1 Introduction

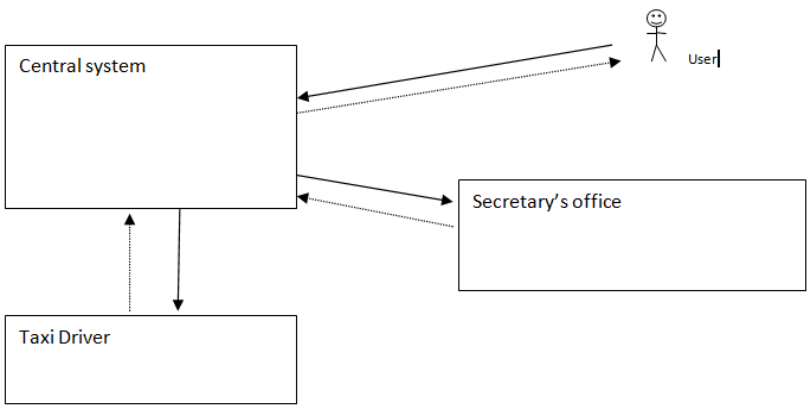
1.1 Purpose

The purpose of this document is to clarify the ambiguity of the natural language which is present in the text document. This clarification will be useful to developers who will have to implement the software and to the client (government) that will know exactly what they will obtain once the system will be completely developed.

1.2 Actual System

We suppose that the actual system is composed and developed in this way: a customer who wants to reserve a taxi ride,calls the operating center of the agency where an employee will reserve to him a taxi ride. To do so the employee insert the data of the client into the central system using an internal app that memorizes all data in a central database.
To optimize the service and to reduce the waiting time at the call center the city government decided to develop a web application and a mobile app. Doing so the customer can reserve himself a taxi ride without calling a call center and waiting a lot of time to be served.
The central system has already a system of API's that allow to us to develop the web application and the mobile application to extend and optimize the already existing central system.
For example,some of these API's are (they are even included those for future implementations e.g the taxi sharing):

-TAXI RESERVATION < username;gps source position;destination;[reservation time]> (reservation time is an optional parameter)
-LOGIN<username; password; [facebook_name];[facebook_password]>
-REGISTRATION<name; surname; gender; e_mail; birth_date; username; password; [address]; [mobile phone]>
-PRENOTATION<username>
-LOGOUT<username>
-DELETE_PRENOTATION<id_prenot>
-MODIFY_PRENOTATION<id_prenot; data; time; source;destionation>
UPDATE_GPS_LOCATION<username; gps_location>
UPDATE_PASSWORD<username; old_password; new_password>
CUSTOMER_RESERVATION<username>
GOVERNEMENT_WORKER_ANAGRAPHIC<username>
SERVICE_STATS<username>
WORKER_STATS<username>
TAXI_SHARING_ADD_USER_TO_RIDE<username; id_ride>

