POLITECNICO
MILANO 1863

MyTaxiService

Requirement Analysis and Specification Document

Davide Citterio, Lorenzo Cunial, Massimo Beccari

November 6, 2015

Contents

1 Introduction

1.1 Description of the given problem

We will project and develop myTaxiService, a service that allow everyone to use taxi in Municipality of LargeCity. People registered into myTaxiService can request a taxi in easy and speed way and taxi drivers can optimize their job. The system will be divided in two different interface: for the common people, the users, that use taxi and for taxi drivers. For users there will be the possibility to request a taxi via web application or via mobile application easily installed on every smartphone. The user interface provide some cool services like the possibility to reserve a taxi by specifying the origin, the destination of the ride and the start time and users can also decide to share taxi ride with other users that make same track. On the other side taxi drivers will have a new device that allow him to interact with the system. In the back-end part the system will be able to guarantee a fair management of taxi queues: the city jurisdiction will be divided in zones that are approximately about 2 km2, and for each zone the system automatically computes the distribution of taxis based on the GPS information it receives them.

1.2 Goals

- The user has to be able to sign up and log in with the mobile application and the web page. Both this two interfaces have to shown all the services provided by the system;
- The user has to be able to require a taxi specifying the starting address;
- The user must be able to use the service of taxi sharing: may be used when a reservation is made. The choice up to the user (through suitable interfaces). Also in this case the service must be accessible from the web app and mobile app;
- For each user asking for a request or a reservation of a taxi is guaranteed the punctuality;
- The system has to guarantee a fair management of taxi queues, dividing the city jurisdiction in several zones;
- The taxi driver has to be able to answer to the calls generated by the system after the users request;
- The administrator has to be able to change the price of the ride, deactivate some services and know the position of the taxi (only the taxi on service);
- The system has to allow user to login using Facebook account;

- The system has to allow user to receive a sms message about the confirmation of registration and for a request/reservation;
- The system has to allow the administrator to manage driver, like insert or delete them.

1.3 Domain Properties

The following properties belong to the analysed domain:

- Taxi locations are known by GPS signal;
- Taxi work within the jurisdiction of Municipality of LargeCity;
- It has to spend less than 10 minutes between when taxi drivers accept the call made by some user and the request of this one;
- Taxi drivers has to take up no more than 4 people simultaneously;
- The connection between the system and taxi drivers devices is made by UMTS technology: all the LargeCity jurisdiction is covered by UMTS signal so it guarantee a very fast connection.

1.4 Glossary

We want to make unequivocal some words that will be often used in our documentation of the project, so we are giving a precise definition of what will be the meaning of these words in the contest of this project.

- GUEST: is a person who hasn't signed up yet to MyTaxiService application. Guests have less power in the system than users, the only function they can use is the sign up;
- USER: is a person already registered in the system. Is the customer of the service;
- DRIVER: is the taxi driver;
- ADMINISTRATOR: is the only person enabled to use the programmatic interface;
- SHARING: is the service consists on reservation a ride sharing it with other users, if it is possible;
- RESERVATION: is the service consists on reserving the ride scheduling it for a specific time;
- REQUEST: is the service consists on request immediately a taxi;

- CALL: it occurs after the system processed a request/reservation made by users. It warnings the driver about a job he has to do;
- ZONE: is one of the areas belongs to the city jurisdiction;
- STATUS: is the current Status of a taxi;
- ADDITIONAL SERVICES: with this terms we identify all the services that the administrator is allowed to change, enable or disable;

1.5 Acronyms

- GPS: Global Positioning System;
- DB: DataBase;
- MTS system: My Taxy Service system;

1.6 Assumptions

We have to assume some facts due to ambiguity of the gave description. We assume that:

- Each zone is reachable in less than 10 minutes by at least one taxi;
- In the previous system users had only one kind of service: the request. This was made by telephone call;
- When a taxi responds to a call, it will reach the meeting location in the shortest possible time;
- Mobile devices of users don't guarantee a precise geolocation service, so we ask to user to insert manually his position (like name address and house number); All taxy drivers that works for the municipality has a device like smartphone or tablet running Android OS, in witch he can install applications.

1.7 Proposed system

We will implement an enterprise application that be composed of a server, which runs the business logic, generates dynamic web pages and access to a DB in which all system's information will be stored. Moreover the server side technology , there will be several clients that interact with the server using a web browser and mobile application. The mobile applications dedicated to users will be coded using Swift (for iphone device) and Java (for android). The application used by drivers in their devices will be coded in Java.

1.8 Identifying Stakeholders

Our "financial" stakeholder is the government of LargeCity who gave us the assignment and expects us to be able to finish the project within 6 months. They ask us to put the attention on two important things; simplify the access of passengers to the service and guarantee a fair management of taxi queues. Those elements involve another two indirect stakeholders. The first ones is the customers - users that have new services. The other ones are the taxi drivers.

1.9 Constraints

1.9.1 Regulatory policies

When a user submit a request that concern himself and other, he can't demand more than two taxi. Because the maximum number of passenger housed in one taxi is 4, the number of the maximum people that can be insert in the request must be less than 9.

1.9.2 Parallel operation

MTS must support parallel operations from different users when working with database and with all operation done by the user after connection.

1.9.3 Documents related

- Requirements and Analysis Specification Document (RASD);
- Design Document (DD).

1.10 API Interfaces

For the navigation system on the taxi drivers devices we have to use the Google Maps APIs(standard version). As well described on the website this API give access to any location (cover 80% of the Earth). Moreover the API calculates the best route based on the traffic condition that is frequently updated. For the login system of users, MTS system use Facebook Graph API. It is a secure authentication system that reduces the burden of login for our users, by enabling them to sign in with their Facebook account.

2 Actors Identifying

The actors of our system are:

- Guest: a guest is someone who hasn't registered yet and wants to use the taxi service, so he can only sign up. He can do the sign up via website or using our mobile application.
- User: a user is someone who has already signed up, so he is a customer of the taxi service. He can use our system through the web site or the mobile application. In particular, he can request a taxi immediately after having done the log in or make a reservation for a specific time.
- Driver: a driver is a person who uses our system to carry out its work. He uses a device on which our application is installed. He needs to login on the application in order to start working.
- Administrator: the administrator is the responsible of the taxi service. He can use our system through a specific web interface that allows him to see where the taxis are in a specific moment or perform actions like activate/deactivate a specific service, add or remove the drivers from the system or set the fees.

3 Requirements

Here follow the main features of our system, with the assumption that the domain properties previously described hold.

- Registration and login of a customer to the system
  - The system has to provide a sign up functionality.
  - The system has to provide a login function with the possibility of recovering a forgotten password.
- Making a taxi request
  - The system has to provide a function that allows a user to make an immediate request for a taxi; the user has to insert his position and the number of people in the request and will receive a confirmation sms in case of successful operation.
  - The user shouldn't be able to make a new request if there is another pending.
- Making a taxi reservation
  - The system has to provide a function that allows a user to make a reservation for a specific time; the user has to insert start place, start time, destination place and the number of people in the request and will receive a confirmation sms in case of successful operation.
  - The user should be able to make more than one reservation but at most six; all the pending reservation times must differ by thirty minutes of each other.
  - The system has to provide a function that shows a user all his reservations with relative information; the user should be also able to filter them by date and price.
  - The system has to make possible for user to modify or delete a reservation, only if the modifying/deleting time is more than ten minutes before the reservation start time.
  - The user should be able to enable the taxi sharing option, if he wants.
- Login and setting the status for the driver
  - The system has to provide a function that allows a driver to login and set his status on-line (waiting for a user request) or off-line.

- Accepting a user request
  - The system has to provide a function to notify a driver when there is a user request that he can serve; the notification includes the information about the start place.
  - The system has to make possible for a driver to accept or deny a request; if the driver doesn't answer the request after 2 minutes, the request is automatically denied.
  - The system has to provide a function that shows the driver the route toward the user destination.
- Shared ride notification and end of a ride
  - The system has to send a notification to a driver when there is a user ready to join his shared ride; the notification has to provide the start and destination places of the new passenger.
  - The system has to provide a function that calculates the fee for each passenger at the end of a ride.
  - The system has to tell the driver in which zone he has to go after ending a ride and show the route for that zone.
- Visualization of taxis in the map in the administrator web interface
  - The system has to provide a function that shows to an administrator a map with the current position of every online taxi and their current status (waiting, driving).
  - The administrator should also be able to see all the information about the current, past and scheduled rides.
- Managing the taxis
  - The system has to make possible for an administrator to set the hourly fee and the fixed day/night tax.
  - The system has to provide a function that allows an administrator to add and remove taxi drivers; in the adding operation the administrator has to insert the name, telephone number, license plate number and driving license number of the driver.
  - The system has to provide a function that allows an administrator to communicate with the taxi drivers.
  - The system has to provide a function that allows an administrator to cancel a ride (if it hasn't started yet).
- Managing additional services
  - The system has to provide a function to allow an administrator to enable/disable additional services (e.g. shared option).

3.1 Functional requirements

After having defined the main features of myTaxiService, we can find the functional requirements about every defined actor:

A Guest can only sign up.

A User can:

- Log in;
- Recover his password;
- Make a taxi request;
- Make a taxi reservation;
- Receive a confirmation sms;
- Modify a taxi reservation;
- Delete a taxi reservation;
- See all his reservations;
- Filter his reservation by date and pric.

A Driver can:

- Log in;
- Set his status;
- Accept or deny a request;
- See the route toward a destination;
- See the fee for a ride;
- See the route toward a zone;
- Being notified when a user wants to join a shared ride;
- Receive an administrator's message.

An Administrator can:

- See the current position of every taxi;
- See the current status of every on-line taxi (driving, waiting);
- See the information about the current/past/scheduled rides;

- Add and remove taxi drivers from the system;
- Set the hourly fee;
- Set the fixed daily fee;
- Set the fixed nightly fee;
- Cancel a driver;
- Enable/disable additional services.

3.2 Non-functional requirements

3.2.1 Performance requirements

The system has to guarantee a good level of usability so the response time has to be close to zero. However the performances are strictly related to the speed of user's internet connection.

3.2.2 Software system attributes

MTS System has to be reachable anytime 24/24h 7/7 days. To achieve this goal is necessary to use a dedicated server hosted by professional company. This solution exempt Municipality to buy and maintain a dedicated server and its infrastructures and it also offers improvements in terms of security, backup of data and availability of the system. This solution is also good in terms of scalability, in case the system enlarge its traffic is possible to switch up to more power hardware without big outgoings. In terms of maintainability, due also to the integration of some API, all the application code has to be well documented.

3.2.3 Security

User-side MTS system implements a login authentication to protect the info of the user. The authentication login asks to user the email address and the password, random generated by the system and sent by SMS to him after the registration. User's password has to be saveb using a hashing mechanism. After the first login, user is allowed to change his password. The password length has to be at least 8, including at least one number and one capital letter. Special character are not allowed.

Application-side On the application side, the system has to prevent any kind of SQL injection attack. System has to use HTTP protocol to communicate with user and driver. It might be better use the HTTPS protocol, to improve security. This solution could be more expensive.

Driver-side Drivers has to do login. Login require the id and the password of the driver. Id and password are given by the administrator when this one insert the driver into the application.

Administrator-side Administrator can access to his interface through a login that requires a password given by the developers and it cannot be modified.

3.2.4 Hardware interfaces

The MTS System has to communicate via internet and via SMS message. For this reason the server that hold the system must have a network interface that allows TCP communication protocol and a integrated board that support at least the GSM technology to allow the sending of SMS message. For this functionality, instead of hardware interface, it could be possible to use a external web service that allow this.

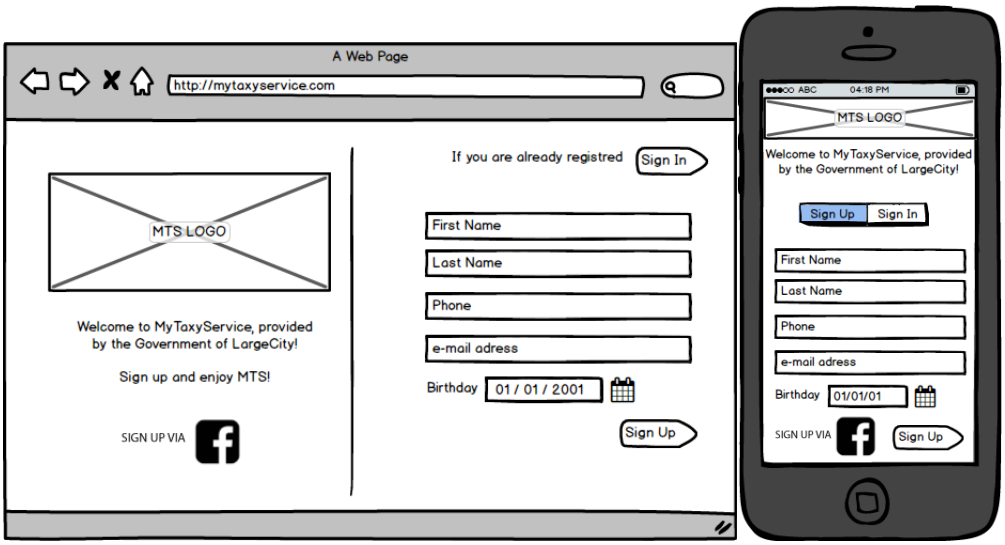
3.3 Mockup and users interface

According to the presence of different actors the system has several kind of interface, each one dedicated to specific actor.

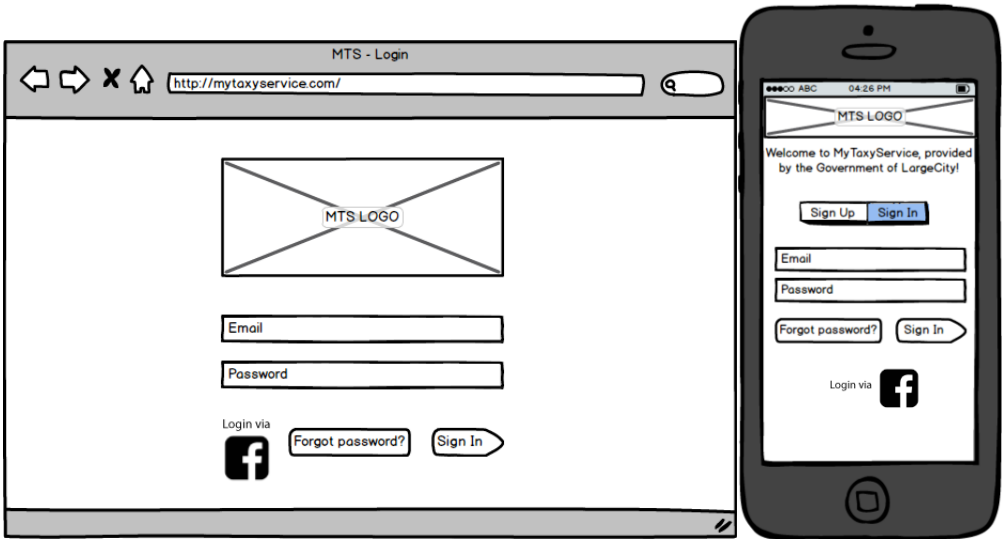
3.3.1 User Interfaces

The user has the possibility to access the system in two ways: via mobile app or via web app. Each one has the same features, only the design template change.

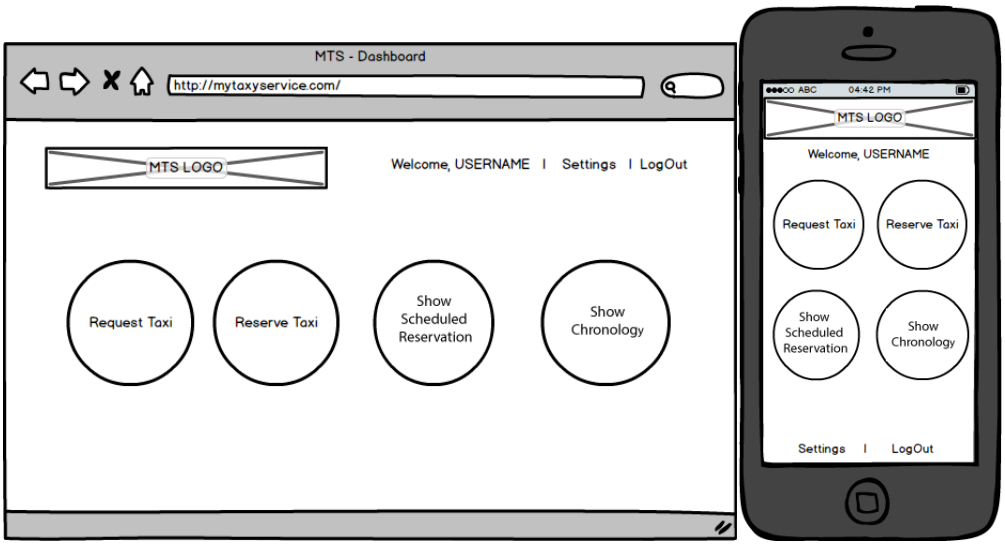
HOMEPAGE for a guest (not logged in) If a customer access to website or open the mobile app, he can do login or sign up, by default sign-up option is enabled. He can also sign up via Facebook.



LOGIN page If the customer is already registered, he can do login. If he made sign up via Facebook he has to do login via Facebook too.

