Project Design Document

Version 1.0

17.04.2015

Prepared by: Martin Brehhov Project

Supervisor: Joseph Kehoe

Contents

Introduction

As mentioned already in specs the project description can be narrowed down to way different users can communicate through android device.

From technical point of view this project can summed as Android device sending or receiving data. Data is stored partly in Java and partly in Databases on web server.

Document overview

Document is continuation for System specifications and should guide developer how to meet project functionality - at least what to build and suggest how to build.

Scope

Document is concentrating particularly on critical part of the document. System uses external API libraries like google play API and MQTT client library.

All those are described in research manual and should be included in working system.

Use cases

Design Document overwrites more detailed those Use Cases described in Functional Specs.

Description of the Actors:

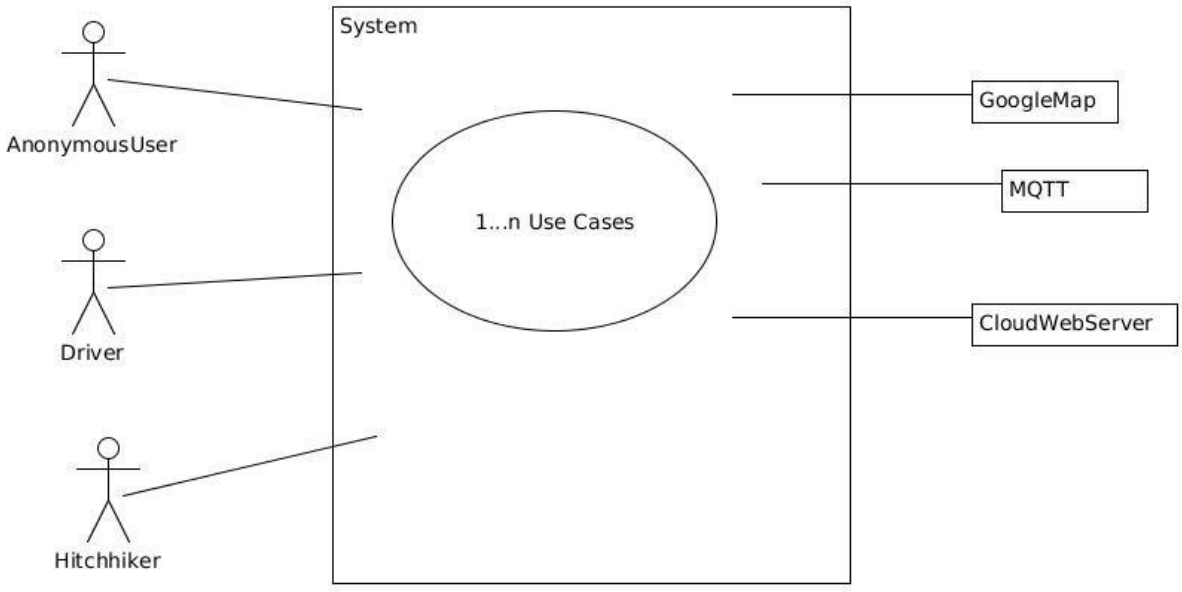
<<System>>

System represents currently developing system and how he responds to external actors.

Anonymous user is user that have a right to observe real world map, and see Hitchhikers. Hitchhikers are represented on google map as android icons without detail information for anonymous user.

Driver user is power user of the system - he has most of the rights. For Example he can observe some Hitchhikers and contact with them. Version 1 of the App are very flexible in term of the user types. So any logged in user can become Driver or Hitchhiker in any given time during runtime.

Hitchhiker user is user of the system. He specifies his location and destination. Waits until any Driver selects him and contact with him. He will be able to contact with driver upon the first Driver message.



General view of the actors. NB: External interfaces are shown on right side of the diagram.

List of Use Cases

Functional specification document suggests following use cases:

Anonymous user use cases:

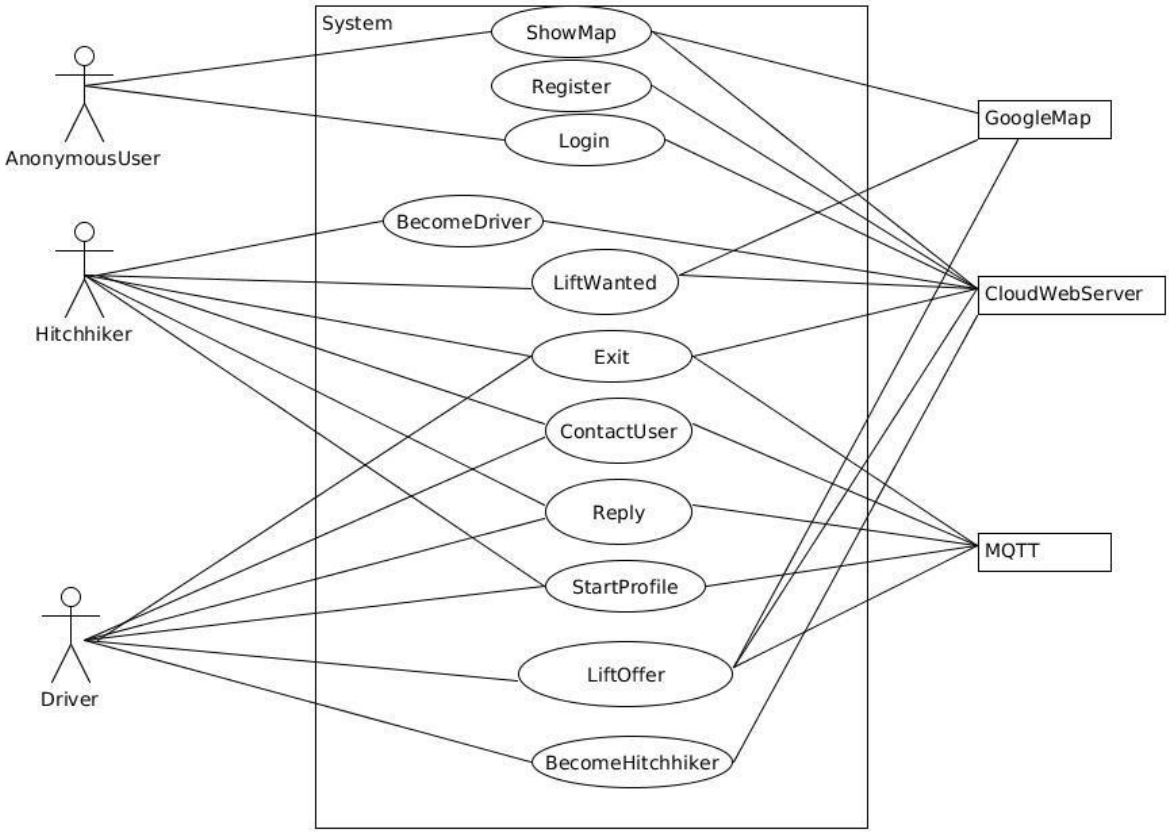
ShowMap, Register, Login

Driver user use cases:

LiftOffer, CheckMail, Reply, StartProfile, BecomeHitchhiker, Exit

Hitchhiker user use cases:

LiftWanted, CheckMail, contactUser, BecomeDriver, Exit



generic view of use cases of the system

Detail Use Cases

System is handled as black box**.**

ShowMap

Actor AnonymousUser

1. This use case begins when user click the "See the MAP" button on main activity.

2. System connects with Cloud Server and retrieves list of stored users GPS coordinates.

3. System connects with google map interface (through google map library) and return initialized map instance.

4. System loops populating map with android icons according to the GPS coordinates derived earlier from web.

5. System Displays map back to user.

Alternatives:
2a. Network error. Either local or backend.
3a. Problems with Google Play service.

Login

Actor AnonymousUser

1. use case begins when actor wants to log in into the system
2. User inputs username and password into system.
3. System sends inputted username and password into cloud web server.
4. Cloud server returns "true" validCode that indicates correctness of user data.
5. System start profile activity for user.

Alternatives

2.b User didn't fill username or password into system.
2.c User only inputs username or password.

3.b Android device experience network error.
3.c Cloud web server experience network error.

4.b server return "false" valid Code into system, that indicates incorrectness of user data
4.c System outputs screen error Message.

Register

Actor AnonymousUser

1. Use case begins when actor wants to become a member of the system and starts Register activity.
2. User inputs username and password and email into system.
3. System sends inputted user data - name, password, and email - into cloud web server.
4. Cloud server returns "true" validCode ,that indicates correctness of user data.
5. System start android login activity for user.

Alternatives

2.b user didn't fill username or password or email into system.

3.b Android device experience network error.
3.c Cloud web server experience network error.

4.b Server return "false" validCode into system, that indicates that username is not unique.
4.c System outputs screen error Message.

StartProfile

Actors: Driver, Hitchhiker

1. use case starts when Actor was successfully logged into the system
2. System start profile activity
3. System make constant connection with external MQTT server
4. System subscribes to MQTT topic with system user name.
5. System starts session.

Alternatives

3.b Network error.

3.c MQTT service is down.

LiftOffer

Actor: Driver

1. Use case starts when actor starts LiftOffer Activity.
2. System retrieves system user GPS coordinates
3. System generates a query to external Cloud with actors current GPS coordinates. 4. Cloud server returns collection of GPS coordinates - which is based on Driver GPS coordinates (more specifically all GPS coordinates with radius of 30km from current user GPS coordinates).
5. System starts new activity ,which requires google map instant
6. google map instant is retrieved from library
7. map is populated with android icons according to GPS coordinates reveived from cloud.
8. Actor picks user to which he intent to contact.
9. ContactUser activity starts.

Alternatives:

2b GPS coordinates not retrieved.

3b Network error

5b Google play service not installed

8b Actor tries to contact users on map without selecting any.

LiftWanted

Actor: Hitchhiker

1. This use case begins when system loads LiftWanted activity to Actor.
2. User determine his current GPS location.
3. System saves current GPS in java
4. User intent to save his destination.
5. User starts new google map frame.
6. user pinpoint on that map his destination
7. System saves destination in java
8. user clicks sendData()
9. System sends data over web cloud server
10. web server response with result (either recorded or not)
11. System outputs result.
12. System starts profile activity.

Alternatives:

2b. GPS service not turned on.

5b. Google Play Service not installed

8b. User hasn't set his current location or destination.

9b. Server not available.

ContactUser

Actor: Driver, Hitchhiker

1. This use case starts when actor request "contact_user_activity"
2. contact_user_activity might contain selected user.
3. Is selected user is additional data. User is saved in system.
4. System outputs users he can contact. By default 1 username appears. System populates that's usernames messages into the screen.
5. User enters message.
6. user sends message
6. System publish message into MQTT cloud.
7. System saves newly created message into System User messages (in Java). 8. System repopulates the screen with the user just sent with added message in the screen.
9. System shows screen to user.

Alternatives

6. b Network error - ex. MQTT connection lost.

