

POLITECNICO DI MILANO

SOFTWARE ENGINEERING 2 PROJECT

TAXI SERVICE



REQUIREMENT ANALYSIS AND SPECIFICATION DOCUMENT

AUTHOR

Pallotta Simone

PROFESSOR

Raffaella Mirandola

Version 1.0

ACADEMIC YEAR 2015/2016

intentionally left blank

CONTENTS

Introduction	5
1.1 Description	5
1.2 Goals	6
1.3 Glossary	8
1.3.1 Definitions	8
1.3.2 Acronyms	9
1.4 Reference	10
1.5 Overview	10
Overall Description	11
2.1 Product perspective	11
2.1.1 System interfaces	12
2.1.2 User interfaces	12
2.1.3 Hardware interfaces	16
2.1.4 Software interfaces	16
2.1.5 Communication interfaces	17
2.1.6 Memory Constraints	18
2.2 Product functions	19
2.2.1 functional requirements	19
2.3 User Characteristics	20
2.4 Constraints	20
2.5 Assumption and Dependencies	20
2.6 Future Enhancements	21
Specific Requirements	22
3.1 Scenario	22
3.1.1 user sign up	22
3.1.2 asking for a ride	22

3.1.3 user sign in	23
3.1.4 check ride status	23
3.1.5 accept or deny a request from taxi driver	24
3.2 Performance requirements	25
3.3 Availability	25
3.4 Security	25
3.5 Error handling	26
3.6 Usability	26
3.7 UML Modeling	27
3.7.1 use case diagram	27
3.7.2 sequence diagram	28
3.7.3 state diagram	29

Appendix 31

4.1 Used Tools	31
----------------	----

Introduction

1.1 Description

TheTaxiService Project is aimed to allow customers to ask for a ride to “Taxi Service” via mobile application, to allow taxi drivers to inform the system about their availability and to permit them to confirm (or refuse) the call issued by the system.

The Application will behave differently based upon user profile, (customer or taxi driver) with different features, interface and functionalities.

The back-end system will handle the incoming request considering the geolocation of both requester and available taxi: the system will dispatch the call to internal queues, called “taxi queues”, each for every zone. Scanning the queues of available taxis, it will finally assign the ride to a specific taxi based upon geolocation of both customer and taxi.

After assigning a job (sending a message to the designated taxi), the system will wait for the confirmation by the taxi driver. In case the driver does not respond before time-out or refuses the job, the system will re-assign the ride to the next available taxi. Only upon acceptance, the system will notify the customer of the number of the taxi in charge of his travel and an the estimated arrival time.

The reservation system is completely automated and doesn’t allow to ask for a ride by phone call to an operator, so the only way to specify the pick-up location is by means of physical coordinates taken from the device in a fully automated way.

Each customer or taxi driver must be authenticated before using the system through standard authentication mechanism (e.g user password, email and a confirmation code, usually referred as 2FA).

The mobile application will implement its functionality when in background or off to notify the user (e.g push notification or similar means.) via dedicated mechanism built in OS that will awake App.

1.2 Goals

The goals of TaxiService application is to simplify the access of passengers to the service, and guarantee a fair management of taxi queues thus reducing travel and wait times.

The main goal is achieved via these tasks:

- acceptance of a ride request from a customer
- notification to every taxi about a pending request
- confirmation (or not) by the taxi driver to accept the ride
- reduction of waiting time using geolocation both for customers and taxis
- notification to the customer about the code of incoming taxi and expected waiting time
- calculation of the the waiting time based upon traffic, distance and other conditions using statistical approximations
- use of the number of seats requested by user to select only taxis with enough seats.
- handling of a dedicated taxi queue for each zone to save (or rotate in case of deny) the taxi identifier
- prevention of multiple requests issued by the same customer
- prevention of sending multiple rides request to the same taxi

NOTE: there is NO direct connection between customer App and taxi App.

This Project will also provide general features such as:

- Registration to the system of customers and taxi drivers
- Authentication to the system of customers and taxi drivers
- Verification code sent via SMS on first registration

The software will not provide a kind of history about previous rides of single customer and doesn't display the cost of the current ride.

In this current implementation, a customer can ask for a ride for multiple persons, but his ride is personal, as if he were alone on the taxi.

