May 27

Software Design Document

The Team Naegling

Team Members:
Johan Nilsson
David M. Szabo
Elsa Wide
Henrik Edholm
Simonas Stirbys
Mikaela Lidström

Supervisor: William Granli

Table of Contents

1. Introduction

The Software Design Document (SDD) describes how the functional and nonfunctional requirements recorded in the Software Requirements Specification (SRS) as well as the preliminary user-oriented functionalities are translated into more technical system design specifications, describing the system architecture and working as a "blueprint" when developing the system. The SDD describes both technical- and user-interface-design goals, considerations and decisions. Likewise it provides a high-level overview of the system architecture and its data structures, as well as specifying the communication between its different components.

1.1 Purpose of the System Design Document

The Software Design document is used to track necessary information required to define architecture and system design in order to guide the team during development of the system. The Software Design document is incrementally and iteratively updated during the system development process, based on the dynamics and iterations of the project and its requirements change. This document may be shared between all the project stakeholders, including the product owner, whenever requested.

2 General Overview and Design Guidelines/Approach

This section describes the principles and strategies to be considered when designing and implementing the system.

2.1 General Overview

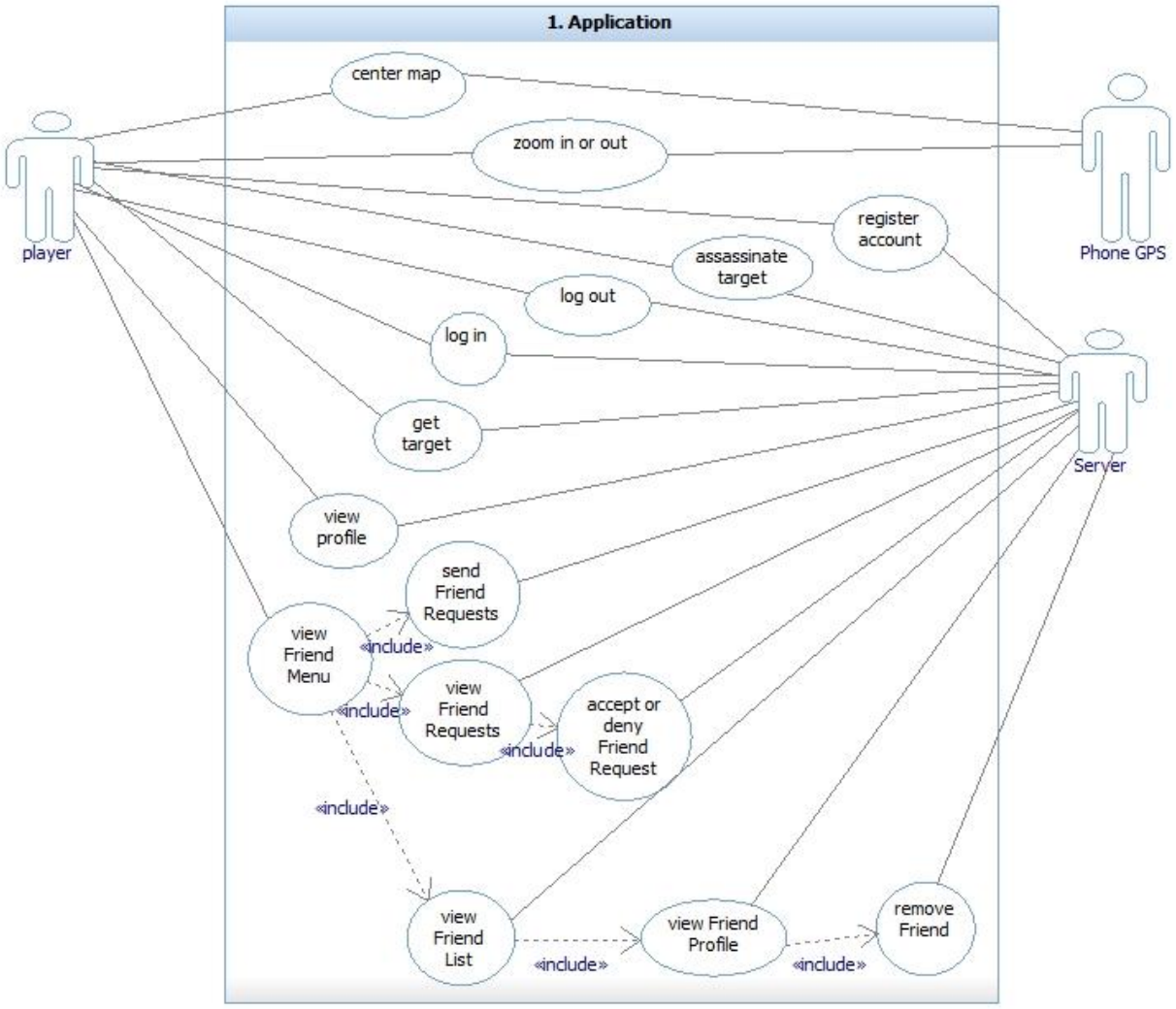
"The system consist of a game for android platforms geared towards providing entertainment for its users, in which players can virtually assassinate each other using a GPS" (Appendix J – SRS)

The general design goals attempts to solve the functionality described in the SRS, with the architectural design goals mainly aiming at the logic functionality, mechanics and high performance of the system, likewise the user-interface design goals are oriented towards providing a greater level of user-friendliness and entertainment/aesthetics for the user.

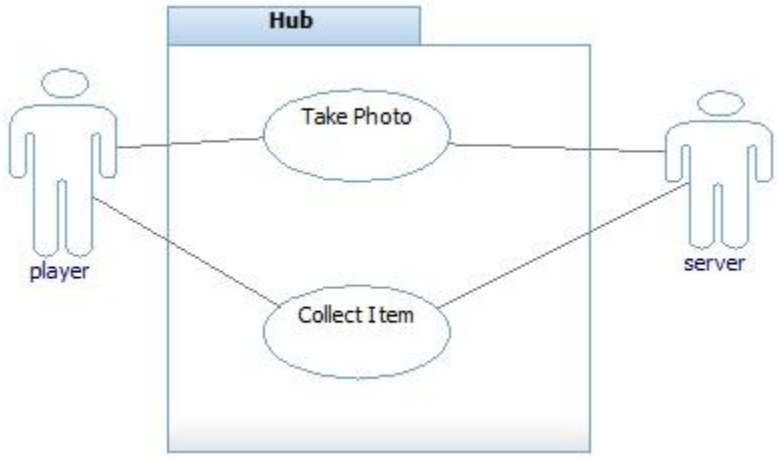
In order to describe what functions are provided by the system, a high-level overview of the system functionality, in the form of use case diagrams (more detailed descriptions of the use cases can be viewed in the SRS) is presented on the next page as follows:

- Picture 2.1.1 presents the use cases within the android application can be viewed, with actors Player, Server and GPS interacting with the system.
- Picture 2.1.2 presents the use cases for the Hub system (Raspberry Pi, NFC Reader, and Camera).

2.1.1 Application Use Cases



2.1.2 Hub Use Cases



2.2 Assumptions / Constraints / Risks

2.2.1 Assumptions & Dependencies

General assumptions & dependencies include the following (From Appendix J – SRS):

- The assumption is made that Google maps are free to use for non-commercial purposes and that we will be able to implement it in the app.
- The assumption is made that the GPS will be accurate and that it will update locations regularly.
- The assumption is made that the server will retrieve, process and display information in a reasonable amount of time.
- The user should have an android device.
- The app is dependent on the internet connection.
- The game is dependent on users having the GPS on.
- The app is dependent on the NFC.

2.2.2 Constraints

General design/implementation designs include the following (Appendix J – SRS):

- Python, C, Java, XML, and PHP languages will be used for programming the system.
- Developed software will run on Android devices.
- The developed system is required to incorporate Raspberry Pi into its function by using it scan phones and collect items.
- The system will be developed using Scrum software process.

2.2.3 Risks

Presented below are the identified risks associated with the system design together with their analyzed probability and proportional effects as well as proposed strategies: (Appendix G - Quality Management Plan, 8. Risk Management):

- Requirements change & inflation, probable risk with serious effects.
    - Strategy: Prioritize within current implementation, documentation.
- Assumptions about users, occasional risk with minor effects.
    - Strategy: Design an easy and logic interface that is understandable.

3 Design Considerations

In this section the reasoning behind our design, and the changes made during the project, will be explained and discussed.

3.1 Goals and Guidelines

When we first decided on our project idea, it was clear to us what we wanted; a game that would be easy to play and intuitive. We set out with the goal to create a product that feels and runs like an existing product, and this also prompted us to aim for quality over quantity. It was also important to make the product portable, it was in fact vital for the game to work at all that we were able to create it as an app. Further goals were such as usability, user security and a

well performing, reliable app.
We also had guidelines and principles within the coding we tried to follow where possible. These were as follows:

- Minimize Hard Coding
- Don't Repeat Yourself Principle
- Naming Conventions
- Comment the Code
- Consistent Programming

3.2 Development Methods & Contingencies

The main method we used during the development was iterative. We worked with small goals were we included several planned finished versions of the product, each one with more features than the last. This way we ensured that if we had any time problems we would still have a functional working product to show, it was also a way to take away pressure for trying too many things at one time and instead focusing on the smaller tasks.
Because of the methods we have chosen any contingencies where more or less eliminated, once we had a first version anything that was not relevant was easy to drop if it seemed it would be more trouble than worth.

3.3 Architectural Strategies

While designing the project we had to make several decisions regarding what would be the most efficient way to go about the project, but we also had to try to keep as true to our original idea as possible. This section will state our design choices and explain the reasoning behind them.
One of the first major decisions we made was what platform to develop the product for. Looking into app development we felt that android had the most useful information to offer and did not have as many restrictions for developers as for example Apple does. The programming language for the app-development was java, since it is obliged by Android and thankfully it was also a language we all already felt comfortable with. Similarly python was the go to language when working with raspberry Pi and PHP for online database and server. We choose to implement Google Maps since it is encouraged by Android and a simple, ready to use service.
Since our application needed connection to internet we decided to use an online server with REST API to return JSON data from the server.
We wanted to have a personal feel to our game by including profile pictures and since we designed the game to have a home base, the raspberry PI, that we wanted to be more included in the game, we decided that the feature should be implemented on the PI. To use the PI camera open source code was common sense.
During the coding and implementing of our design we decided to have a bug report that was constantly updated as a way for us to keep track of the errors and how long it took to fix them.
On the server the MYSQL database we choose was default and since it was easy to work with PHP to access the information and since it fitted our needs we saw no need to look for anything else.

4 System Architecture and Architecture Design

This section outlines the system and hardware architecture design of the system.

The system as a whole consists of: an android application with a local SQLite database, a hub station (consisting of a raspberry pi connected to a NFC scanner and a camera) and a webserver with a MySQL database.

The application retrieves user input and provides the user with a GUI and output in the form of results from actions. The user input consists of personal, social & integrity data for the user profile as well as for triggering game and NFC functions. The application communicates with the server through JSON in order to query the MySQL database where the data shared between different instances of the app are stored. The application also uses a local database on the android device for temporary storage of user data.

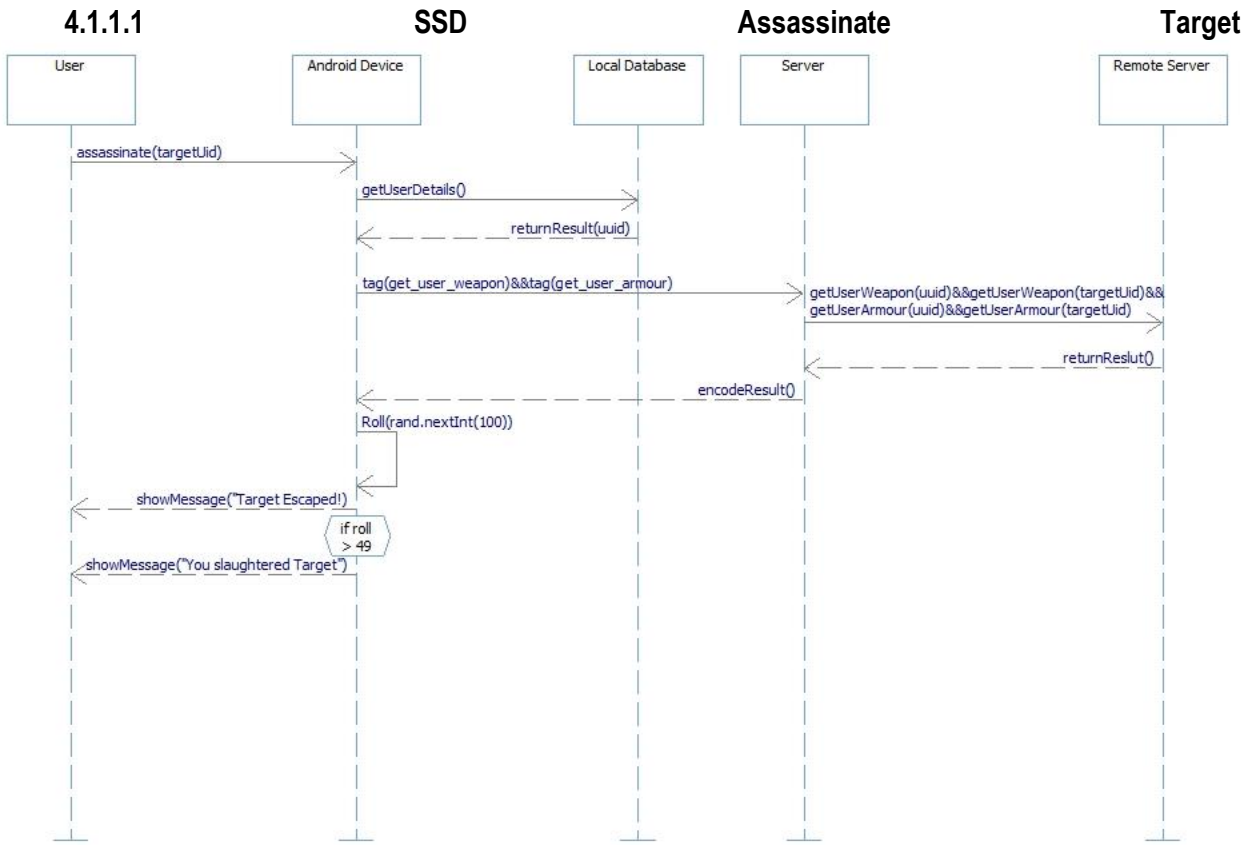
The application retrieves data from the android device's internal GPS in order to locate the positions of the different users, the data is stored in the database as integers representing Latitude and Longitude coordinates.

The NFC scanner of the Hub sends and retrieves NFC signals which are parsed within the app as "Tags" through Android NFC APIs.

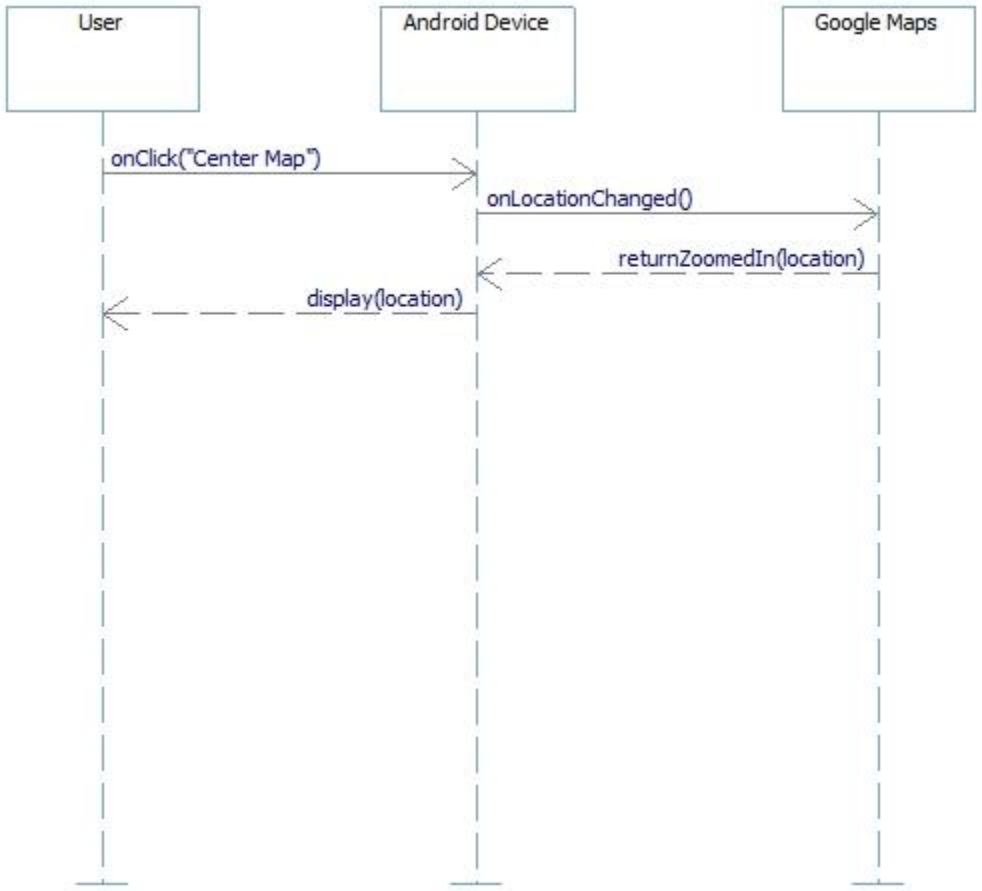
4.1 Logical View

The logical view presents the functionality that the system provides to the end-users in the form of System Sequence Diagrams, Communication Diagrams and Class Diagrams.