Exit

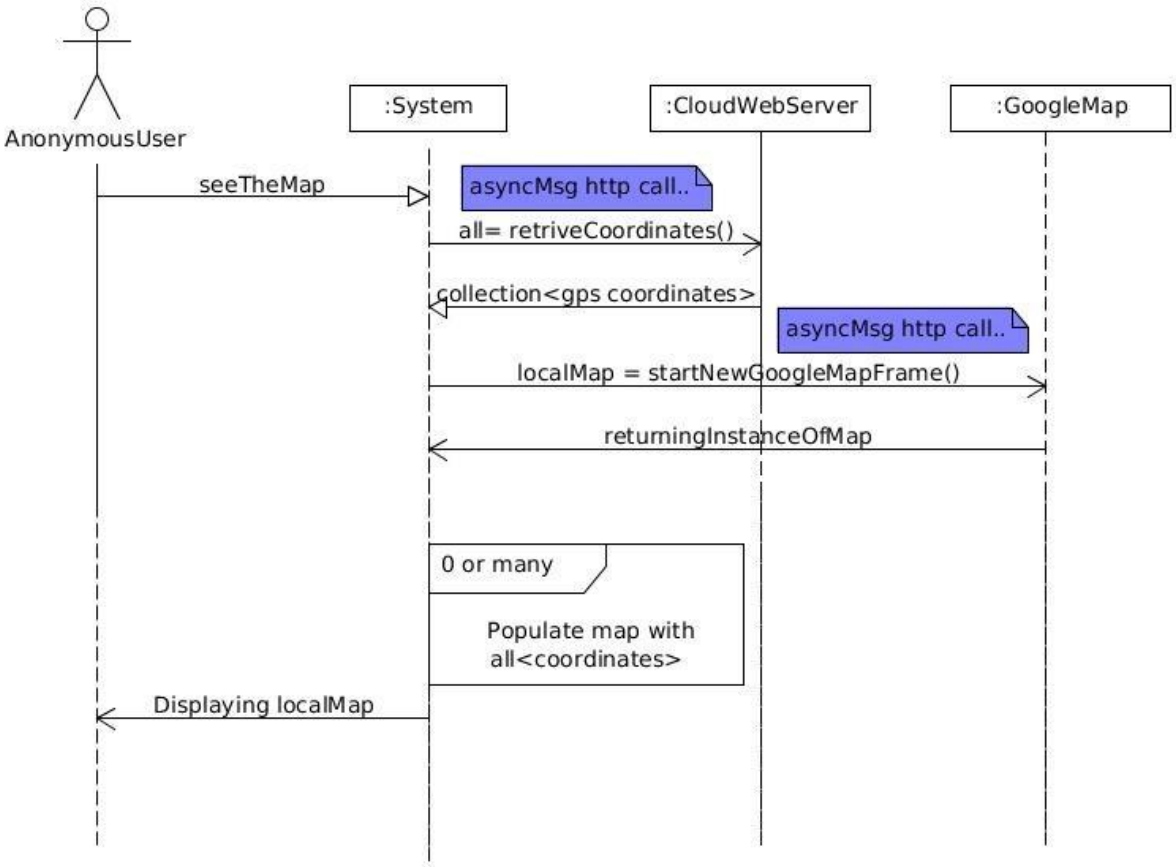
Actor: Driver, Hitchhiker

1. this use case starts when user wants to finish use of app.
2. system clears shared preferences
3. system stops service that connects to MQTT server
4. System clears stack
5. System starts main_activity.

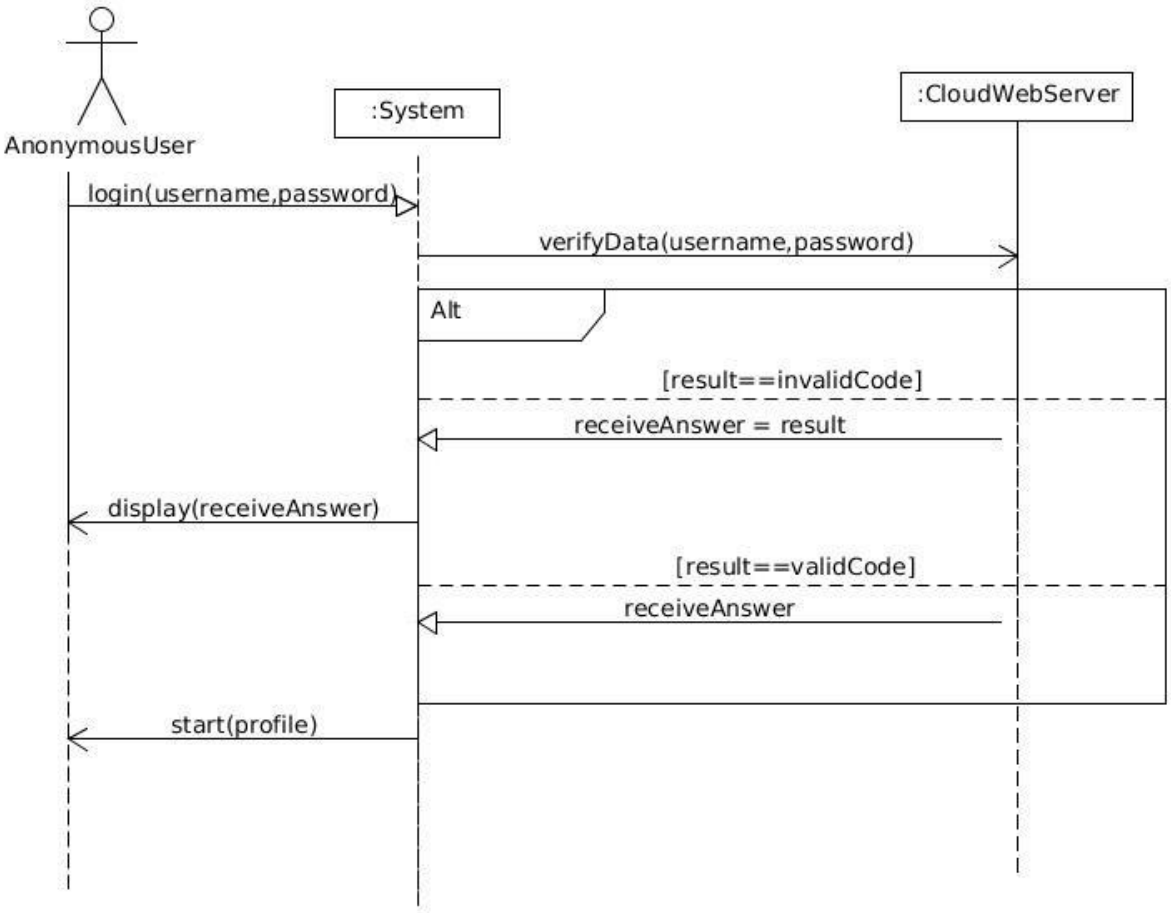
System Sequence Diagrams

One System Sequence Diagram per use case.