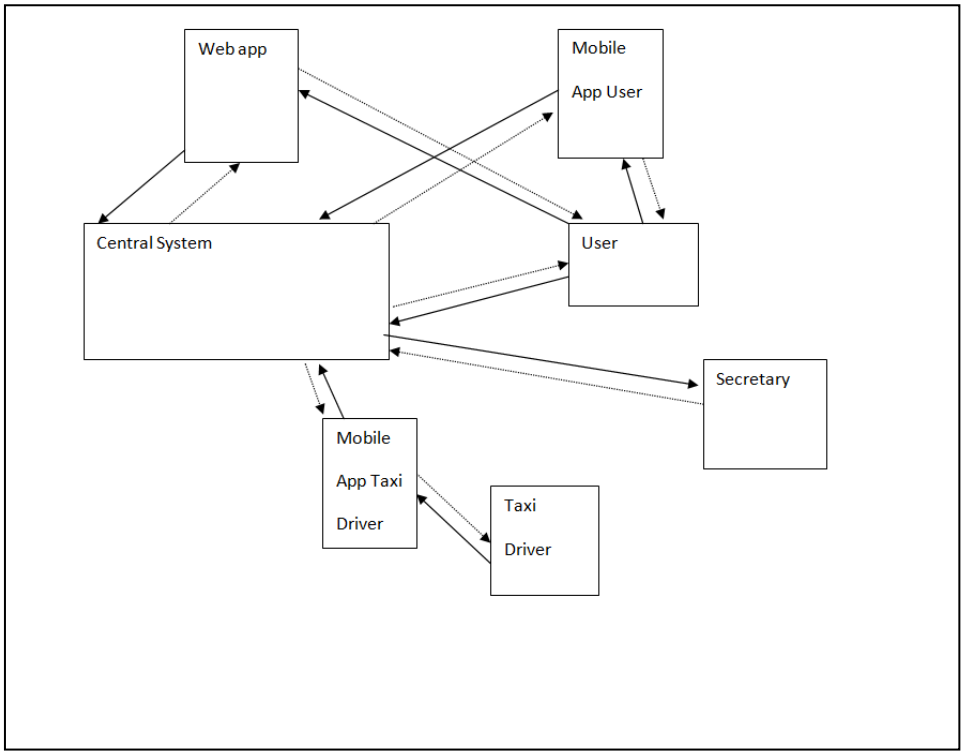


1.2 Scope

The scope of the system is to allow a fair management of taxi queues and to provide a simple and fast way to customers who want to reserve a taxi.
As **fair management** we mean that taxi queues are well balanced based on the distribution of customer's requests in the various zones of the city during a day,to ensure that it is very unlikely that there isn't a taxi in a client's zone when he asks for it. With the mobile app it will be easier to meet this goal because in each moment the system knows exactly where each taxi is at every moment and so the central system can decide in which zone a taxi must be assigned to.
As **simple and fast way** we mean that with mobile and web apps customers can reserve a taxi without having to wait at the call center and the apps will have user friendly interfaces to make them simple to use. The client's mobile app is thought to be used when the client is out of home and he needs an optimized interface for mobile and in the future will also be used to support other functions based on the position of the customer (for example taxi sharing). The web app instead is thought to be used when the customer is at home.The new system will be faster also because,thanks to the gps used by the taxi driver's mobile app, the central system will assign the nearest taxi to the client who's making the request,decreasing in this way the needed waiting time to the taxi driver to reach the customer.
In our system there will be two types of client :
-**Registered Customer**: This type of client can make two different types of reservation:

- **Instantaneous**: The aim of this type of request is to take a ride immediately (in the meaning that the taxi driver will come to the user as soon as possible to the client's specified source to pick him up)
- **Future**: The scope of this type of request is to book a taxi ride that will have to be taken at the specified date and time in the future.

This type of user can also see the position of the incoming taxi and his past reservation.
-**Occasional Customer:** This type of client isn't registered to the system but can take a taxi ride asking directly to the taxi driver employeed of the agency.



1.4 Actors and stakeholders

1.4.1 Actors

- Customers: People who need the taxi service. Customers can be of two types:
    - **Occasional users**: Customers that see a taxi in a zone and ask directly to the taxi driver for the ride.
    - **Registered customer**: This customers can also use the mobile and web app to reserve a taxi ride.
- Taxi drivers: People who guarantee services to the customers. Taxi drivers are employees of the government taxi agency. Each taxi driver has his own company account that uses in a mobile app. Taxi driver only access to the mobile app and they don't access to web application.
- Government employees: The employees of the government who work in the agency to analyze the the profit and the usage of the service. They are considered as **special users** in the system

1.4.2 Stake holders

The stakeholders are all the actors and the government of the city that asked to us to develop this system and it's interested in the services that the city offers and the revenue from those services (taxi service included)