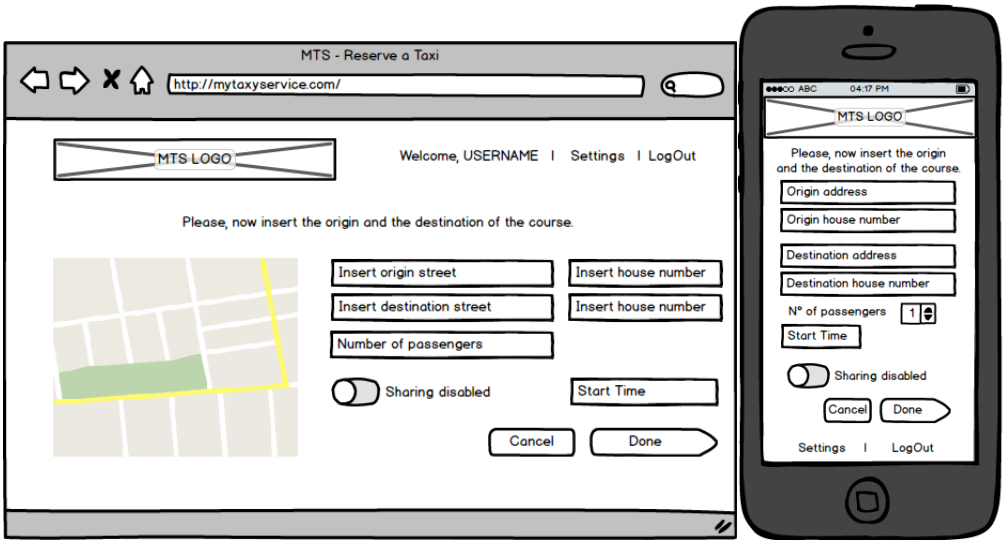


HOMEPAGE for authenticated user After the login, a user can choose between two services, call taxi now or schedule a ride. He also opens the scheduled reservation page and the chronology of his travels.

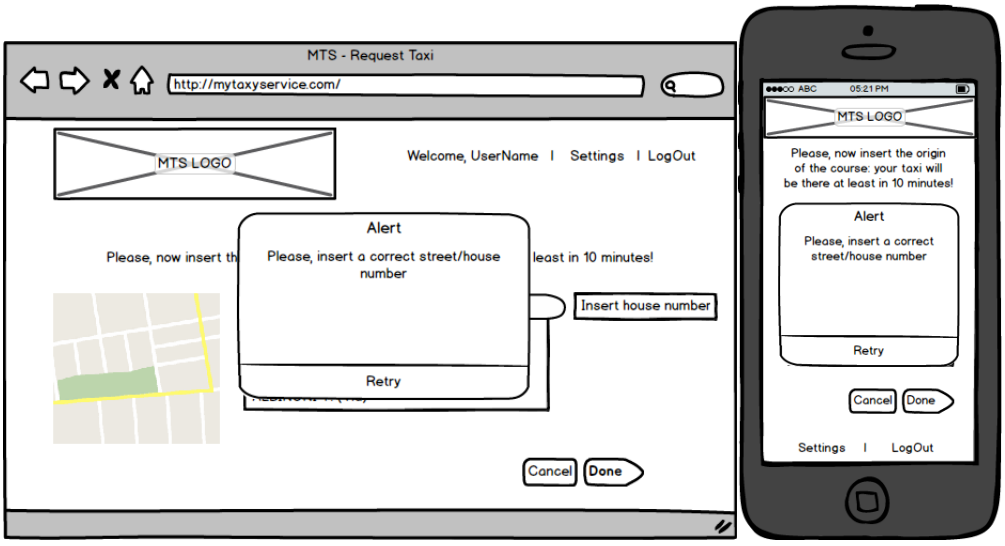


TAXI REQUEST page If a user chooses the "request taxi" option, he has to entry the address where taxi will waiting for him and how many persons has to take.

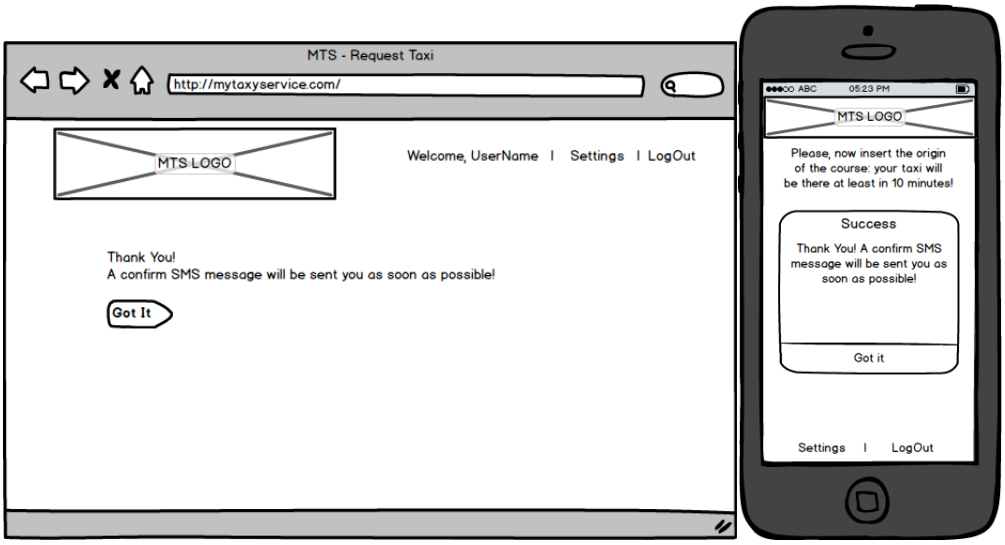
TAXI RESERVATION page If a user chooses the "make a reservation" option he has to entry the origin and destination addresses where taxi has to bring him, the number of passengers and the start time. He can choose the "sharing option", by default it is disabled. If the customer switches "sharing option" on, the system automatically computes sharing course and the customer hasn't to do anything.



ERROR page When customer insert addresses that are not registered in system, a message will notify that.



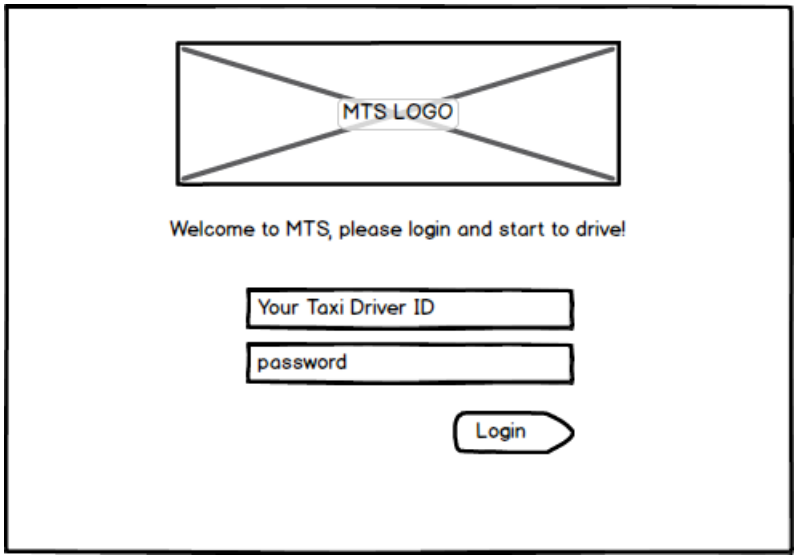
CONFIRMATION screen After submission of the request, the system shows the confermation alert.



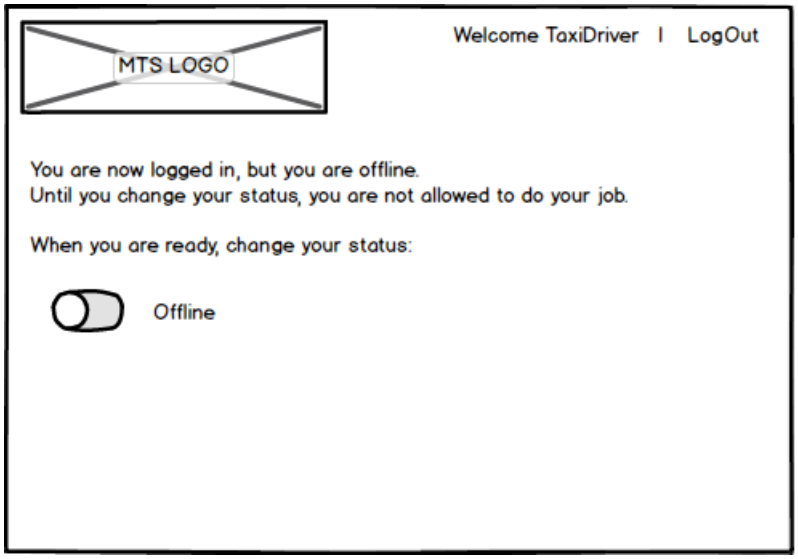
3.3.2 Taxi-driver interfaces

Taxi drivers are allowed to access the system by their device. It has the below interfaces.

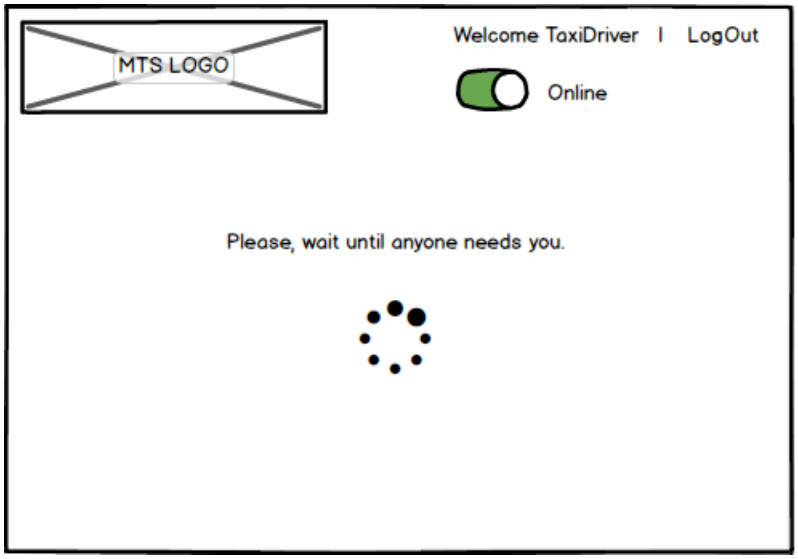
LOGIN page When a taxi driver switch on the app on his device, he system asks to log in, using his credentials.



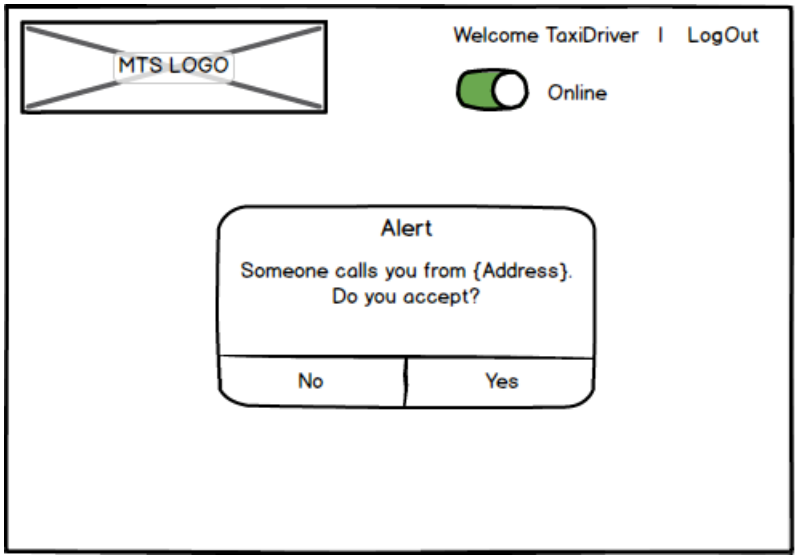
ONLINE/OFFLINE page After the login, the driver has to switch on the "online" button. In this way, the system can assign him some courses.



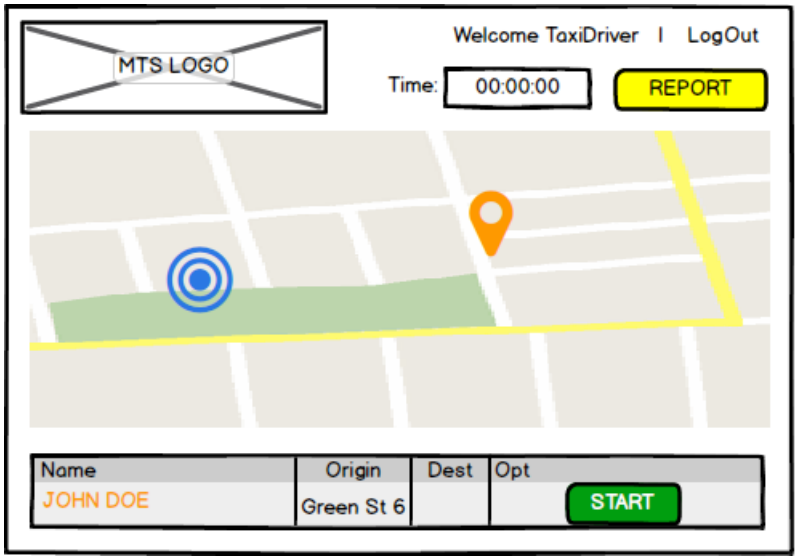
WAITING-FOR-A-CALL page Until a user makes a request, a waiting-for screen appears on the device.



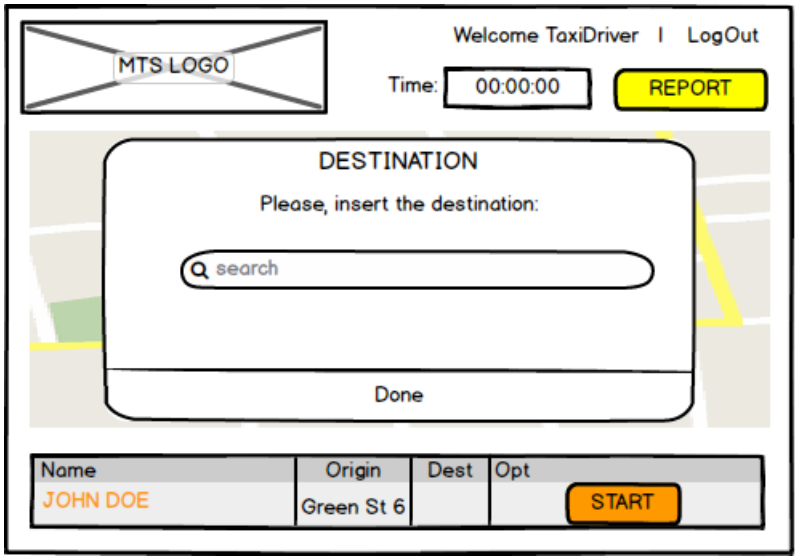
ACCEPTANCE SCREEN When a user makes a request, system asks to taxi driver if he want to accept that.



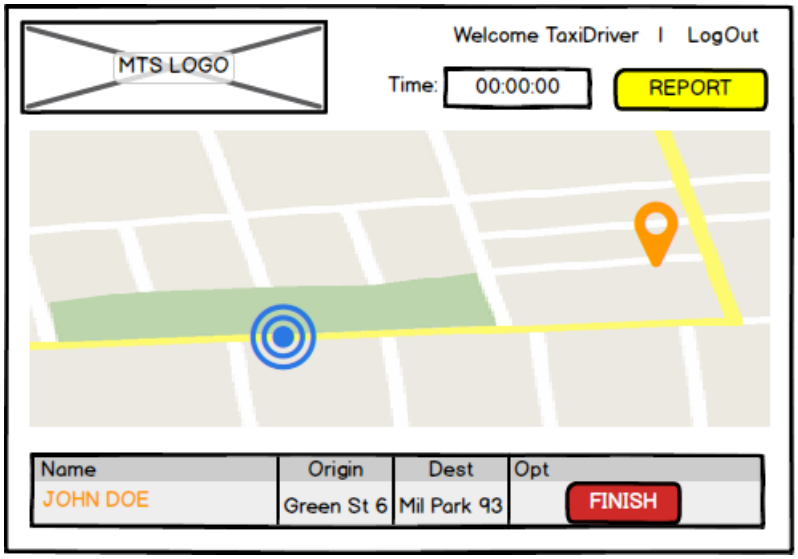
MAP ROUTE to the passenger page (REQUEST TAXI CASE) When taxi driver accepts the request, on the device appears the map with the indications of the start place. When the driver arrives at the origin place and he finds the user he taps the start button, otherwise if there aren't users he taps the report button.



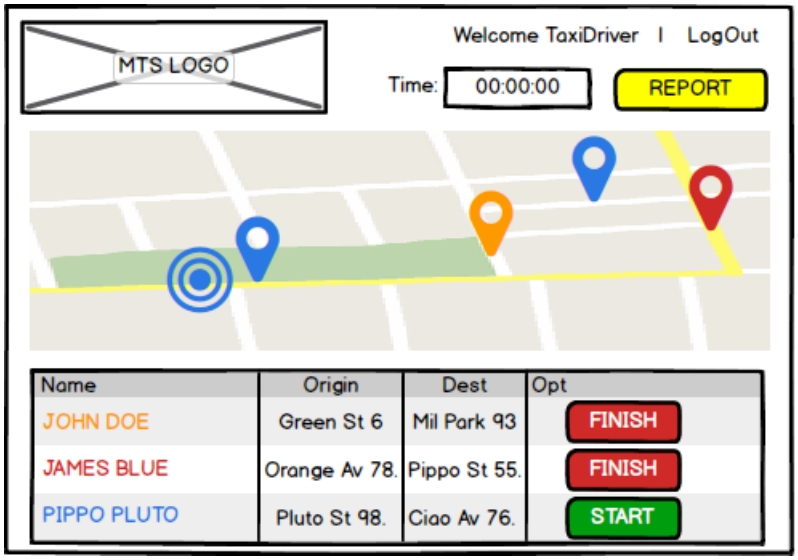
INSERT DESTINATION PAGE (REQUEST TAXI CASE) If the call is for a request, then after the start button is pressed system asks to driver to insert the destination.