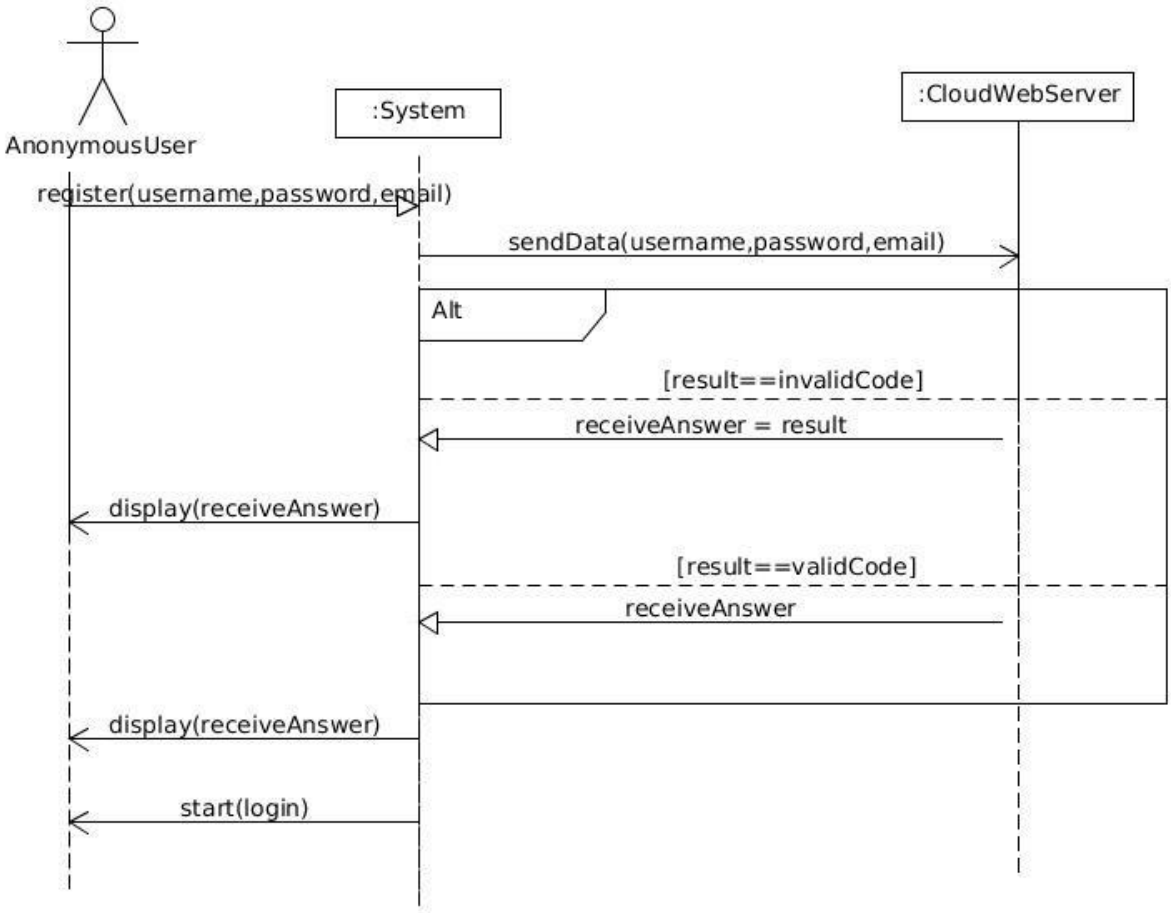
Name: ShowMap



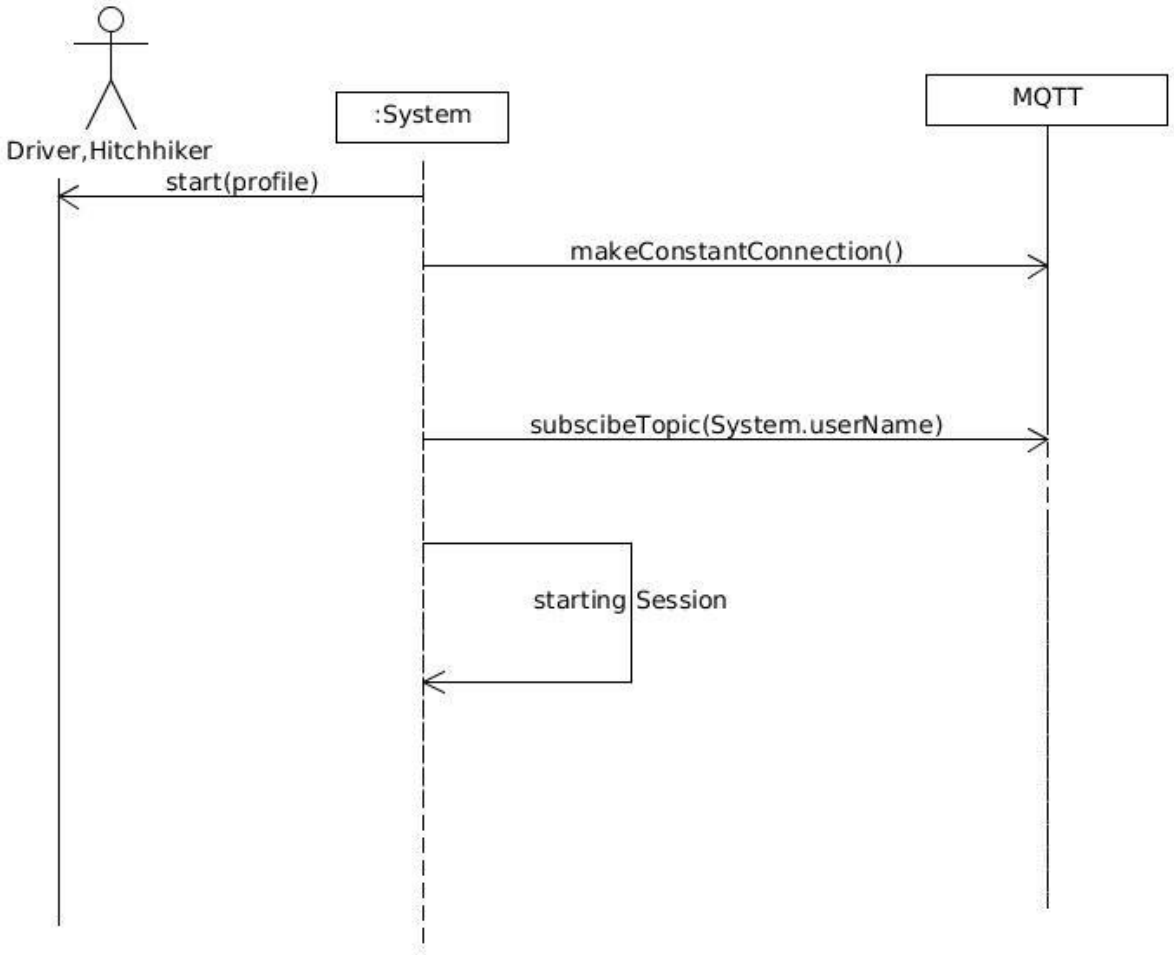
Login


AnonymousUser
:System
:CloudWebServer
login(username,password)
verifyData(username,password)
Alt
[result==invalidCode]
receiveAnswer = result
display(receiveAnswer)
[result==validCode]
receiveAnswer
start(profile)