1.5 Goals

[WG]=web app goal [MBG]= mobile app goal [CG]=common goal

1.5.1 Client apps goals

- [CG1] A user can login in the system by writing his credentials.(username and password)
- [CG2] A user can register himself to the system if he hasn't an account.
- [CG3] A user can book an instantaneous taxi ride
- [CG4] A user can book a future taxi ride
- [CG5] A user can cancel a previous reserved taxi ride.
- [CG6] A user can view his previous reservations.
- [CG7] A user can view his position and the incoming taxi driver position.
- [MG8] When the system has assigned to the customer a taxi the mobile app must notify this fact to the user so he can view how much time he will have to wait for the taxi.
- [CG9]A user can see his personal info
- [CG10] A user can update a previous made reservation
- [WG11] A use see the notifications of incoming taxi when he opens the web page

1.5.2 Taxi driver mobile app goals

- [MG1] The taxi driver can login himself to the system.

- [MG2] The taxi driver can view his past served rides
- [MG3] The taxi driver can accept or deny the customer requests.
- [MG4] The application must periodically communicate the taxi driver position.
- [MG5] The taxi driver can set the occupied/available state
- [MG6] The taxi driver can view how much he has already earned this month as variable month salary
- [MG7] The taxi driver can view the assigned zone by the system where he will have to go once a ride is over.
- [MG8] The taxi driver can accept an occasional user ride

1.5.3 Government employee app goals

- [WG1] The employee of the government can access as a special user to the central system
- [WG2] The employee of the government can view the agency employees and their personal data
- [WG3] The employee of the government can view the service statistics
- [WG4] The employee of the government can view the worker statistics

1.6 Reference documents

- IEEE Standard For Requirement Specification.pdf
- Assignments 1 and 2 (RASD and DD).pdf

2 Overall description

2.1 Product perspective

The applications we will develop will be composed by three different applications:

- Client applications:
  - Mobile app
  - Web app
- Taxi driver applications:
  - Mobile app
- Employee app:
  - Web app

All these three apps will be integrated to the central system by using the exposed APIs.

2.2 User characteristics

As specified in the actor section there will be two types of users in our system:

- The taxi drivers: they want a simple way to manage client request and go to the position where the client wants to be picked up. Also they want to be updated of their currently earned variable part of the salary
- The clients: they want an easy and fast way(see definition of fast in the scope section) to manage taxi reservation.
- Goverment employees: They want an app to monitor the performance and situation of the system

2.3 Constraints

2.3.1 Regulatory policies

The confidential data inserted from the users(name, address, telephone…)must be stored in a secure way as specified from the privacy laws. Also the position of the users must be authorized before by the users themselves.

2.3.2 Interfaces to other applications

Our apps interacts to the Api's interfaces and therefore we can access to the central system only by using functionalities exposed by these APIs.

2.4 Assumptions and dependencies

2.4.1 Assumptions

- The system is not unfeasible. Failures and faults can happen.
  If the system crashes it's able to roll up to the previous same state from the crash.
  The roll up technology is already implemented in the existing central system.
- If one of the three apps in our system crashes, the user must login in the system again.
- If there are no taxies available to satisfy a customer request, then a message from the application is shown to the user saying to trying again in a future moment.
- At the start of the day, taxies are disposed as the central systems decides. Each taxi driver can see from the app the assigned zone
- We suppose that the taxi driver has a battery charger for the mobile in order to allow the system to track at any time the taxi driver position.
- We suppose that the taxi driver's smartphone has a stable internet connection within the city that allows to communicate with the central system.
- The payment of the ride will be done in cash at the end of the ride, directly to the taxi driver. The payment will be registered with the taximeter and at the end of the day the taxi driver will transfer all data to the central system.
- If a taxi goes out from the city,the taxi is not available in any zone of the city but it's inserted in an external taxi's list with the other taxis that are not in the city.
- Taxi drivers have a minimum month salary and an extra amount of this salary is earned on a euros/km model. This is done to boost taxi drivers to work more and consequently to obtain a better service for the clients.
- All taxi drivers have already a personal account given from the agency at the time of the hiring,so they only have to login,not to register.
- Taxi drivers use only the mobile app to work and interact with the central system and not also the web app.
- Taxi drivers that accept a client request stay in the same position of the zone's queue before they accepted the request.This assumption is due to the fact that these taxi drivers are offering a good service to the customers and so it's not correct to modify their priority in the list
- If an occasional user sees a free taxi in the zone he is than he can ask for a ride directly to the taxi driver that will simply set his state as busy/taken and will send the destination data to the central system.
- At the end of a ride,if a taxi driver doesn't receive or accept any customer request,the central system will assign to him the nearest zone where in the future there will probably be the need of an extimated number of taxi (this exstimation is done by the knowledge of the traffic in a day by the central system basing on past experience) to satisfy the customers and to avoid that they have to wait too much.