Also, this first version will not manage any feature about the programmatic interfaces for additional services (e.g taxi sharing), even if it will be designed keeping these expansions on mind to easily implement them.

1.3 Glossary

1.3.1 Definitions

Geo Location: The physical position expressed by longitude and latitude.

Ride: the geo-localised movement of a vehicle (taxi) among two geo locations.

Customers: registered human being that can make a ride/transportation request for one or more persons.

Taxi: the vehicle that can carry customers.

Taxi Identifier: an alphanumeric sequence that uniquely identifies a vehicle.

Taxi Status: current condition of a taxi. Status can be:

- out of service
- waiting
- reaching a customer
- riding

ID an alphanumeric sequence that uniquely identifies a customer or a taxi driver.

Taxi Driver: registered human being that can accept a ride request and fulfil it and communicate the status after a ride. His **ID** is matched with a single Taxi ID a once.

Ride executor: the taxi that will execute a ride.

Chosen ride executor: the taxi chosen by the system for a specific ride: if accepted, it gets the Ride executor and its Taxi ID is associated with the ride.

Push Notification: a short service message sent to mobile Apps with no assurance of delivery. Such message can be received even if the specific app is not running. The system take charge to save and deliver it to user / to the app later.

Push Notification Service: system to deliver mass Push Notifications to mobile Apps

Taxi zone: a square area where a Taxi rides.

User Credential: tuples of values that identifies uniquely a customer/taxi driver

Waiting Time: number of minutes the customer should wait for his taxi.

Estimated Arrival Time: the time in local time zone the taxi will be (hopefully..) present.

Taxi queue: an ordered list of taxi (each one identified in queue by its "Taxi Identifier") from where the system picks a designated ride executor.

Confirmation Code: numerical code generated by system upon registration needed to allow first authentication in the device after sign up

SMS Gateway: external system in charge of sending confirmation code via telephone company

1.3.2 Acronyms

APNS: Apple push notification system/ servers

DBMS: database management system

DB: database

ETA: Estimated Time of Arrival

SOAP: Simple object access protocol

1.4 Reference

- IEEE Recommended Practice for Software Requirements Specifications

1.5 Overview

In the come after will be introduced an overall description of requirement to provide an initial background; starting from chapter 3 we will describe in details the requirements the system must guarantee, shown in a UML diagram.

This Document is structured in four section, following the hint 5.1.4 stated in document IEEE Std 830-1998:

- **Chapter 1:** Introduction, Goal description of software
- **Chapter 2:** Overall requirement's description
- **Chapter 3:** Detailed requirement's description
- **Appendix:** Used tools