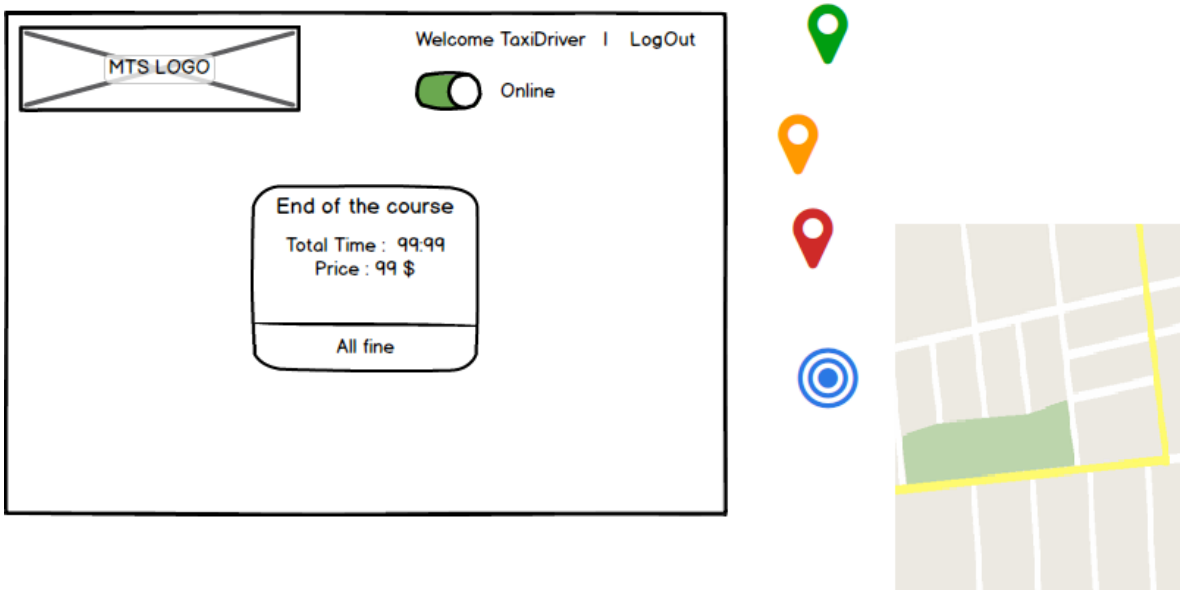
MAP ROUTE page After the tap of start button, on the screen appears the map of the route. When taxi arrives at destination, the driver taps the finish button. This screen is also shown immediately after the acceptance alert if the call is for a reservation, because the system already knows the destination.



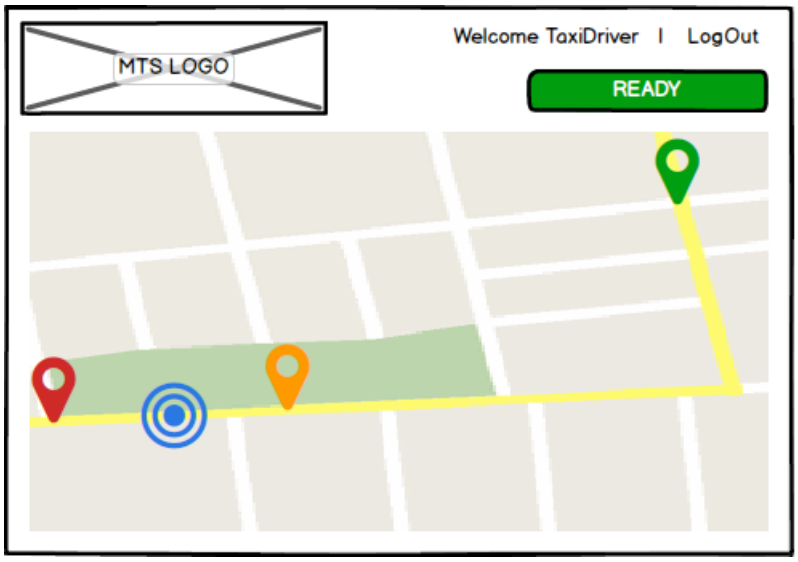
MAP page (SHARING CASE) When arrives a sharing call, after the acceptance, on the screen appears the map and the list of user. When taxi takes on board one user, driver has to tap the start button. When a user is arrived on destination driver has to tap the finish button.



END OF COURSE page When the taxi driver tap the finish button system shows him the total time and price.



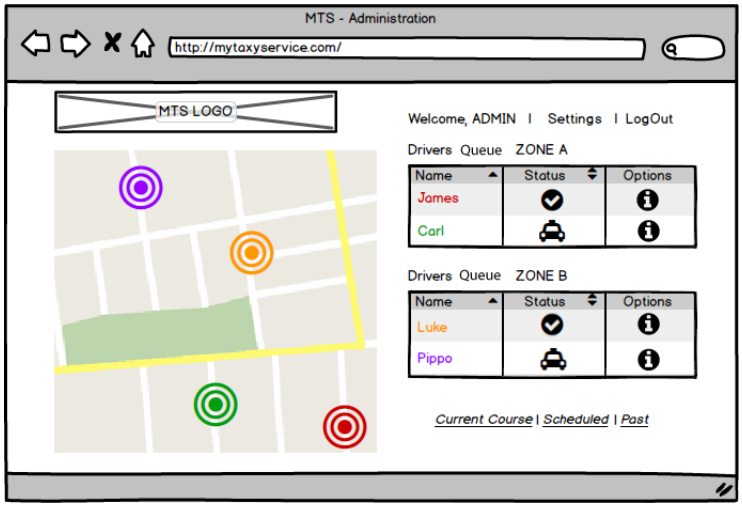
COME BACK page After the payment (when driver taps All Fine button), system shows to driver the map of the new destination of the waiting-place. When the driver arrives at destination he taps the ready button and the waiting for a call page appears.



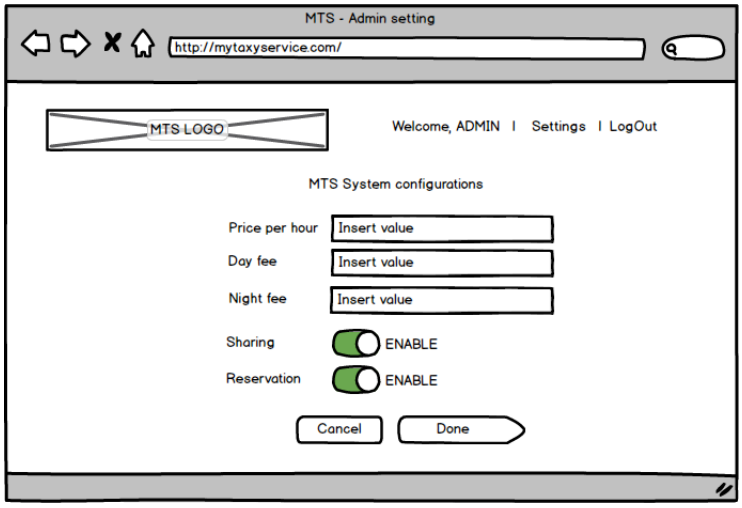
3.3.3 Administration interfaces

The admin of the system has the possibility to see in real time the distribution of taxi on the town.

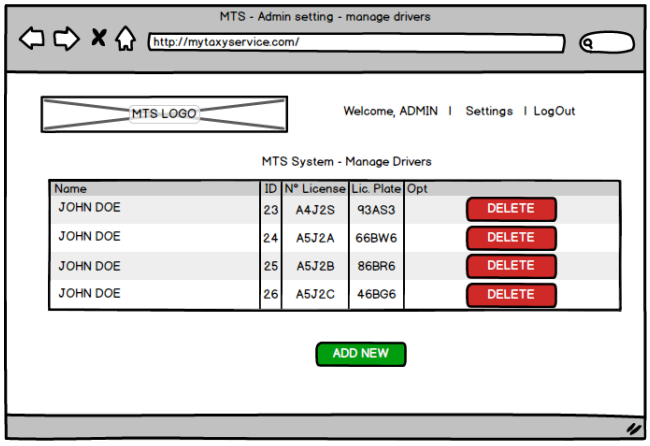
MAIN-SCREEN There is the map of the town and system shows the position of each taxi that are logged in and online. The System shows the list of all taxi online, listed by zone with relative status, and the button for having the info of current ride and position. There is also links to the list of real-time course, the scheduled ones and for the past ones

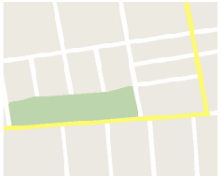


SETTING PAGE Here the administrator is allowed to change some parameters used by the system.



MANAGE DRIVERS Here the administrator can see all the drivers inserted in MTS system. He can delete or add a driver.





4 Scenarios Identifying

1. Louisa is usual to travel in LargeCity in her free time. She usually takes taxi because they are very fast and cheap. She reads on the newspaper that the Municipality has released a new application that allow users to call taxi in a easy way. So she goes to her mobile app market and she downloads that app. After she has opened it, the app asks her to sign up, so she compiles the form and submit it. In a few minutes she receives a e-mail with the confirmation of the registration and a SMS message with the new password for the access.

2. Louisa has just finished her shopping in downtown and she wants to go home. As usual, she wants to take a taxi. So she opens the MTS app that she is already downloaded, she does the login and tap the request for a taxi, specifying the address where taxi has to take her and how many people taxi has to take. After the submission of her request a SMS message arrives at her mobile with the confirm that the taxi number 9098 will be there in less than ten minutes.

3. Erik arrives with his 6 friends at LargeCity train station. They have to go to the hotel and they decide to take a taxi. So he makes the login on MTS mobile app and tap the request taxi button. He insert the station address and the number of the components of the group. He submits the form and a SMS confirm the request. Than the system allocates two taxi for the request.

4. Lukas has to go to the airport for a flight this afternoon. He is already registered into the MTS system and he wants to schedule a taxi that bring him to the airport. So he opens the browser and goes to the MTS website, he makes the login and he taps the "make a reservation" button. He inserts the origin and destination of the course, the start time and the number of passengers. He also taps the sharing option because he wants to save money, then he submits the form and a confirm alert appears, he also receives a SMS message with the confirm. Few minutes later, Daniel opens his MTS mobile app and makes a reservation for a taxi in shared mode, he inserts origin and destination addresses that are near the Lukas's one. So he submits form and as Lukas he sees the confirm-reservation alert and SMS When the system registered the second request automatically computes that is possible to make a single course for both of them. Ten minutes before the course it reserves the first taxi in queue in the correspondent zone.

5. John is a taxi driver of LargeCity. He starts to work at 8am, so opens the MTS system app installed on his device. He does the login using the id and password gave by the administrator of the system and he switches on the online button. Now he wait for a call.

6. John is logged in the MTS and he is waiting for a call. An alert appears on his device, someone call him not far from his position, he has the possibility to accept o refuse the call. He accept and on the device the map of the route is shown. The first destination is the customer waiting point, he takes on his customer, he asked to user about the destination, he inserts the destination on the device and he goes on the final destination reported on the map. When he arrives at destination he taps the "end of the course" button and the total price of the course is shown. When the customer pays the service he taps a button and system shows him a new route where there is the place where he has to wait for a new call. When he arrives at the new destination he taps the ready button and the waiting-for-call screen appears.

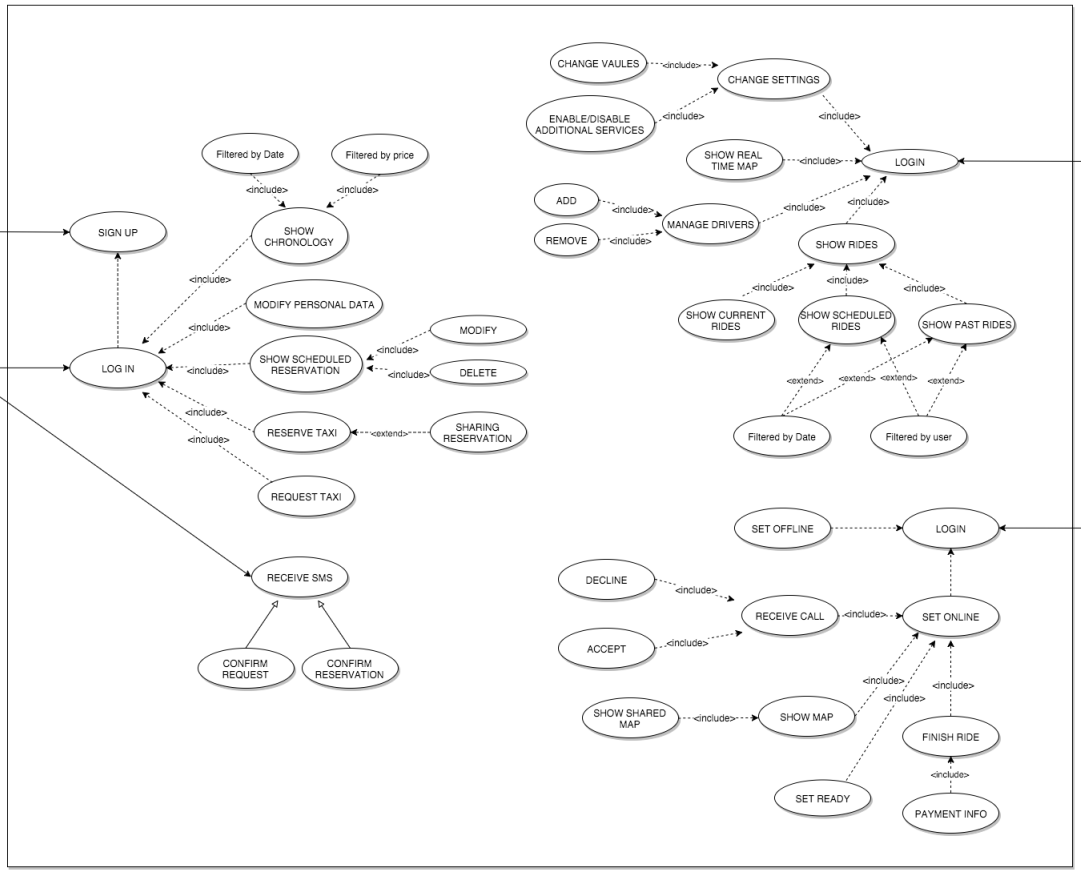
7. John is logged in the MTS and he is waiting for a call. An alert appears on his device, someone call him not far from his position, he has the possibility to accept o refuse the call. He has too few fuel and he has to refuse the call.

8. John is logged in the MTS and he is waiting for a call. An alert appears on his device, someone call him not far from his position, he has the possibility to accept o refuse the call. He accept and on the device the map of the route is shown. The first destination is the customer waiting point. When he arrives on place he doesn't find anyone, so he taps the report button and notifies the problem. The system cancelled the course and shows him the wait-for-call.

9. Linda is the taxi-service boss and she can access to the admin pages of the MTS system from her office PC. Today there is high traffic and taxi takes long time to do their job. To ensure that every call is served in less than 10 minute she has to disable the sharing option. So she makes login and goes in setting page, she swap the correspondent button and she click confirm. Now the change is active.

5 UML Models

5.1 Usecase



5.2 UseCase description

We can derive some use cases from the scenarios identified in the previous paragraph:

5.2.1 USER SIGN UP

Name	Sign Up
Actors	Guests
Entry Conditions	The Guest isn't registered into the system
Flow Events	• The guest access to MTS via mobile app or web app • If he has a Facebook account he can signup via Facebook, in this case after the authorization he has to insert the phone number and than he has access to the homepage. • If he want to register without Facebook he has to compile the form with the following info: - First Name - Last Name - Phone number - Email address - Birthday • He clicks Sign Up bottom • The system show him a successful operation message and it send him an email with all data inserted, a password random generated by the system and the instruction about the first login.
Exit Conditions	Registration successfully done.
Exceptions	Guest is already registered (the email address is already in use). He doesn't compile all fields of the registration form. He inserts a not valid birthday. He inserts a not valid mail address.

5.2.2 USER LOG IN

Name	Log in
Actors	User
Entry Conditions	The User is already registered.
Flow Events	• The user access to MTS via mobile app or web app • He clicks the login button • If he has registered via Facebook he tap the" login via Facebook button" • Otherwise he inserts email address and password • He clicks Login bottom
Exit Conditions	The system shows him the homepage
Exceptions	He inserts wrong username/password

5.2.3 RESERVE TAXI

Name	Reserve a Taxi (no sharing)
Actors	User
Entry Conditions	The user is already logged in and he is on the homepage
Flow Events	• User clicks "reserve taxi" button • He compiles the form with the following info: - Origin address - Origin house number - Destination address - Destination address number - Number of passengers - Date - Start time • He click done button
Exit Conditions	System accept the reservation. It shows him a confirmation screen and it sends him a SMS message about the confirm.
Exceptions	User insert a not valid address. User insert a not valid date/time (according to reservation time limits)

5.2.4 RESERVE WITH SHARING

Name	Reserve a Taxi with sharing
Actors	User
Entry Conditions	The user is already logged in and he is on the homepage
Flow Events	• User clicks "reserve taxi" button • He compiles the form with the following info: - Origin address - Origin house number - Destination address - Destination address number - Number of passengers - Date - Start time • He switched on the sharing button • He click done button
Exit Conditions	System accept the reservation. It shows him a confirmation screen and it sends him a SMS message about the confirm.
Exceptions	User insert a not valid address. User insert a not valid date/time (according to reservation time limits) User insert a not valid number o passengers (according to specifications)

5.2.5 REQUEST TAXI

Name	Request a Taxi
Actors	User
Entry Conditions	The user is already logged in and he is on the homepage
Flow Events	• User clicks "request taxi" button • He compiles the form with the following info: - Origin address - Origin house number - Number of passengers • He click done button
Exit Conditions	System accept the request. It shows him a confirmation screen and it sends him a SMS message about the confirm.
Exceptions	User insert a not valid address. User insert a not valid number o passengers (according to specifications)

5.2.6 SHOW SCHEDULED RESERVATION

Name	Show scheduled reservation
Actors	User
Entry Conditions	The user is already logged in and he is on the homepage
Flow Events	• User clicks "Show scheduled reservation" button • If he has made reservation and if he is allowed according to specifications he can delete or modify it. • If he taps modify button he can modify the start time and date • If he taps delete button he deletes the scheduled ride. • Only reservation that respect the specification parameters can be modified/cancelled, the other are shown without any possibility to change them.
Exit Conditions	If user modify the star time/date then system re-scheduled the ride, else if he taps delete, than the system delete the scheduled ride. After the change/delete a SMS message is send to the user to confirm the action.
Exceptions	User insert a not valid time/date.