This meets also the goal of fair management because the queues are statistically well balanced

- We suppose that the client is at the specified source when the taxi arrives or otherwise the client calls the agency call center and an employee will cancel his reservation.

2.5 Future possible implementations

- Taxi sharing: this could be done inside the mobile app where users can dynamically add themselves to an existing ride visualizing how many persons are already in the taxi and the destination of this ride. With this information the users can decide to share the taxi or not.

- Service Survey: At the end of the ride a client can compile a short survey about the quality of the ride

- History of rides: A user can see all the rides he took in the past.

- Facebook LogIn: A user can access with his facebook credentials

- Places of Interest: If a user is there for visiting the city and he's taking a taxi,if there are important places nearby the mobile app suggest the user to visit them.

- Forum: In the mobile app e web app the users can comment the quality of the service

Scenarios

Bob: client
Alice: taxi driver.
Tom: second taxi driver
Jane: government employee

1: Bob is visiting a new city and he needs a taxi. He opens the mobile app on his smartphone. Given that is not registered in the system he has to register himself by inserting personal data. Doing so he can log himself in the system and he can ask for an instantaneous taxi ride (see **section 1.3** for types of requests) and then the app will send to the central system bob's username,gps location and the instantaneous type. The central systems takes care of this request and sends a notification to the first taxi driver in Bob's zone queue.
Alice is the notified taxi driver and she deny the request because she's temporanely busy. Therefore the system puts Alice in the last position of her zone's queue and asks the same request to the taxi driver in the second position after Alice,whose name is Tom.
A notification message is sent by the central system to Bob's smartphone and at that time he can see the position of Tom arriving and the corresponding waiting time.
Once the taxi is arrived pays Tom for the ride and Tom notify that the ride is over.

2:Bob needs to go to an appointment the next day and so he opens at home the web app and after the login reserve a taxi ride to pick him up at his home and bring him to the appointment location. Bob specifies this two locations (the address of his home and the address of the appointment) and confirms. The web app verify that the actual time it's at least two hours before the appointment

time and the check is successful sends this information to the central system who memorize the reservation and sends the acknowledge that the reservation has been stored correctly.
10 minutes before the appointment,the central system notifies Alice (which is the first in the queue) to take carry of the ride. She accepts and then a notification is sent to the client to his mobile app where can see the estimated waiting time and where Alice is and for the last steps it's equal to case 1.

3: Bob is at home and he opens the web app to see the rides he took in the past with information about the ride time and the price paid. With this information he can organize himself to reserve another appointment to arrive in time the next time he asks for the service.

4: Alice is at service in a normal working day and she decides to open her mobile app to see how many clients has taken care of that day an she can also see how many clients he took care of in the last months of the current year and the client based monthly salary reached until now. Basing upon this information she decide that this month has taken too few clients and then she accepts more requests.

5: Jane is a government employee and wants to see the agency's workers salary and their performance. To do so, she access to the web app where she can see different information regarding the agency employees to evaluate the performance of the agency itself.

6:Alice receives a notification from the application. There's a new customer request. She decides to accept the request and so she taps on the accept button in the app. At this point she starts her road to pick up Bob. Once arrived to Bob's location she takes him to his destination and she receives the payment, which will be registered on the taximeter.