4.1.1 System Sequence Diagrams

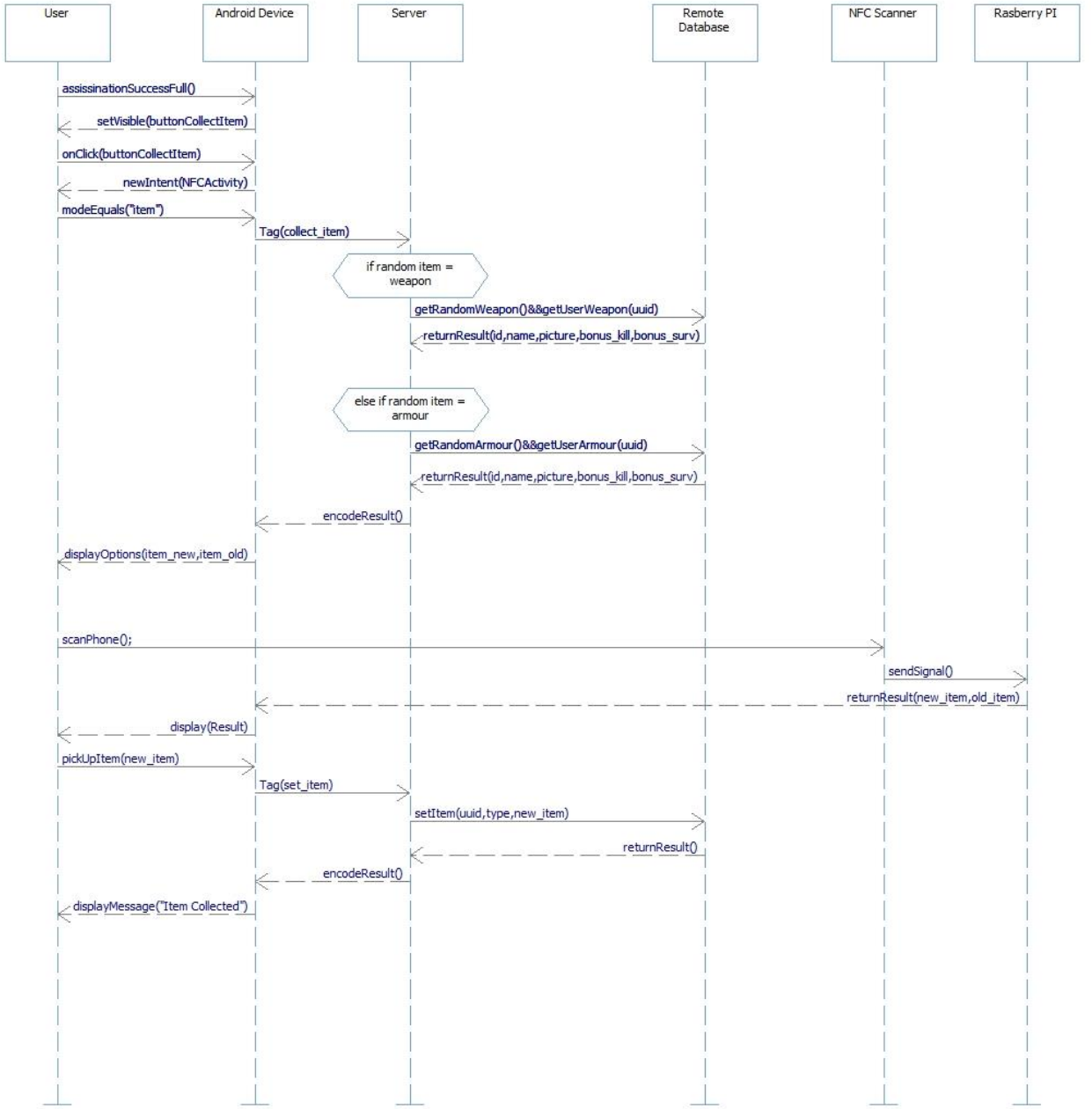


4.1.1.2 SSD Center Map

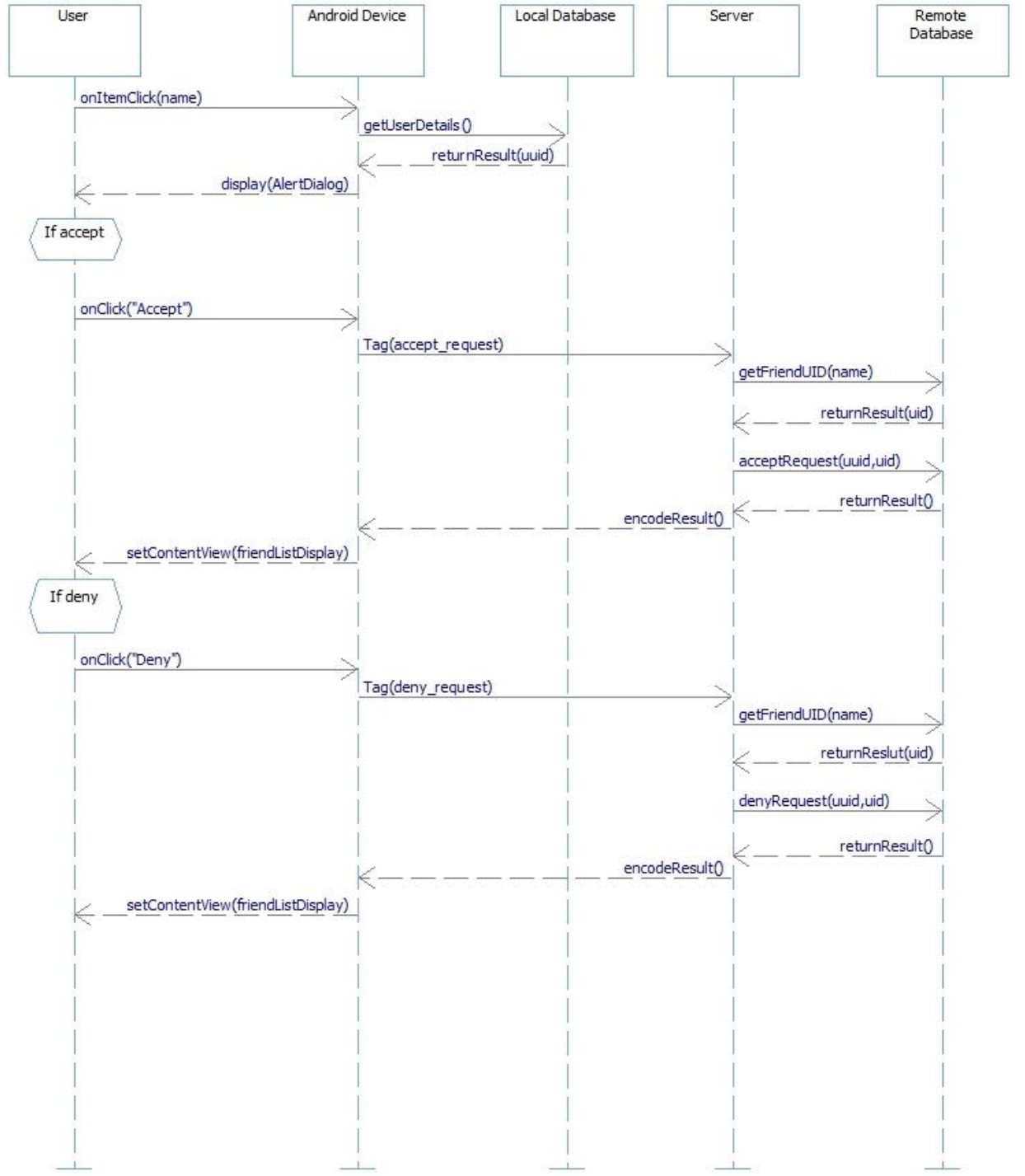


4.1.1.3 SSD Change Password

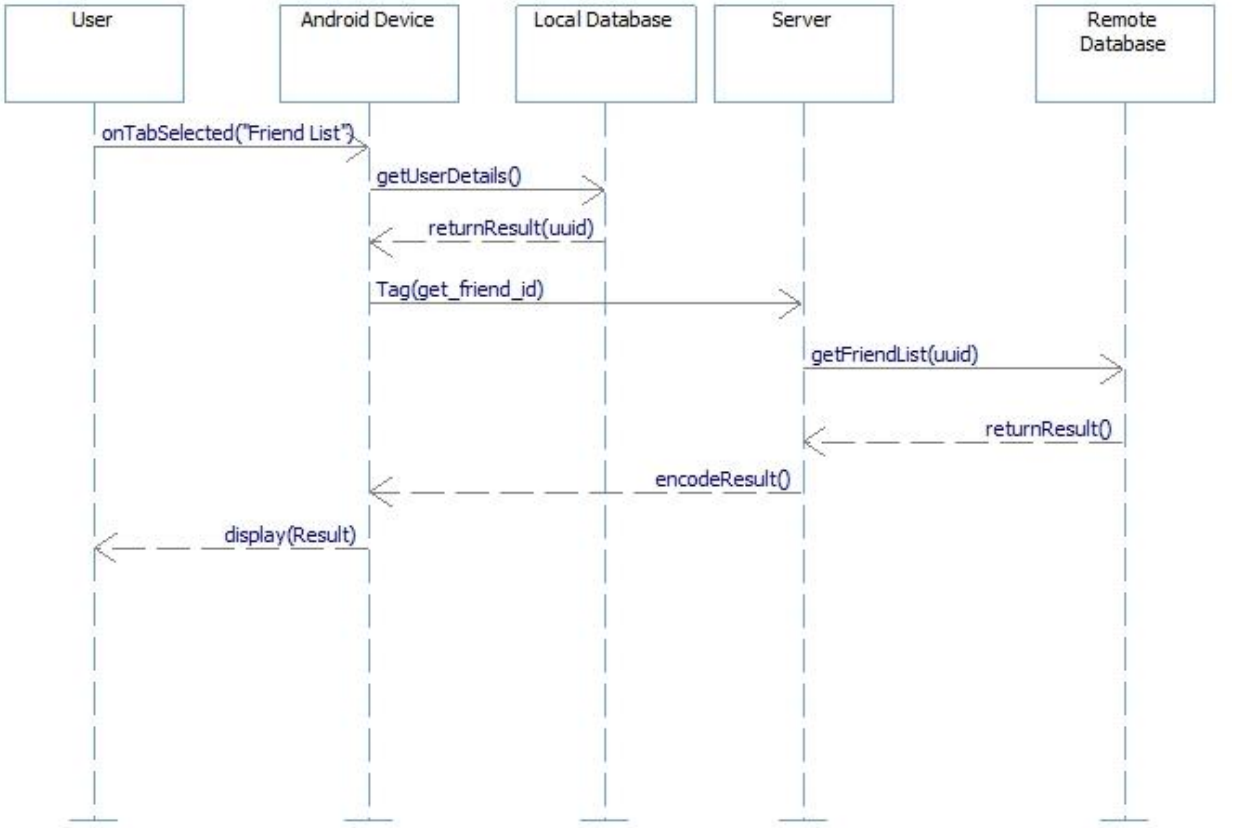
4.1.1.4 SSD Collect Item



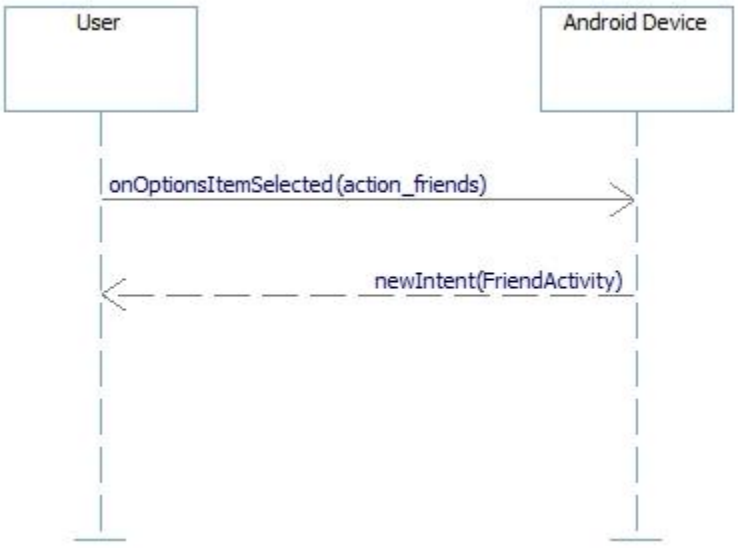
4.1.1.5 SSD Deny/Accept Friend Request



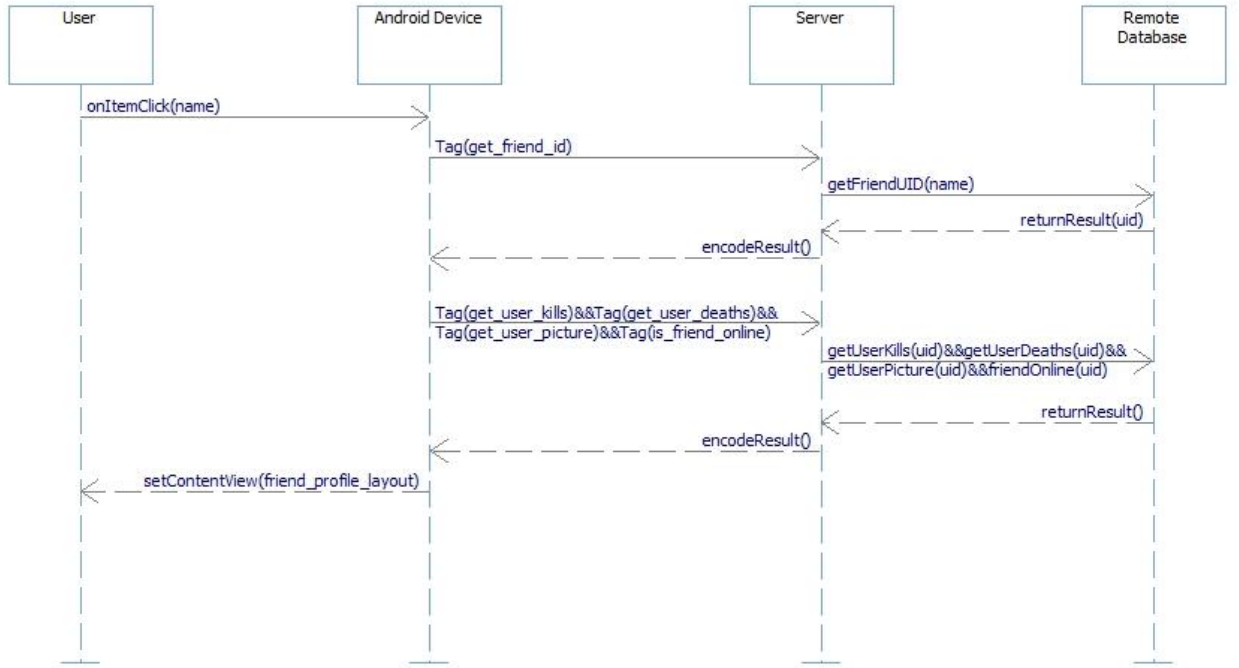
4.1.1.6 SSD View Friend List



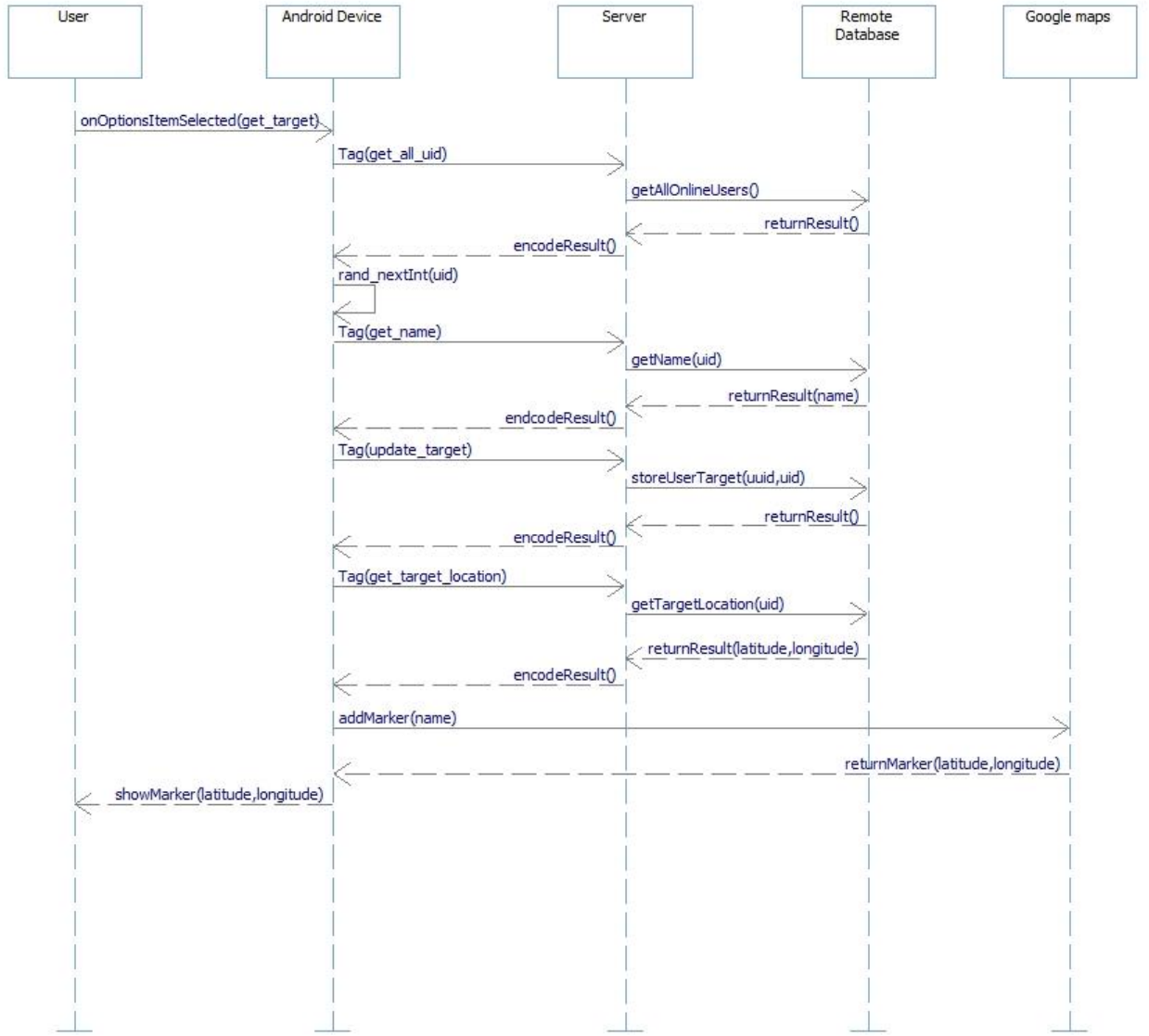
4.1.1.7 SSD View Friend Menu



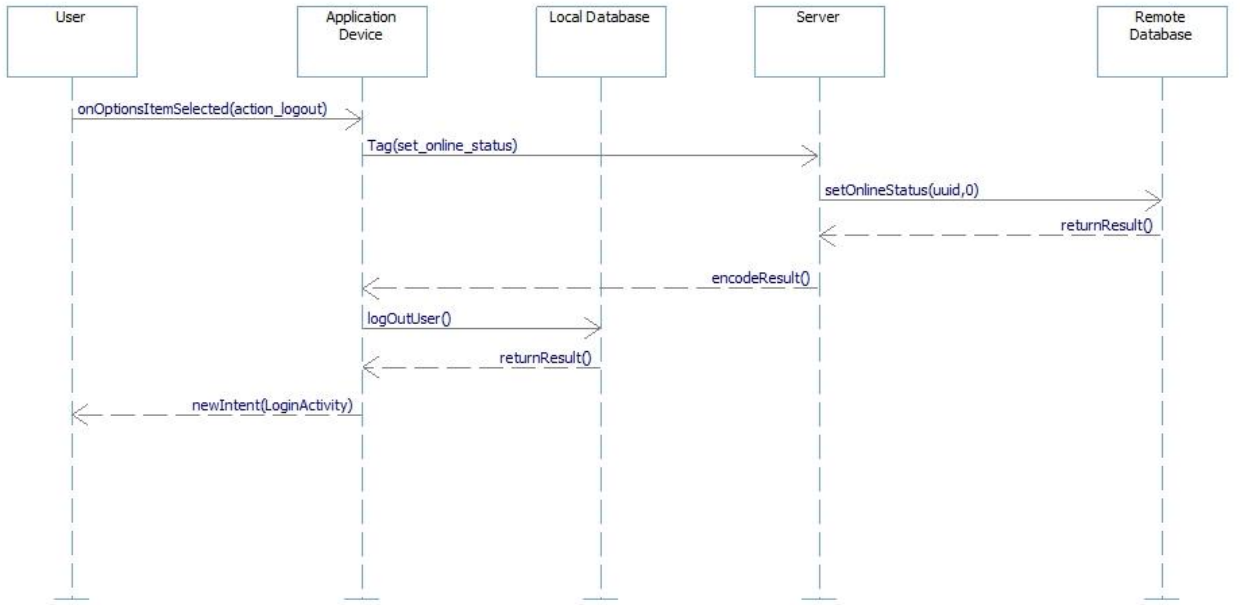
4.1.1.8 SSD View Friend Profile



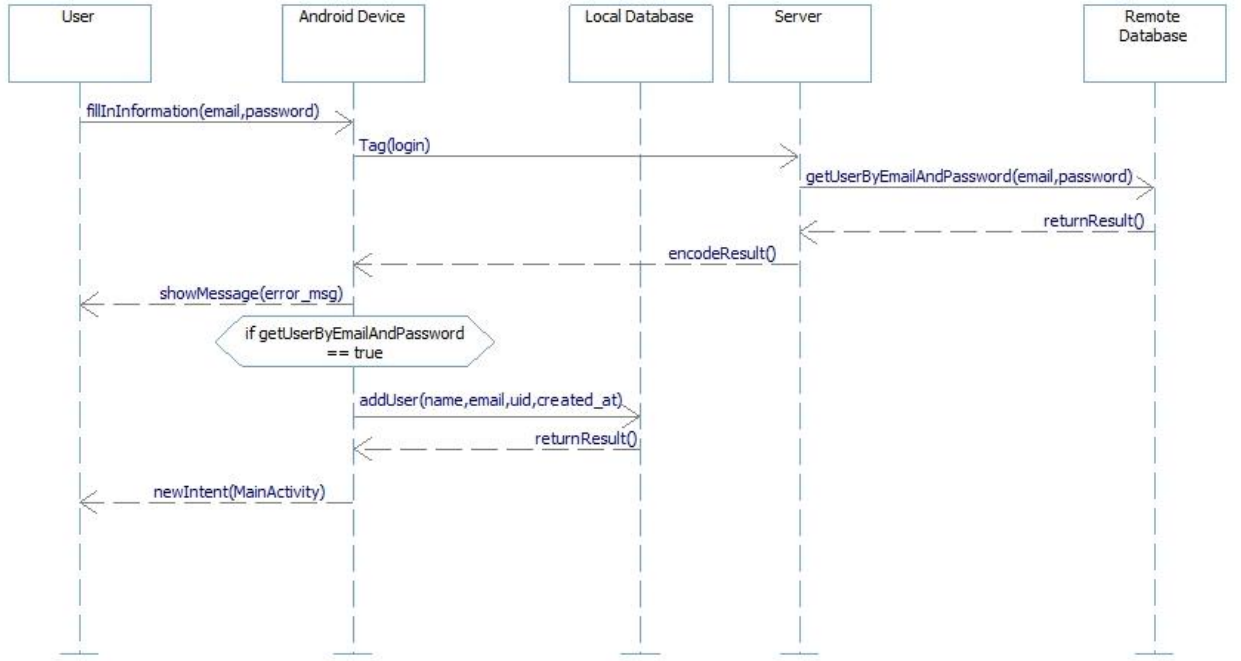