5.2.7 SHOW CHRONOLOGY

Name	Show Chronology
Actors	User
Entry Conditions	The user is already logged in and he is on the homepage
Flow Events	• User clicks on chronology button • He can see all the request and reservation he has made, filtering them by date or price paid.
Exit Conditions	None
Exceptions	None

5.2.8 DRIVER LOGIN

Name	Driver Login
Actors	Driver
Entry Conditions	Driver switch on his device
Flow Events	• Driver inserts his ID • He inserts the password • He taps login button
Exit Conditions	System shows him the online/offline page
Exceptions	Driver insert a not valid id. Driver insert a not valid password

5.2.9 RECEIVE CALL

Name	RECEIVE A CALL
Actors	Driver
Entry Conditions	The driver is already logged in and online. On the device appears the alert.
Flow Events	• Driver sees the alert • He tap the confirm/denied button
Exit Conditions	After he presses the confirm button on the device appears the map After he pressed the denied button on the device appears the waiting for a call screen
Exceptions	If he doesn't tap the button within 2 minutes, the system automatically decline the call and ask for some other driver.

5.2.10 SHOW MAP

Name	SHOW MAP
Actors	Driver
Entry Conditions	The driver has already accepted a call
Flow Events	• If the call is for a request so on the device appears the map with the route the drive has to follow. The destination is the starting point where take passenger. When he arrives and find the passenger he tap the Start button, he insert the destination street (that passenger has told him) and on the device appears the map with route to destination. When taxi arrives at destination, the driver tap the finish button and a alert with total time and price is shown. If he doesn't find anyone at the origin street he taps the report button and the ride is cancelled. After that a new map is shown with the new waiting for a call destination that driver has to reach. • If the call is for a reservation, system already known the destination so on the screen appears both the origin and destination street. When taxi arrives at destination, the driver taps the finish button and a alert with total time and price is shown. • If the call is for a sharing reservation on the screen appears the map and the list of user. System shows the route to take all users, when a user is taken on board driver has to tap the start button, when a user arrives at his own destination, taxi has to tap finish button and an alert with the relative price appears.
Exit Conditions	After the payment of the user (after the last user in sharing case) the system shows a new map with the new waiting for a call destination that driver has to reach.
Exceptions	None.

5.2.11 SET READY

Name	SET READY
Actors	Driver
Entry Conditions	The driver has already logged in and he has just finished a ride. On the screen appears the alert with price and time of the course.
Flow Events	• Driver tap done button • On the screen appears a map with a new destination and the route about where he has to go. • When driver arrives at destination he tap the ready button
Exit Conditions	After the press of ready button the waiting for a call screen appears.
Exceptions	None.

5.2.12 ADMINISTRATOR LOGIN

Name	Administrator Login
Actors	Administrator
Entry Conditions	Administrator opens the MTS Admin page
Flow Events	• He insert id • He insert password • He clicks login button
Exit Conditions	System shows him the real-time map
Exceptions	Administrator insert a not valid id. Administrator insert a not valid password

5.2.13 CHANGE SETTINGS

Name	CHANGE SETTINGS
Actors	Administrator
Entry Conditions	Administrator has already made login
Flow Events	• Administrator clicks settings button • He changes one or more values among the ones available • He switch on/off the additional services buttons • He clicks on done button
Exit Conditions	System shows him the same page with new values.
Exceptions	none

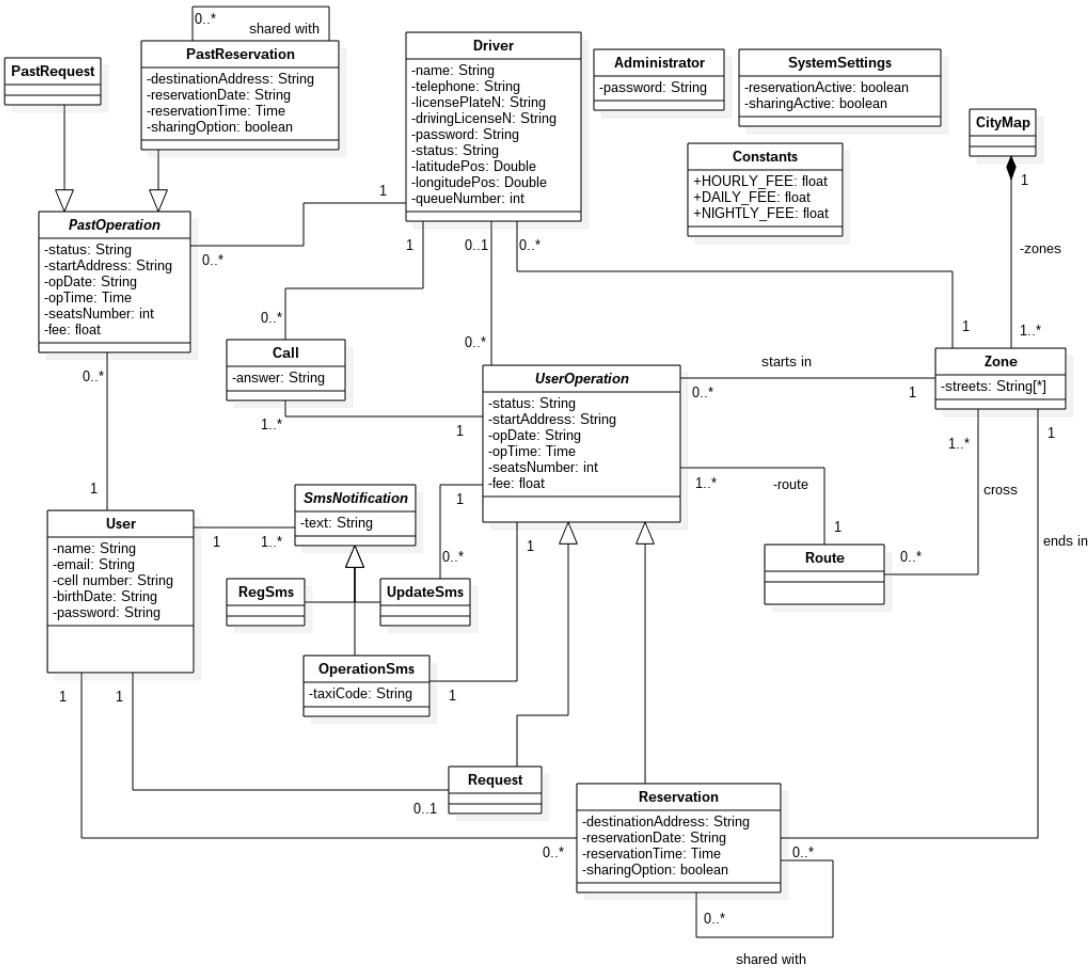
5.2.14 SHOW RIDES

Name	SHOW RIDES
Actors	Administrator
Entry Conditions	Administrator has already made login
Flow Events	• Administrator can visualize 3 kind of list: current rides, scheduled rides and past rides • If he clicks on current rides he sees a list of all current rides, with all info about driver, user (or users if share), origin and destination. • If he clicks on scheduled rides he can see the scheduled reservation with info of users, if it is sharing, starting time and the driver if it is already allocated. • If he clicks on past rides he can see all rides made by drivers with all info and price.
Exit Conditions	none
Exceptions	none

5.2.15 MANAGE DRIVERS

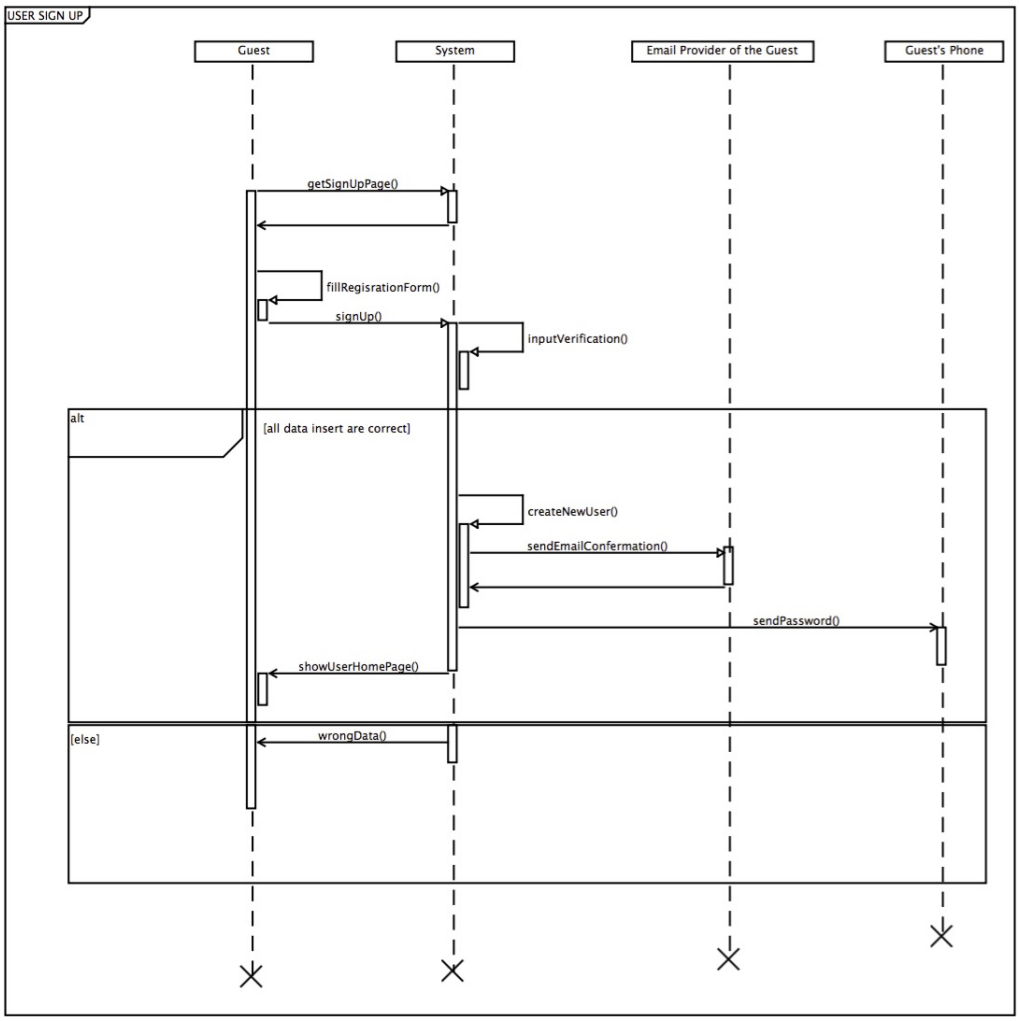
Name	MANAGE DRIVERS
Actors	Administrator
Entry Conditions	Administrator has already made login
Flow Events	• Administrator clicks on manage drivers button • System shows him the list of all drivers. • He can delete someone or he can add one driver • If he wants to add a new driver he has to click on add button, he compiles all fields of form (name, mobile phone number, number of license, and car plate. Then he clicks on add button and the system shows him a confirmation page with id number and password that administrator has to communicate to driver. • If he wants to remove someone he has to click the x button next to the selected name and system automatically removes it.
Exit Conditions	If there is a new driver: system shows the new id and password of the driver. If there is a removing, system delete driver and shows the new list without the selected driver.
Exceptions	Not all of the required fields are compiled.