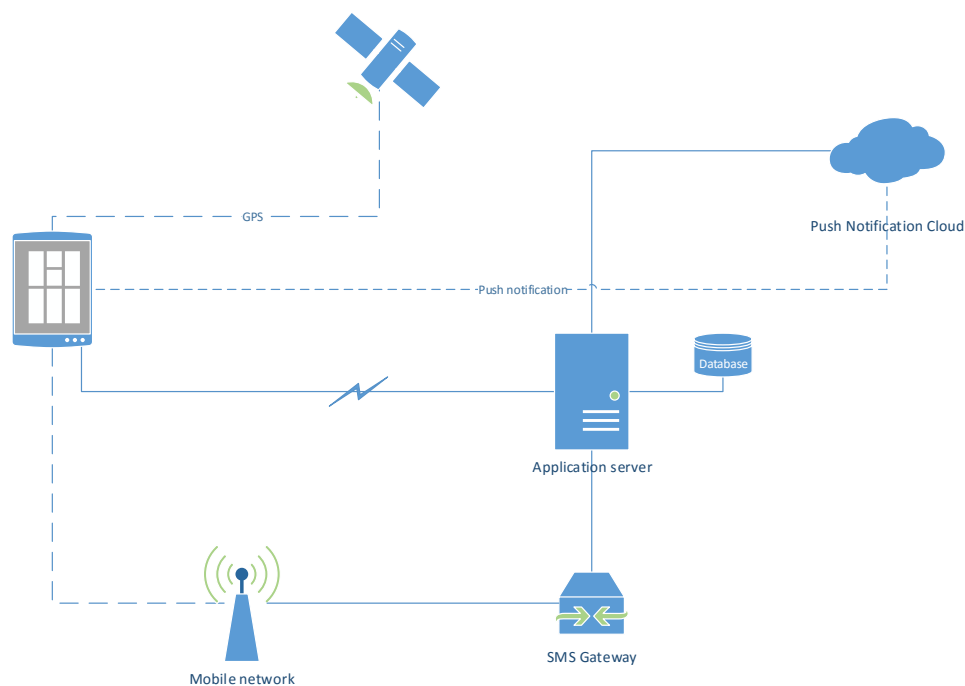
Overall Description

2.1 Product perspective

The application we will release is a mobile application which is not integrated with other existing system, such as travel systems, maps, or payment service providers.

That application depends on remote system via secure network connection but also it relies on geo-location and push notification.

Data for “geo-loc” are delivered directly by the HW subsystem to OS, and similarly Push Notification system sends messages directly to OS that will dispatch then to App.



2.1.1 System interfaces

Current the only system interfaces are towards APNS and SMS gateway.

We do not consider the system interface to geolocation system, as it's managed entirely by operating system.

In the same way we do neither consider interfaces between APNS and single devices (as it's directly managed by the system), nor from messaging system and device (confirmation code is received directly by system application and manually typed in by user.)

Interface to APNS is modelled following the Apple specification for iOS devices and Google Specs for Android devices.

2.1.2 User interfaces

This project will be a mobile application.

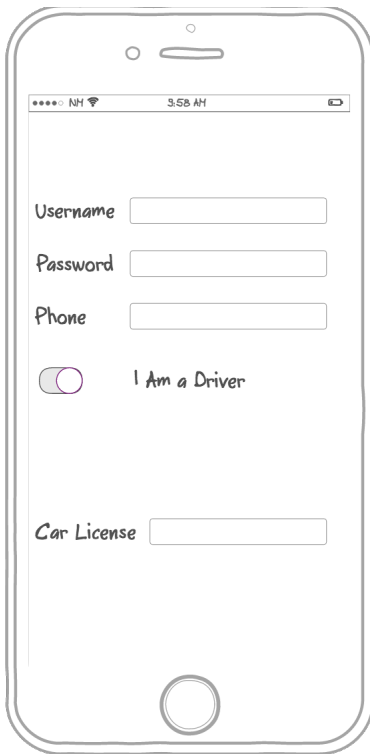
In the following we introduce the UI of application both customers and drivers:

Sign in View:



At launch the application checks the existence of an existing profile, if found will bring to appropriate to the main view that is different for taxi driver and customer. if a profile is not found, the app will show the sign up sequence.

Sign up View:



A mobile app screen showing a sign-up form. The status bar at the top displays 'NH' and '3:58 AM'. The form contains the following fields and elements:

- Username**: A text input field.
- Password**: A text input field.
- Phone**: A text input field.
- I Am a Driver**: A toggle switch (currently off) and a label.
- Car License**: A text input field.

During sign up phase the user fills a form with his own credentials such as username, password and other data such as email and cell phone.

If you are a taxi driver, you must type your taxi car license, too. The presence of a valid car license automatically qualify user as a Taxi driver.

After signing up, the user will be brought back to the verification view.

Verification View:



A mobile app screen showing the verification view. The status bar at the top displays 'NH' and '3:58 AM'. The screen contains the following elements:

- SMS Code**: A heading.
- Insert code received on your phone**: A text label.
- code...**: A text input field.
- Verify**: A button.

This view is shown only during registration process.

The user must type the code received from SMS.

When user tap "Verify", the code will be sent to the server, that can answer confirming the code or denying in case of mistyping. Note that the App will not send the code alone, but all the 3 values needed: user/pass/Code.

Upon receiving the confirmation message, the app will go directly main view(different between customer and taxi driver) after saving data in profile in persistent storage.

"Request a Taxi" View:



In this scenario the customer is able to ask for a ride, eventually specifying the number of seats requested.

This view show also the map, with a pin, indicating the position of the customer.