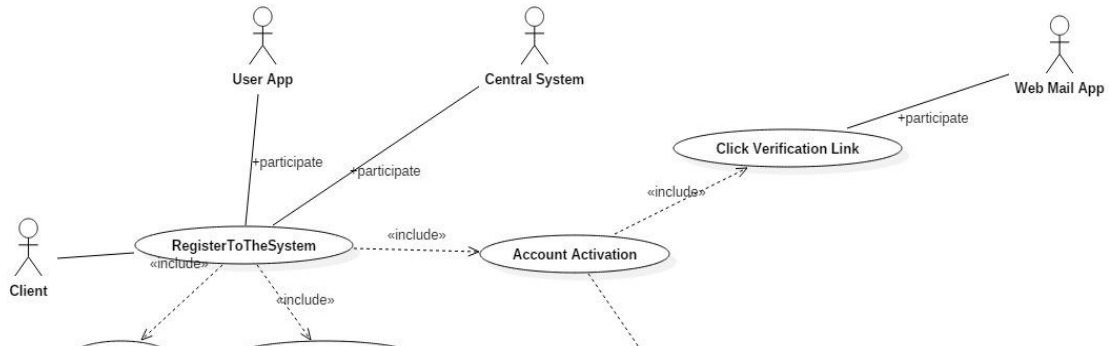
7. Alice is an available taxi driver and the central system assigns her a new zone to go. She receives a notification on her mobile app and she follows the indication to reach this zone. When she arrives to the destination she taps on the "arrived" button and sends a notification to the system. This button is useful in the case in which the gps doesn't work and the taxi position is not clear to the central system.

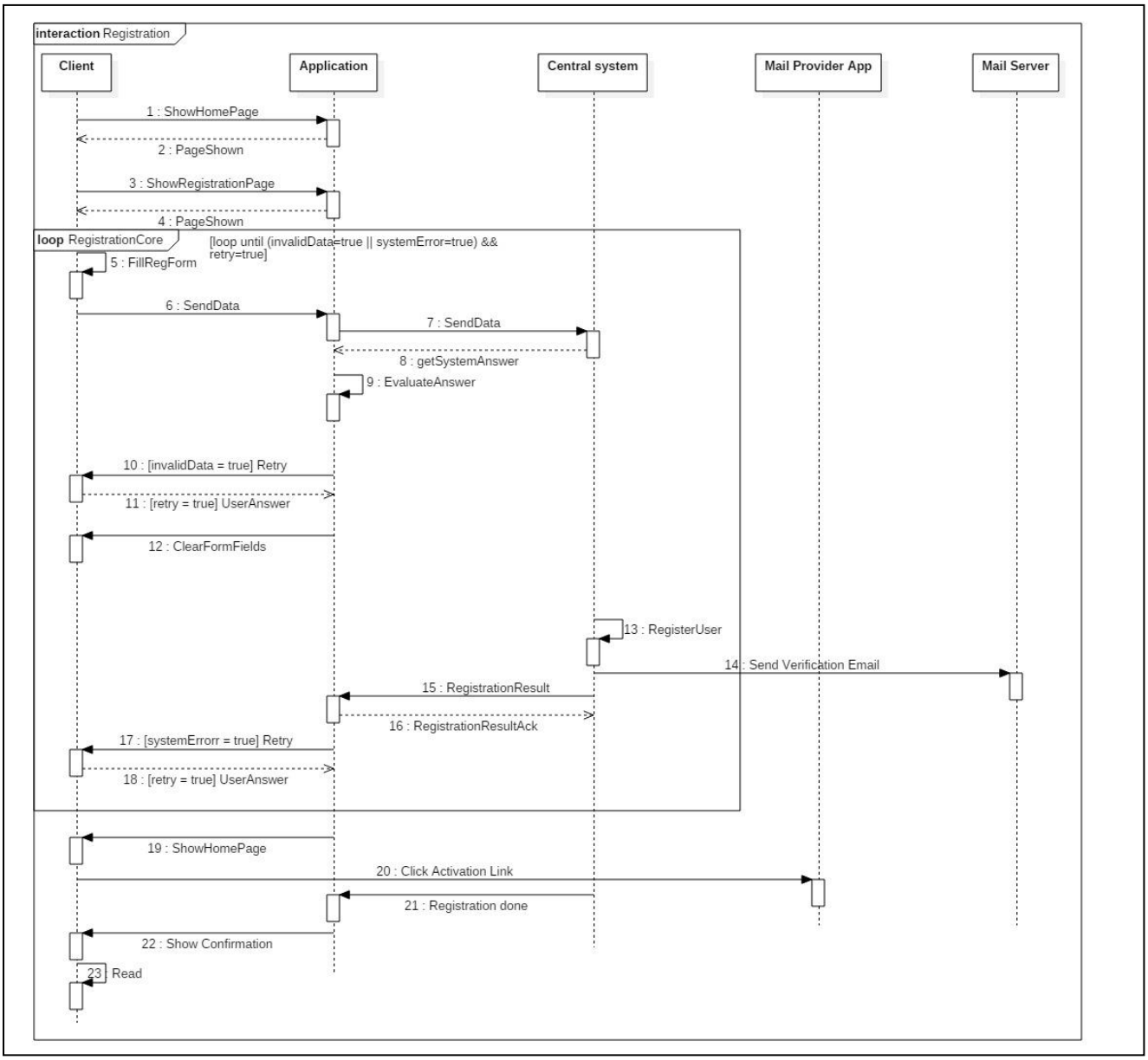
8. Bob arrives in the city by train and he has to go to an important work appointment. So he decides to take one of the agency's taxi available there. He sees that Alice is available and he asks to her if she can take him to his appointment's place. She accepts to take the ride. Alice specifies Bob's destination within the mobile app an she rides to the destination. When the ride is over Alice tells the price of the ride to Bob that pays for it and at last Alice register the price with the taximeter.

