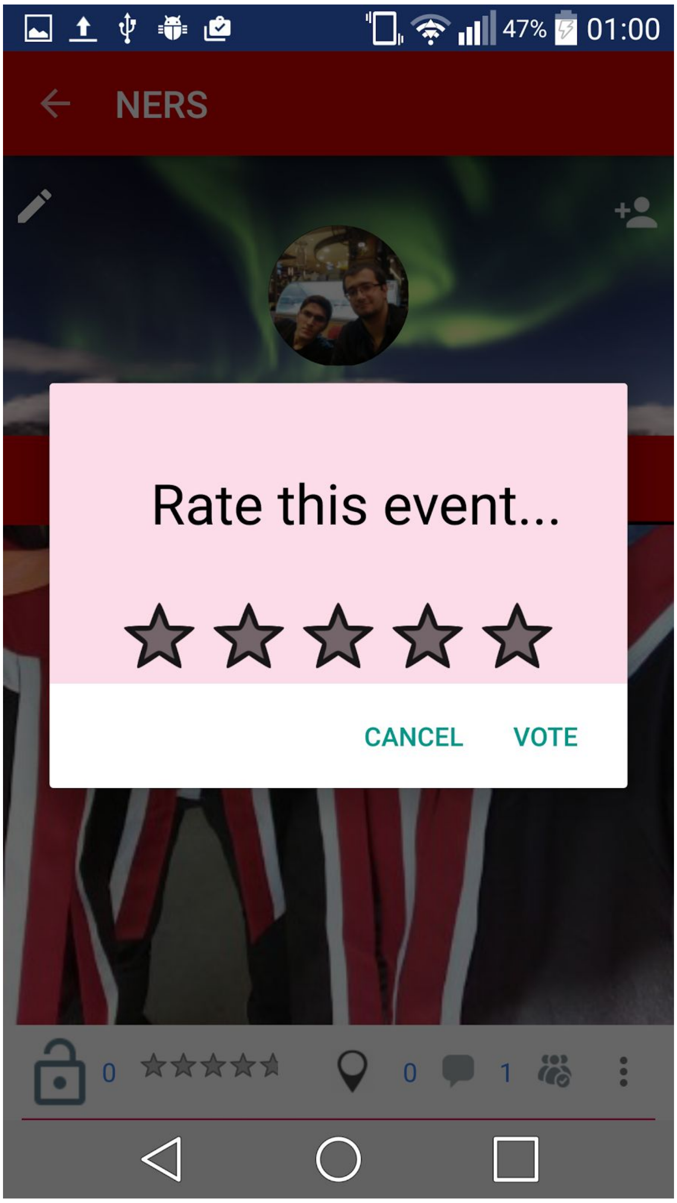
ners test

Ahmet Gedikli 1 2 3 test
2016-05-28 13:16:55

Ahmet Gedikli Denene deneme
2016-05-28 13:16:59

Your comment

Figure 5.6.9.1



47% 01:00

NERS

Rate this event...