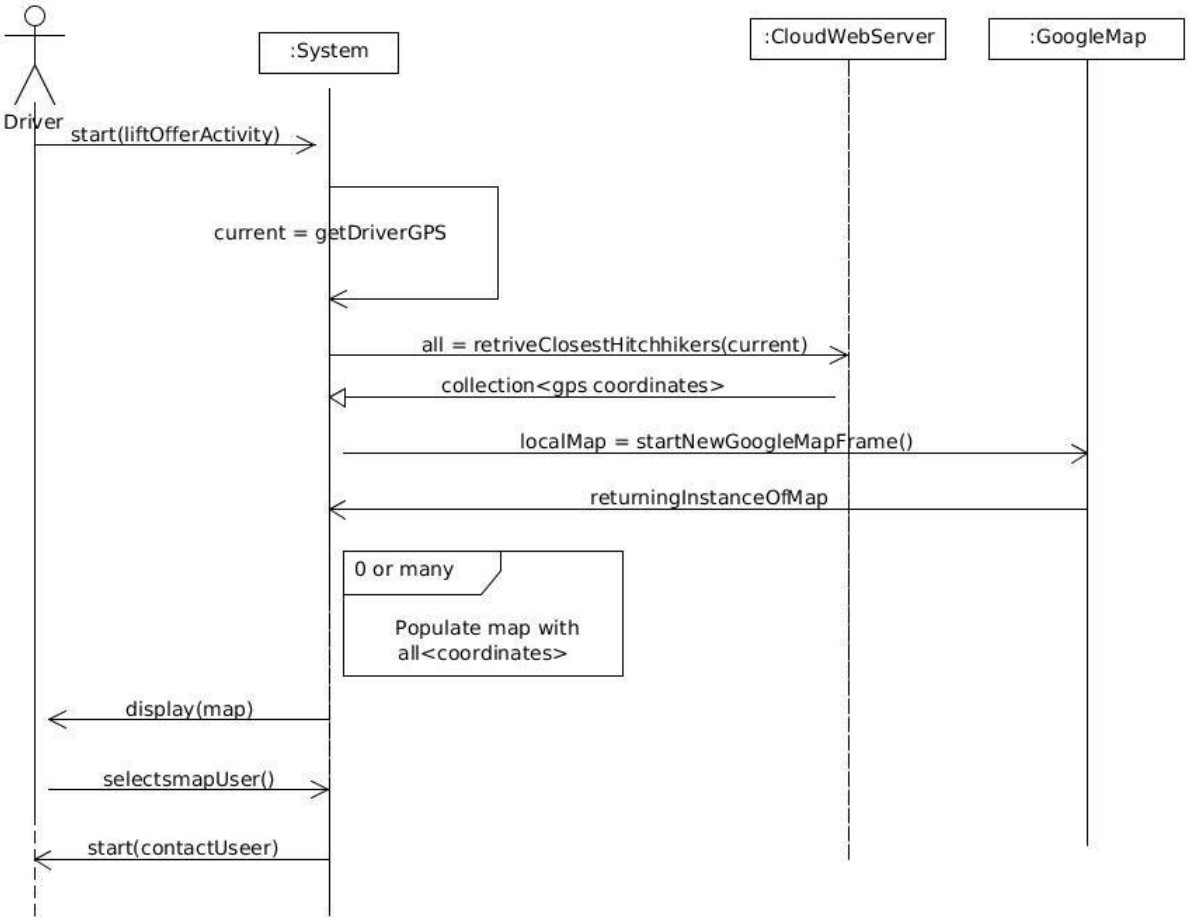
Register


AnonymousUser
:System
:CloudWebServer
register(username,password,email)
sendData(username,password,email)
Alt
[result==invalidCode]
receiveAnswer = result
display(receiveAnswer)
[result==validCode]
receiveAnswer
display(receiveAnswer)
start(login)