5.3 Class Diagram

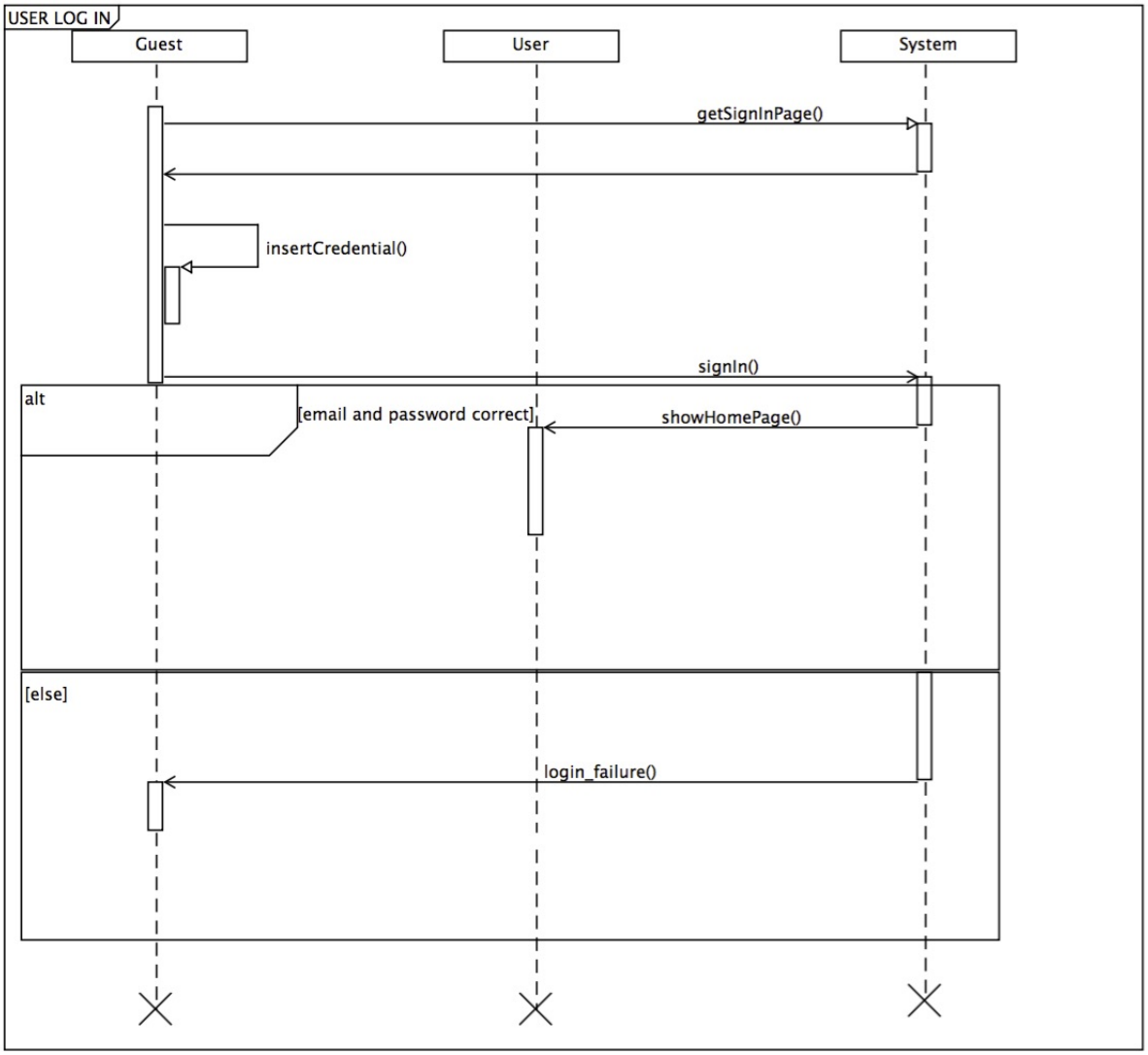


5.4 Sequence Diagram

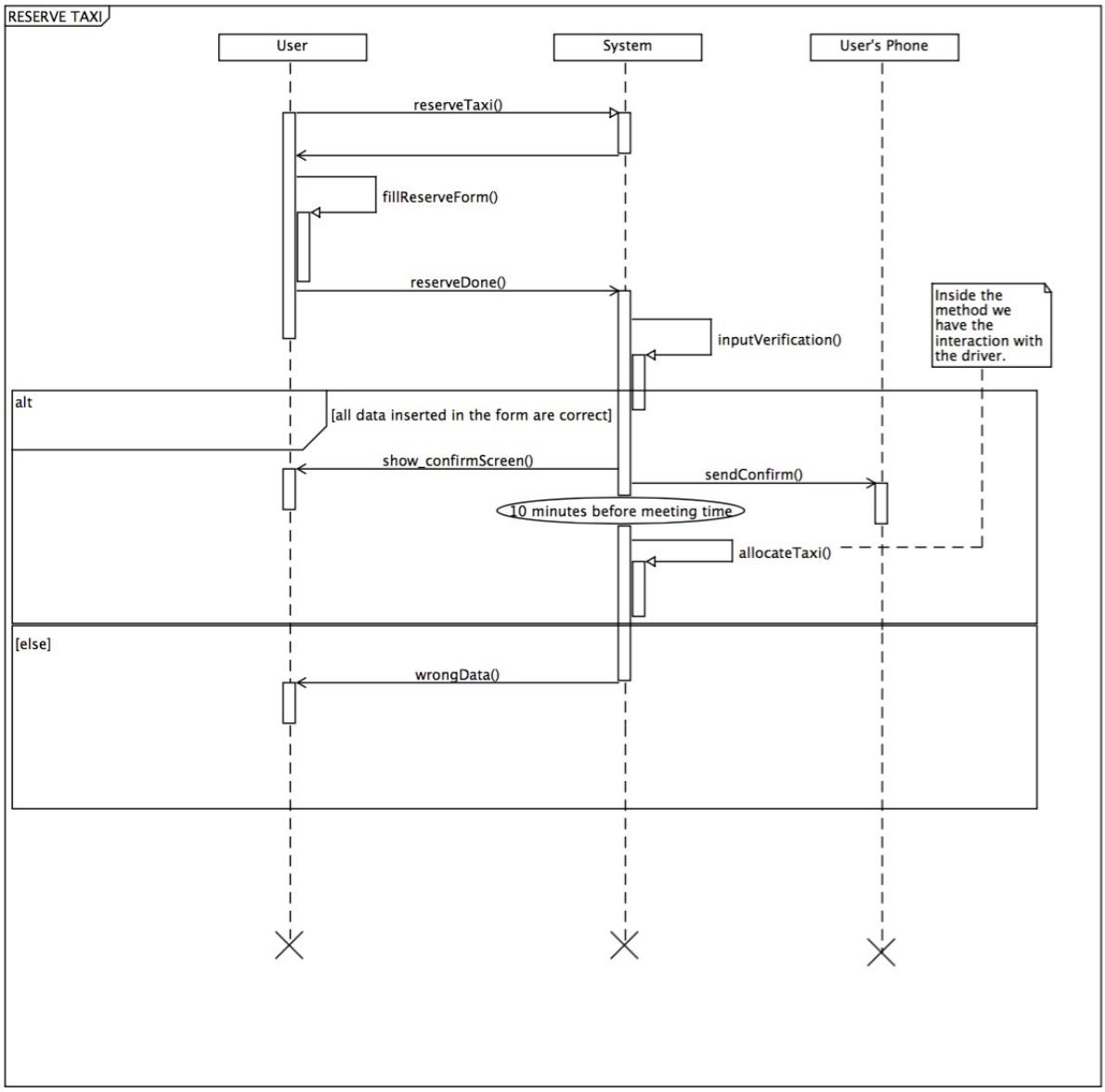
5.4.1 User Signup



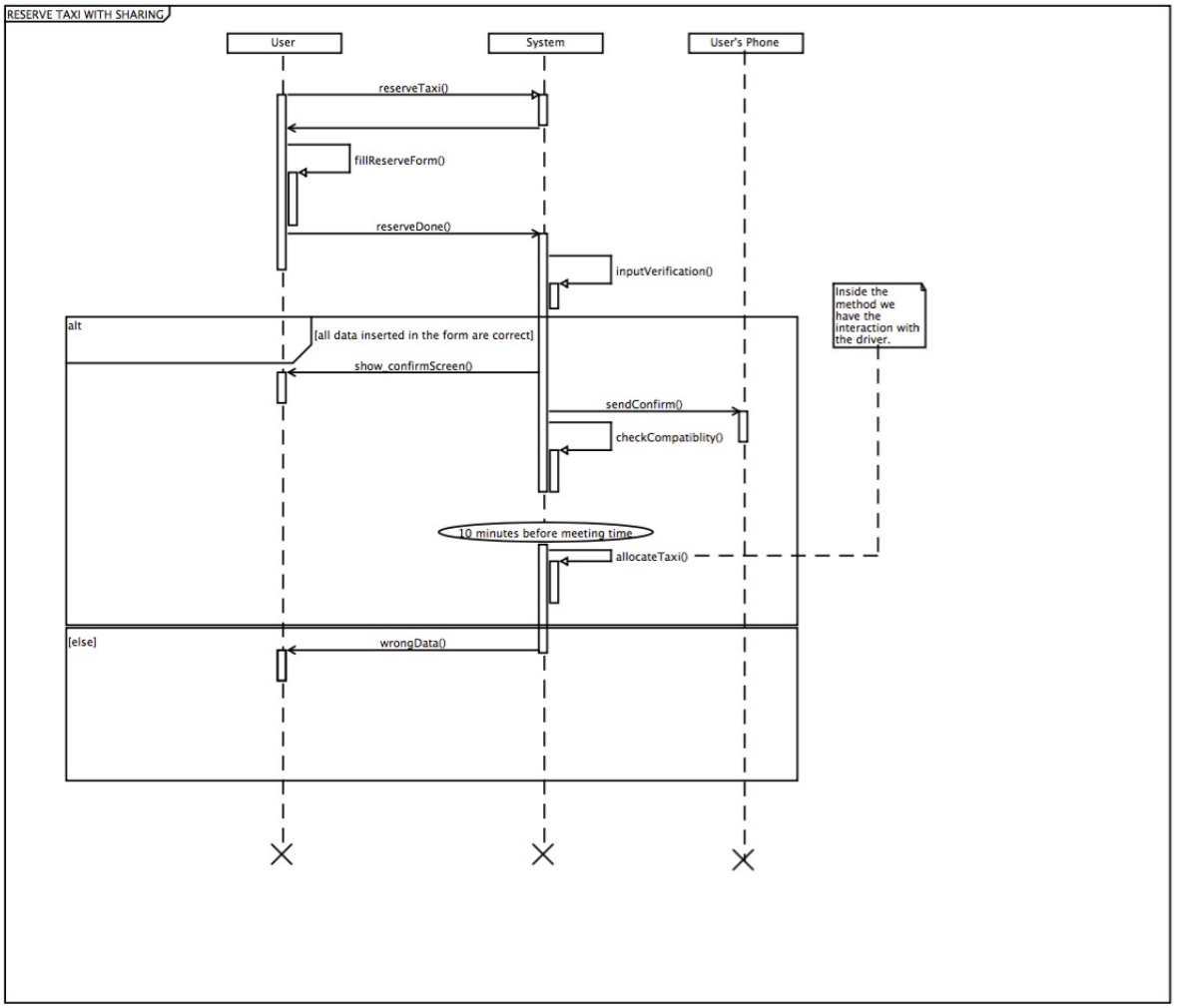
5.4.2 User Login



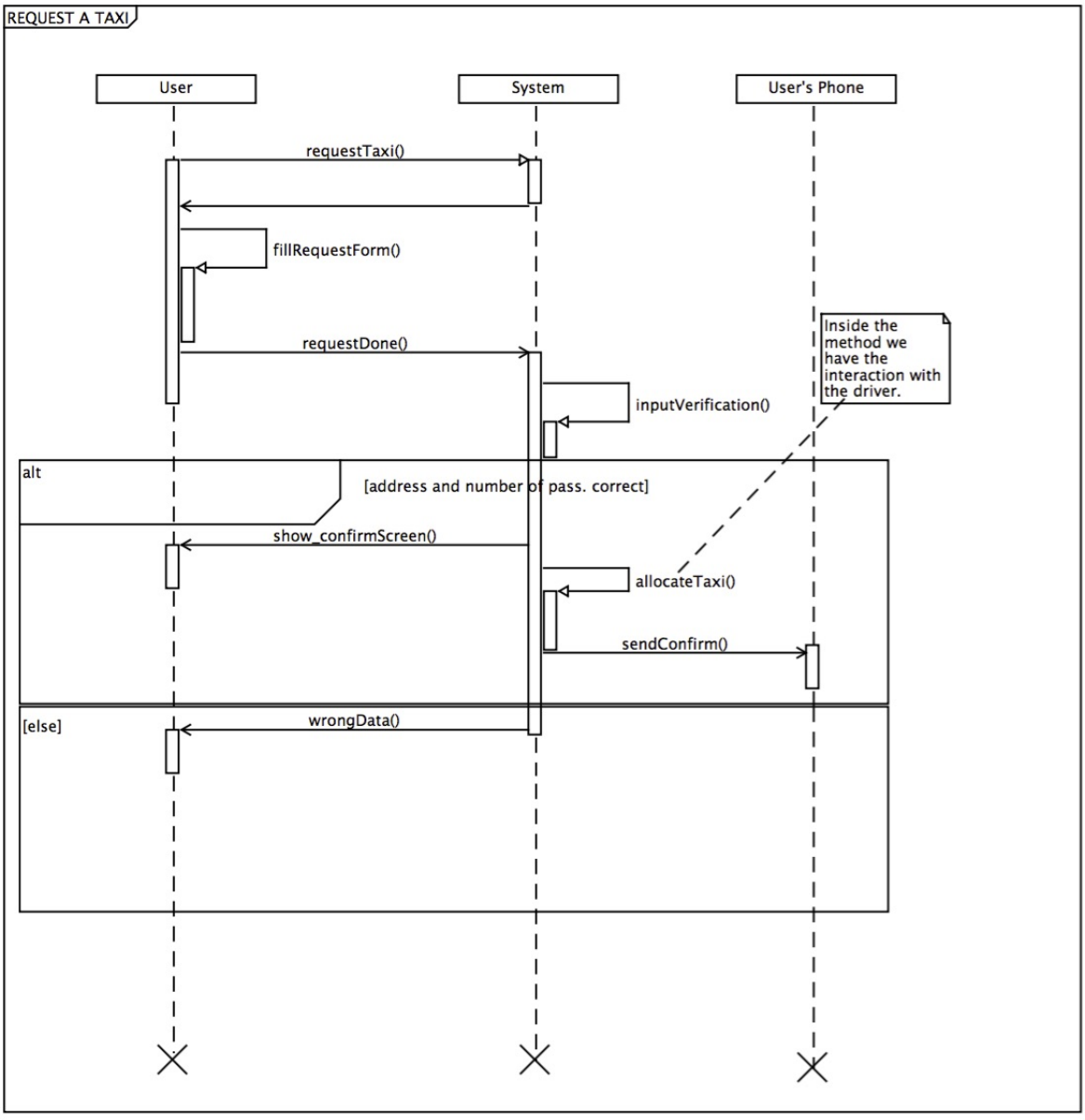
5.4.3 Reserve a Taxi



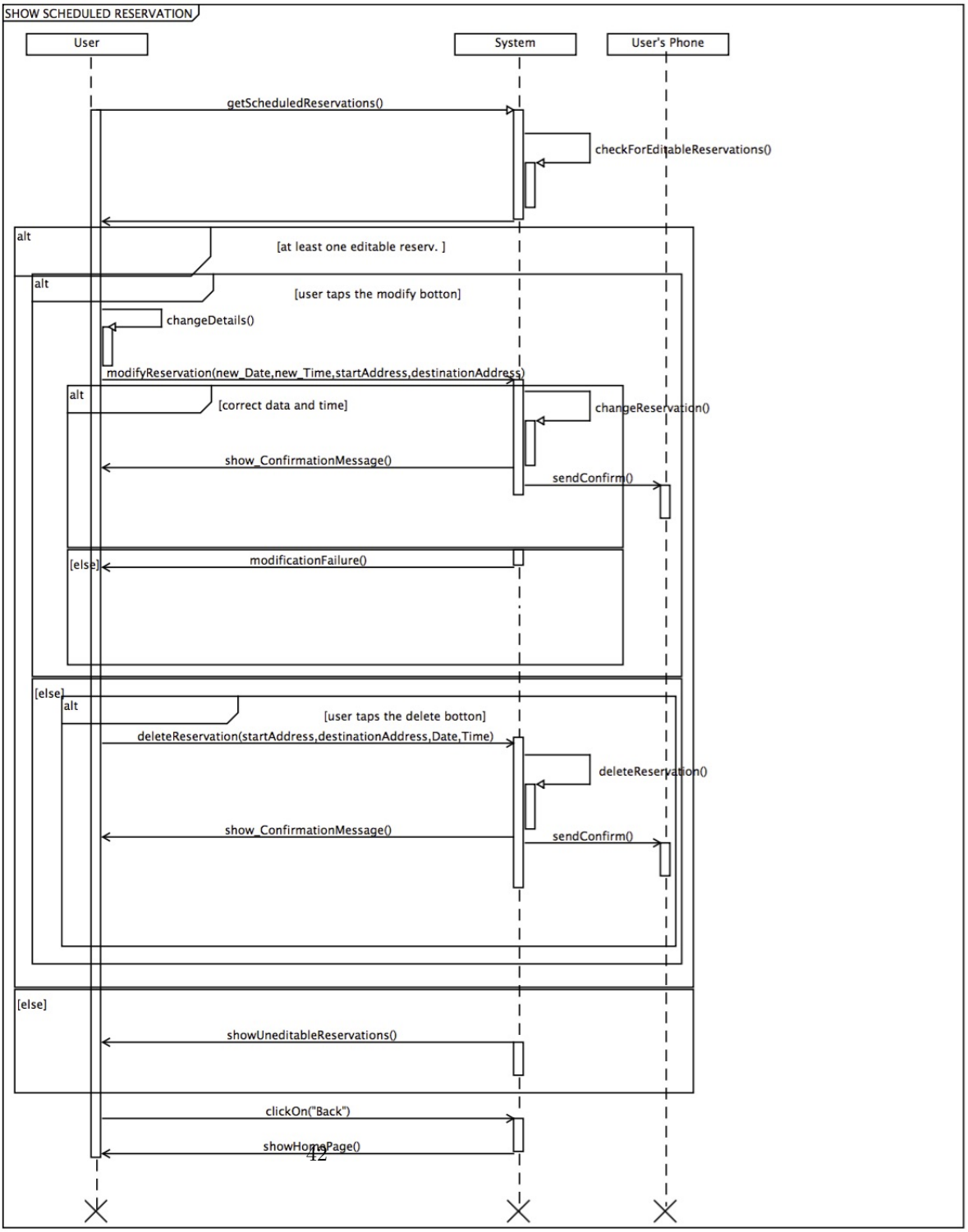
5.4.4 Reserve a Taxi with Sharing



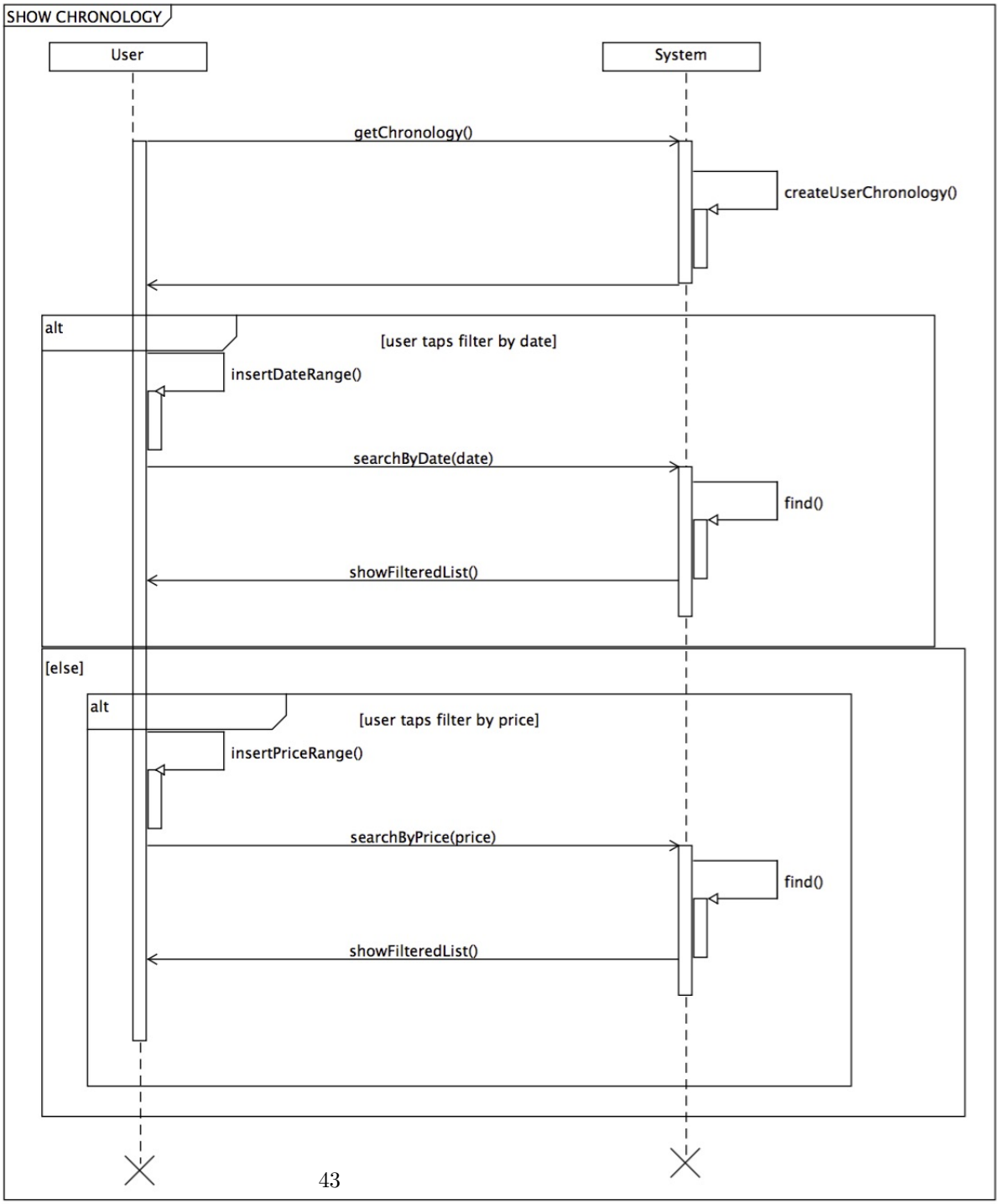
5.4.5 Request a Taxi



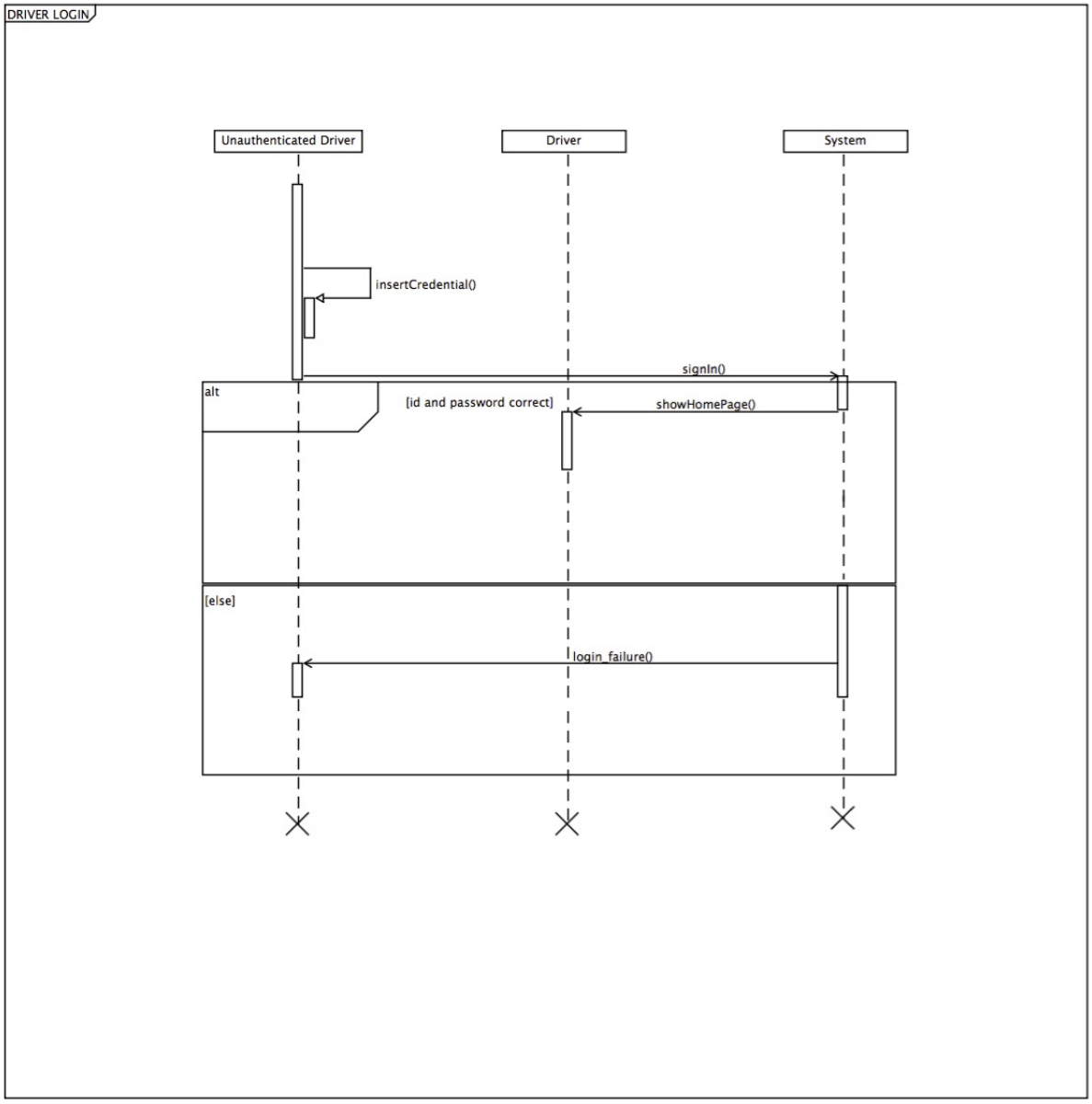
5.4.6 Show Scheduled Reservations



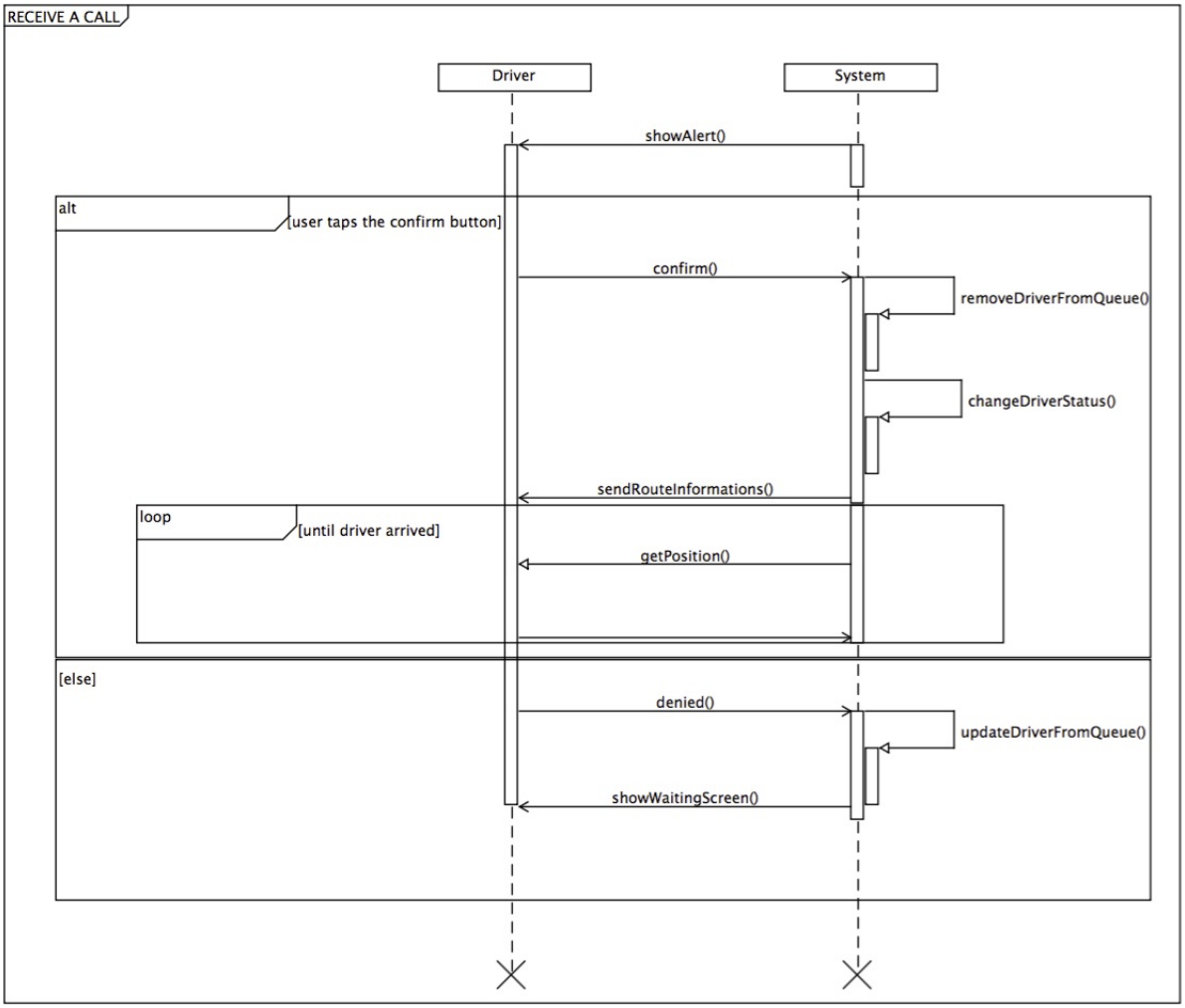
5.4.7 Show Chronology



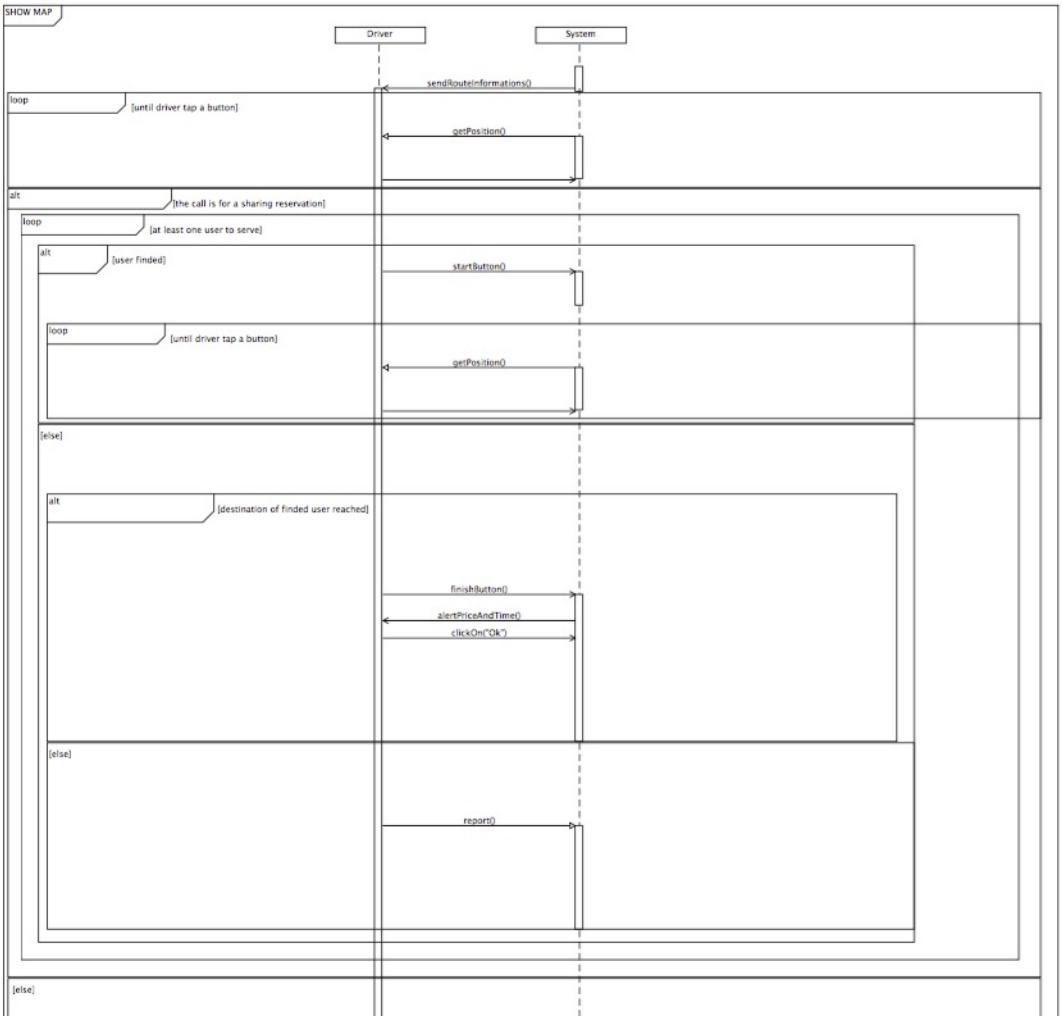