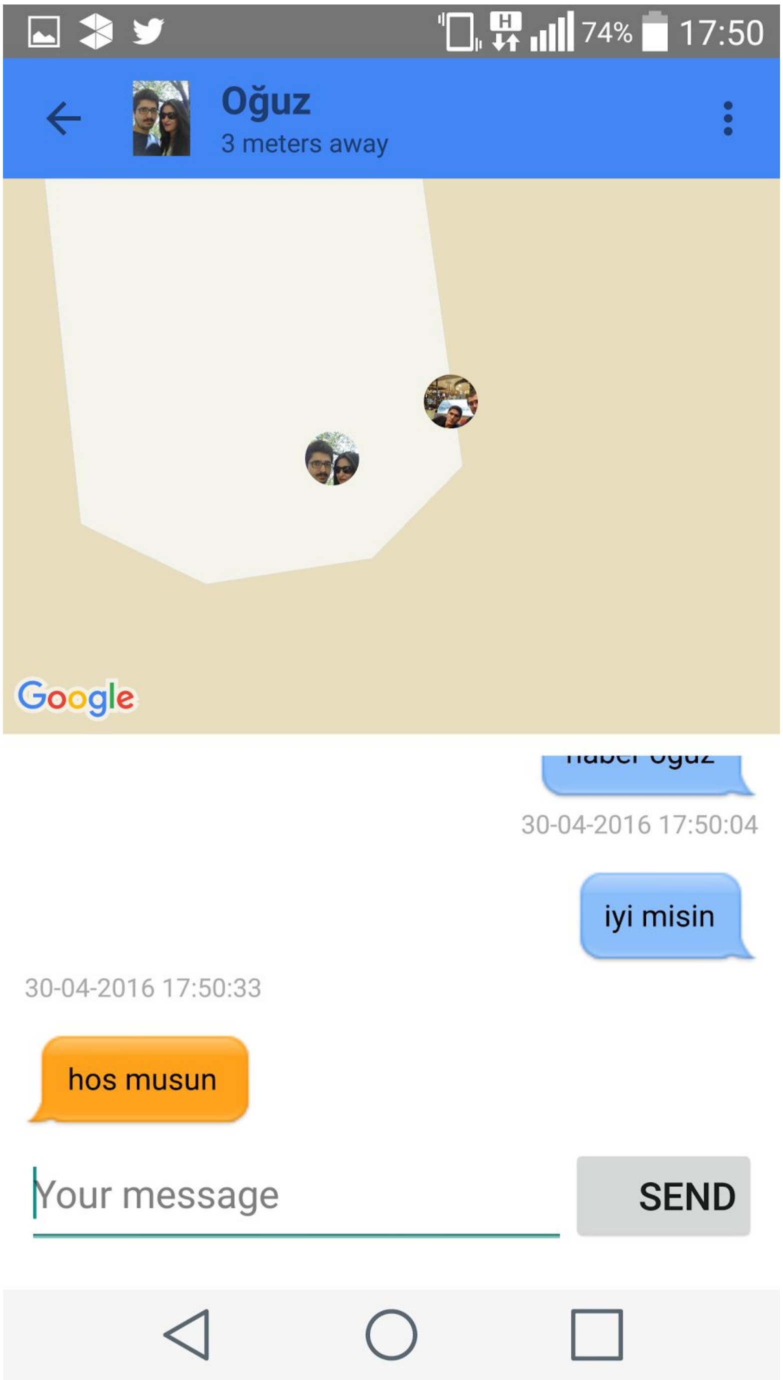
☆☆☆☆☆

CANCEL VOTE

Figure 5.6.9.2

5.6.10 Send Message Interface

In send message interface users will be able to see their friends in real time and they will be able to chat with them synchronously. The relevant interface example can be seen in figure 5.6.10.1.



naber oguz

30-04-2016 17:50:04

iyi misin

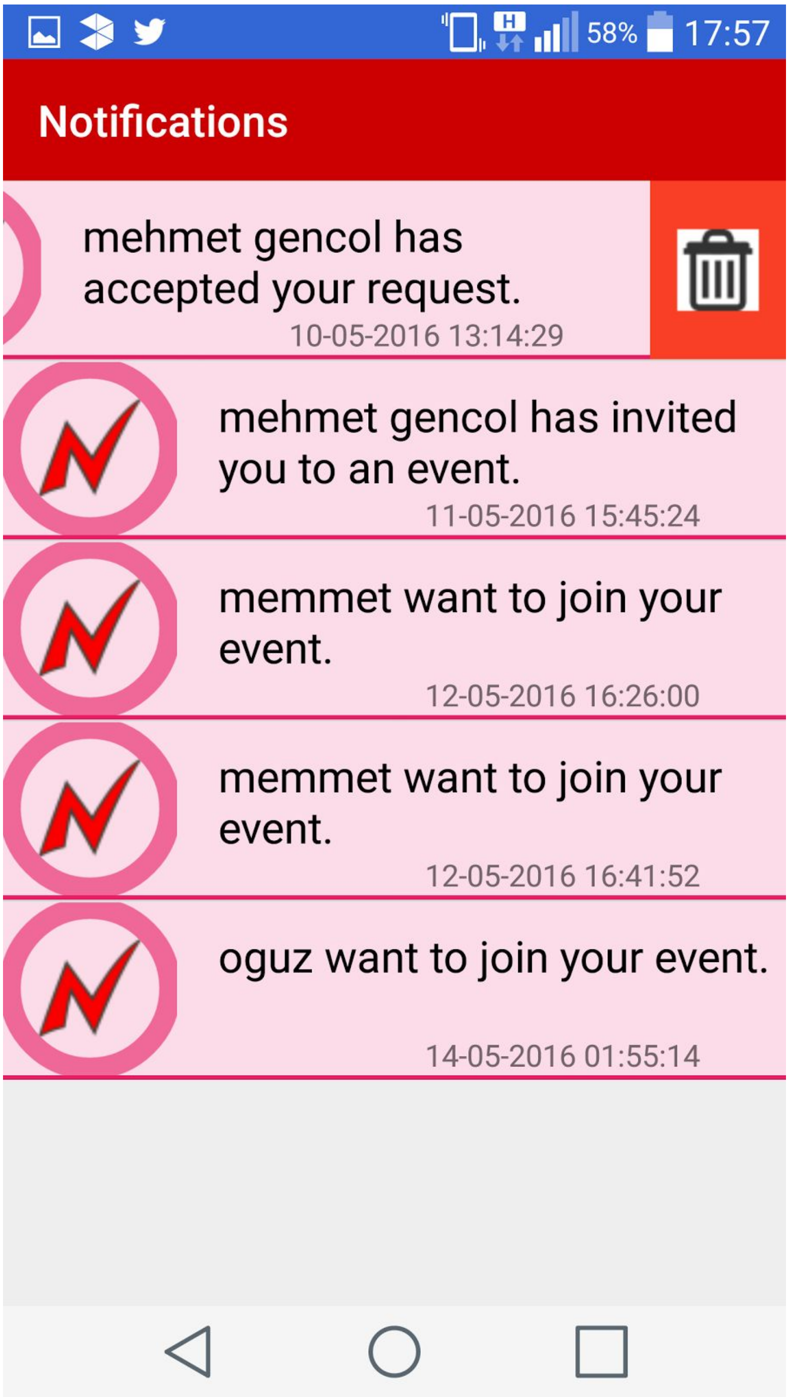
30-04-2016 17:50:33

hos musun

Figure 5.6.10.1

5.6.11 Notification Interface

Users will be able to view the notifications which are relevant them like getting message and invitation. An example interface is provided in figure 5.6.11.1.



Notifications

mehmet gencol has accepted your request.
10-05-2016 13:14:29

mehmet gencol has invited you to an event.
11-05-2016 15:45:24

memmet want to join your event.
12-05-2016 16:26:00

memmet want to join your event.
12-05-2016 16:41:52

oguz want to join your event.
14-05-2016 01:55:14

Figure 5.6.11.1