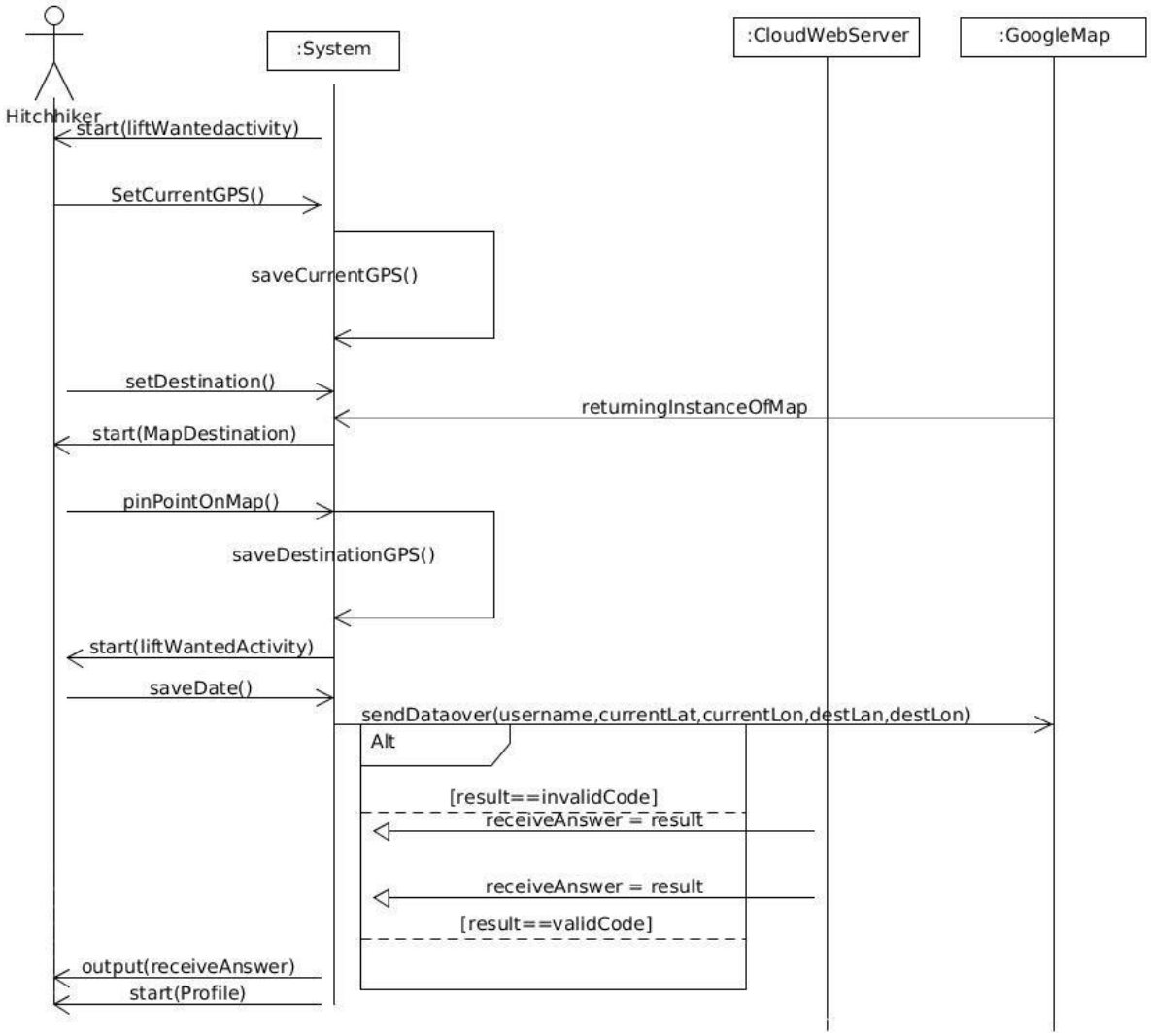
StartProfile



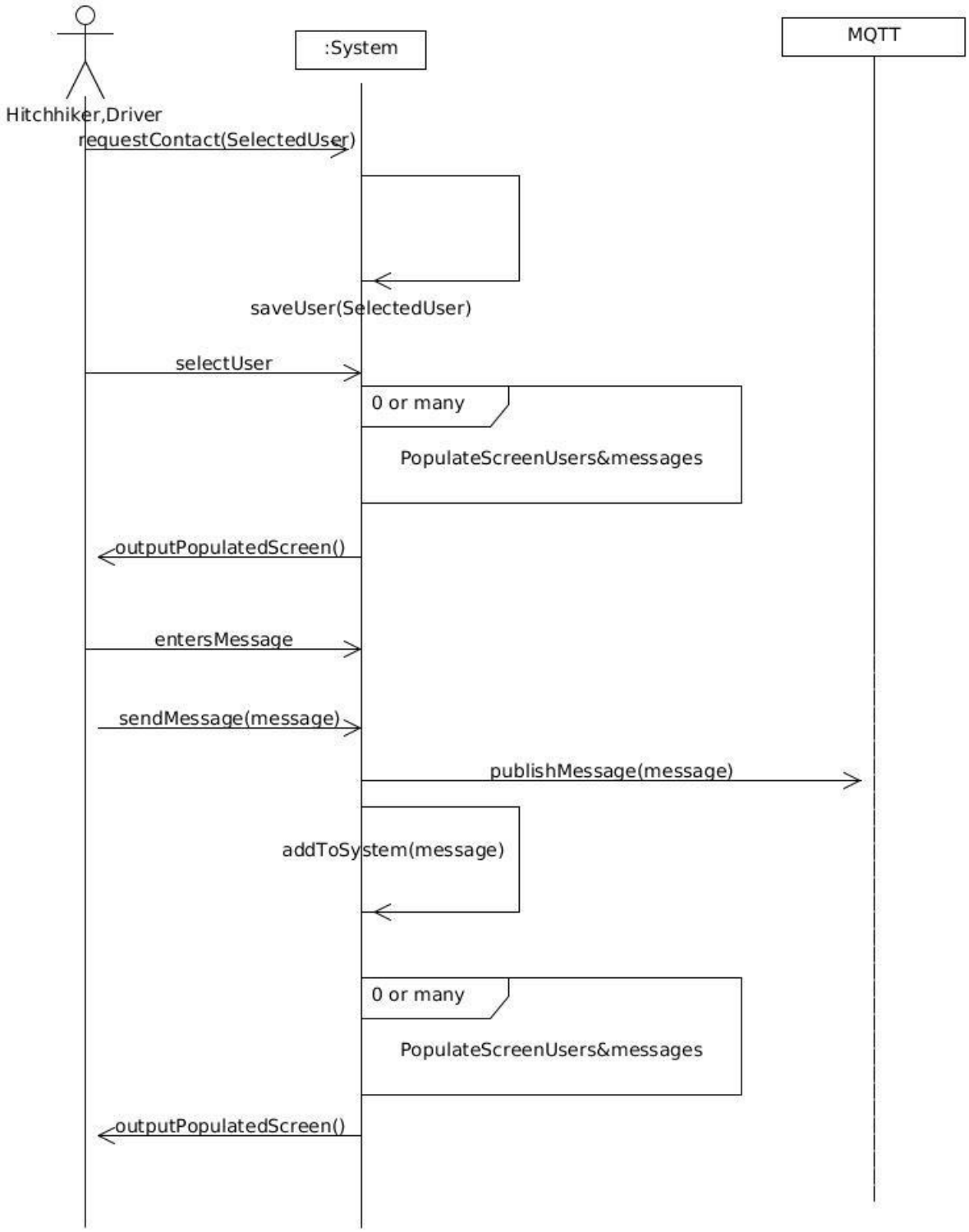
LiftOffer


Driver
:System
:CloudWebServer
:GoogleMap
start(liftOfferActivity)
current = getDriverGPS
all = retriveClosestHitchhikers(current)
collection<gps coordinates>
localMap = startNewGoogleMapFrame()
returningInstanceOfMap
0 or many
Populate map with
all<coordinates>
display(map)
selectsmapUser()
start(contactUseer)