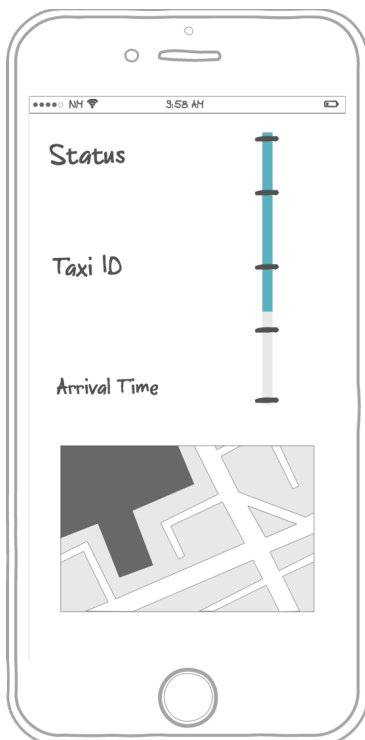
The "Ask for Ride" button will send a request to the server sending infos such as:

user ID

current geo-position

After waiting for the server response, (that does NOT contain confirmation, yet) the app will bring user to the next view.

Taxi ride status View:



In this view the system polls the server to get the current status of the request .

After the driver has accepted, system will poll for the ID and position of the taxi.

We do not use APN to send these data to the App, as doing so will require too much traffic.

Push notifications will be be sent upon confirmation and we have two scenarios:

- App is open on "Taxi ride status" view: view is updated and App starts polling for updates.
- App is closed: push notification will awake the App and the App will bring user to "Taxi ride status" view, as seen in previous scenario.

Accept request:



This view is available only to driver.

Once opened, the driver will see the current request (or none) and the position of the customer, if received through APN. (see below)

The driver has two options:

- "Deny": the driver deny request and system will update the queue.
- "Accept": the system send back a message to confirm current status and send also message to the customer.

Since accepting, the App is sending periodically its geo-position to the system to allow updating of taxi position to be sent to the customer. The status of the taxi passes to "reaching a customer" and app now shows a button "picked up" that must be tapped on when arrived to the location of the customer. When tapped, the app will change the button title, showing "end of ride".

Tapping on, the App signals to the system that the taxi status now is "waiting". If APN is received when the app is open, App will simply show the message

Usage of APN:

If App is off, a push notification will signal to the driver of a pending request of ride, that will be shown in detail when opened.

If app is open, APN is used to send request and trigger status transition: no poll to the server occurs.

2.1.3 Hardware interfaces

No hardware interfaces are required

2.1.4 Software interfaces

Database Management System:

Name: Mysql

Version: 5.6.10 Community Server

Ruby Framework

Name: Ruby on Rails

Version: 4.2.1

Package Manager:

Name: Ruby Gem:

Before your Rails application can connect to a MySQL server, you need to install the MySQL adapter. The gem provides this functionality.

iOS Operating system:

Name: iOS

Version: 9.0.1 (minimum required)

Android Operating system:

Name: Marshmallow

Version: 6.0 (minimum required)

*2.1.5 Communication interfaces***Client/application server:**

All the communication will be performed via HTTPS (http over TLS) using standard certificate chaining.

At the application layer every client request will implement a fully compliant REST approach based on JSON format.

Application Server/DB:

The connection will be performed using TCP protocol based on 3006.

The DB will listen only on local port for security reason.

APNS/mobile app:

All the messages will use standard protocols based on TLS as specified by push notification Specs:

Apple:([ApplePushService.html](#)) and similar protocols for Android equivalent.

TCP port 5223: used for communication with APNs

TCP port 2195: used for forwarding notification to APNs service

TCP port 2196: used for feedback APNs service

Application server/SMS gateway:

SMS Gateway enables customer applications to access most of the main features of the messaging platform through the SOAP protocol

The service is available over HTTP and HTTP SSL/TLS.

2.1.6 Memory Constraints

Mobile application generally has no memory issues; on some low end devices maps can hog a lot of memory.

On Application Server there are no special requirements as request will be generally managed by the database with little processing overhead.

On Database there are no special requirements as queries will be very simple.

The only real memory constraints can be considered when dealing with the payload of push notification messages where payload is generally limited to less or equal to 2 kilobytes.

(see [CommunicatingWithAPS.html](#))

2.2 Product functions

We introduce the functional requirement that application and system must guarantee:

2.2.1 *functional requirements*

- Customer:
 - can sign up
 - activate his account using activation code
 - can sign in and sign out
 - can issue a ride request once at time
 - check the status of his request
 - can see on the map the current position of taxi (if assigned) and his own
- Taxi driver:
 - can sign up
 - activate his account using activation code
 - can sign in and sign out
 - can accept a ride request once at time
 - can refuse a ride request
 - can see on the map his current position and the customer's pickup location

2.3 User Characteristics

We distinguish two type of user: customer and taxi driver, both can have a mobile device with a valid mobile connection.