4.1.1.9 SSD Get Target



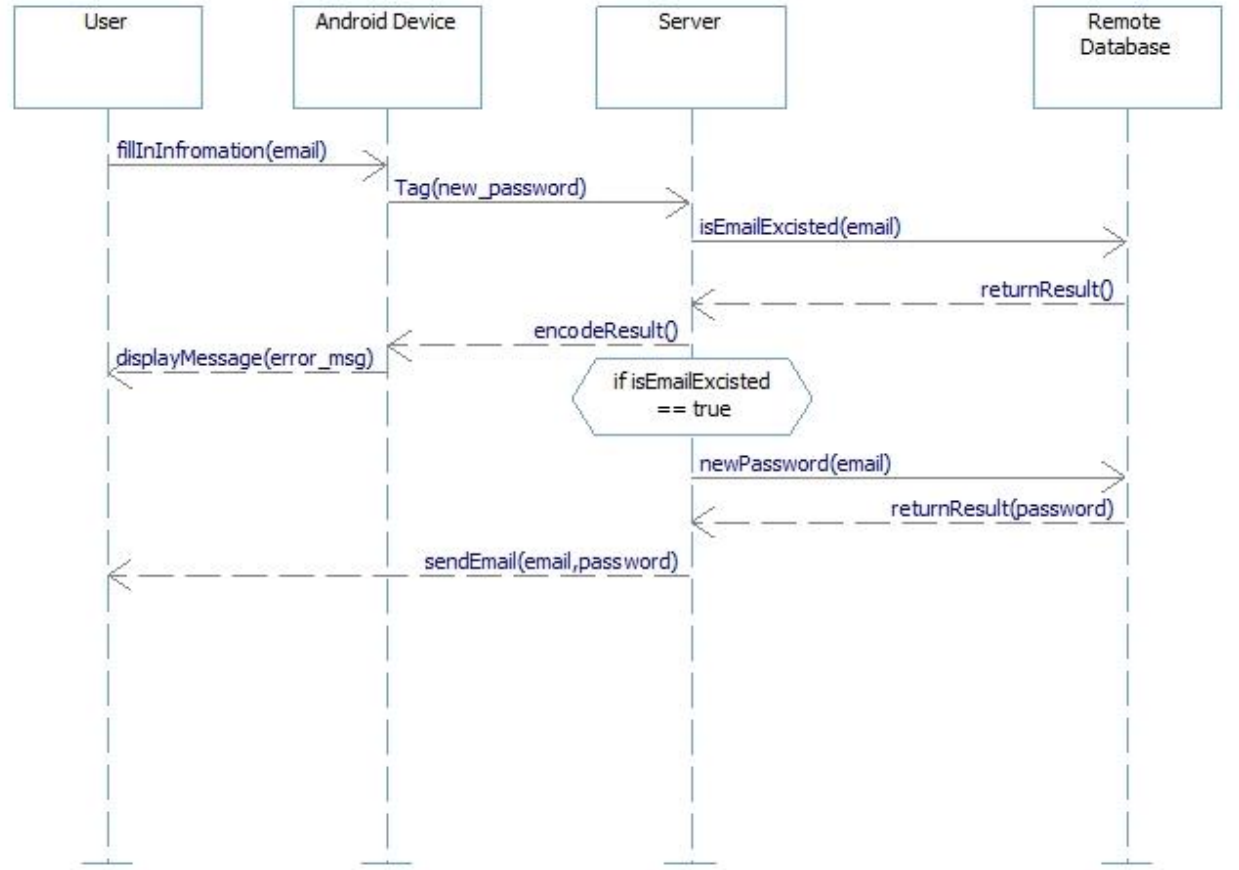
4.1.1.10 SSD Log Out



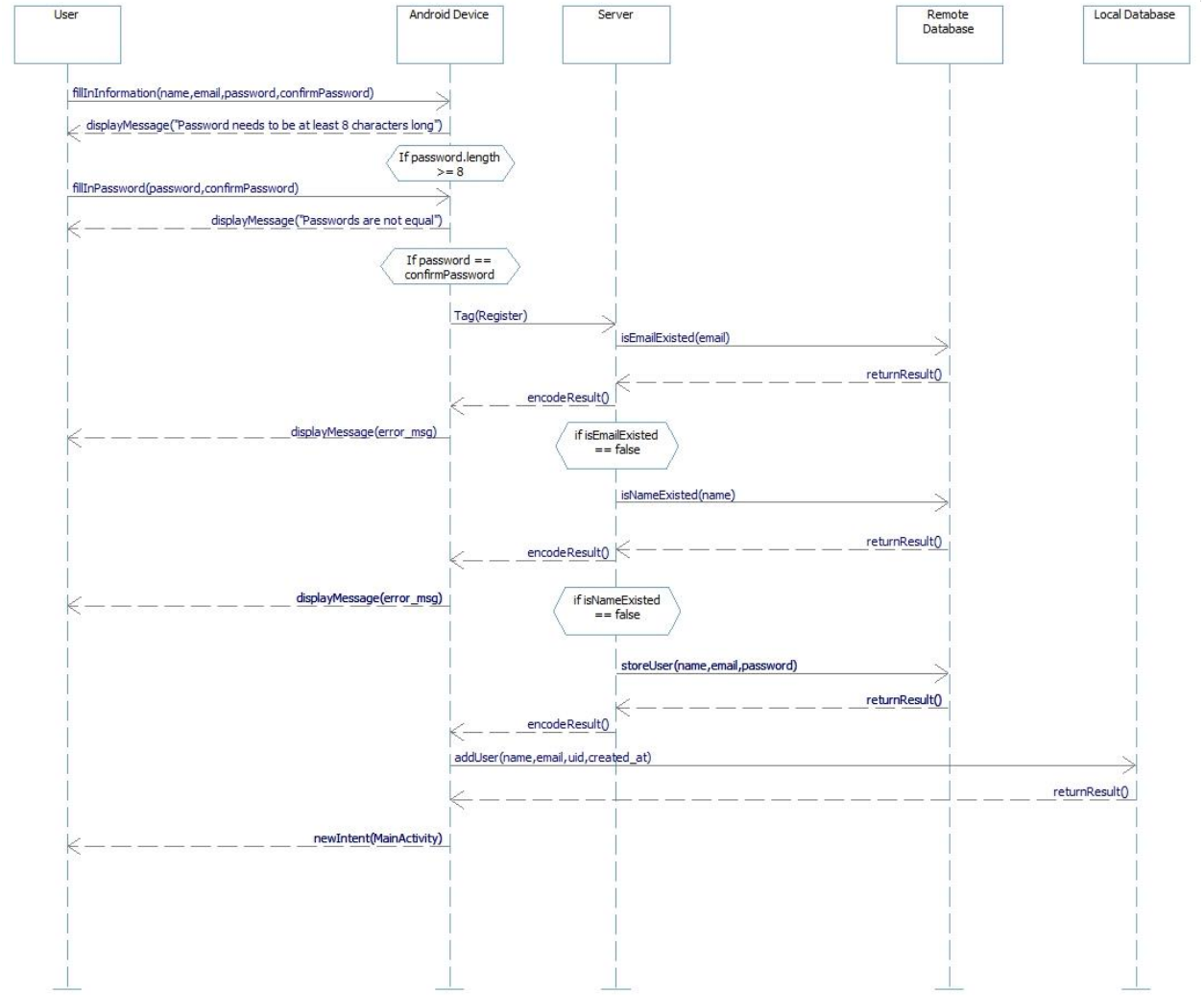
4.1.1.11 SSD Log In



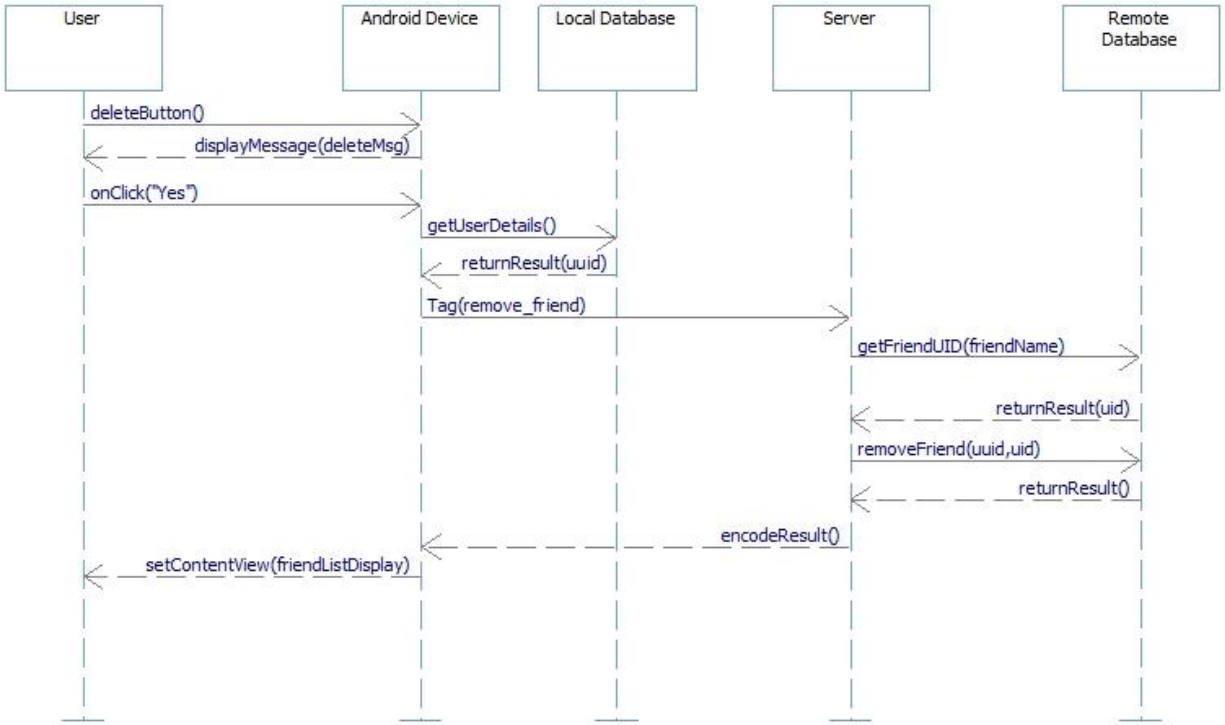
4.1.1.12 SSD New Password



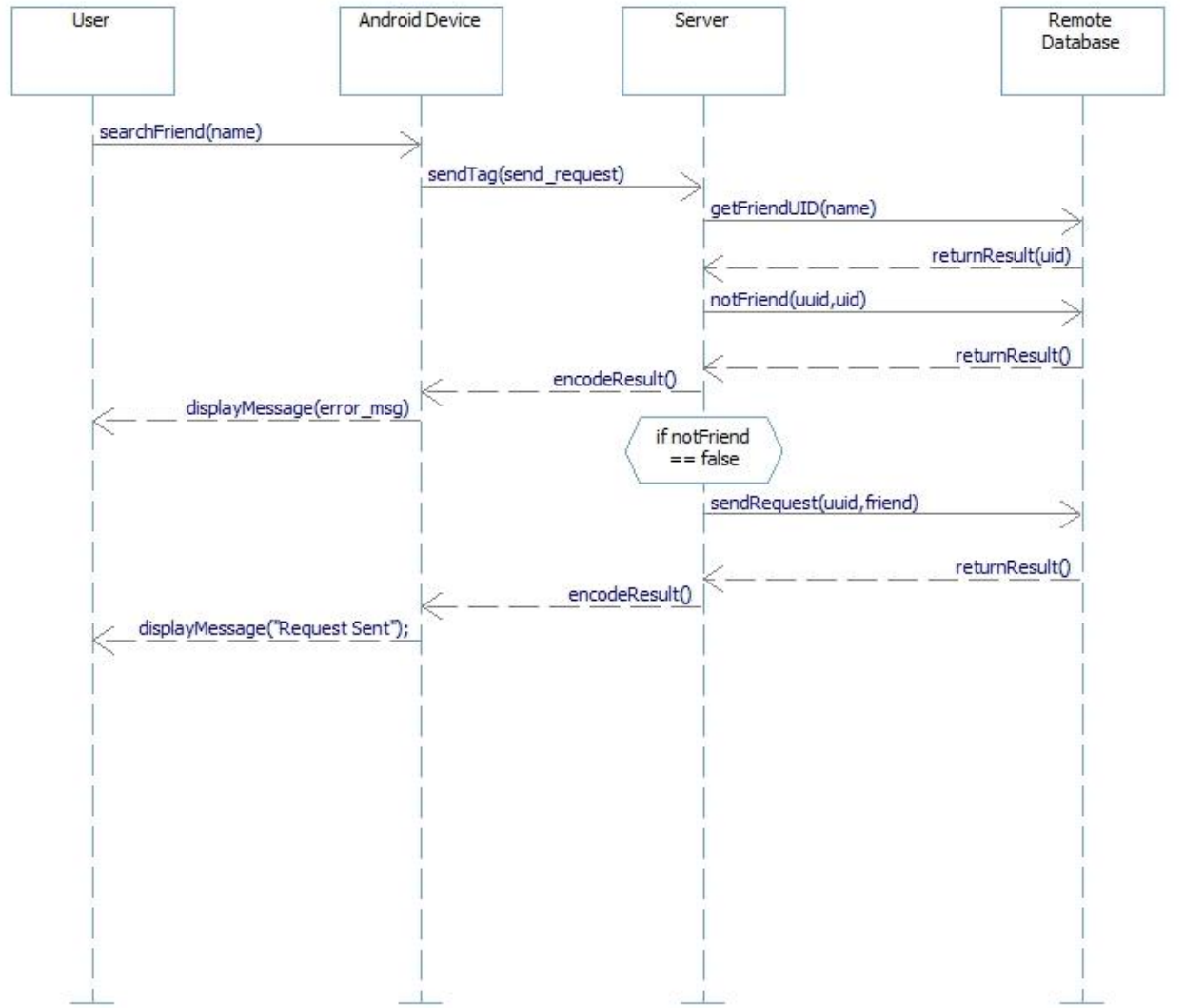
4.1.1.13 SSD Register Account



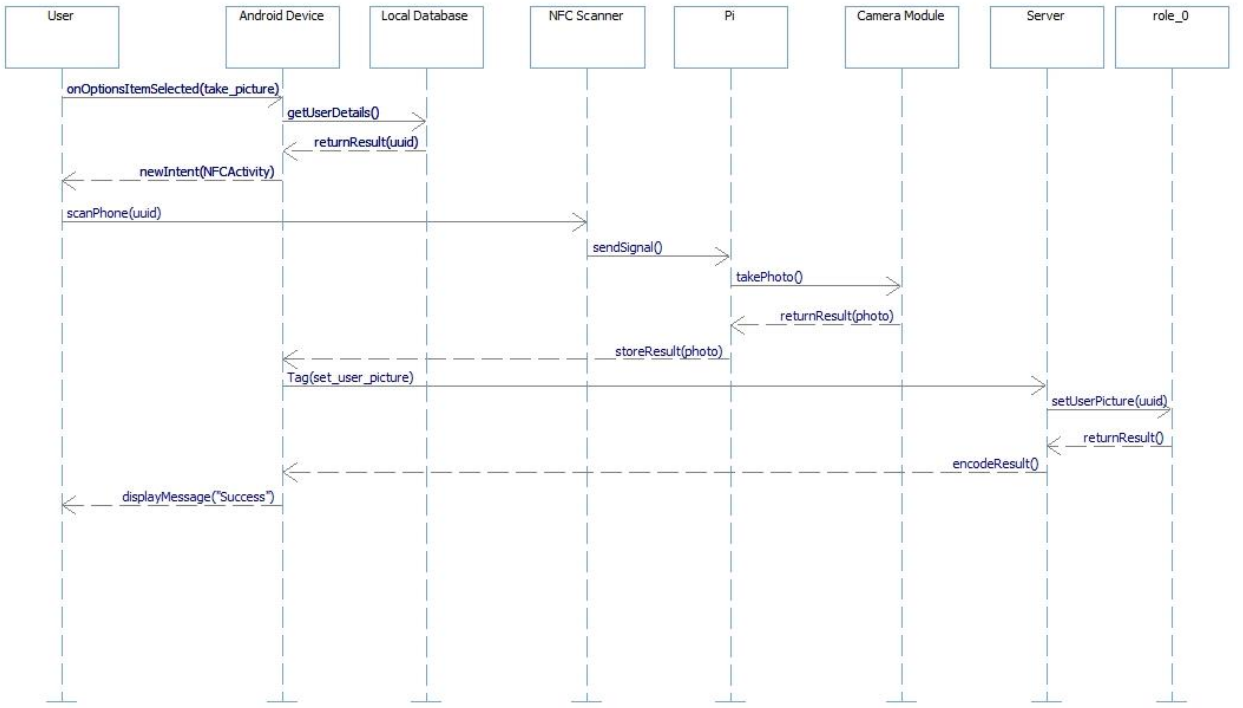
4.1.1.14 SSD Remove Friend



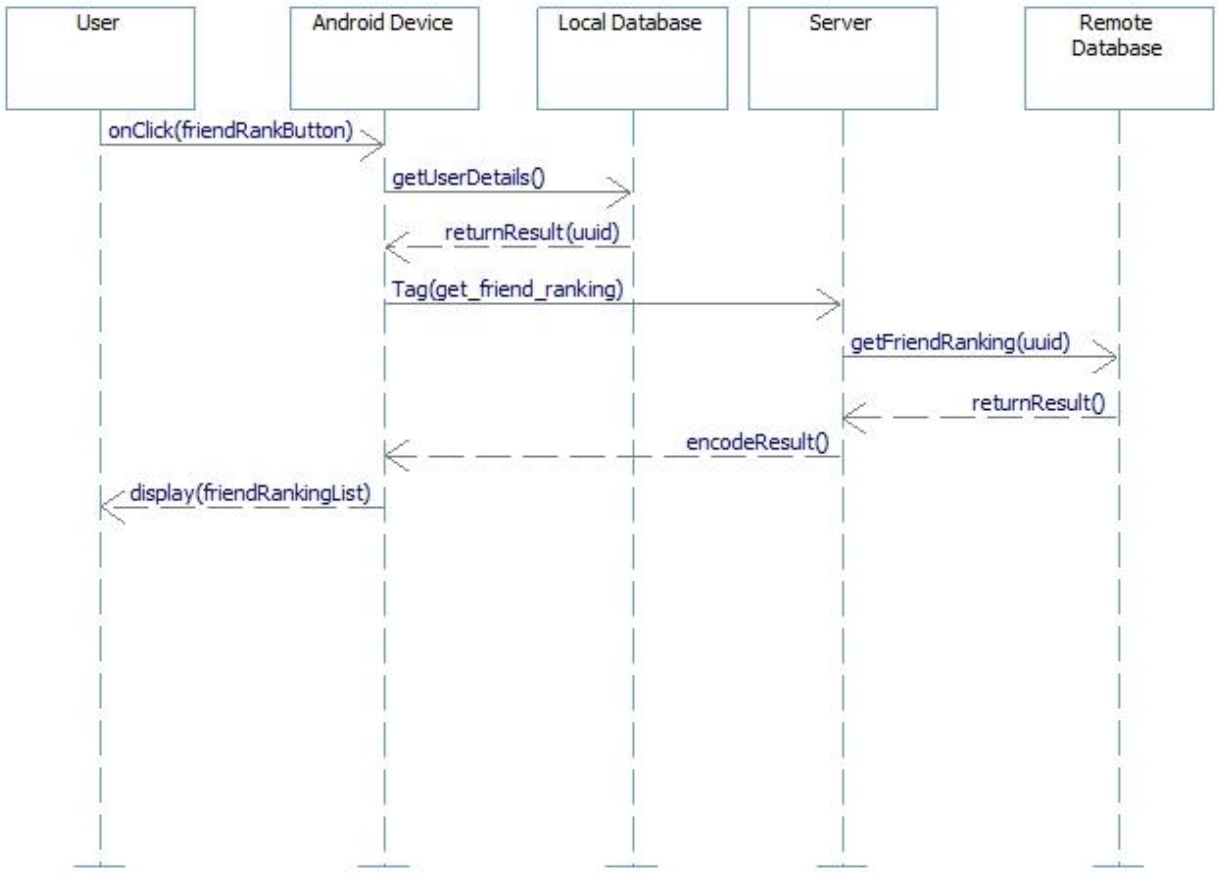
4.1.1.15 SSD Send Friend Request



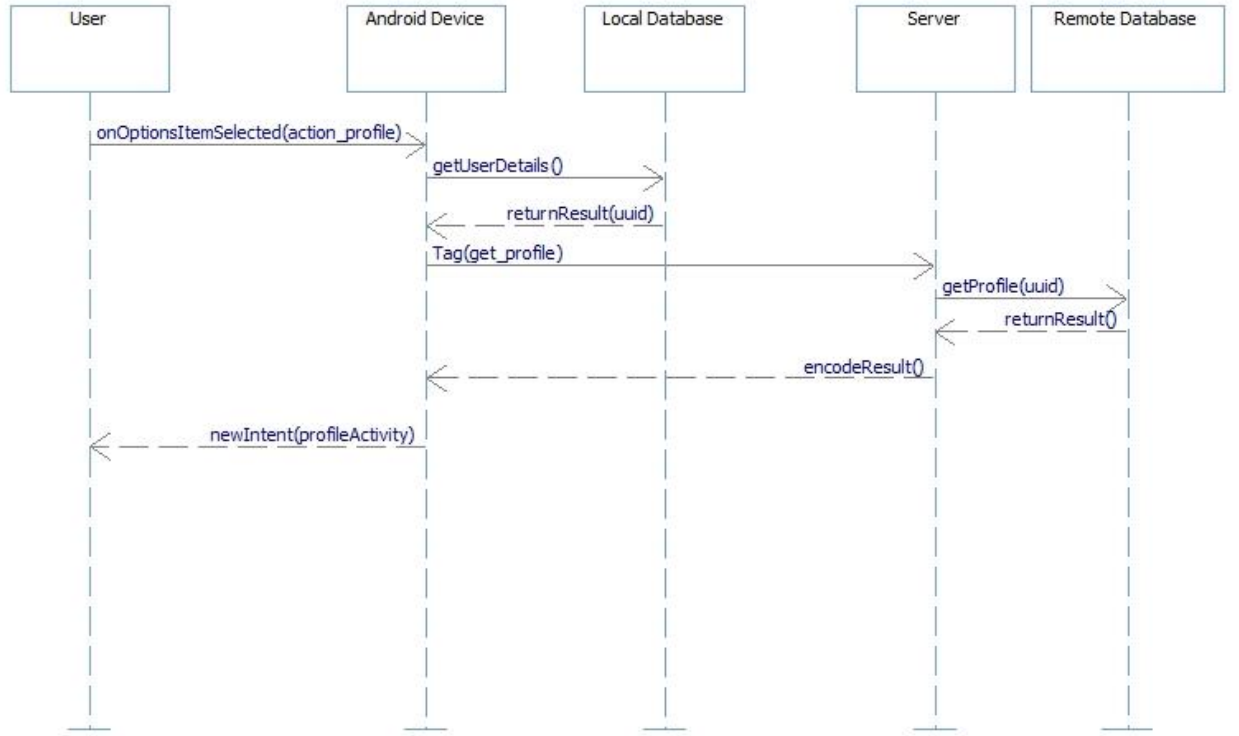
4.1.1.16 SSD Take Photo



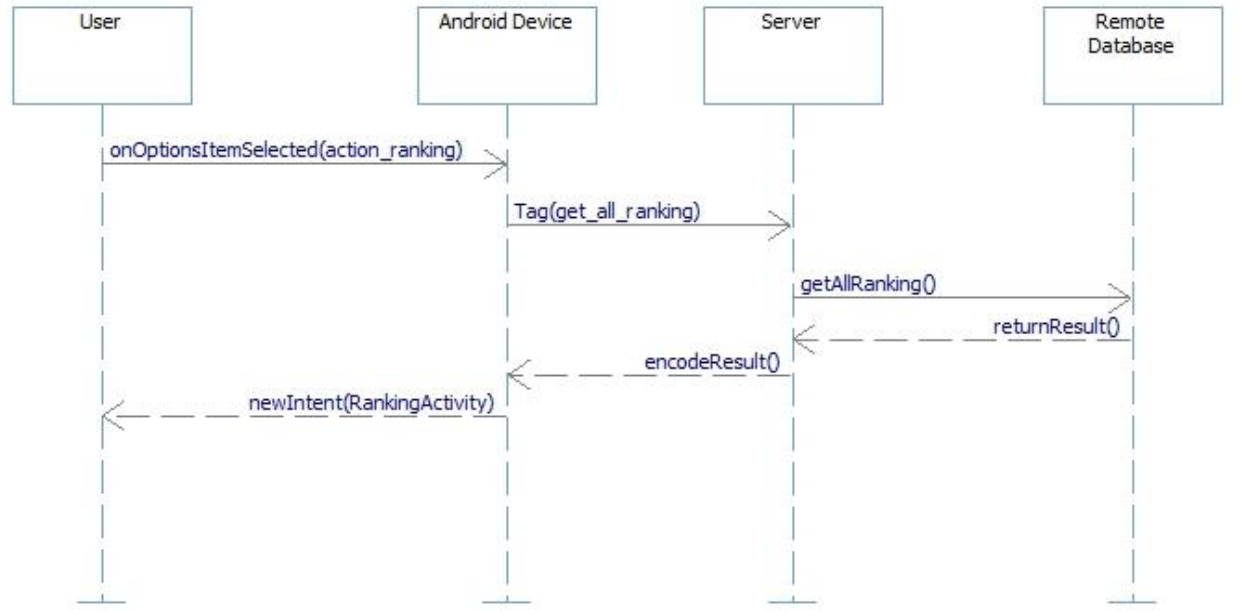