2.6 Models

Registration

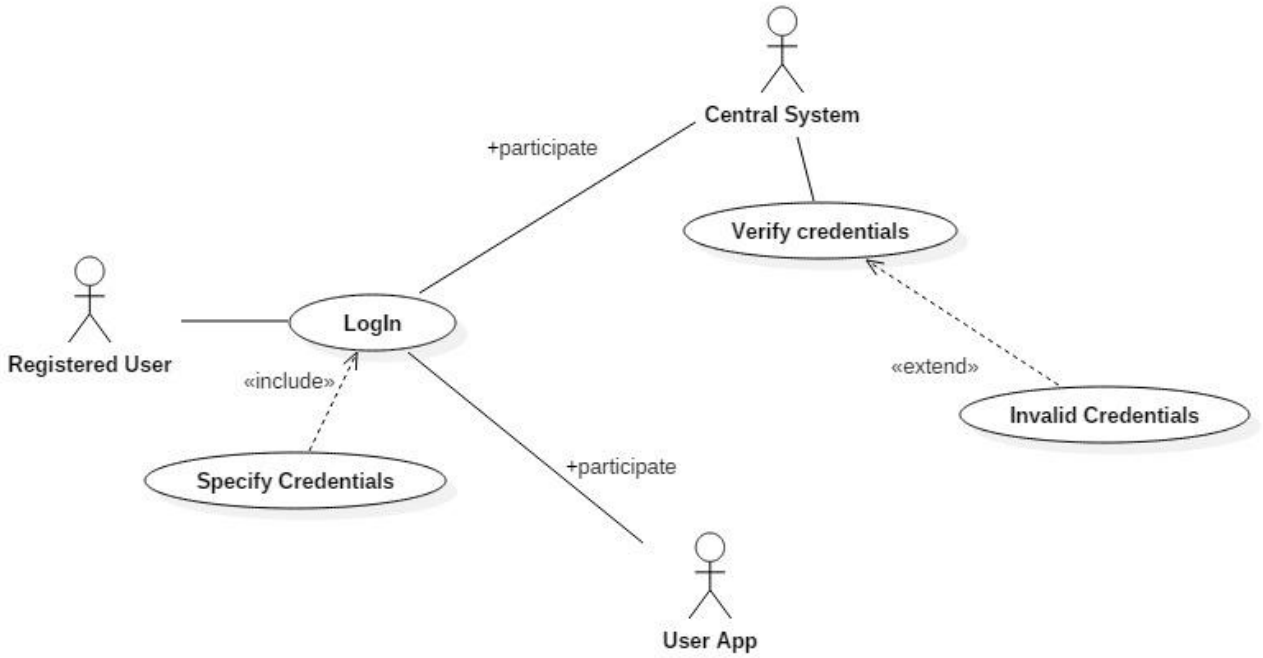
Actor	Client,User App,Central system,Web Server, Web mail app
Goal	CG2
Pre- Condition	NULL
Event Flow	1. Opens the app 2. Opens the register form 3. Insert the required data to register 4. Sends the data to the central system
Po	is
Ex	d field) or