The user must know the UI of used device and able to navigate on the standard interface.

To the user is not request any specific language knowledge as the application will be localised.

No technical expertise is required to use application.

2.4 Constraints

The application cannot works offline.

All the application flow works only with mobile connection.

As the messaging is based on remote notification, notifications must be enabled on the devices and firewall ports must be open to allow receiving APNS messages.

2.5 Assumption and Dependencies

For the correct running of system is need to user satisfy the following requirements:

- user must have a valid mobile subscription
- user must have an appropriate data plan
- user must enable push service on his device
- If connecting thought the firewall, ports must be open
- application server must have static IP, but is not required to be registered in DNS
- server must have a valid certificates installed for push notification
- the server must have valid certificates to deliver HTTPS connection. (application can use self-signed certificate, but for iOS 9 must be avoided self-signed certificate and instead used standard certificates from CA authorities)

- the request are inserted on FIFO queue stored in DB; a task under cron will periodically try to assign requests to a free taxi and will modified the taxi queue. The same cron task will trigger sending push notification to notify mobile app of changes in the ride status. (cron will ne modified to run periodically ruby script, possibly with low privilege user level for security reasons)
- As not earlier, updates to taxi positions will not be managed via push notification but the mobile apps will poll server to know the position of the taxi. Taxi will send its position via standard web request. (REST PUT command).

2.6 Future Enhancements

The next version of this application may provide new features:

- taxi sharing
- payment on board
- password recovery
- history of past rides
- allow customer to cancel a request not yet accepted
- update and modified customer credentials
- Reports about rides and utilisations of every taxi and average journey

Specific Requirements

3.1 Scenario

3.1.1 user sign up

Scenario	user registration
Actors	customer or taxi driver
IN state	user has not signed up yet
<p>To use a mobile application, the user needs to authenticate.</p> <p>He must click on sign up button and insert his own credentials.</p> <p>The system will generate an activation code and through the SMS gateway the user will get it to authenticate.</p>	
OUT state	the user will be signed up

3.1.2 asking for a ride

Scenario	ask a ride
Actors	customer
IN state	user is signed in
<p>The customer must fill form with some information (minutes, number of seats..) and click the "ask" button.</p> <p>The system will receive the call and will forward it to the taxi driver waiting for acceptance.</p>	
OUT state	customer send a request for a taxi ride

3.1.3 user sign in

Scenario	user sign in
Actors	customer or taxi driver
IN state	user has not signed in yet
<p>The user must fill in a form with credentials (username or mail and password).</p> <p>The system will verify the correctness of credentials and will allow the user to use mobile application.</p>	
OUT state	the user will be signed in

3.1.4 check ride status

Scenario	see the status of a ride
Actors	customer
IN state	the customer has already requested a taxi
<p>After the acceptance by the taxi driver the customer will be warned by push notification about the acceptance of the ride from taxi driver.</p> <p>The customer will be able to monitor the status of a ride.</p>	
OUT state	the user see the status of a ride

3.1.5 *accept or deny a request from taxi driver*

Scenario	accept or deny a request by taxi driver
Actors	taxi driver
IN state	taxi driver receive a request
<p>The system notifies taxi driver about the request from customer.</p> <p>The taxi driver will see the map with the location of the customer and two buttons (deny or accept): the driver can accept or not.</p> <p>If accepted, the system notifies the customer.</p> <p>If the taxi driver denies the ride, no information will be notified to the customer.</p>	
OUT state	the system will inform the customer about the accept decision of a taxi driver

3.2 Performance requirements

On the mobile application, all the network connection must be performed in a asynchronous mode to prevent blocking UI.

Th response times are very dependant on network connection: DB/application server response time are usually one or two order of magnitude lower.

On mobile performance must satisfy a fluid and responsive experience.

Typically we can assume response time on server must be within 5 seconds.

3.3 Availability

Special care must be put to grant constant availability of application server.

The system cannot tollerate downtime larger than 1 minute.