4.1.1.17 SSD View Friend Ranking



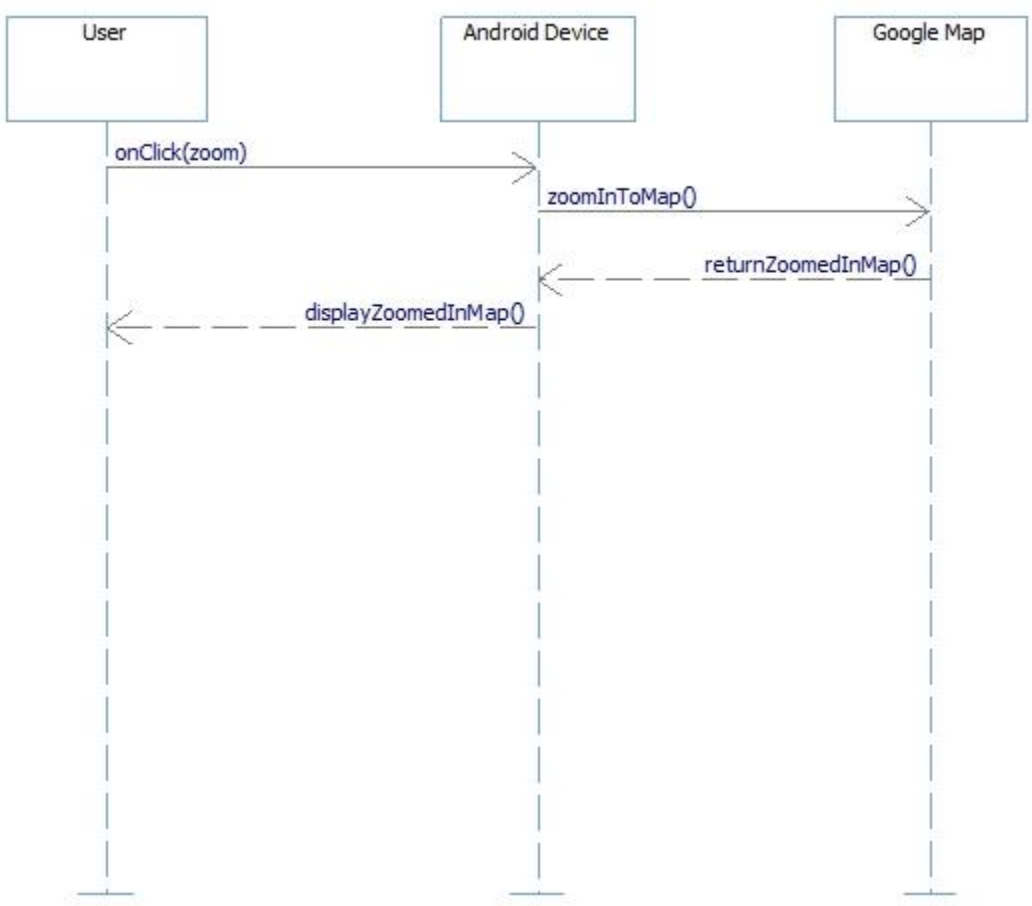
4.1.1.18 SSD View Profile



4.1.1.19 SSD View Ranking



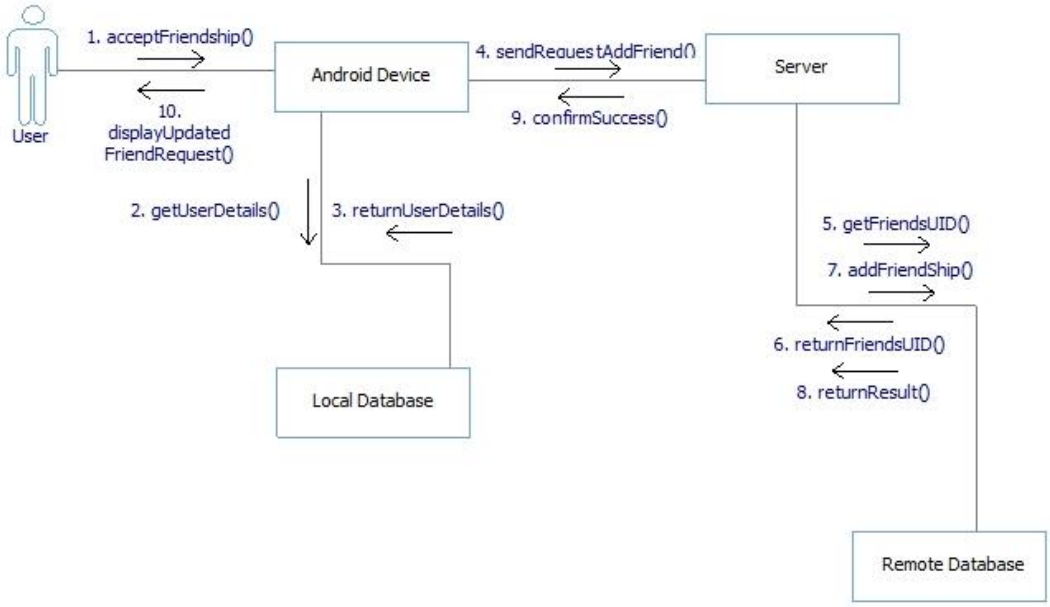
4.1.1.20 SSD Zoom Map



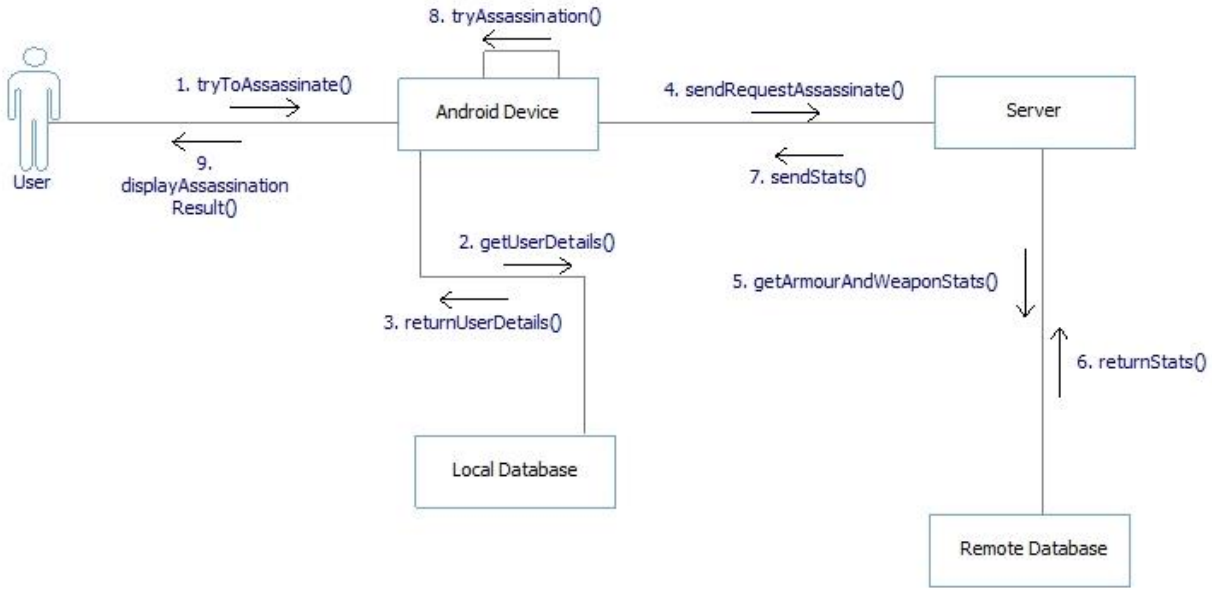
4.1.2 Communication Diagrams

A communication diagram, like a sequence diagram, is a vision of the user case. It is however more focused on the amount of communication between different objects rather than the time stream of the interactions. This is a way to see between what objects the most interactions are taking place.