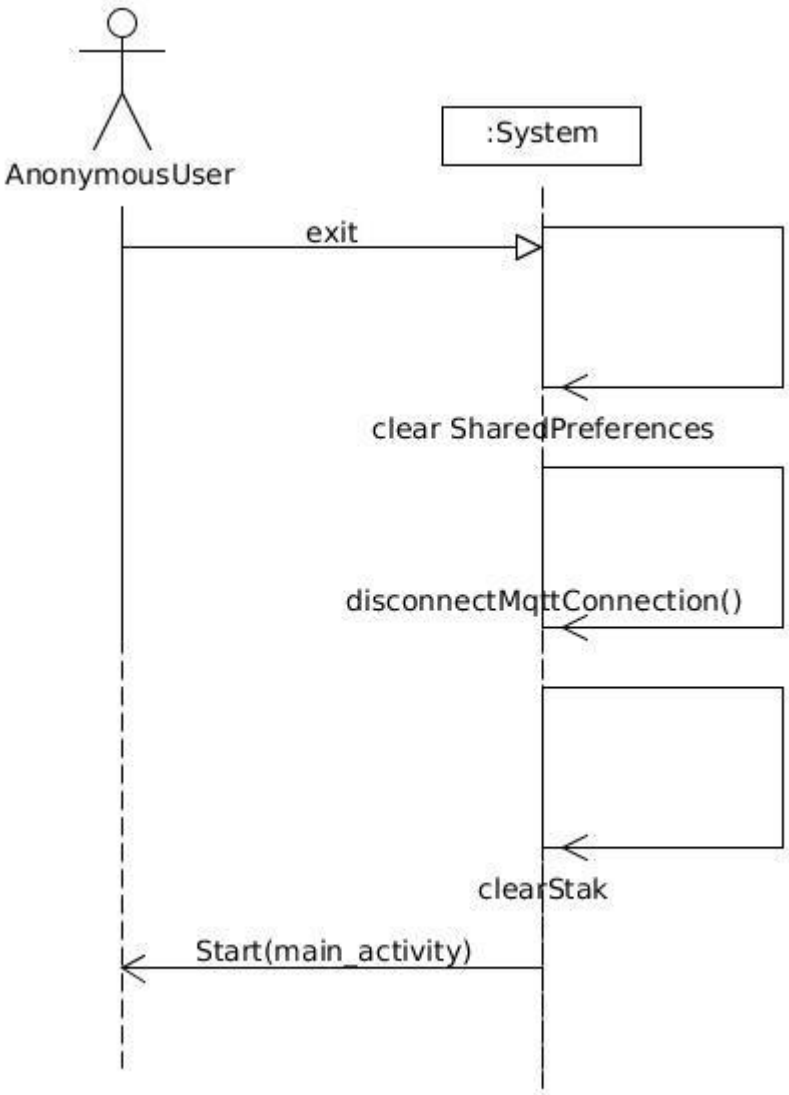
LiftWanted


Hitchhiker
:System
:CloudWebServer
:GoogleMap
start(liftWantedactivity)
SetCurrentGPS()
saveCurrentGPS()
setDestination()
returningInstanceOfMap
start(MapDestination)
pinPointOnMap()
saveDestinationGPS()
start(liftWantedActivity)
saveDate()
sendDataover(username,currentLat,currentLon,destLan,destLon)
Alt
[result==invalidCode]
receiveAnswer = result
receiveAnswer = result
[result==validCode]
output(receiveAnswer)
start(Profile)