On the server boot scripts will restart automatically all the services in case of power loss and the following restart.

Administrator can access remotely (SSH) the server to monitor and restart services.

3.4 Security

- Mobile Application:

The credential must be stored securely on keychain on mobile.

User must authenticate using user/password mechanism and SMS confirmation code

All the communication will flow via HTTPS /TSL connection

- Server application

Server accept request only by HTTPS on port 443

root access vis SSH is disabled

must be hardened using stem level mechanism such as SELinux

- Database

accept connection only on local port from application server

- APNS

guarantee via TLS transport mechanism and guarantee by APNS provision profile and the same for mobile app and APNS

3.5 Error handling

In case of connection lost, the application will not notify the user until the user will perform a network request.

In this case the application will show pop-up message asking to check the connection status.

3.6 Usability

The User interface must be as clear as possible.

The application must be usable from a heterogeneous public

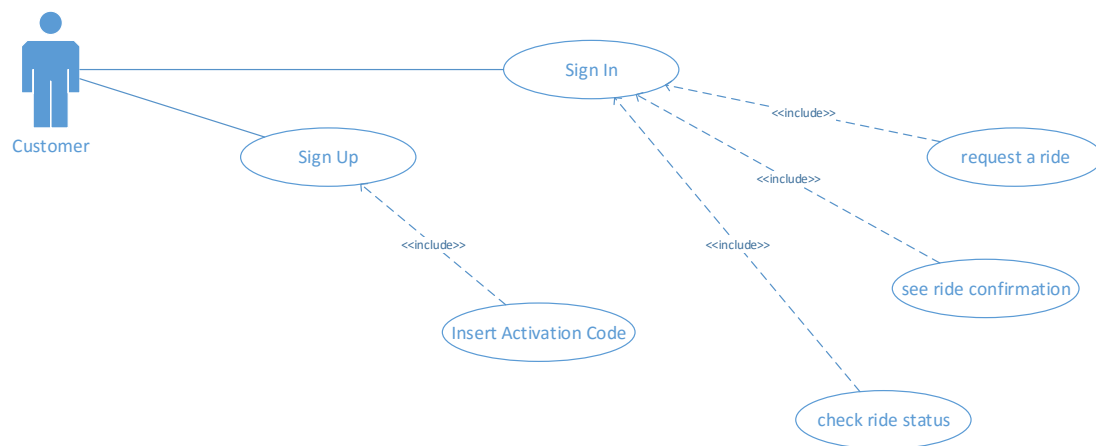
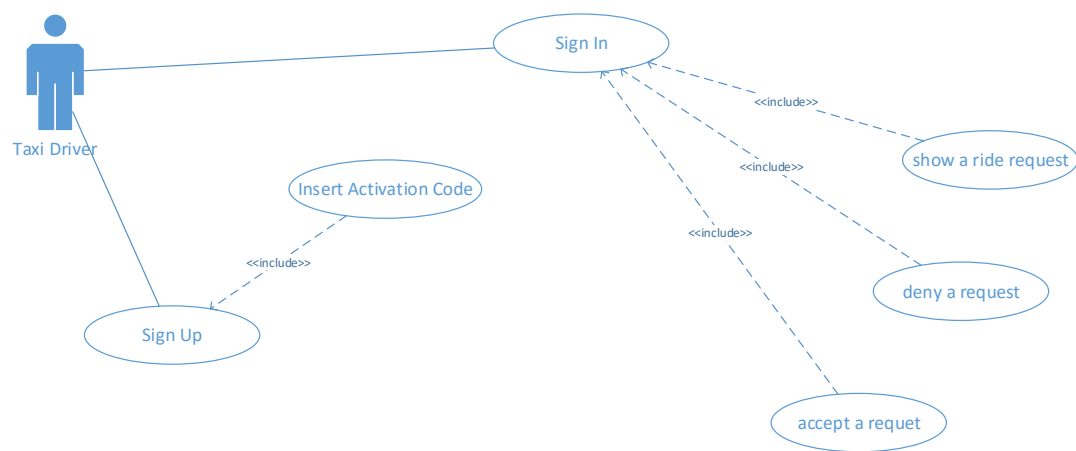
The app will use the typical “paradigm” of every platform and ecosystem.