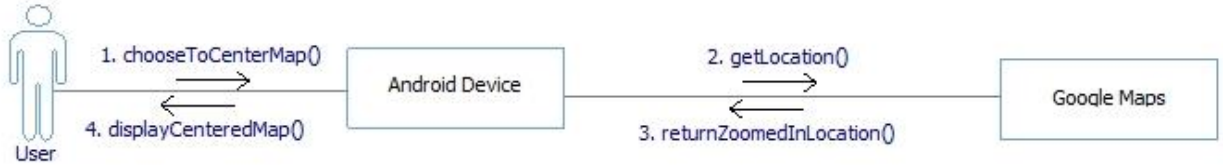
4.1.2.1 Accept Friend Request



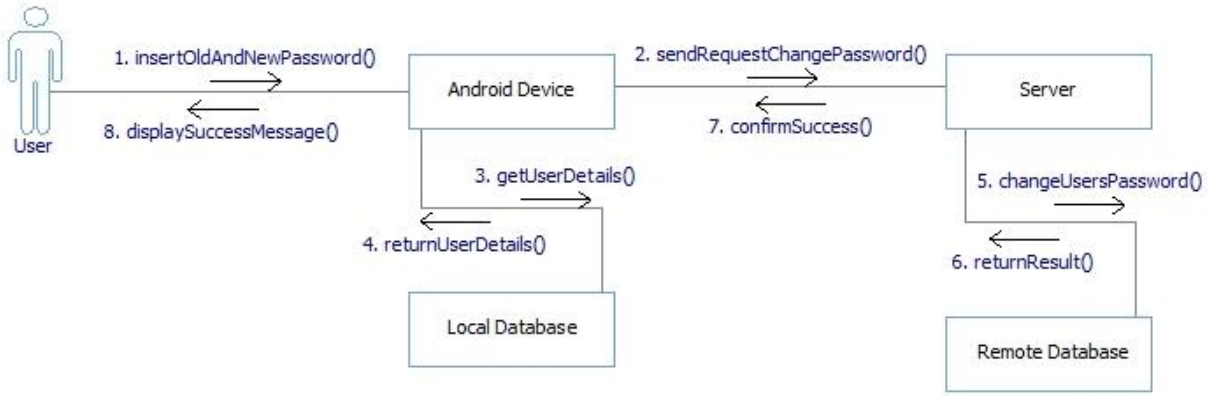
4.1.2.2 Assassinate Target



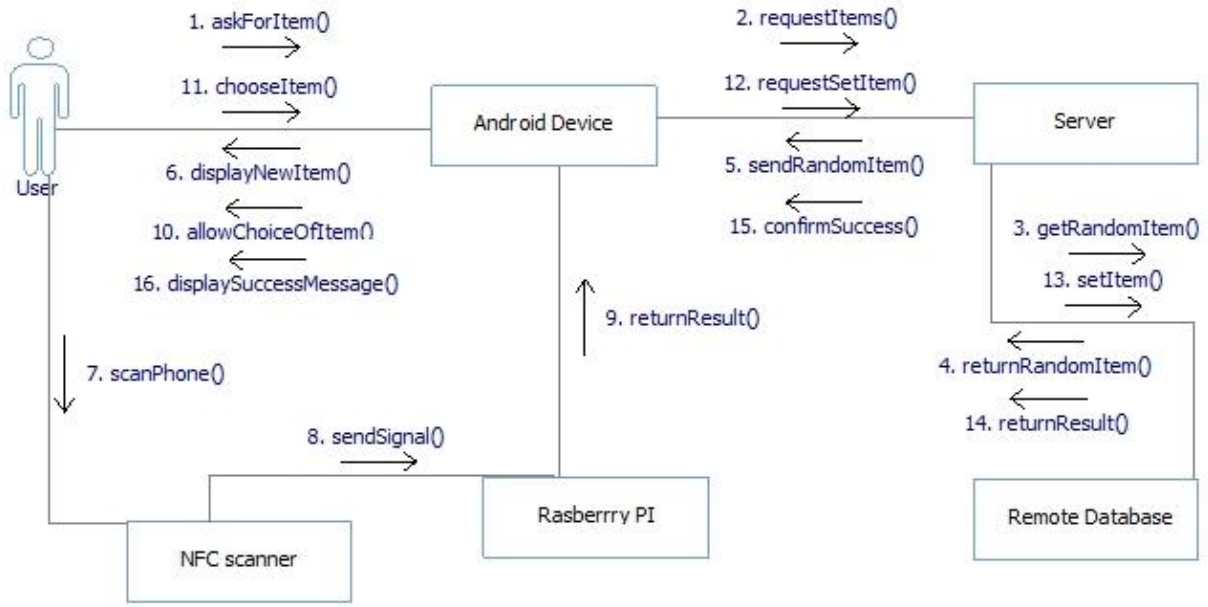
4.1.2.3 Center Map



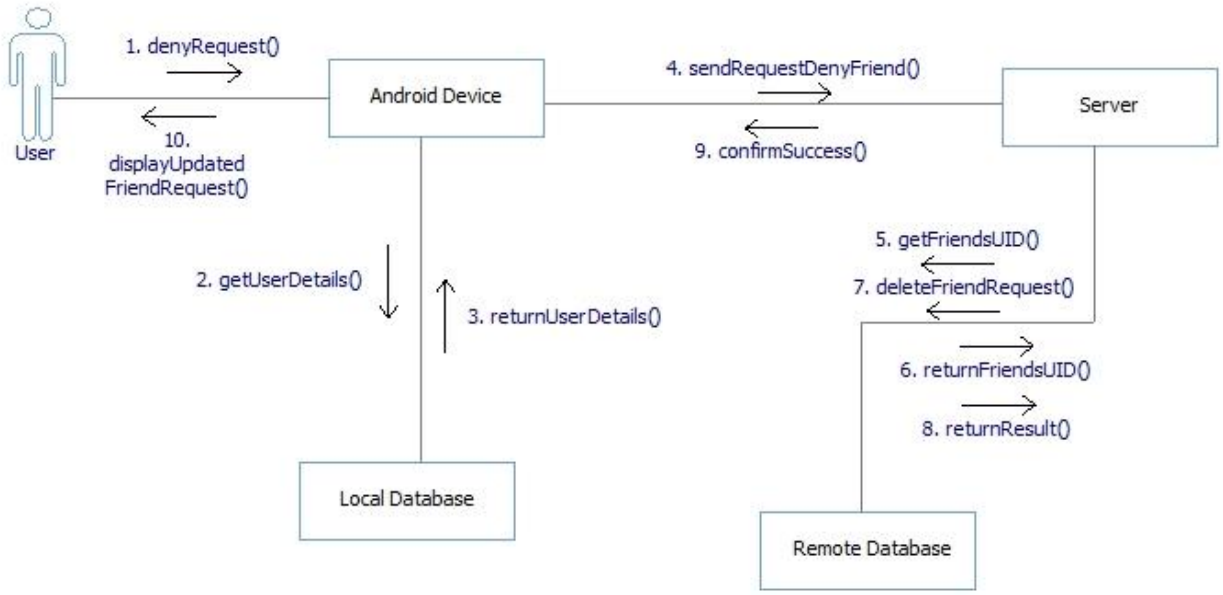
4.1.2.4 Change Password



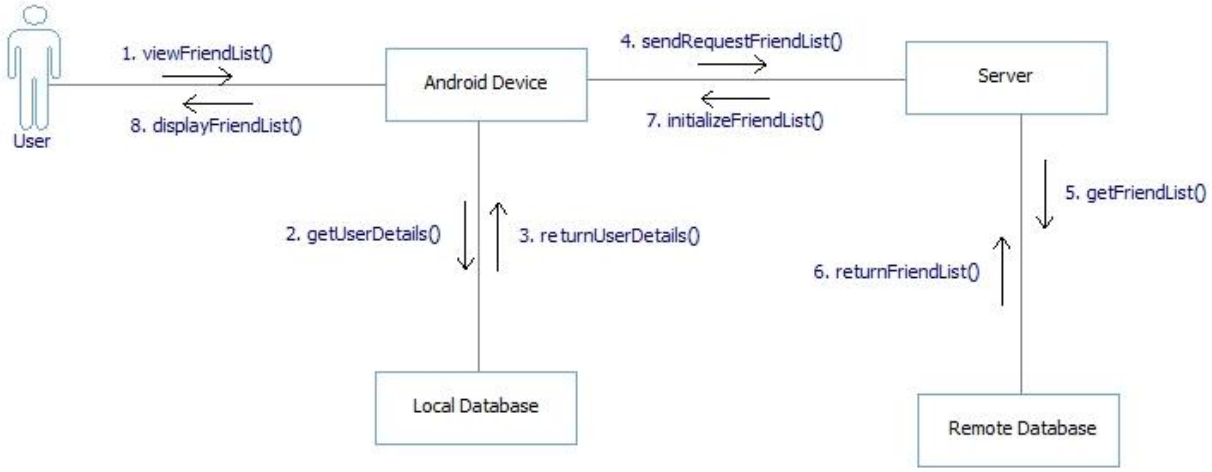
4.1.2.5 Collect Item



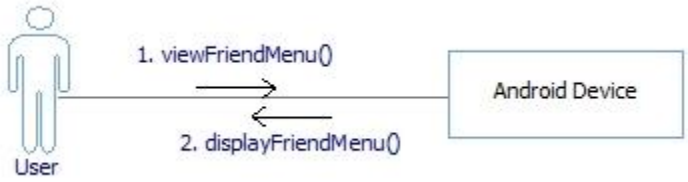
4.1.2.6 Deny Friend Request



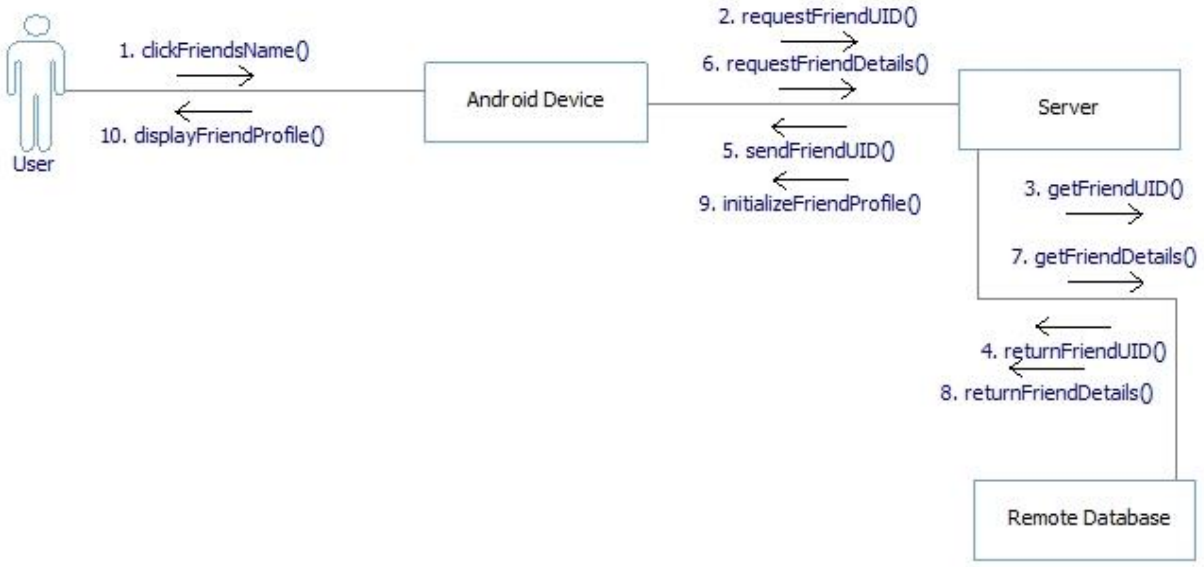
4.1.2.7 View Friend List



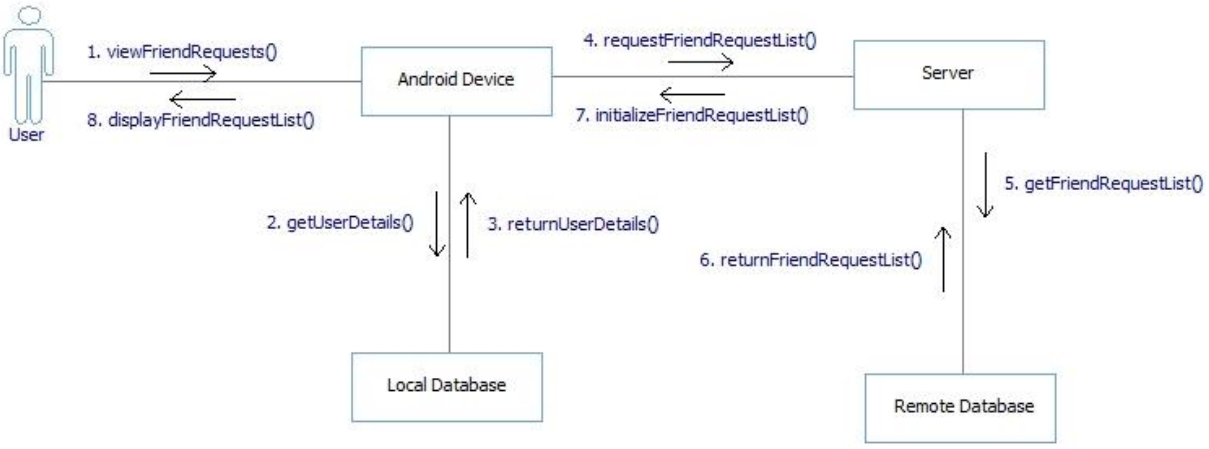
4.1.2.8 View Friend Menu



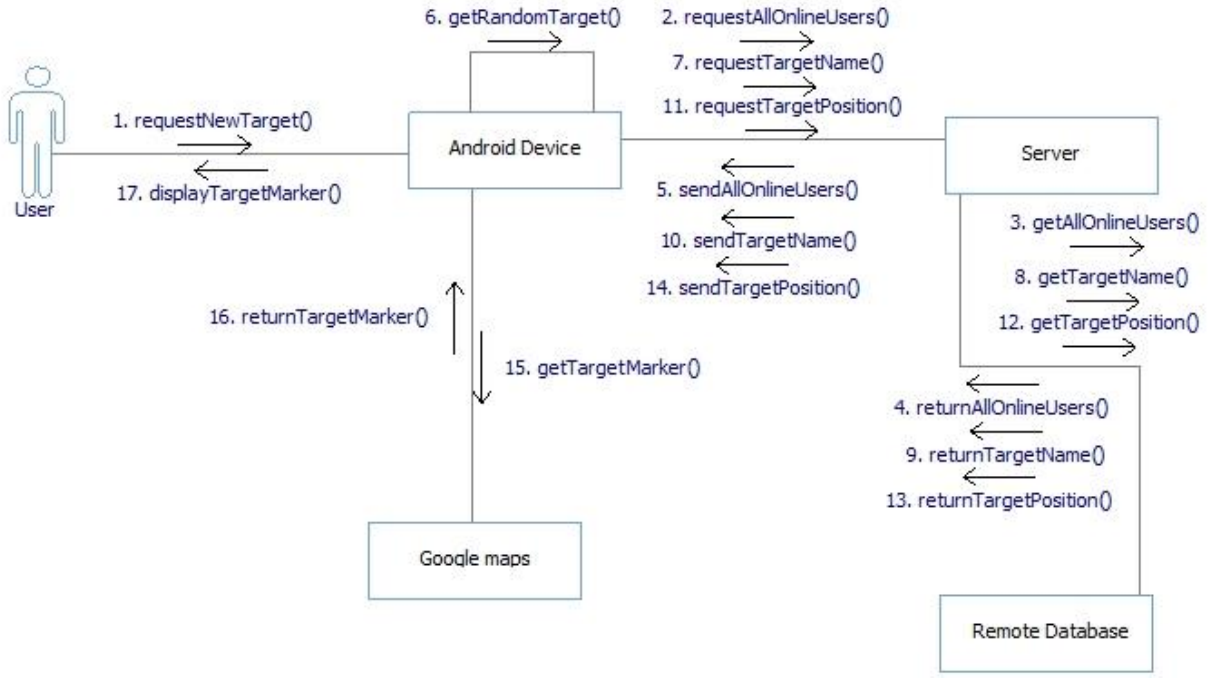
4.1.2.9 View Friend Profile



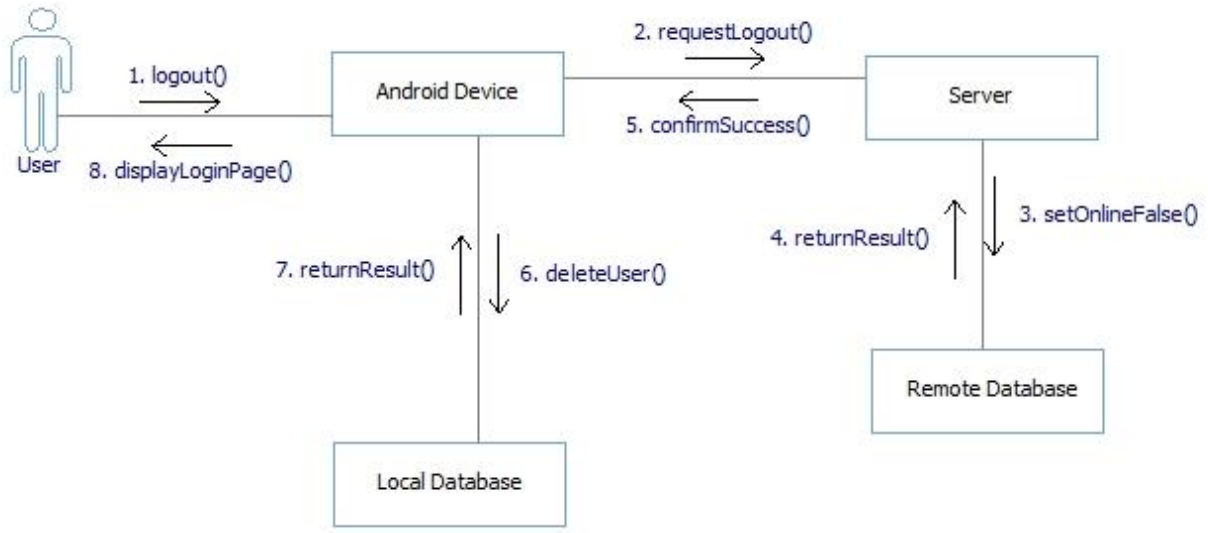
4.1.2.10 View Friend Request



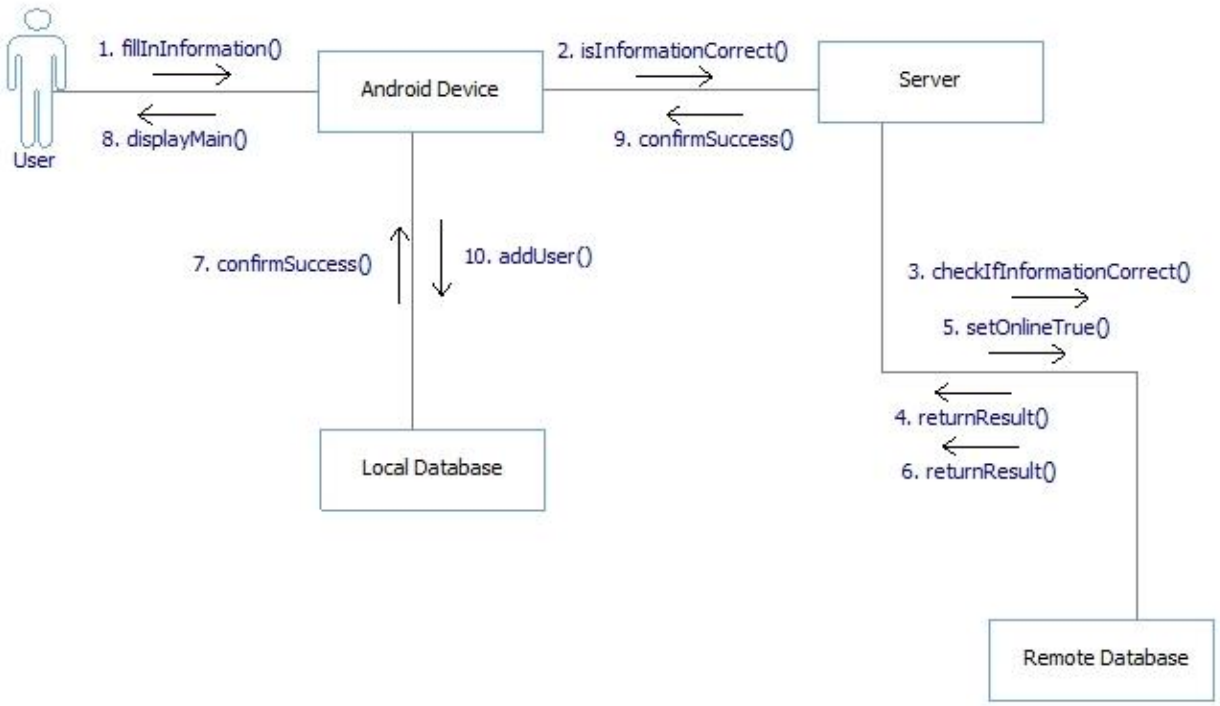
4.1.2.11 Get Target



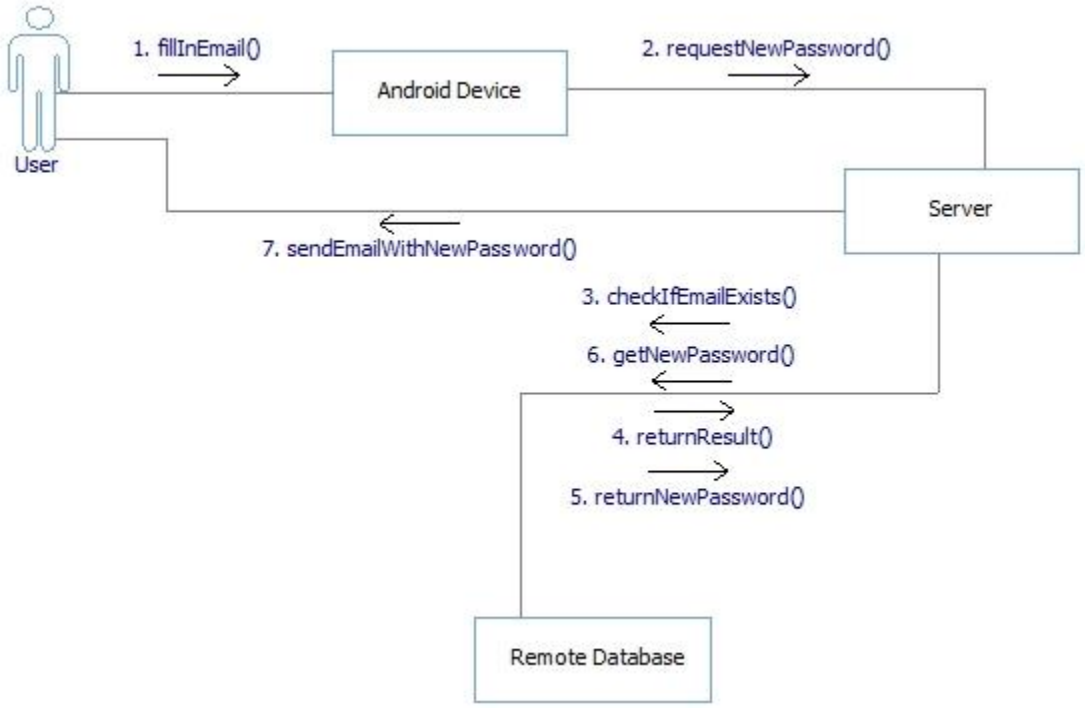
4.1.2.12 Log Out



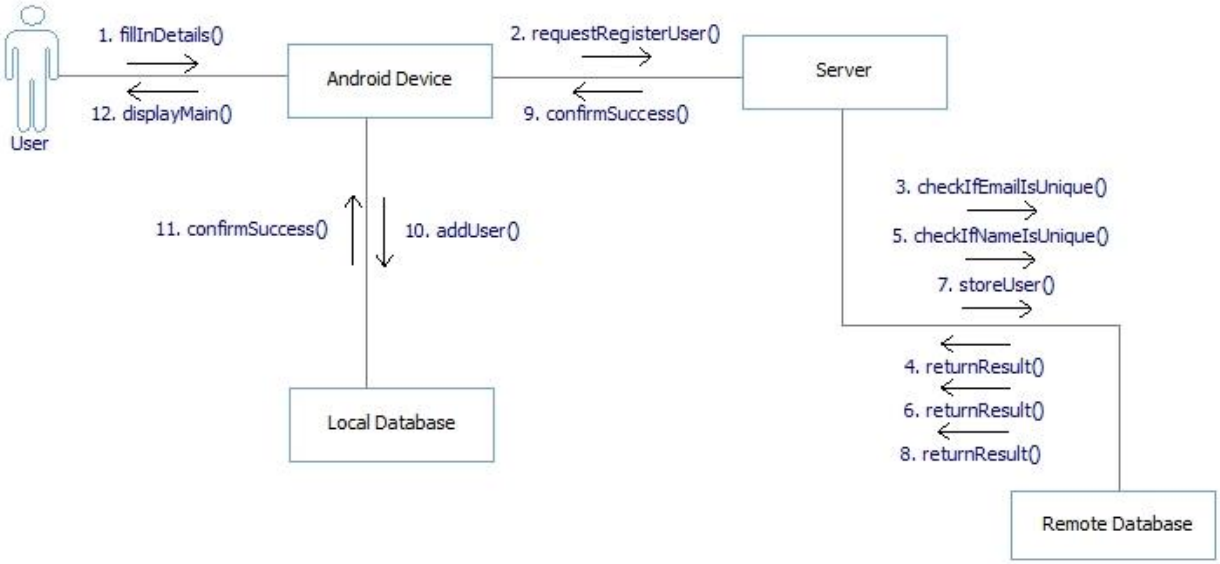
4.1.2.13 Log In


User
1. fillInInformation()
8. displayMain()
Android Device
2. isInformationCorrect()
9. confirmSuccess()
Server
7. confirmSuccess()
10. addUser()
3. checkIfInformationCorrect()
5. setOnlineTrue()
4. returnResult()
6. returnResult()
Local Database
Remote Database