5.4.8 Driver Login

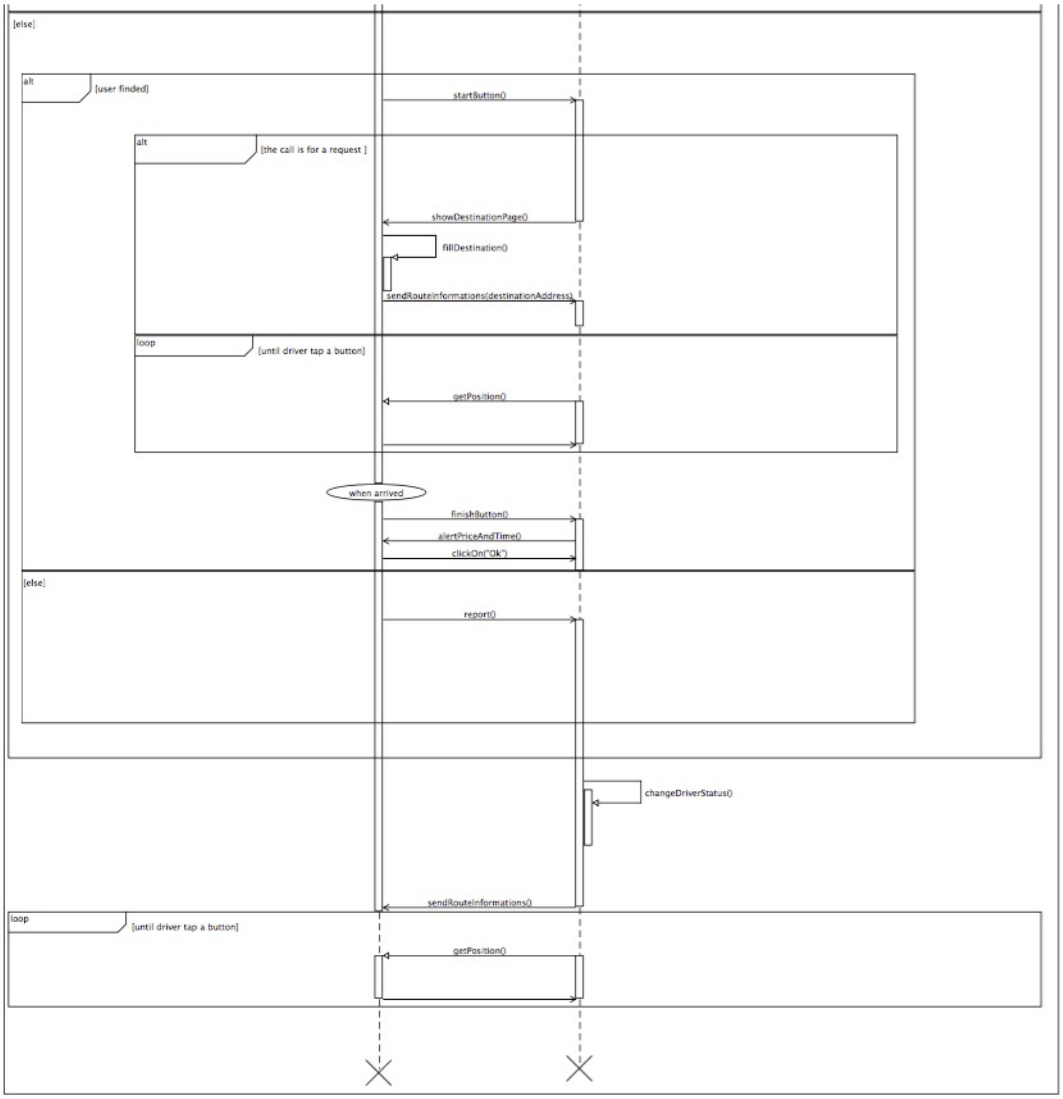


5.4.9 Receive a Call

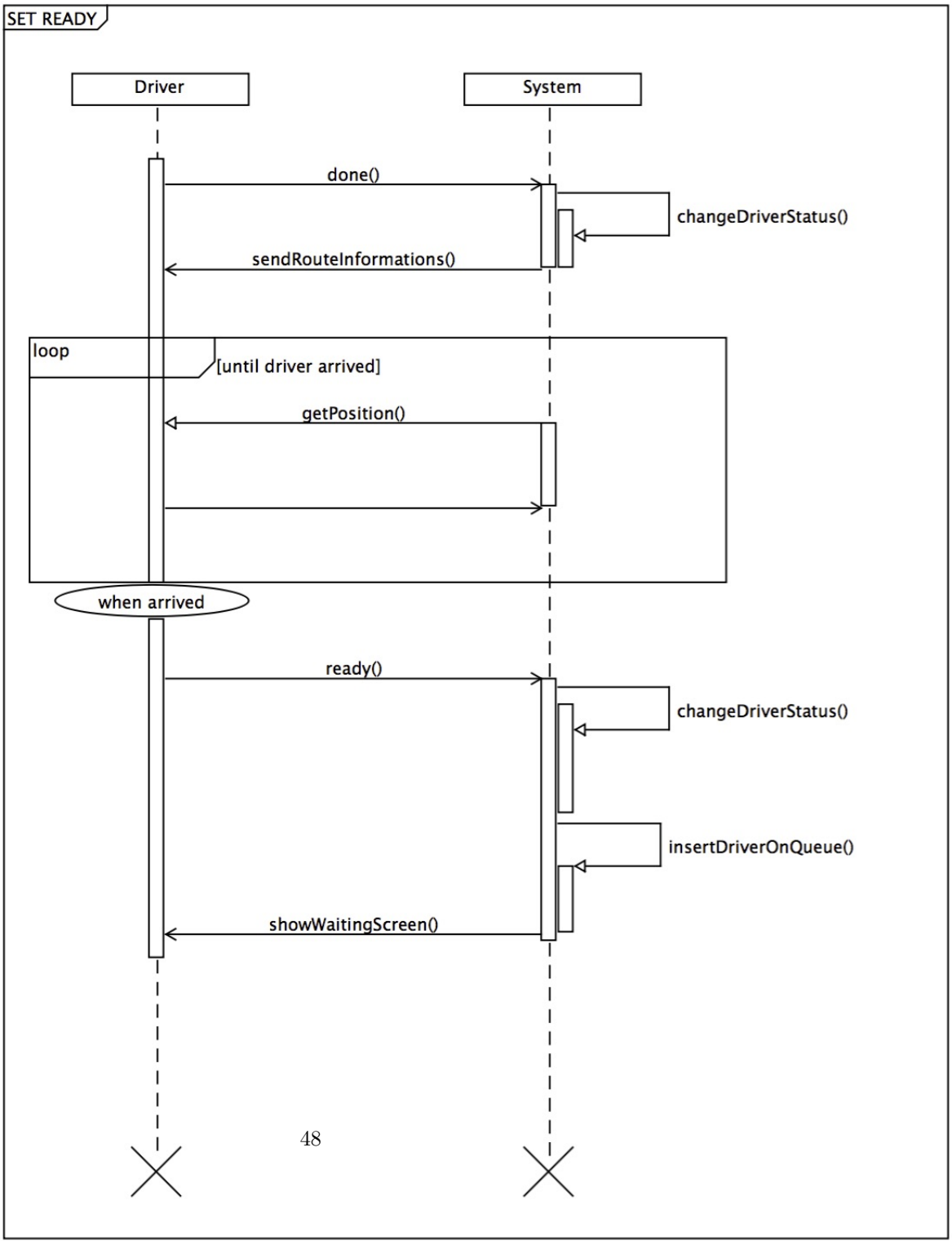


5.4.10 Show the Map

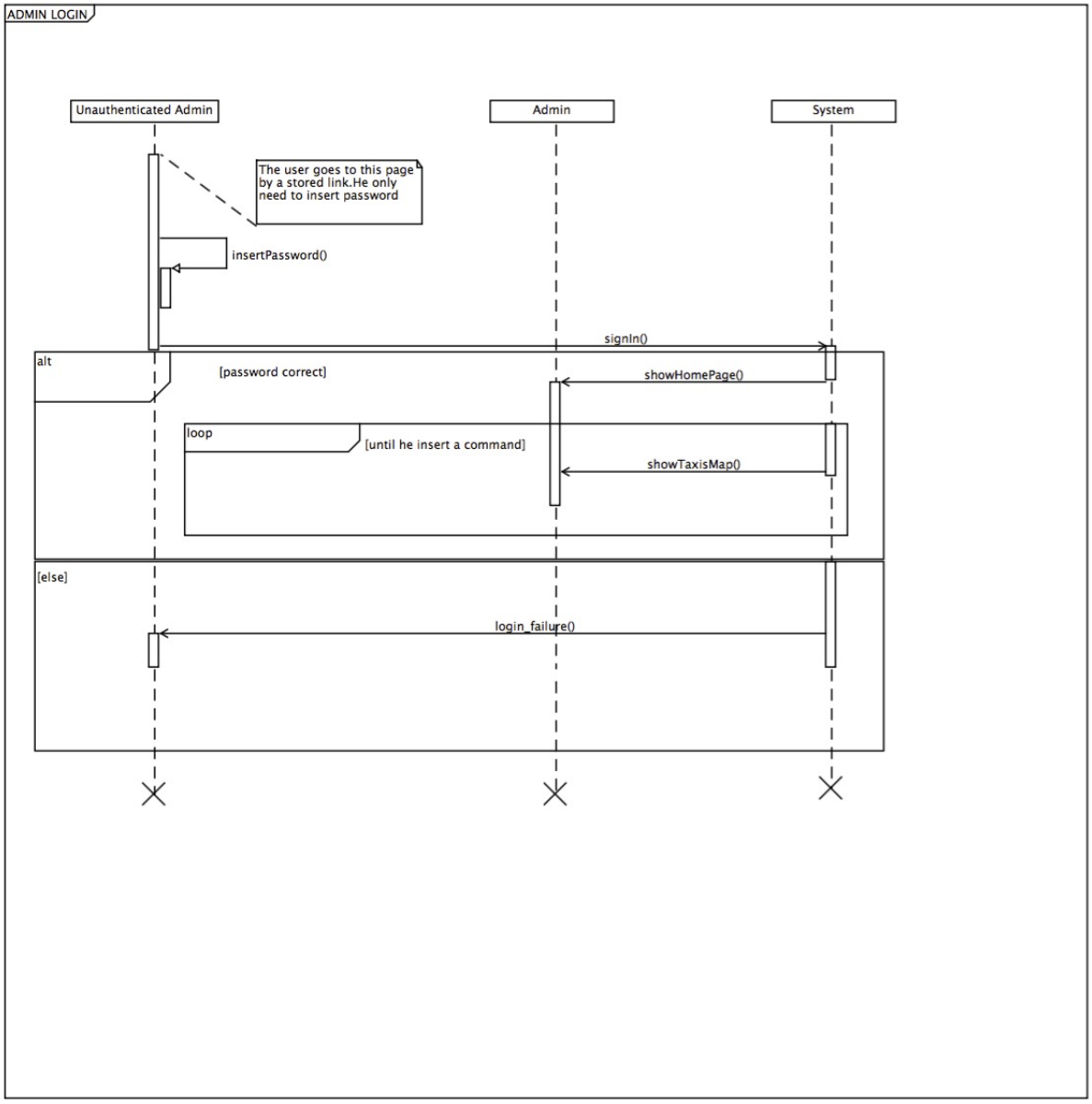



[else]
alt
[user finded]
startButton()
alt
[the call is for a request]
showDestinationPage()
fillDestination()
sendRouteInformations(destinationAddress)
loop
[until driver tap a button]
getPosition()
when arrived
finishButton()
alertPriceAndTime()
clickOn("Ok")
[else]
report()
changeDriverStatus()
sendRouteInformations()
loop
[until driver tap a button]
getPosition()