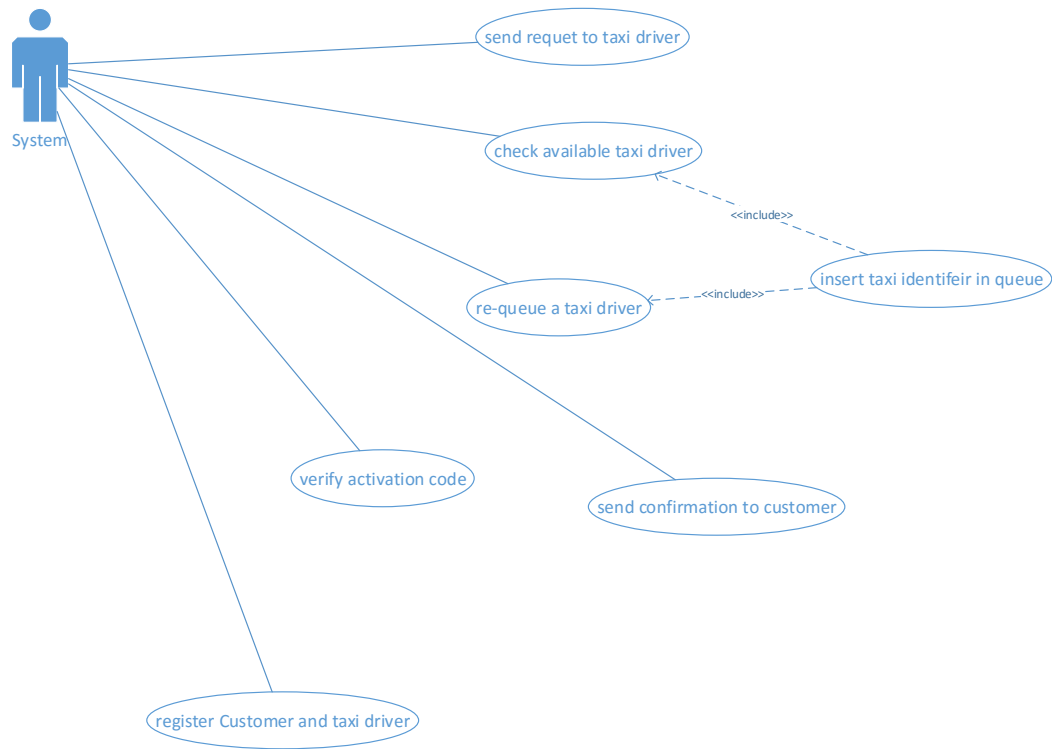
3.7 UML Modeling

3.7.1 use case diagram

On follow is shown the use case diagram that includes the main actors:

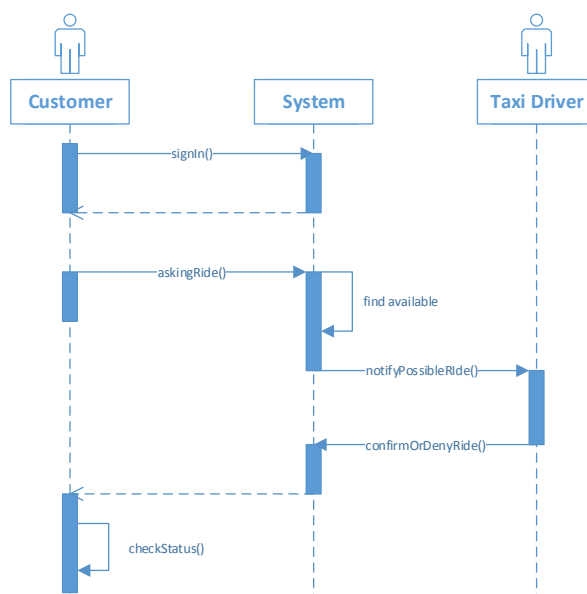
- Taxi Driver
- Customer
- System

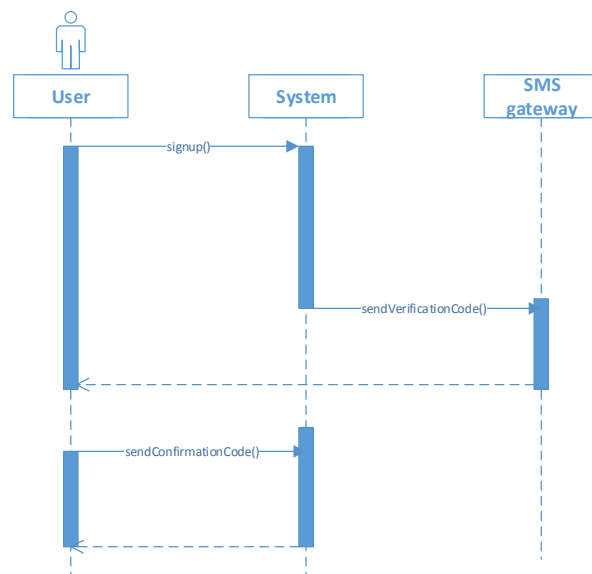




3.7.2 sequence diagram

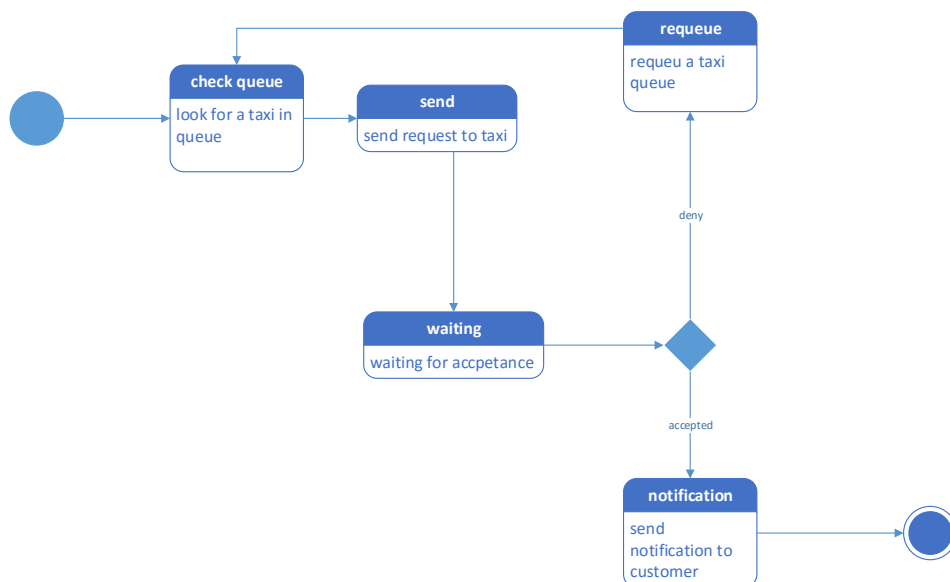
in the following we show the sequence diagrams:



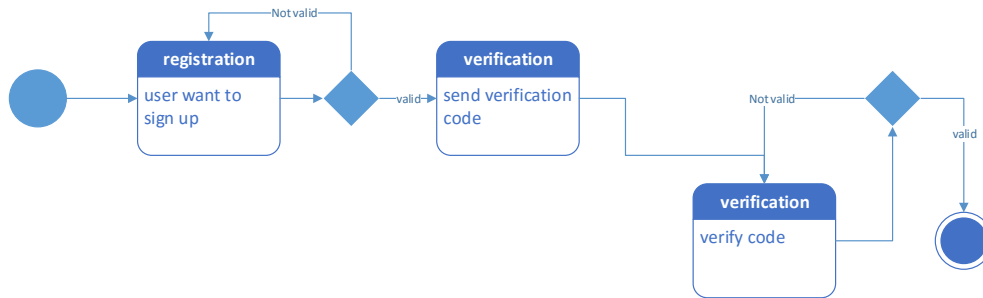


3.7.3 state diagram

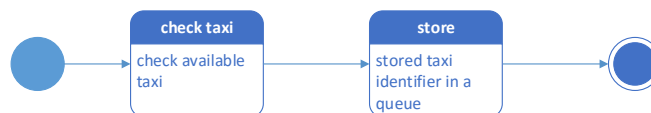
Asking for a ride:



User Registration:



Storing an available taxi:



Appendix

4.1 Used Tools

- Apple Pages to write and format this Document
- Visio Professional 2016 to create the UML state, sequence and case diagrams
- NinjaMock - free web tool for mobile app wireframes and website