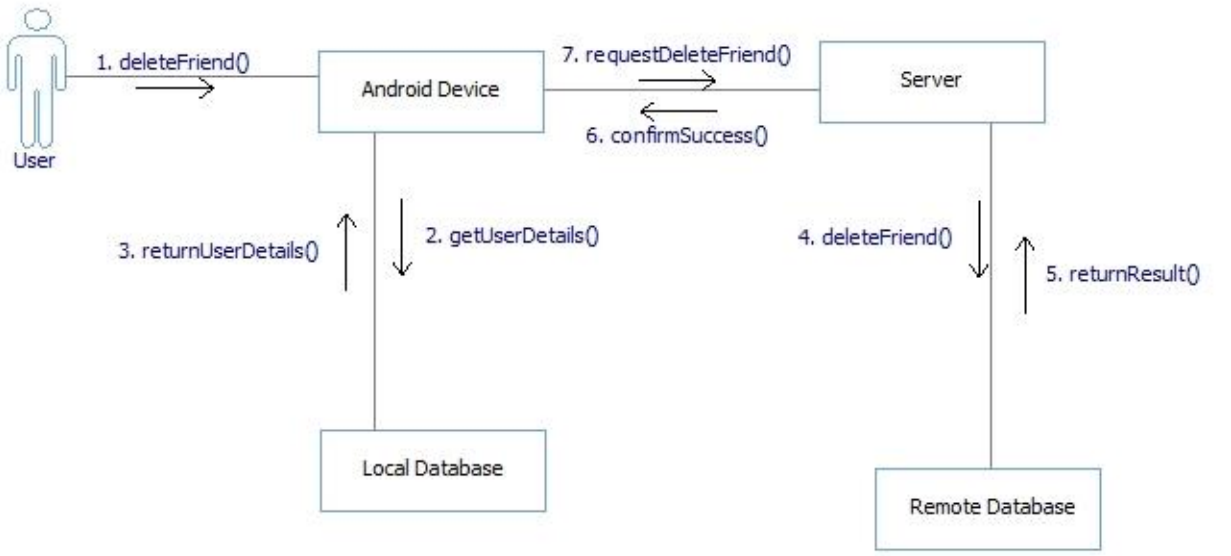
4.1.2.14 Set New Password



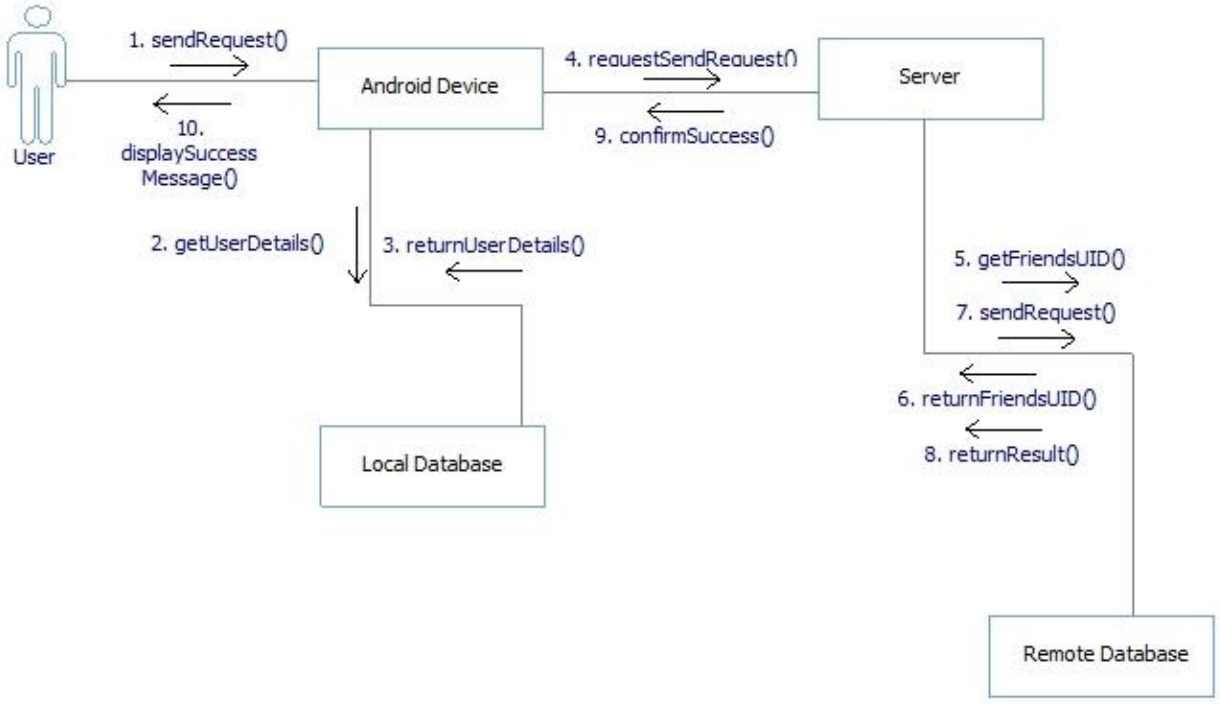
4.1.2.15 Register



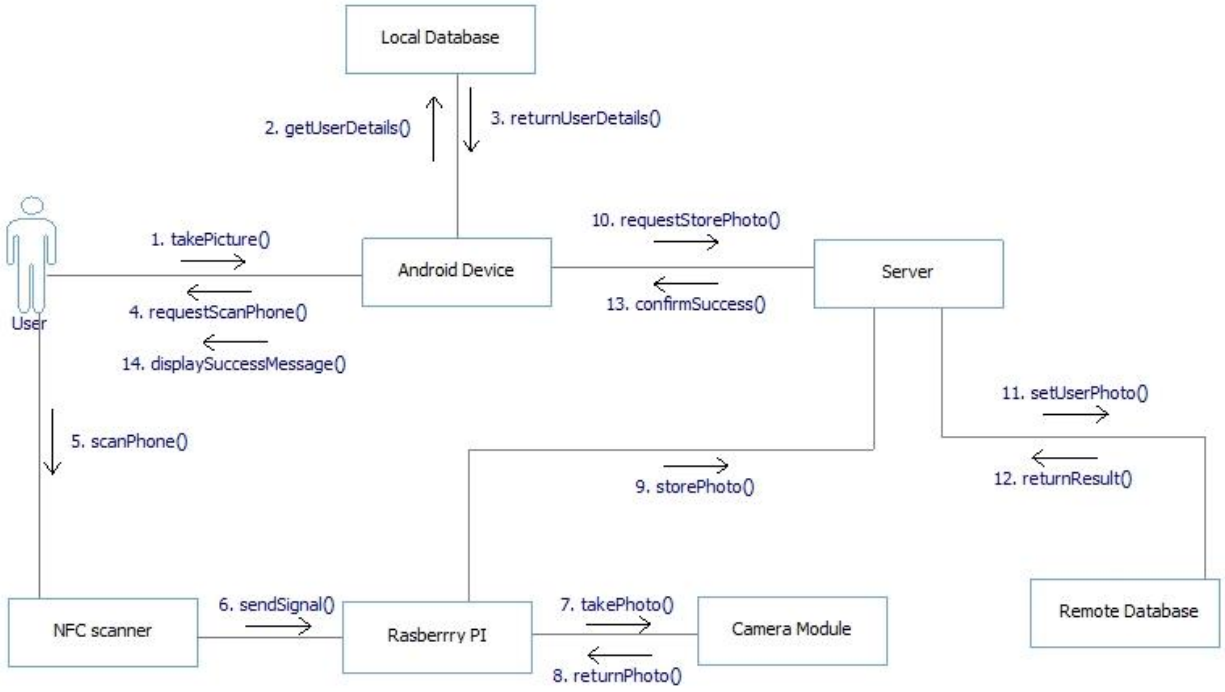
4.1.2.16 Remove Friend



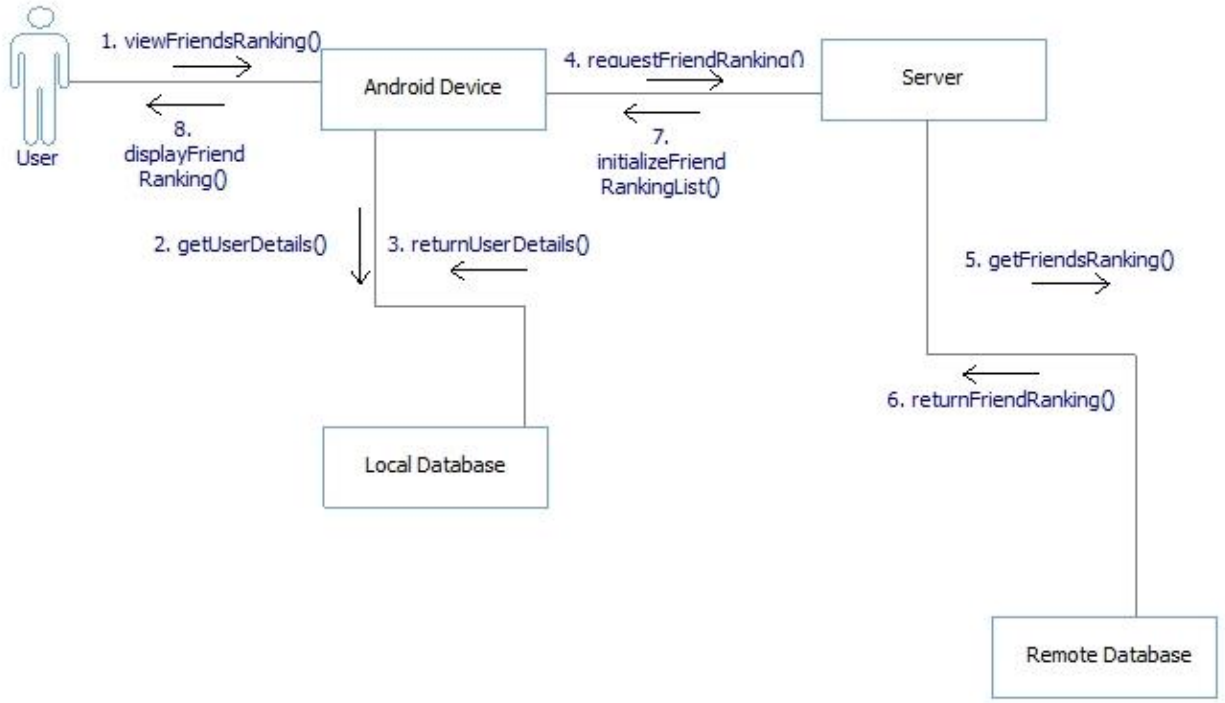
4.1.2.17 Send Friend Request



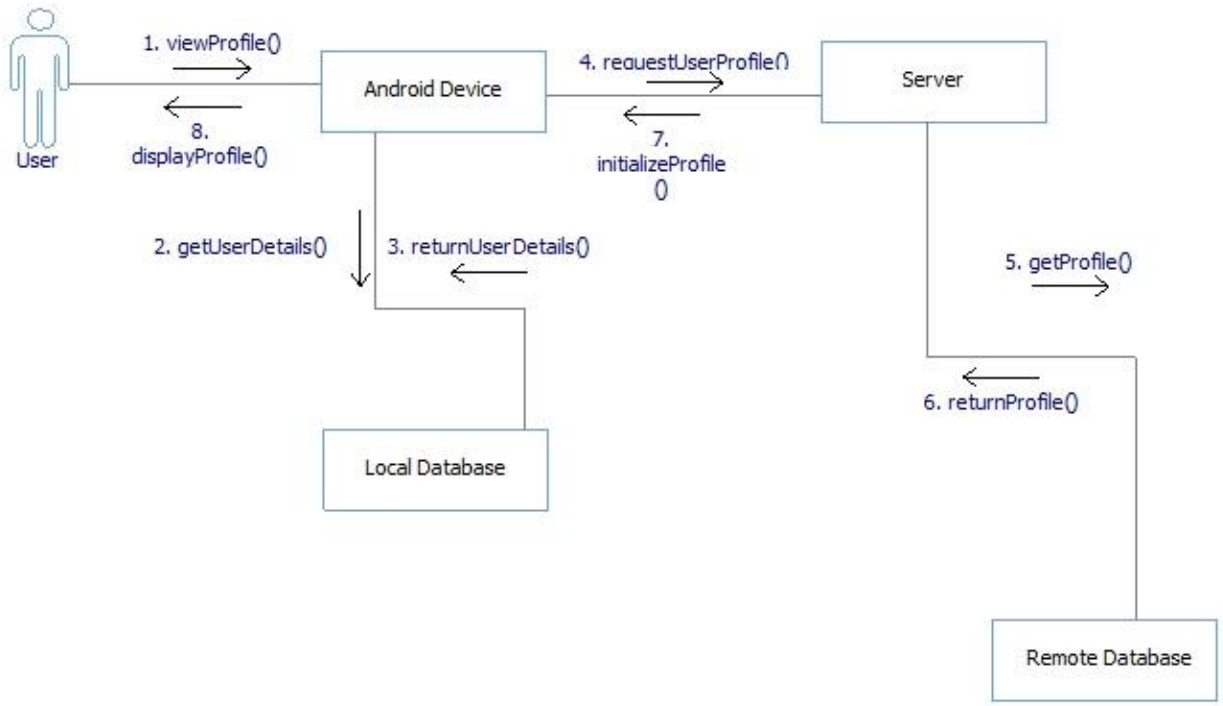
4.1.2.18 Take Photo



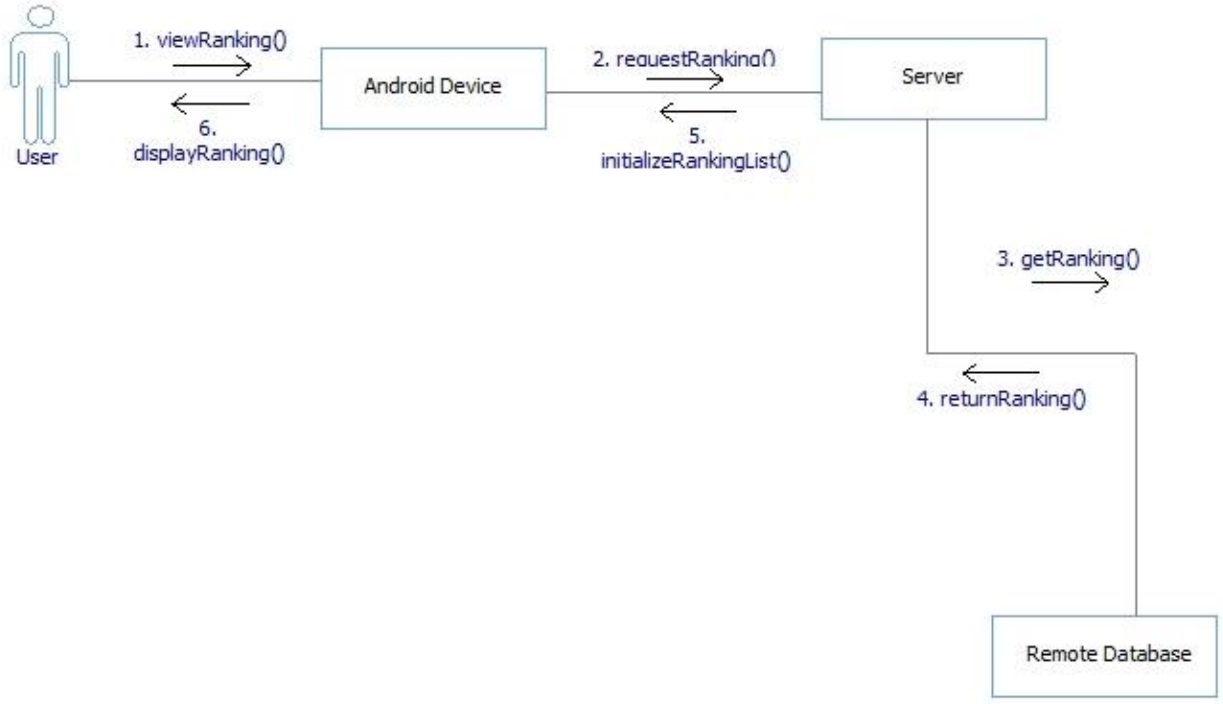
4.1.2.19 View Friend Ranking



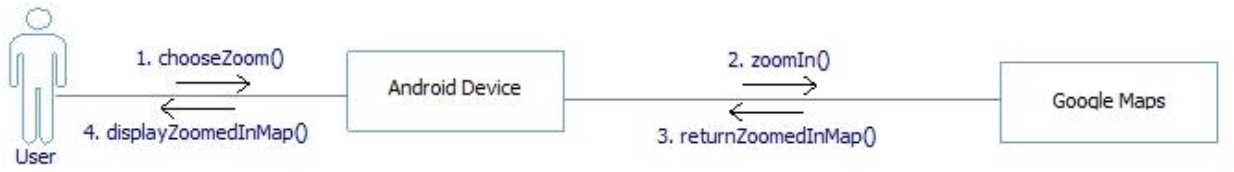
4.1.2.20 View Profile



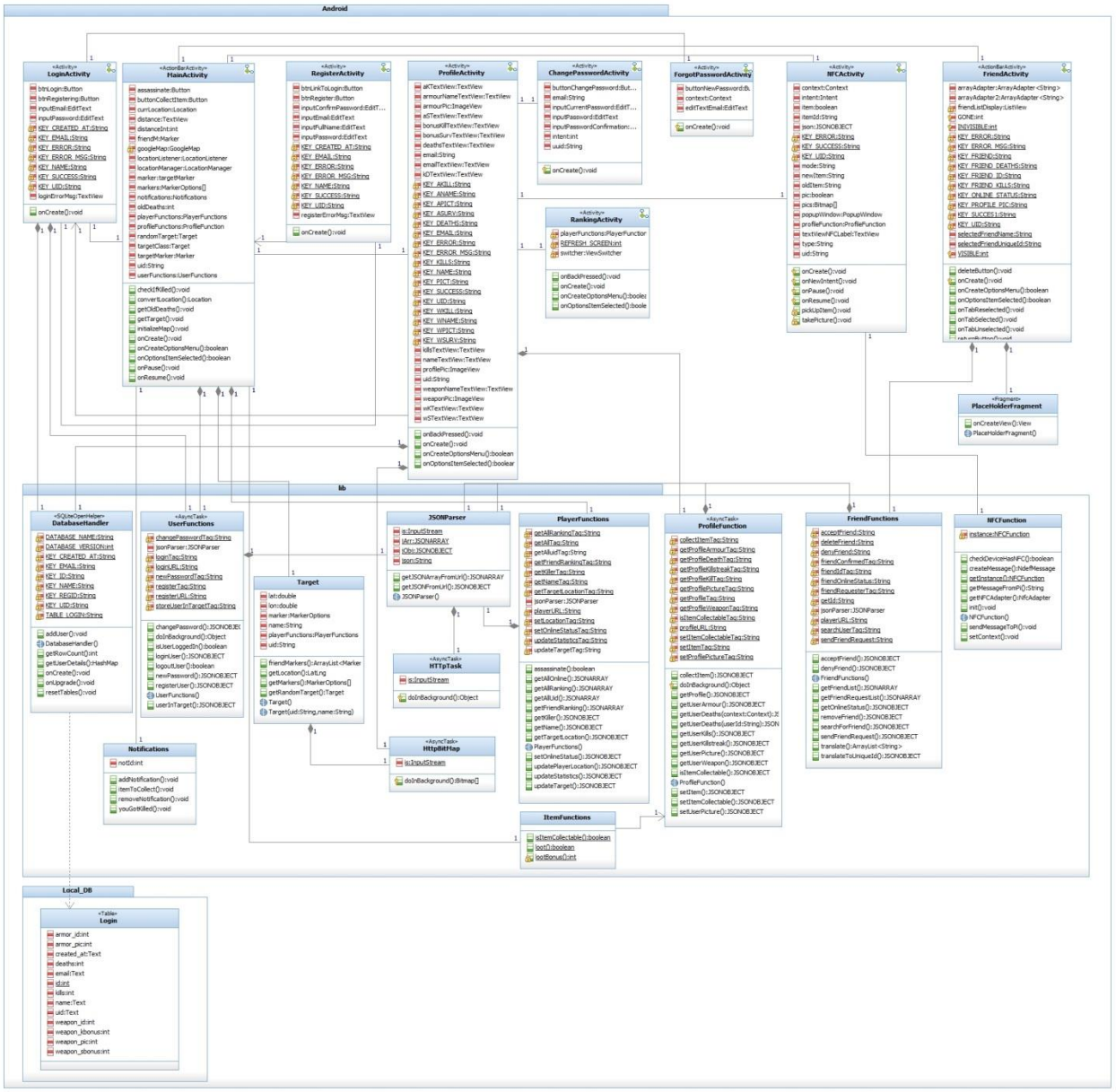
4.1.2.21 View Ranking



4.1.2.22 Zoom In Map



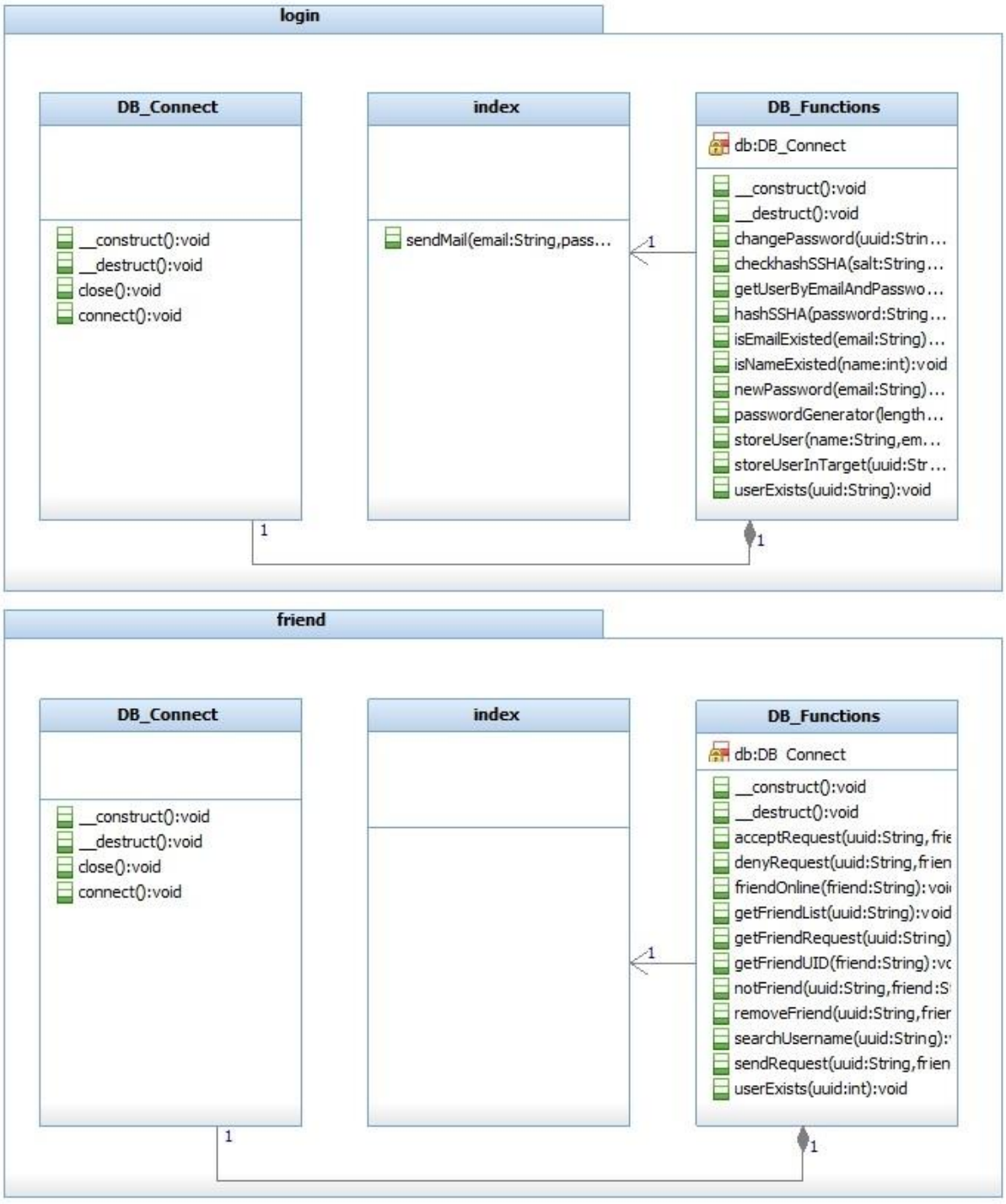
4.1.3 Android Application Class Diagram



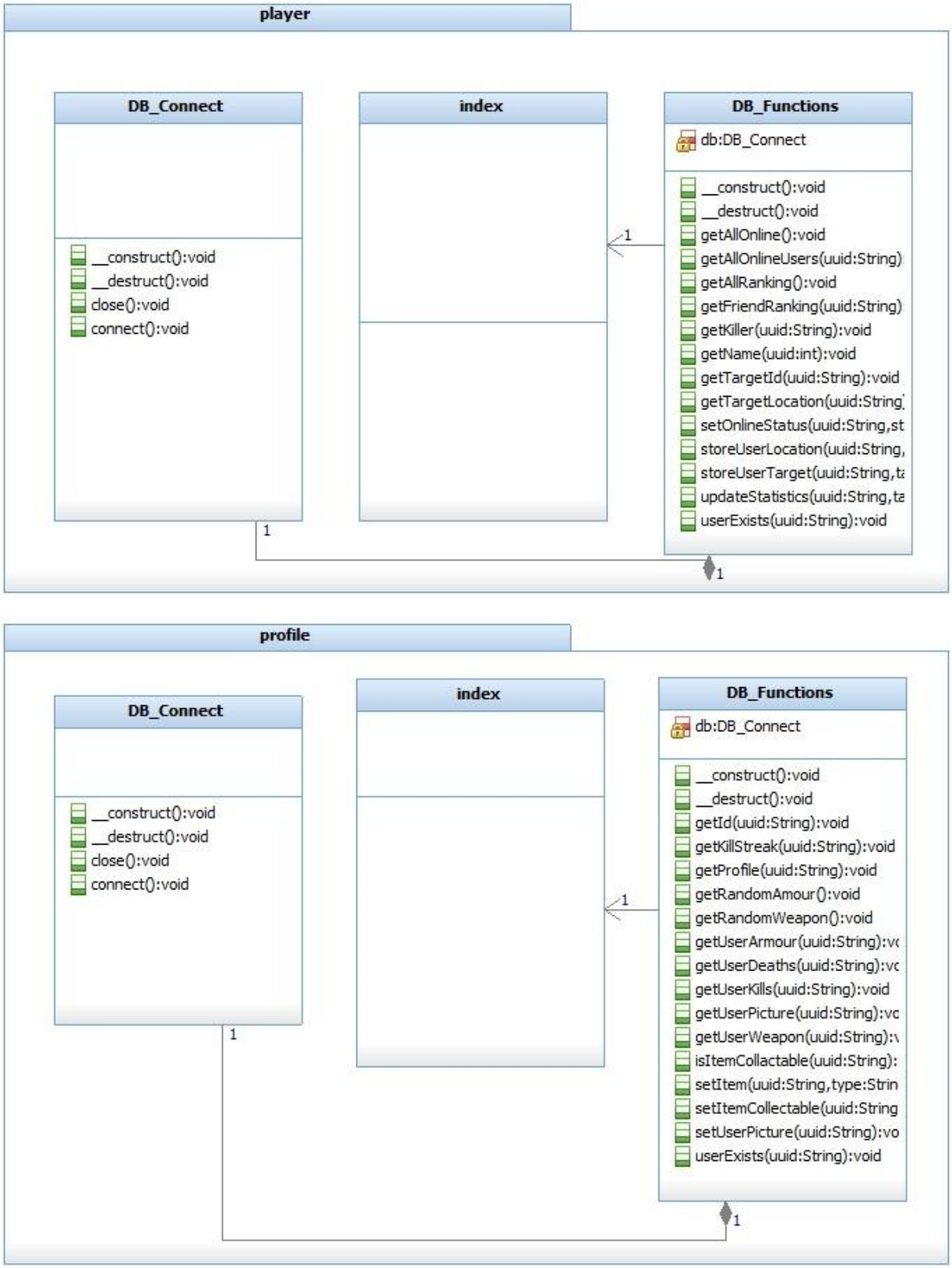
4.1.3 Can be viewed separately in ClassDiagram_Android.jpg

4.1.4 HTTP Server Class Diagrams

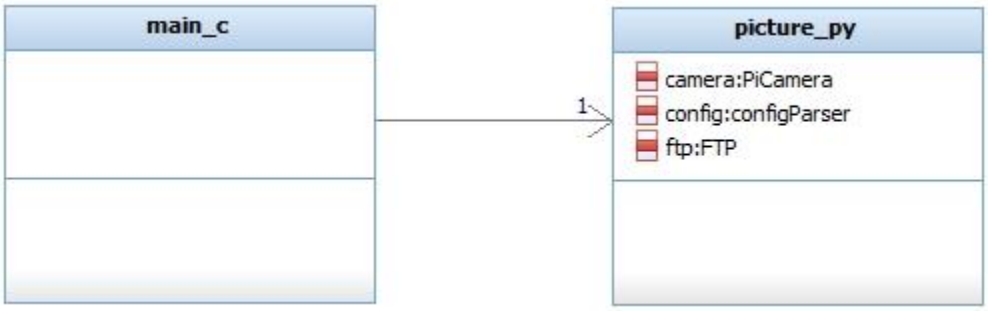
4.1.4.1 HTTP Server Class Diagram 1



4.1.4.2 HTTP Server Class Diagram 2



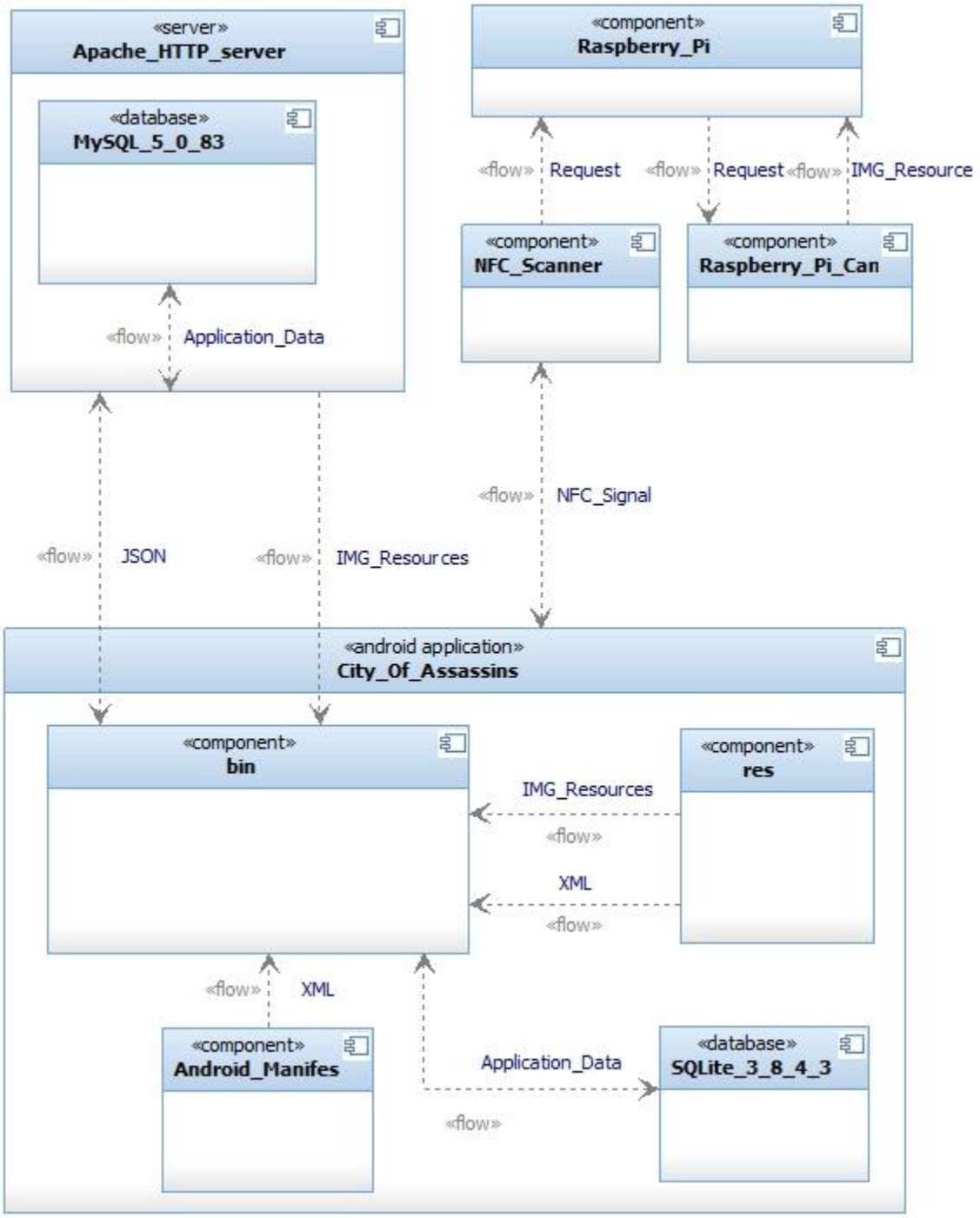
4.1.5 Hub Class Diagram



4.2 Development View

The development view illustrates, in the form of a component diagram, what and how the different system components communicate with each other and in what order.