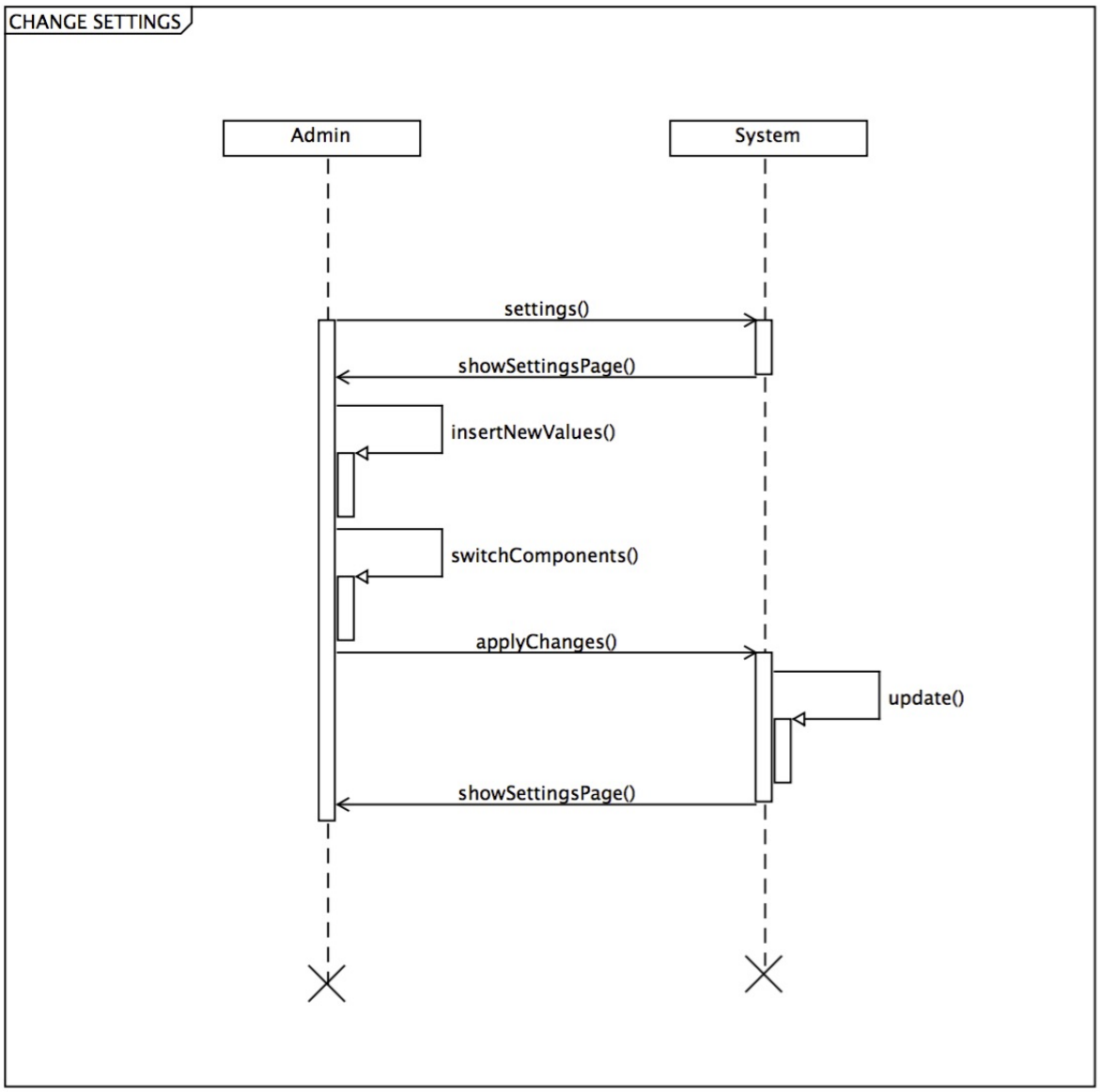
5.4.11 Set the Driver Status Ready



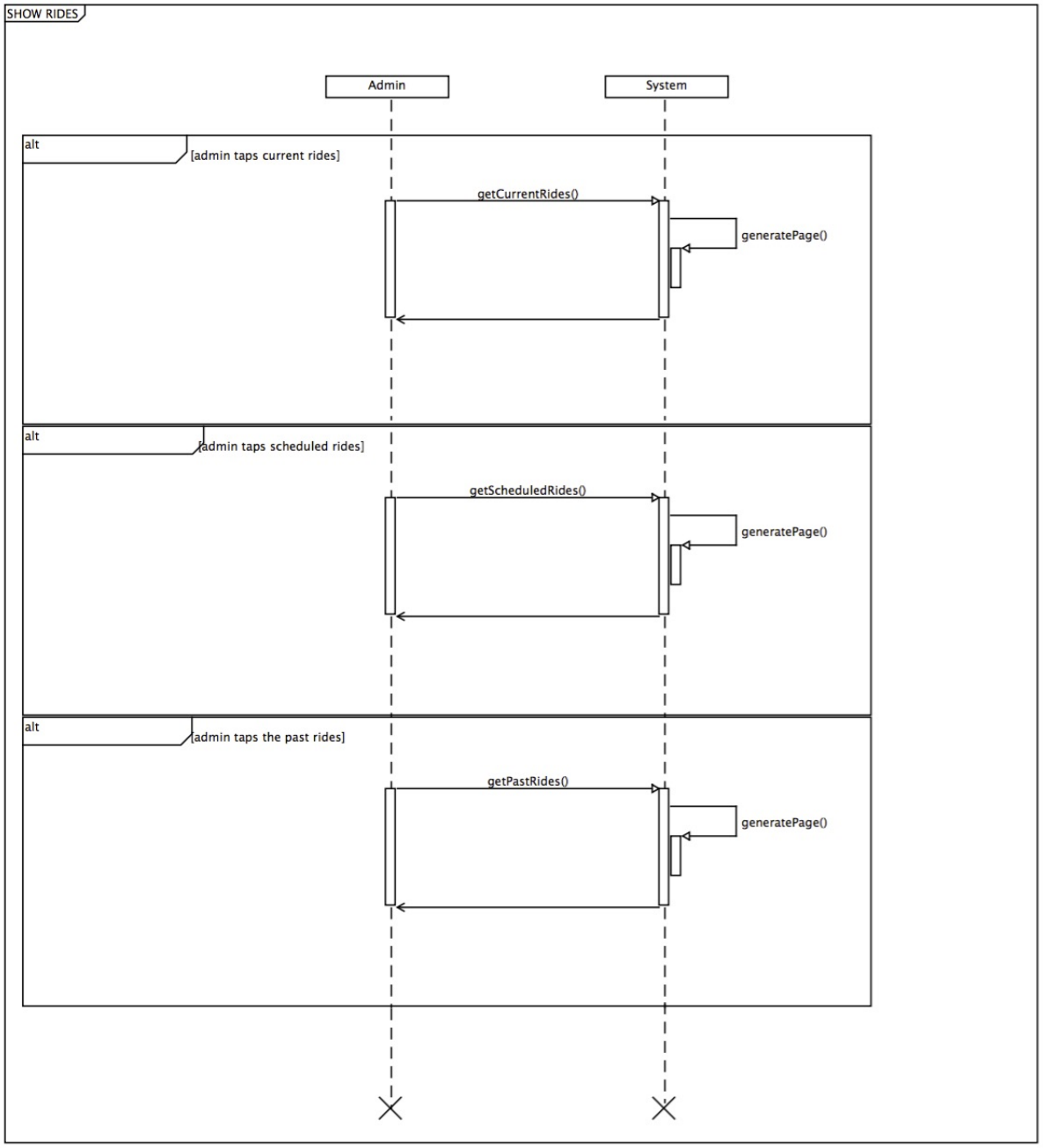
5.4.12 Administrator Login



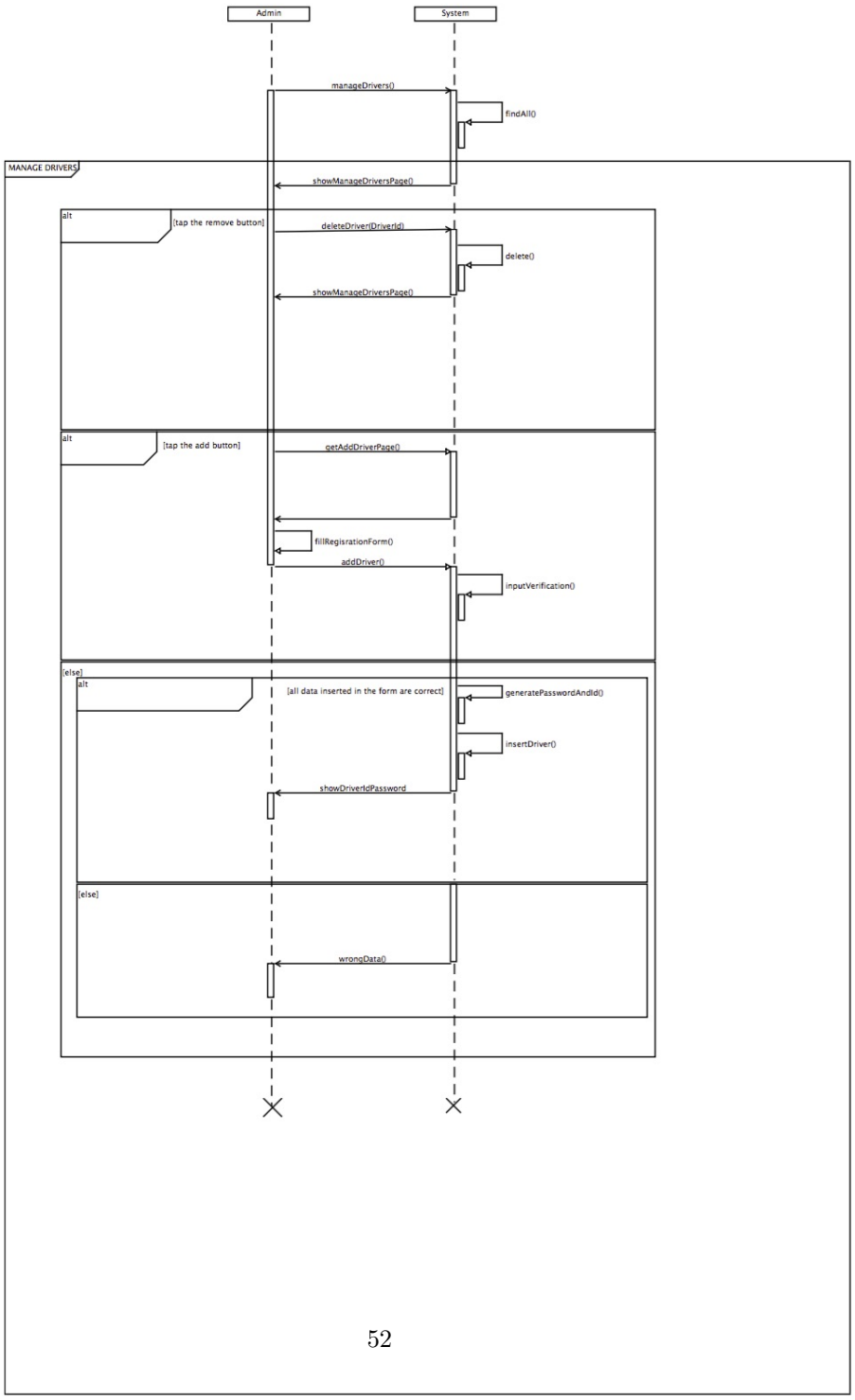
5.4.13 Change Administrator Settings



5.4.14 Show Current Rides

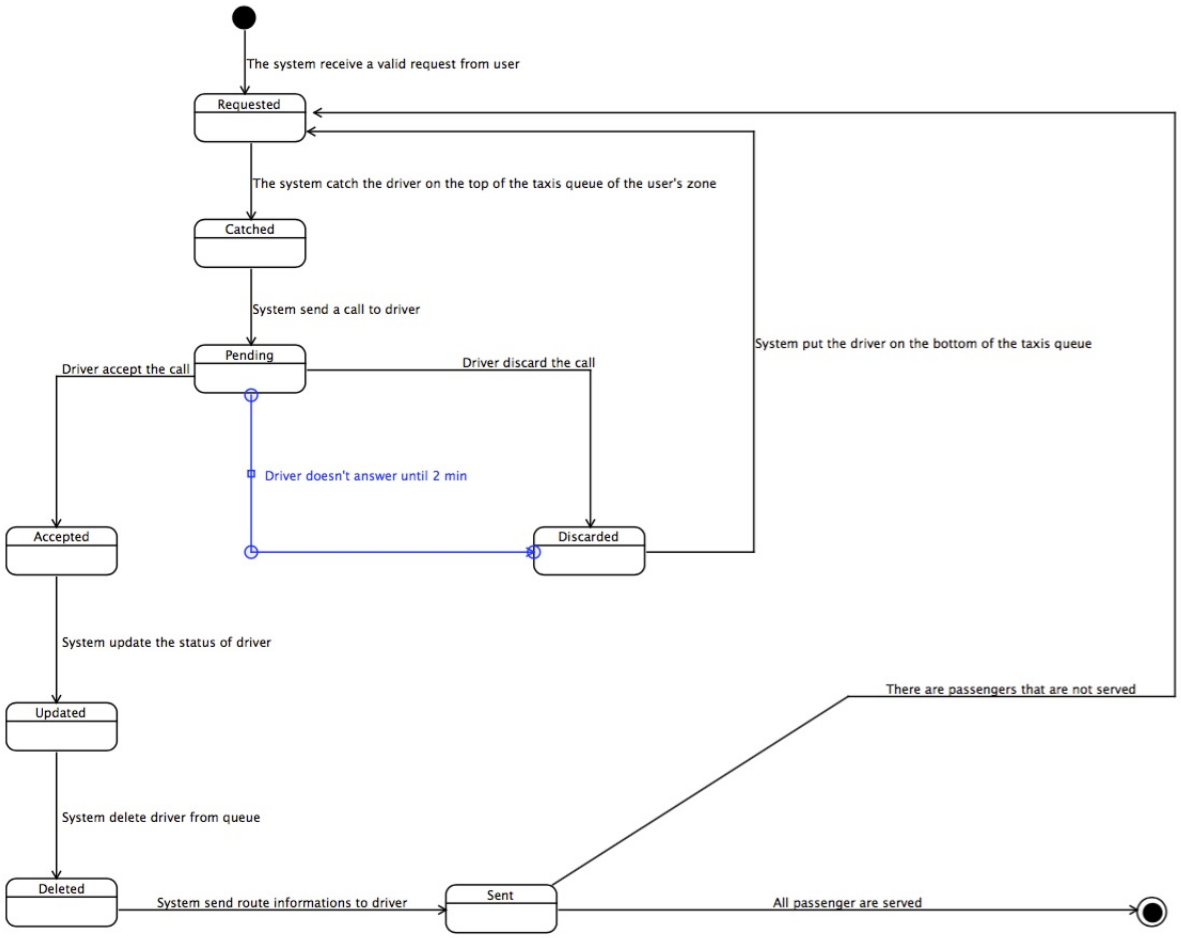


5.4.15 Manage Drivers

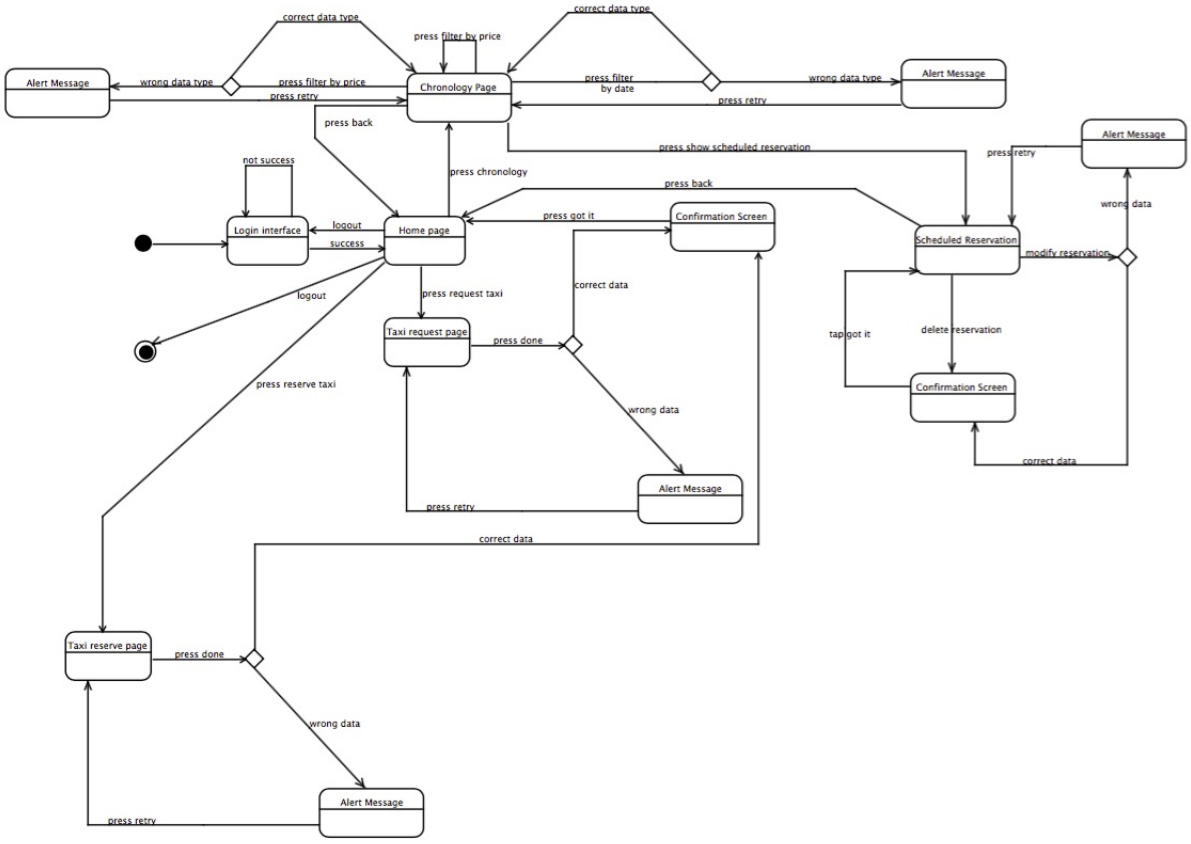


5.5 Statechart Diagram

5.5.1 Allocate a Taxi



5.5.2 User View



6 Alloy Modeling

6.1 Alloy Code

```
//SIGNATURES

sig User { }

sig Driver {
        zone: one Zone
}

abstract sig UserOperation {
        creator: one User,
        server: lone Driver,
        startPlace: one Zone,
        route: one Route
}

sig Request extends UserOperation { }

sig Reservation extends UserOperation {
        endPlace: one Zone,
        sharedWith: set Reservation
}

sig Route {
        crossedZones: set Zone
}

sig Zone { }

sig Call {
        receiver: one Driver,
        object: one UserOperation
}

abstract sig SmsNotification {
        receiver: one User
}

sig RegSms extends SmsNotification { }

sig OperationSms extends SmsNotification {
        object: one UserOperation
```

```
}

sig UpdateSms extends SmsNotification {
        object: one UserOperation
}

//FACTS

fact RequestAndDriverInTheSameZone {
        all r: Request | one r.server implies r.startPlace = r.server.zone
}

fact MinimumPath {
        all r: Reservation | (r.startPlace = r.endPlace) implies
                (no z: Zone | (z in r.route.crossedZones)
                and (z != r.startPlace))
}

fact OneRequestPerTime {
        no u: User, r,s: Request | (r != s) and (r.creator = u)
                and (s.creator = u)
}

fact SharedRidesInTheSameZones {
        all r, s: Reservation | (r in s.sharedWith) and (r != s)
                implies ((s.startPlace in r.route.crossedZones) or
                (r.startPlace in s.route.crossedZones))
        all r, s: Reservation | (r in s.sharedWith) and (r != s)
                implies ((s.endPlace in r.route.crossedZones) or
                (r.endPlace in s.route.crossedZones))
}

fact SharedNoReflexive {
        all r: Reservation | some r.sharedWith implies
                (not(r in r.sharedWith))
}

fact SharedSimmetric {
        all r, s: Reservation | (s in r.sharedWith) iff
                (r in s.sharedWith)
}

fact ReservationOfSameUserNotShared {
        no r, s: Reservation | (r in s.sharedWith) and
                (r.creator = s.creator)
}
```

```
fact DriverServeOneOperation {
        all r: Request, d: Driver | (r.server = d) implies
                (no s: Request | (s.server = d) and (s != r))
        all r, s: Reservation | ((r in s.sharedWith) or (r = s))
                iff (r.server = s.server)
        no r: Request, s: Reservation | r.server = s.server
}

fact RouteContainsStartAndEndZone {
        all r: Reservation |
                (r.startPlace in r.route.crossedZones) and
                (r.endPlace in r.route.crossedZones)
}

fact OneOperationSmsPerUserOperation {
        no o, s: OperationSms | (o.object = s.object) and (o != s)
        no u: UserOperation | no m: OperationSms | m.object = u
}

fact AtLeastOneCallPerUserOperation {
        no o: UserOperation | no c: Call | c.object = o
}

fact NoMultipleCallForSameOpAndSameDriver {
        no c, d: Call | (c != d) and (c.object = d.object)
                and (c.receiver = d.receiver)
}

fact DriverServeOperationIfReceiveCall {
        all o: UserOperation | (some o.server) implies
                some c: Call | (c.object = o) and (c.receiver = o.server)
}

fact UserReceiveSmsForHisOperation {
        all s: OperationSms | s.receiver = s.object.creator
        all s: UpdateSms | s.receiver = s.object.creator
}

//ASSERTIONS

//Check that there are no driver serving a request and a
//reservation at the same time
assert NoDriversServingRequestAndReservation {