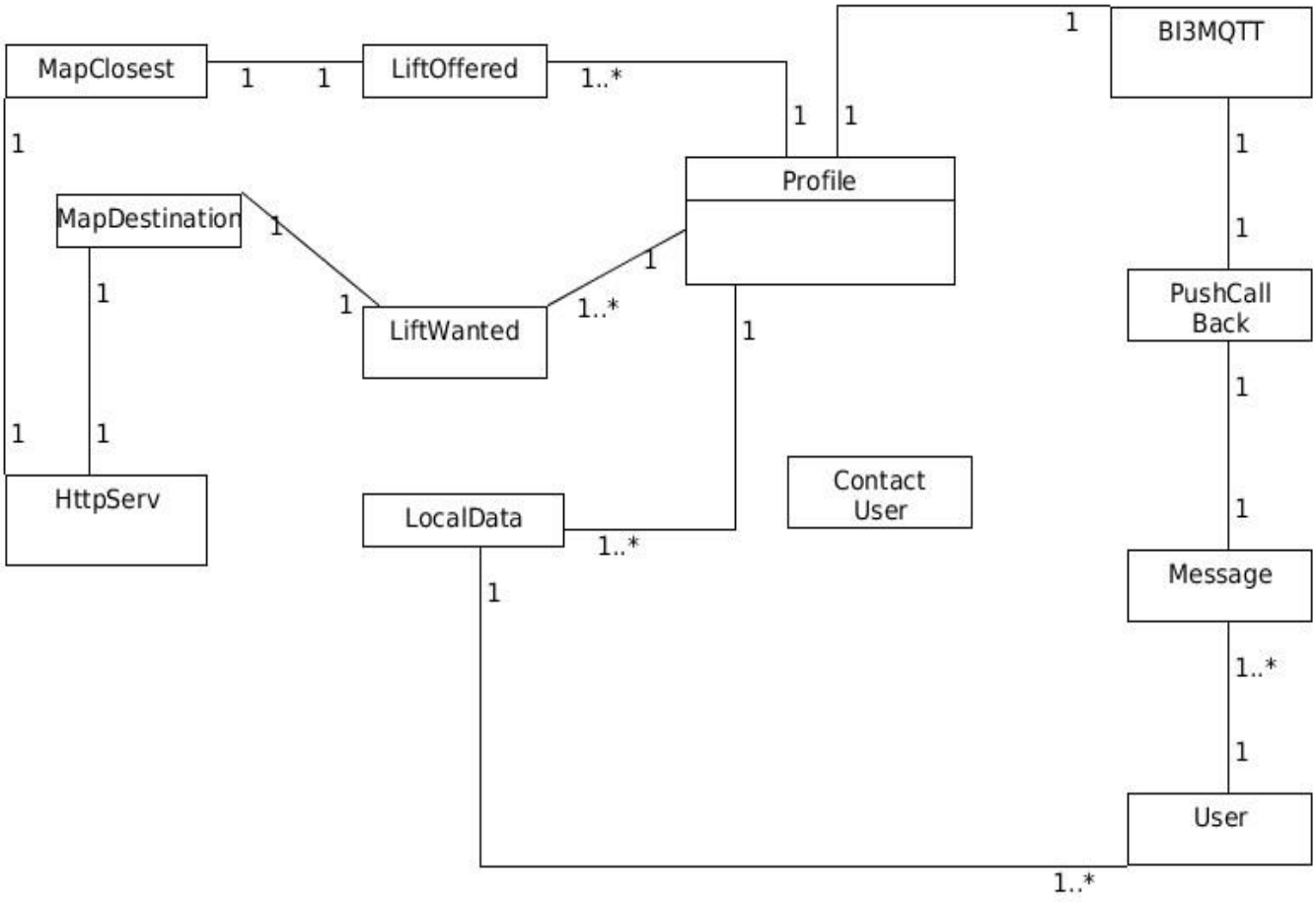
ContactUsers



Exit

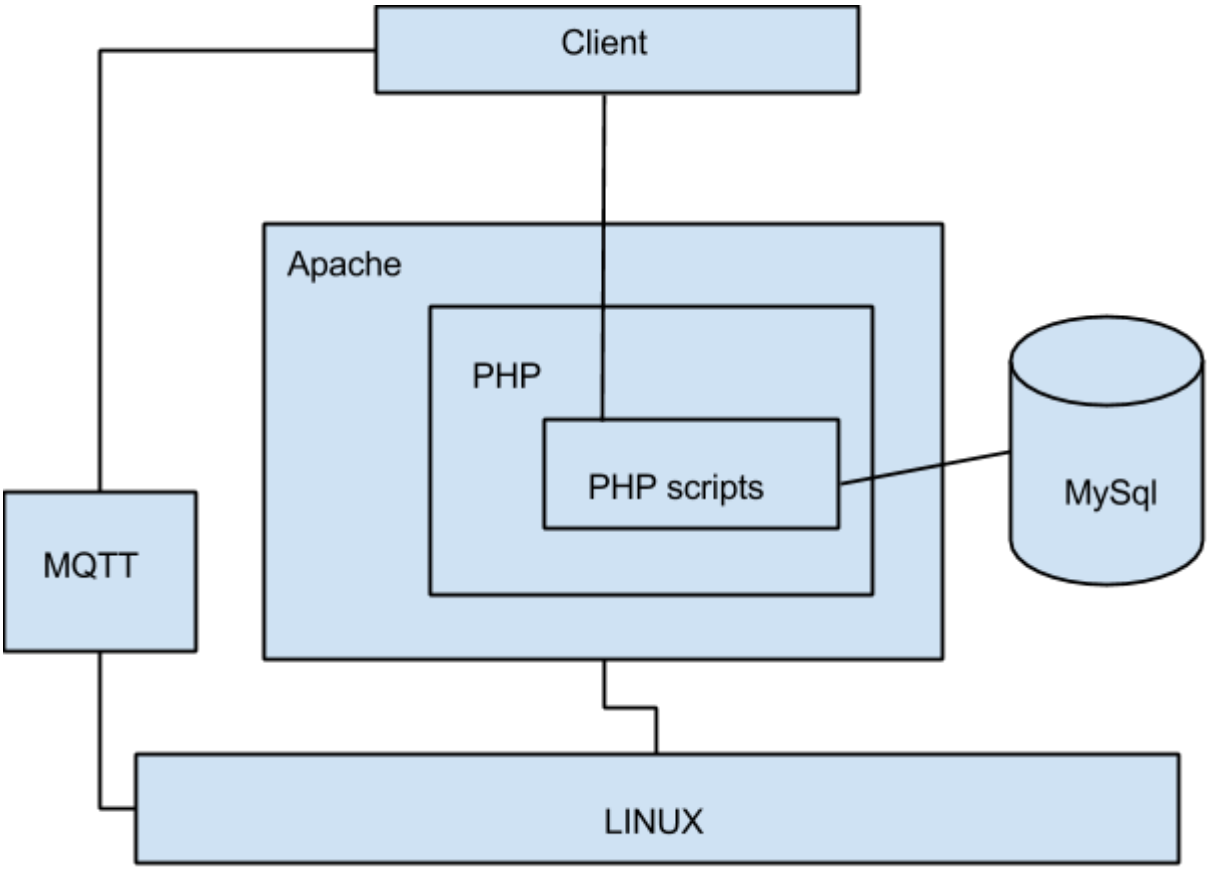

AnonymousUser
:System
exit
clear SharedPreferences
disconnectMqttConnection()
clearStak
Start(main_activity)

Domain Model Diagram



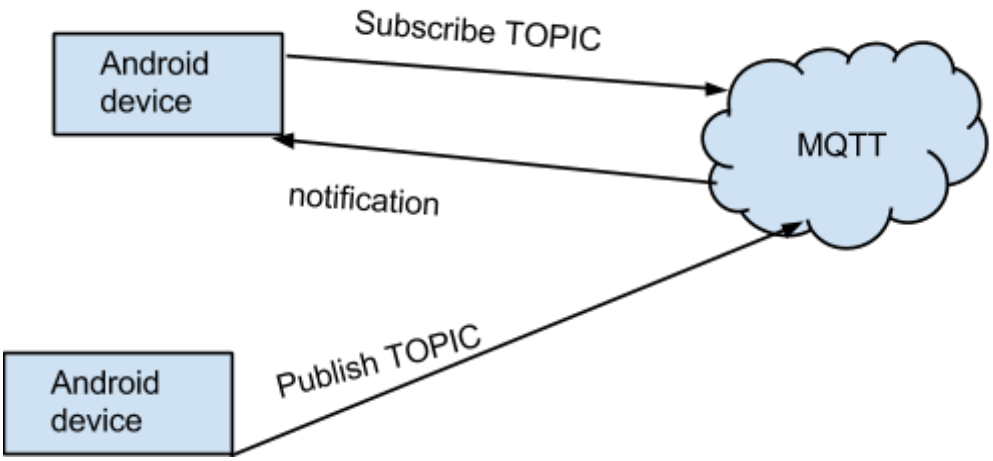
Architecture

Observation of Hardware architecture and 3th party application involved. More details in Research manual.



Interaction between client and cloud.

Client is android device. Minimum support version is “11” and target version “21”.
Apache is an HTTP web server and most widely used open source web server software.
PHP is a server side scripting language.
MySQL is open source relational database management system, using SQL
MQTT or Message Queue Telemetry Transport was designed and implemented by IBM as open source “light weight” messaging protocol for devices with limited battery power or limited network bandwidth, which makes ideal for our android project as notification of the events. See the drawing below.



Interaction between client activity and MQTT tcp/ip server

Multiple mobile clients can access the data without affecting each other.
App is using MQTT for receiving notifications. For updating MySQL DB App is making “HttpRequests” into PHP scripts.

Software for developing the system

- Eclipse IDE for Android Developers,using Java programming language
- SDK Emulator. Better to use real android phone.
- GIMP some drawing might occur.
- Git - Software Version control,
- Apache2 Web server. Running and providing web service functionality.
- Php5 installed and configured as Apache module.
- MySQL installed and configured as Apache module
- MQTT server.

More on that in Research manual.

Database

Originally 3 tables is replaced with 2 tables for Version 1 App.

User table has User related data.

RidesWanted table is Hitchhiker related data. Hitchhiker can generate many request, but last one will be active (see the field Active). Each time User generates Rides Wanted record the active field become 1 and any previously generated records by that user is voided (by record becoming 0)

Database also contained Tabled Rides Offered for Car driver, but were deprecated as version 1 of the software doesn't really require memo of drivers picking up hitchhikers.

User table user id is used as foreign key in RidesWanted table.

User table

☐	**userId**	int(11)			No		auto_increment
☐	**userName**	varchar(20)	latin1_swedish_ci		No		
☐	**password**	varchar(20)	latin1_swedish_ci		No		
☐	**email**	varchar(15)	latin1_swedish_ci		No		
☐	**phone**	int(15)			No		
☐	**validAccount**	tinyint(1)			No		
☐	**sex**	char(1)	latin1_swedish_ci		No		
☐	**age**	int(3)			No		

RidesWanted table

	Field	Type	Collation	Attributes	Null	Default	Extra	
☐	**id**	int(11)			No		auto_increment	
☐	**userId**	int(11)			No			
☐	**Date**	date			No			
☐	**StartLocationLat**	varchar(15)	latin1_swedish_ci		No			
☐	**DestinationLat**	varchar(20)	latin1_swedish_ci		No			
☐	**StartLocationLon**	varchar(15)	latin1_swedish_ci		No			
☐	**DestinationLon**	varchar(20)	latin1_swedish_ci		No			
☐	**active**	tinyint(1)			No			