4.2.1 Component Diagram

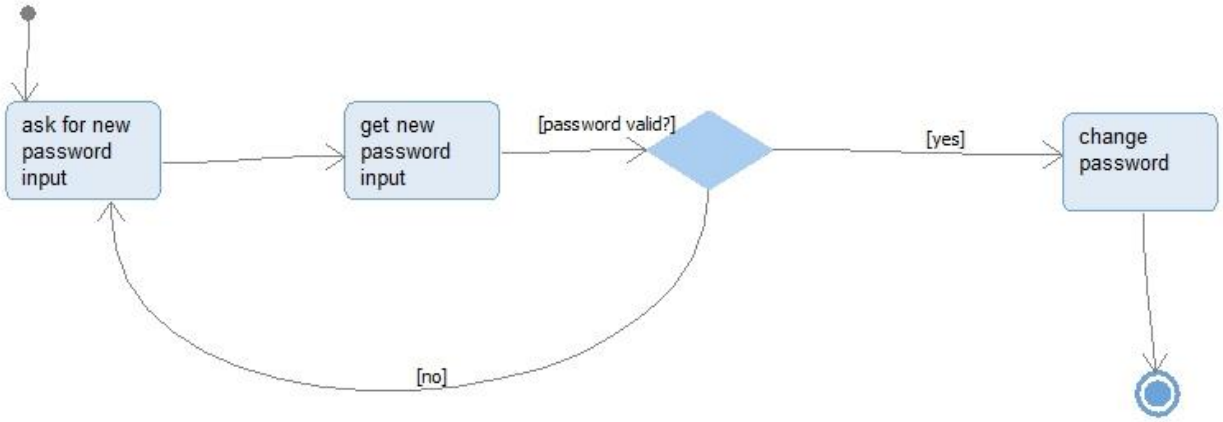


4.3 Process View

The process view presents the dynamics of the system after user input. It's illustrated in the form of different Activity Diagrams for every one of the different graphical views of the user.

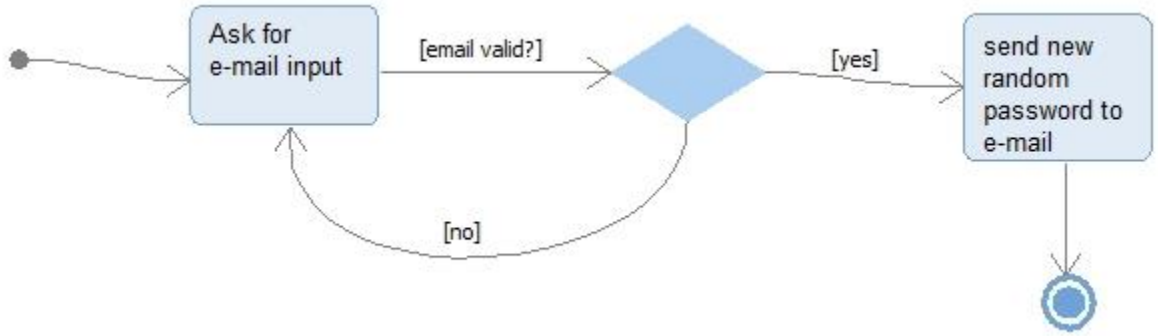
4.3.1 Activity Diagrams

4.3.1.1 ChangePasswordActivity Activity Diagram



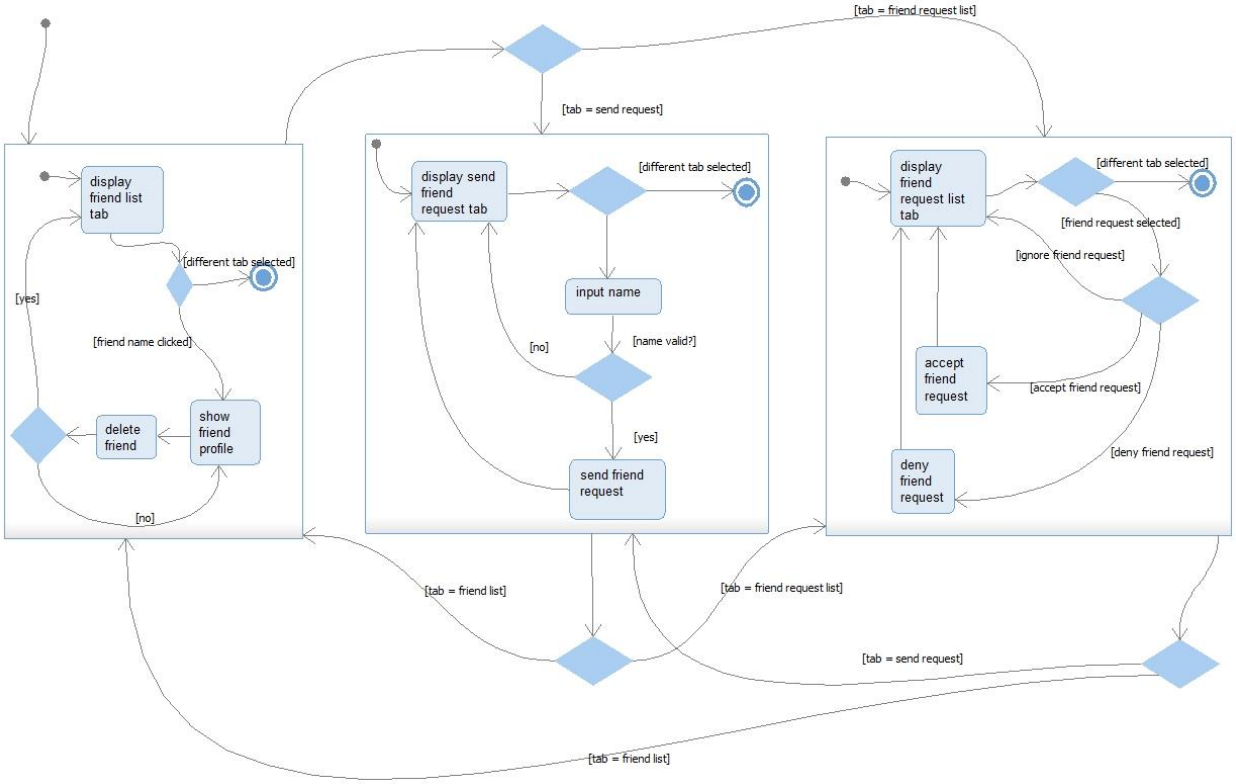
This activity diagram covers the process of user changing his password. It begins with the user choosing to change his password. He is asked for a new password input. When the user inputs a new password, the system checks if it is valid or not. If the new password is invalid, user is asked to input it again. Else, if it is valid, the system changes it and the diagram ends.

4.3.1.2 ForgotPasswordActivity Activity Diagram



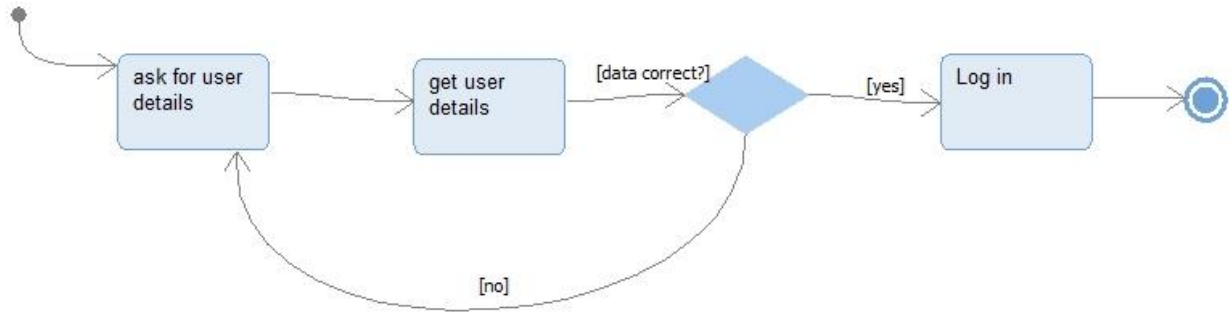
This activity diagram covers the process of user requesting a new password in case he forgets it. The activity diagram begins by system requesting for user's e-mail input. It checks if the input is valid. If not, it asks for e-mail input again. Otherwise, it sends a new randomized password to the provided e-mail and the diagram ends.

4.3.1.3 FriendActivity Activity Diagram



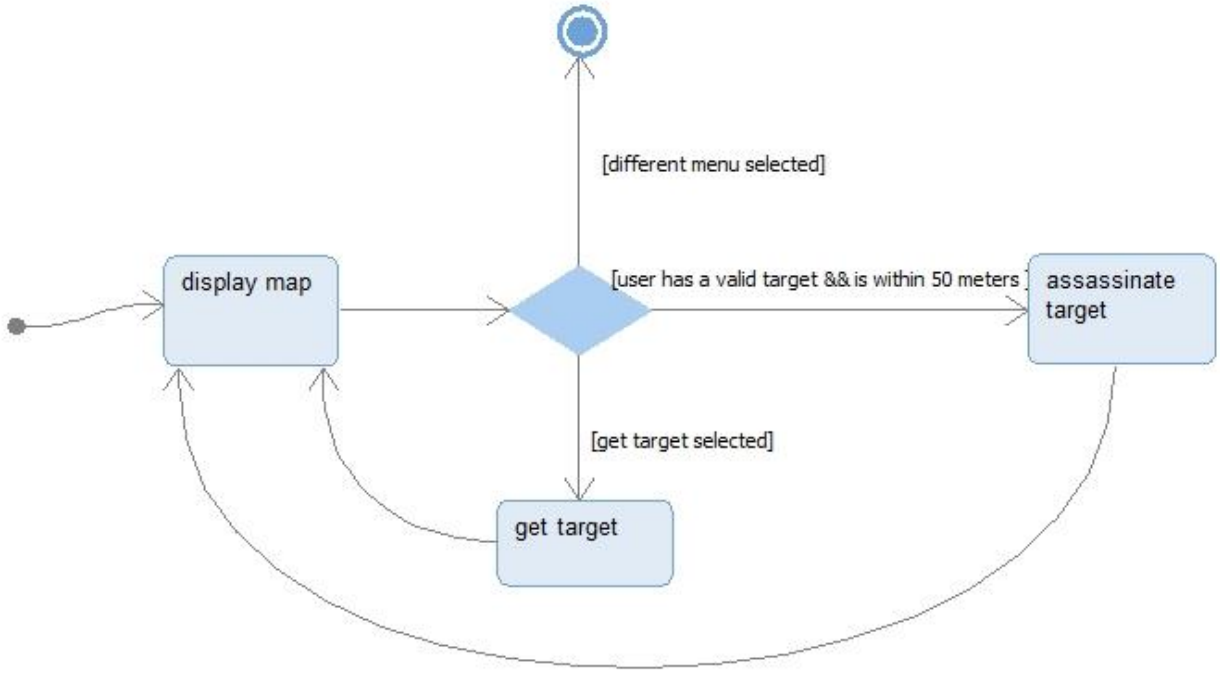
This activity diagram covers the process of user navigating between the three tabs provided to him in the friends menu, as well as handling features offered in those tabs. The activity diagram starts by showing the user his friend list. The user can select a friend's profile, and if he does that, he can choose to delete his friend (which requires a confirmation). The user can also choose to select two other tabs as well. If he chooses to send a friend request, the user will be displayed a send friend request tab, from where he can input a name of his choice. The system checks if the name is valid or not, and if it is valid, it send the friend request to the specified person. If user chooses to go to the friend request list tab instead, he will be displayed a list of people who want to be his friends. If the user selects a friend request, the system will ask if he wants to deny, accept or ignore the request for now, and will take appropriate actions.

4.3.1.4 LoginActivity Activity Diagram



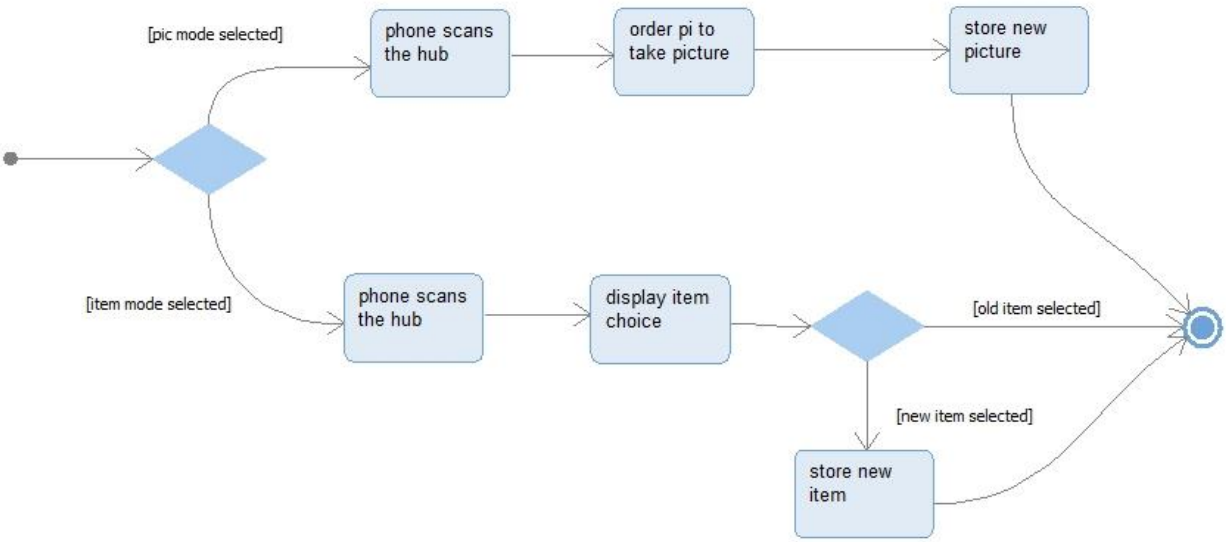
This activity diagram covers the process of user logging in into the game. The activity diagram begins with the system asking for user details. Once it gets user input, it checks if the data is valid or not. User is logged in if the data is valid, or asked to enter the details again if it is not.

4.3.1.5 MainActivity Activity Diagram



This activity diagram covers the process of showing the user the game map as well as acquiring and assassinating targets. Activity diagram begins with user being displayed the map of the game. From there user can choose to get a new target, or assassinate a target if he has a valid one already and is within a specified distance of his target. The activity diagram exits when the user chooses a different menu.

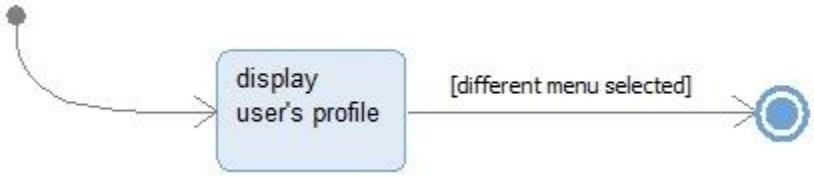
4.3.1.6 NFCActivity Activity Diagram



This activity diagram covers the process of user retrieving an item from the hub or taking a picture using the hub. When the activity diagram starts, user has to choose between selecting a picture mode or item mode. In picture mode, the user scans the hub with his phone, which then orders the Raspberry Pi to take a picture. Then it stores the new picture in the database and the activity diagram ends.

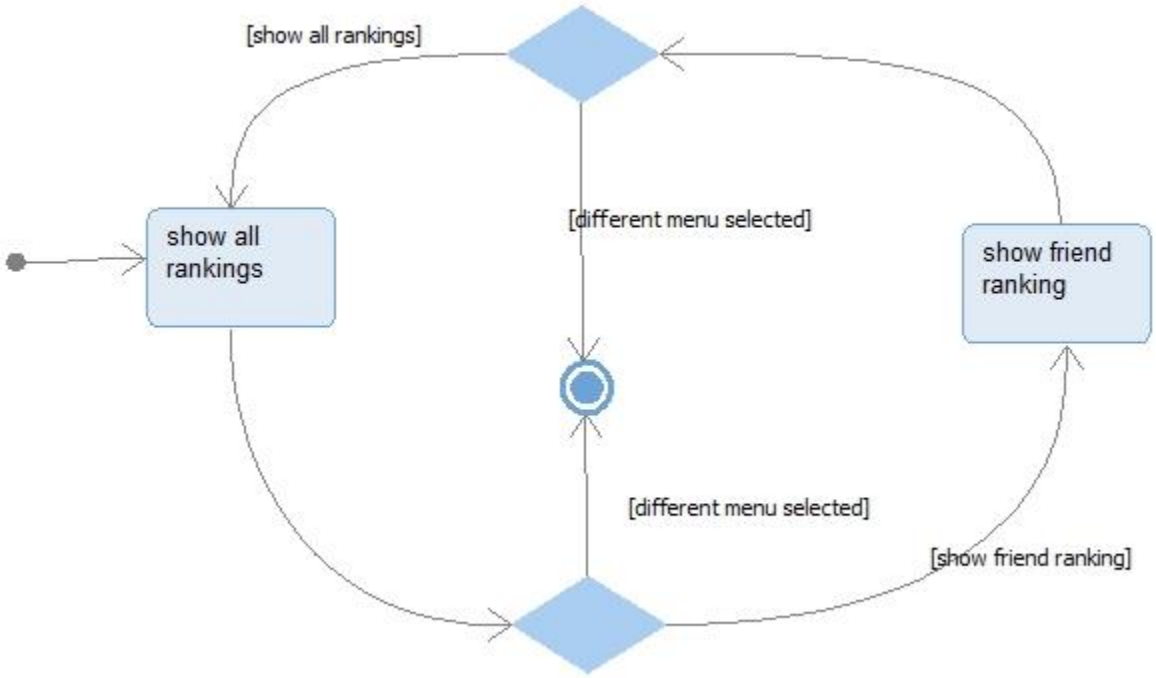
If the user selects item mode instead, he uses the phone to scan the hub again, and the system displays an item menu for him. The user then can choose to keep old items, or select a new one (which is then stored in the database). The activity diagram ends then.

4.3.1.7 ProfileActivity Activity Diagram



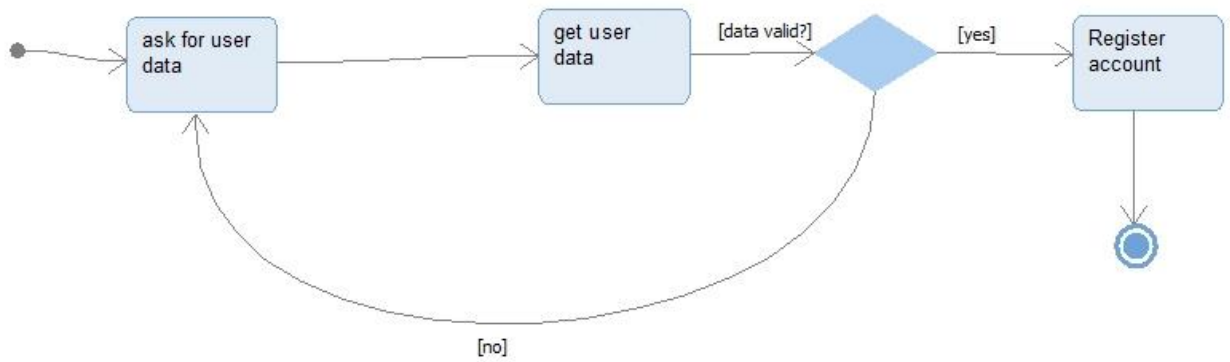
This activity diagram visualizes the process of user viewing his profile. Activity diagram begins when user chooses to view his profile. The system displays the profile, until a different menu is selected, at which point the activity diagram ends.