        no d: Driver | some r: Request, s: Reservation |
                (r.server = d) and (s.server = d)
```

```
}

check NoDriversServingRequestAndReservation

//Check that there are no operations that are served
//by a driver who hasn't received a call for that operation
assert NoOperationServedWithoutCall {
        no o: UserOperation | (some o.server) and (no c: Call
                | (c.object = o) and (c.receiver = o.server))
}

check NoOperationServedWithoutCall

//Check that all the shared rides has at least a common
//zone in their route
assert NoSharedRidesInFarZones {
        all r, s: Reservation | (s in r.sharedWith) implies
                ((s.startPlace in r.route.crossedZones) or
                (s.endPlace in r.route.crossedZones)) or
                ((r.startPlace in s.route.crossedZones) or
                (r.endPlace in s.route.crossedZones))
}

check NoSharedRidesInFarZones

//PREDICATES

//Show a world with at least one request that hasn't a server yet
pred showRequestWithoutServer {
        some r: Request | no d: Driver | r.server = d
}

//Show a world with at least two shared reservations
pred showSharing {
        some r, s: Reservation | (r in s.sharedWith)
                or (s in r.sharedWith)
        #Request = 0
        #Reservation = 2
        #UpdateSms = 0
        #RegSms = 0
}

//Show a world that contains at least two calls
//for the same operation
pred showTwoCallsForOneOperation {
        some c, d: Call | (c != d) and
```

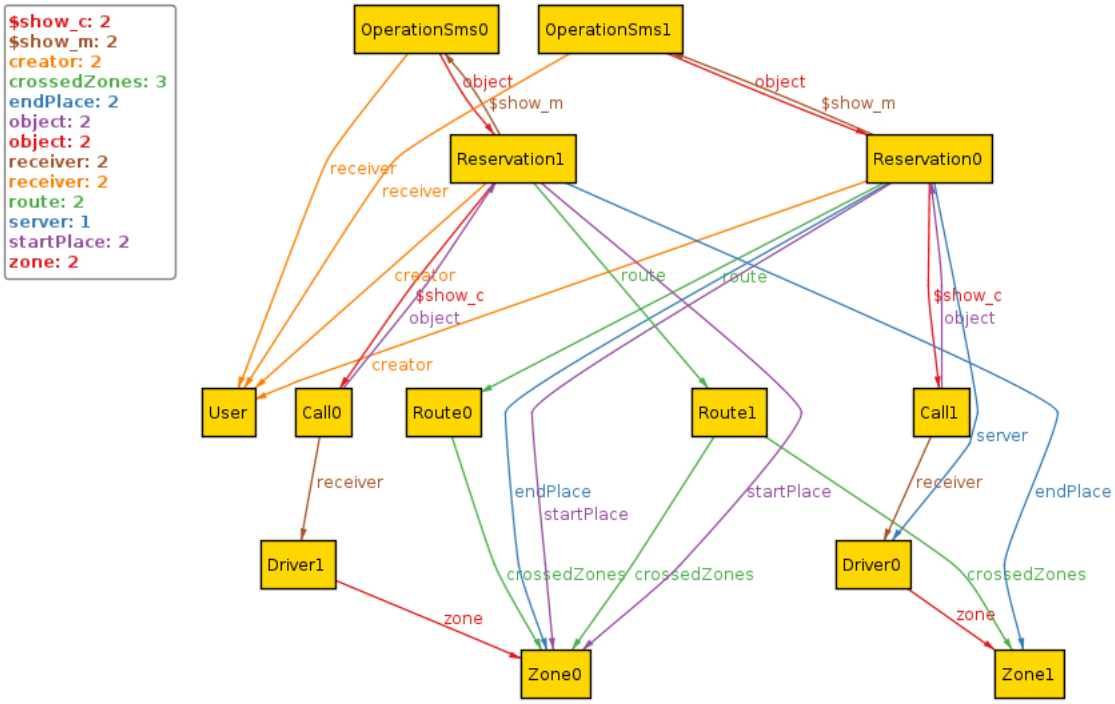
```
                (c.object = d.object)
        #Reservation <= 1
        #Request <=1
        #UpdateSms = 0
        #RegSms = 0
        #Driver <=2
}

//Show a generic world
pred show { }

run show for 2
```

Here the result of the analysis:

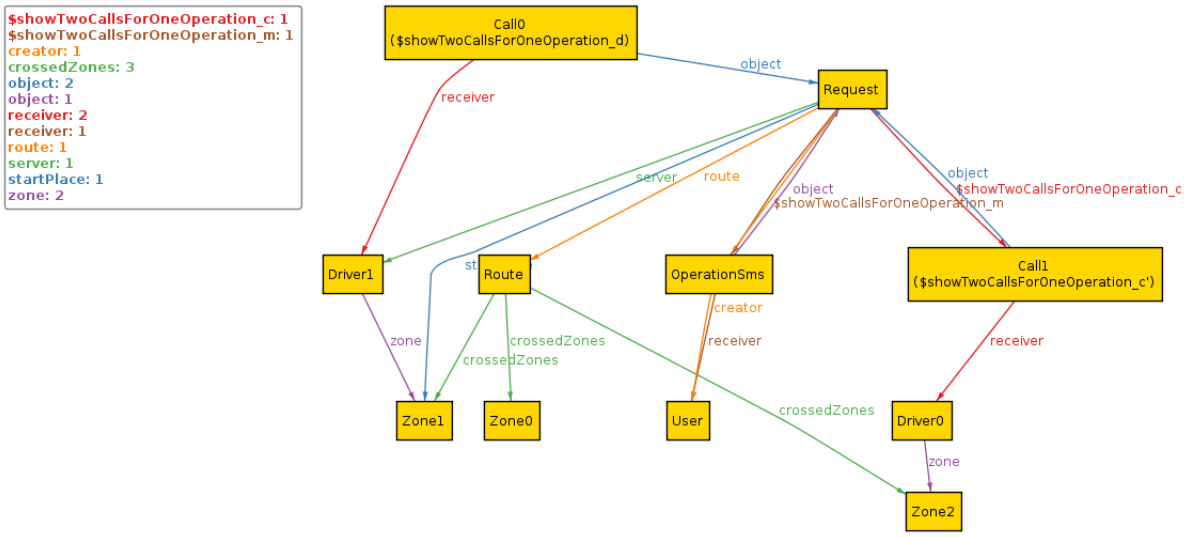
```
Executing "Check NoDriversServingRequestAndReservation"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  2457 vars. 174 primary vars. 4196 clauses. 280ms.
  No counterexample found. Assertion may be valid. 36ms.

Executing "Check NoOperationServedWithoutCall"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  2447 vars. 168 primary vars. 4165 clauses. 129ms.
  No counterexample found. Assertion may be valid. 38ms.

Executing "Check NoSharedRidesInFarZones"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  2524 vars. 171 primary vars. 4274 clauses. 99ms.
  No counterexample found. Assertion may be valid. 38ms.

Executing "Run showRequestWithoutServer"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  2394 vars. 168 primary vars. 4075 clauses. 53ms.
  Instance found. Predicate is consistent. 41ms.

Executing "Run showSharing"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  2453 vars. 171 primary vars. 4202 clauses. 44ms.
  Instance found. Predicate is consistent. 18ms.

Executing "Run showTwoCallsForOneOperation"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  2479 vars. 171 primary vars. 4280 clauses. 29ms.
  Instance found. Predicate is consistent. 14ms.

Executing "Run show for 2"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  991 vars. 80 primary vars. 1552 clauses. 18ms.
  Instance found. Predicate is consistent. 9ms.
```

6.2 Alloy Worlds

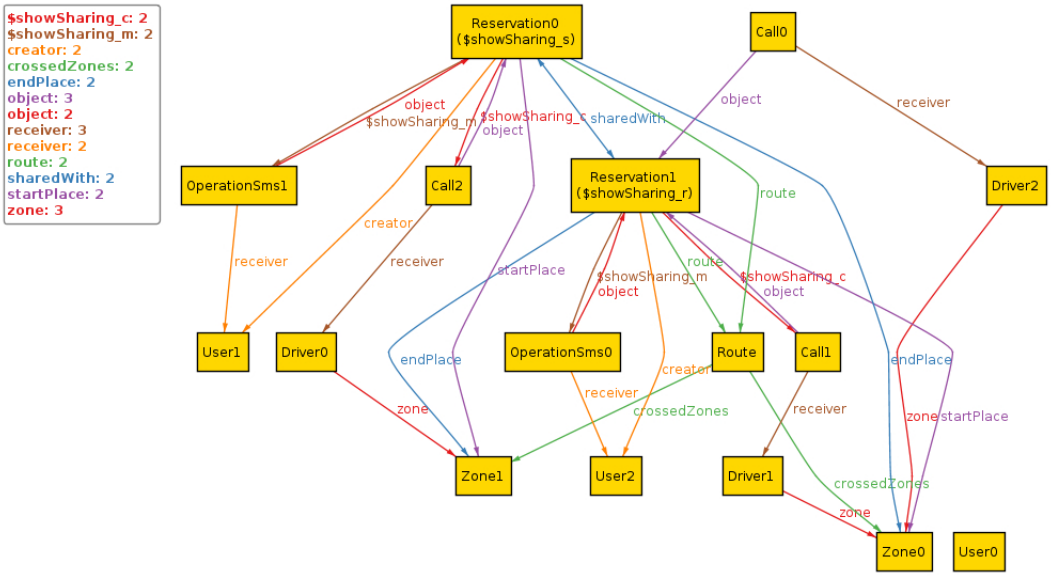
6.2.1 Generic World



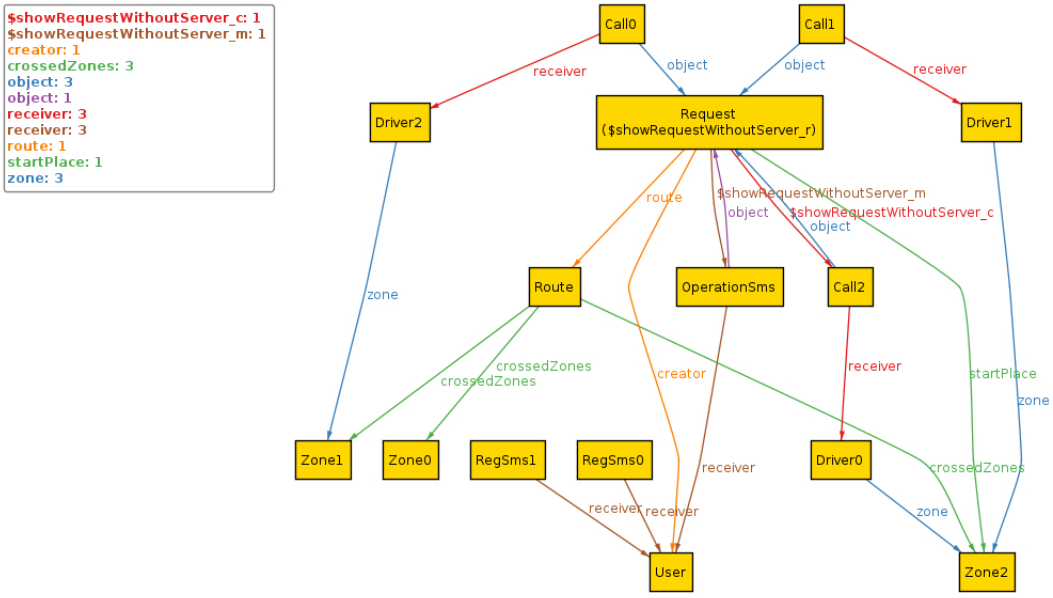
6.2.2 Multiple Calls



6.2.3 Sharing Rides



6.2.4 Requests and Servers



7 Used tools

The tools we used to create this Require Analysis and Specification Document are:

- Microsoft Office Word 2013: to redact the draft of the document
- Balsamiq Mockups 3 to create mockup images
- Google draw.io to create use case diagram
- StarUml to create the class diagram
- UmLet to create sequence and state-chart diagrams
- Alloy 4.2 to write our model and analyze its consistency
- Lyx to redact this document

To redact this document, each of the authors has worked about 25 hours.