4.3.1.8 RankingActivity Activity Diagram



This activity diagram covers the process of user viewing different ranking. The activity diagram starts with user choosing to view rankings. He is displayed a ranking of all players, from where he can choose to view only friend ranking. When the user is shown his friend ranking, he can choose to view the ranking of all players. While the user is shown either his friend ranking or all player ranking, he can choose to view a different menu, at which point the activity diagram exits.

4.3.1.9 RegisterActivity Activity Diagram



This activity diagram covers the process of user creating a new account. The activity starts by asking user for data. Once it gets user data, it checks if it is valid or not. If it is not valid, the user is asked to input the data again. Otherwise, the account is registered and the activity diagram ends.

4.4 Physical View

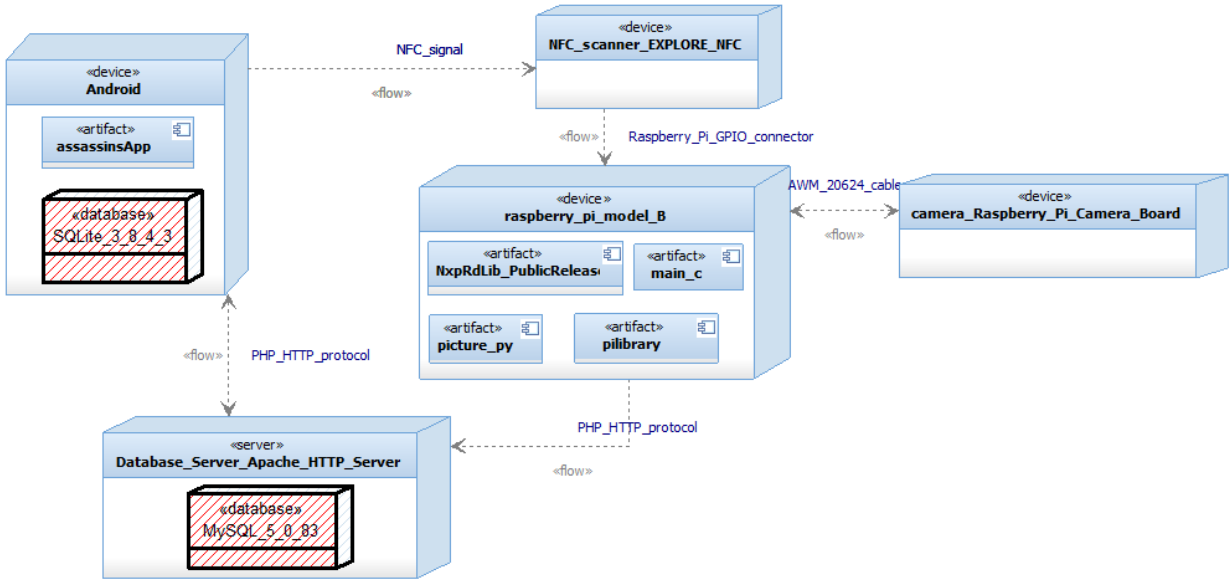
4.4.1 Deployment Diagram

The deployment diagram displays how physical aspects (nodes) of the developed system interact.

There are two types of nodes:

- Device Node: a physical computing resource with processing and memory services to execute, such as a typical computer or mobile phone.
- Execution Environment Node: this is a software computing resource that runs within an outer node (like a computer or a phone) and executes other executable software elements.

4.4.1 Deployment Diagram



As presented in this diagram, when the user scans his/her Android device, the Android device node uses an NFC signal as a communication path to the NFC scanner. Then the NFC scanner, using a Raspberry Pi GPIO connector, communicates to the Raspberry Pi that a signal has been received. At this point, the Raspberry Pi sends an order to the Raspberry Pi Camera Board to take a picture via an AWM 20624 cable. Raspberry Pi Camera Board takes the picture and sends it back to Raspberry Pi via the same cable. Then, the Raspberry Pi uses PHP HTTP protocol to upload the picture in an Apache HTTP database server, which stores it in a MySQL 5.0.83 database.

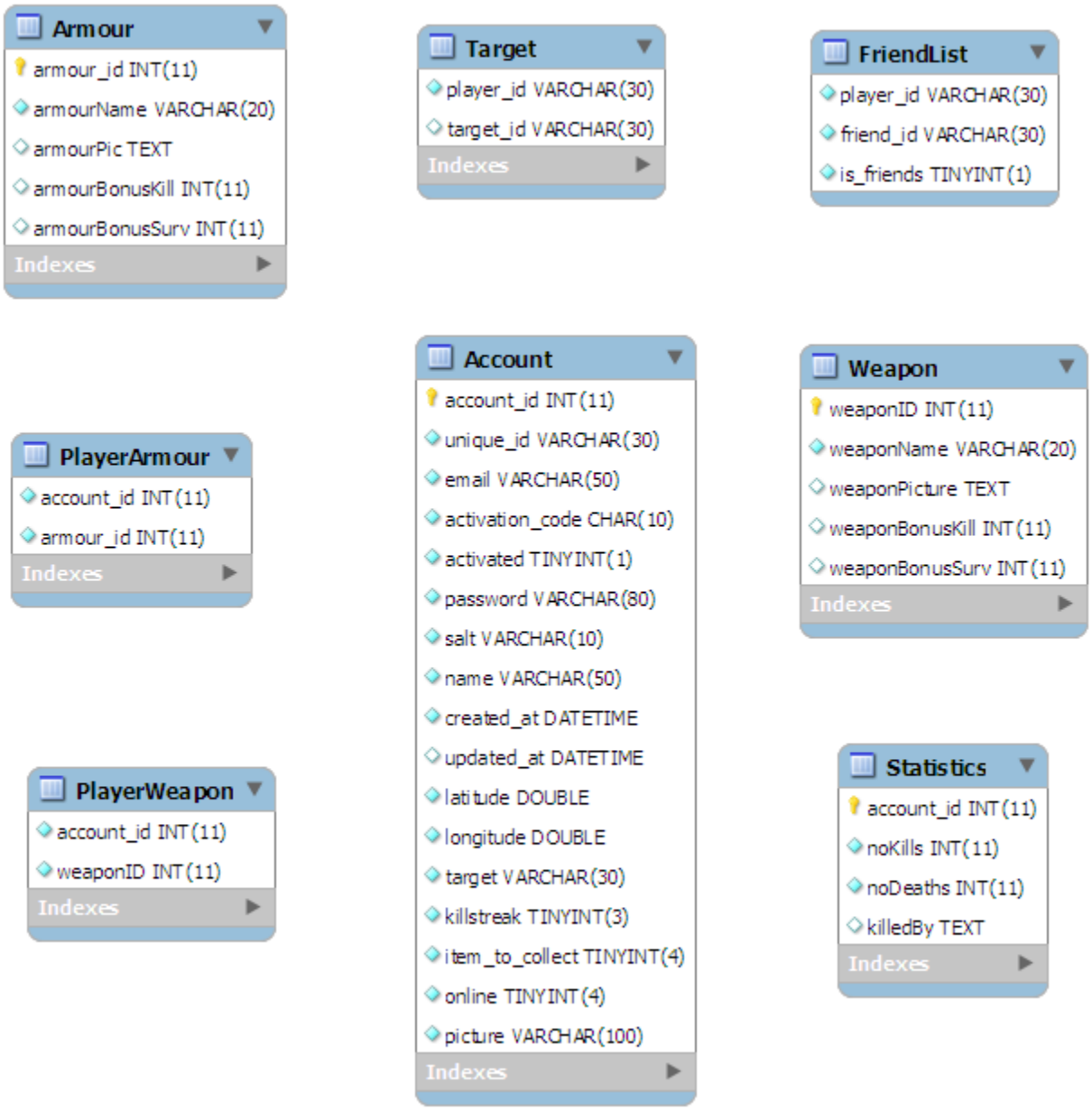
When users log into our game on their Android device, the device stores their log in information in a SQLite 3.8.4.3 database. When they actually play the game, they use the AssassinsApp artifact on their Android device. The Android device uses PHP HTTP protocol to send and retrieve information from the Apache HTTP server.

5 System Design

5.1 Database Design

An E/R Diagram displaying the different tables and the non-existing relations of the Server MySQL Database.

5.1 Server MySQL Database E/R Diagram



5.2 User Interface Design

5.2.1 Introduction

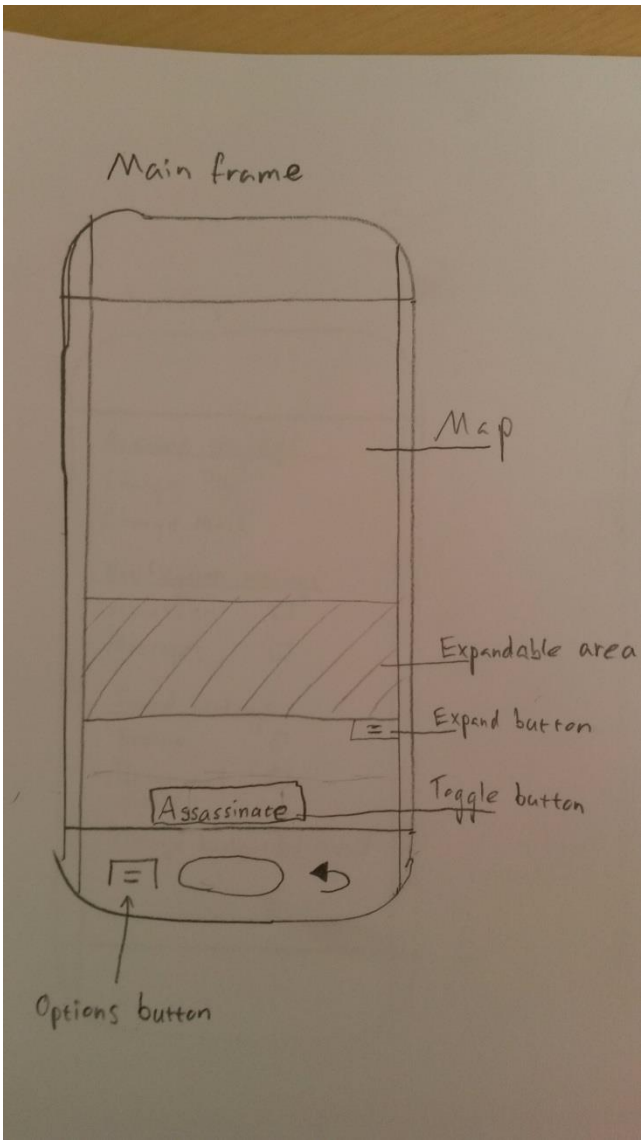
Throughout the interface design we have always tried to think through the user's perspective; how does it feel to have a button here? does any vital information disappear if something is displayed on this section of the screen? etc. All to make the user feel as much affection to the base design as possible. This part of the document will present more in detail how we were thinking when sketching up the different frames.

5.2.2 Main Frame

We knew from the start that this would be the most crucial design to get right since this is where the player will spend most of his time while playing. We started off by discussing how the user would assassinate his target and the top two ideas were either to make the user press on his targets marker and then an assassination button or to simply have a separate button for assassination. To come up with a decision for this we started drawing on a whiteboard-app on our phones to get a feeling of how the user would feel when assassinating in the different ways. Since the separate button required less clicking and let the user keep a good overview of the map we choose that idea.

After that we started to think about how the user would be able to see an event flow and a message board. As we want the user to be able to see as much of the map as possible at all times, we came up with the conclusion that an area which could be expandable would be the best way to go. This would let the user see more of recent events if wanted.

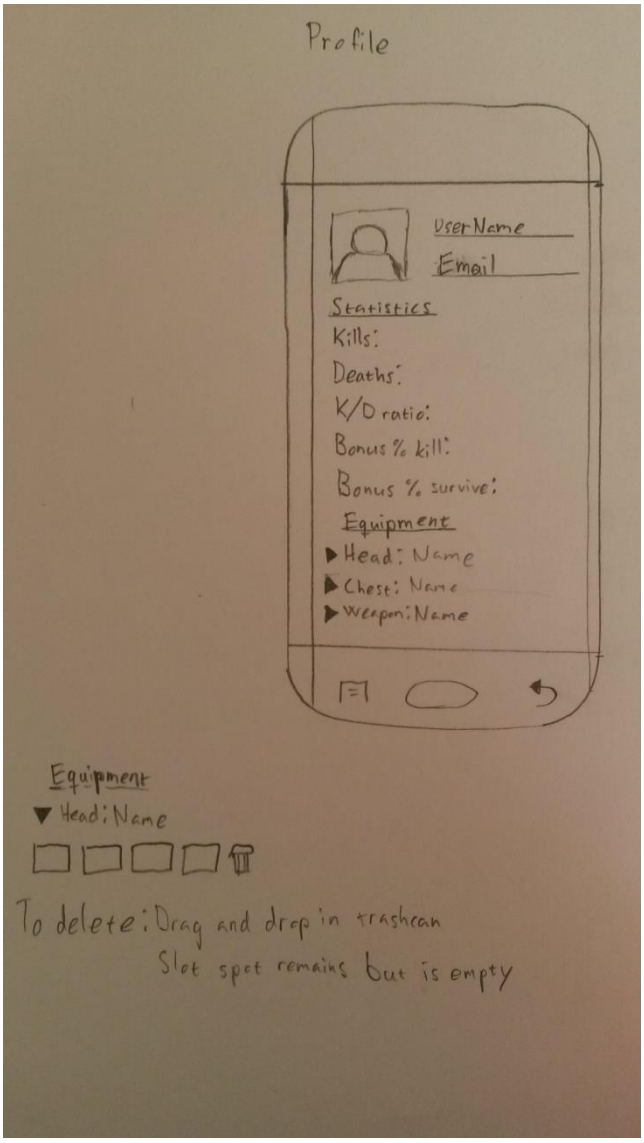
5.2.2 Main Frame Sketch



5.2.3 Profile Frame

For the profile we wanted the user to be able to get a good overview of his/her statistics, the equipment worn as well as the standard things: picture, name and email. Most of it was very straight forward but when it came down to how we wanted the equipment to be displayed we ended up with a lot of ideas. One idea we had, for example, was to display a chest and let it have its name displayed next to the type. The user would then have to press on an arrow to display a drop-down menu with all the items of that type. Another idea was to restrict the capacity of the user's items into one item per type and display them directly next to their bonuses and name.

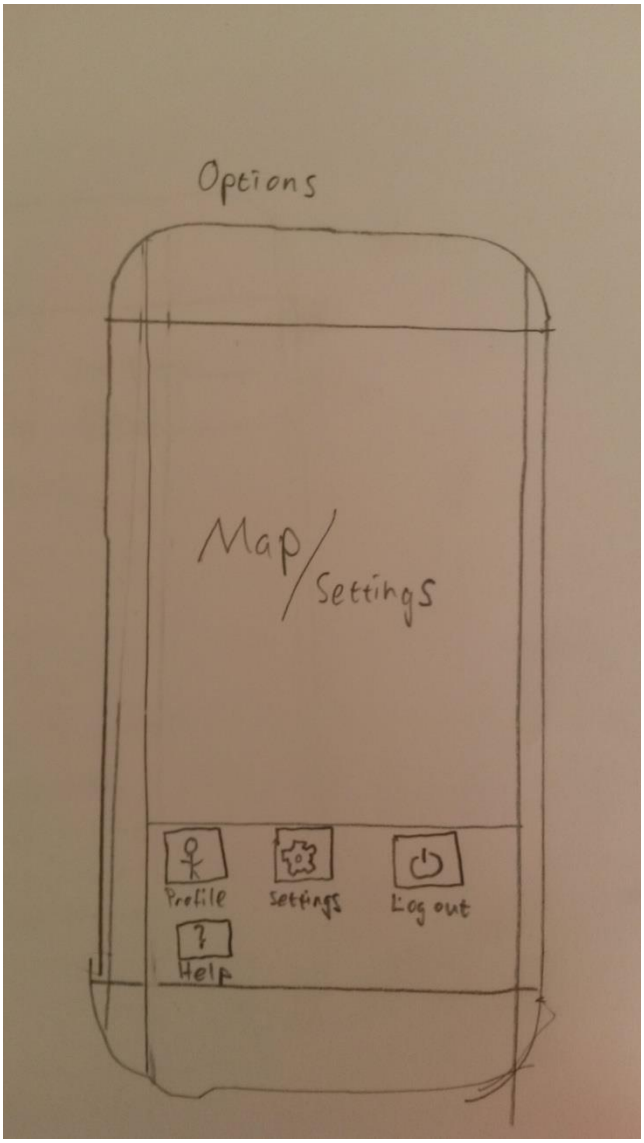
5.2.3 Profile Frame Sketch



5.2.4 Options

When we decided on how we wanted the options menu to look we took inspiration from other apps, e.g. Facebook, with appropriate icons for the various functions, simple yet stylish and as always, a good overview of the frame in the background.

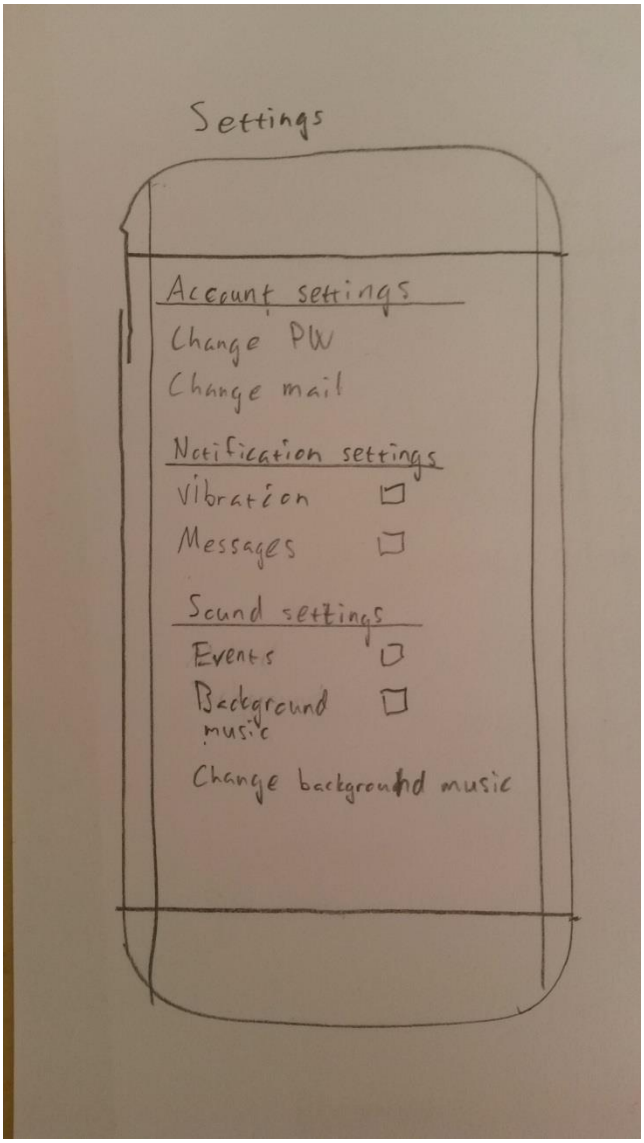
5.2.4 Option Frame Sketch



5.2.5 Settings Frame

The setting frames design was based on quite a few other apps. We wanted the user to have an option for how the notifications and the sound settings should work. Furthermore, we wanted our users to have a possibility to change their password and mail, all in one place.

5.2.5 Settings Frame Sketch



5.2.6 Friend Frame

When we decided to include friends we had very big ideas of what we wanted. There should be three tabs which would display list of friends, the option to send a friend request and a list of friend request respectively. The plan was to be able to to add friends for the purpose of chatting with them, so what we wanted was a friend list that you could interact with. It became evident quite quickly though that just adding friends was more complex than first thought so we decided to dismiss the chat function and use the friend list to display which friends where online. This was going to be done by different colors for online and offline but the result was very unattractive so a third idea was implemented instead. We used the functions for Profile Frame to create a small friend profile that would show their basic information – picture, name, statistics and online status. The friend profile also gives the user the option to remove a friend.

5.2.7 Reflection

When looking back on these frames it becomes clear that a lot have changed throughout the project. Entire frames have been scrapped, some plans have been moved into the future and some have been implemented. We have even added new frames to give the app a higher usability to our users.

When taking a look on the main frame today it is quite similar to what was planned in the beginning, the button got placed as intended and we still have a great overview of the map. However, a lot happened along the way, the event flow can't be seen anywhere within the app, but is something intended to be added to the app in the future so that users can see everything that happens to the other players in their area.

For the profile frame a lot have changed, the username and email are moved to the top of the frame, the statistics are in the middle of the screen and in the background of the top-section you can see the profile picture. We cancelled the idea of a drop-down menu, restricting the users to only have two item types. This gives our users an incentive to play the game in order to gain more items since they can't store up on items.

The options menu is in a simpler shape than firstly intended but is also planned to be revised in the future to be more appealing to our users. When it comes to the settings menu it doesn't exist for now since we currently have no sounds within the app and the email we wanted to keep static during the beginning of the app. Change password have been moved to the profile frame since we still want our users to have that option.

6 Operational Scenarios

Use Case Scenarios are presented in the SRS together with all the Use Case information. Below is an excerpt scenario from the SRS:

6.1 Example Scenario

John is playing the game and he has acquired a target. He approaches his target until he is within 50 meters or closer. The “Assassinate Target” button on his screen becomes enabled. John presses the button and the System does a random roll. The roll comes out in favorable to John and he succeeds in assassinating his target. He and his target are notified of the result, and John’s kill count is increased by 1, while his target’s death count is increased by 1. John is then given a new target. (Appendix J – SRS, Use Case 10. Assassinate Target)

7 System Integrity Controls

Since the product uses web server with MySQL for storing data and login credentials we needed a way to keep these safe. We decided to use a base64 encryption on the password to get a layer of security for the users. The data passed from the application is using POST method in HTTP so it will not be visible to others. All encryption is done on the server side in the PHP method.

Glossary

GUI

Graphical User Interface

HTTP

Hyper-Text Transfer Protocol

JSON

JavaScript Object Notation

NFC

Near Field Communication

PHP

Hypertext Preprocessor

SRS

Software Requirement Specification

SDD

Software Design Document

Referenced Documents

Appendix G - Quality Management Plan

Appendix J – SRS