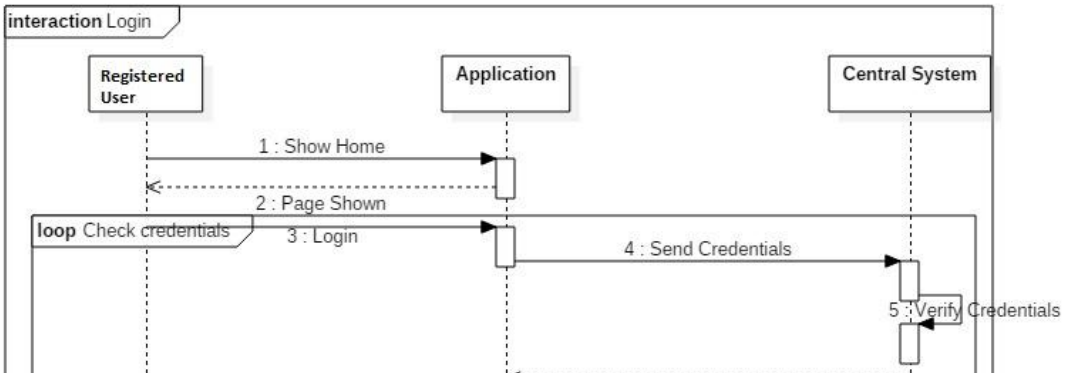




LogIn

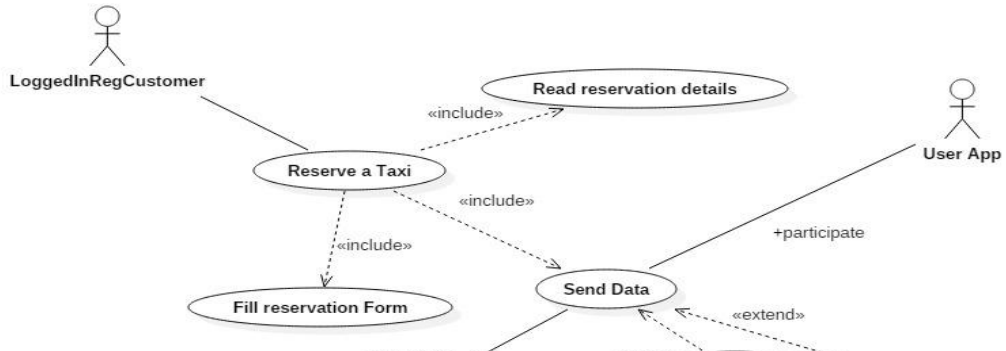
Actor	Registered User,Central System,User app
Goal	CG1
Pre- Condition	NULL

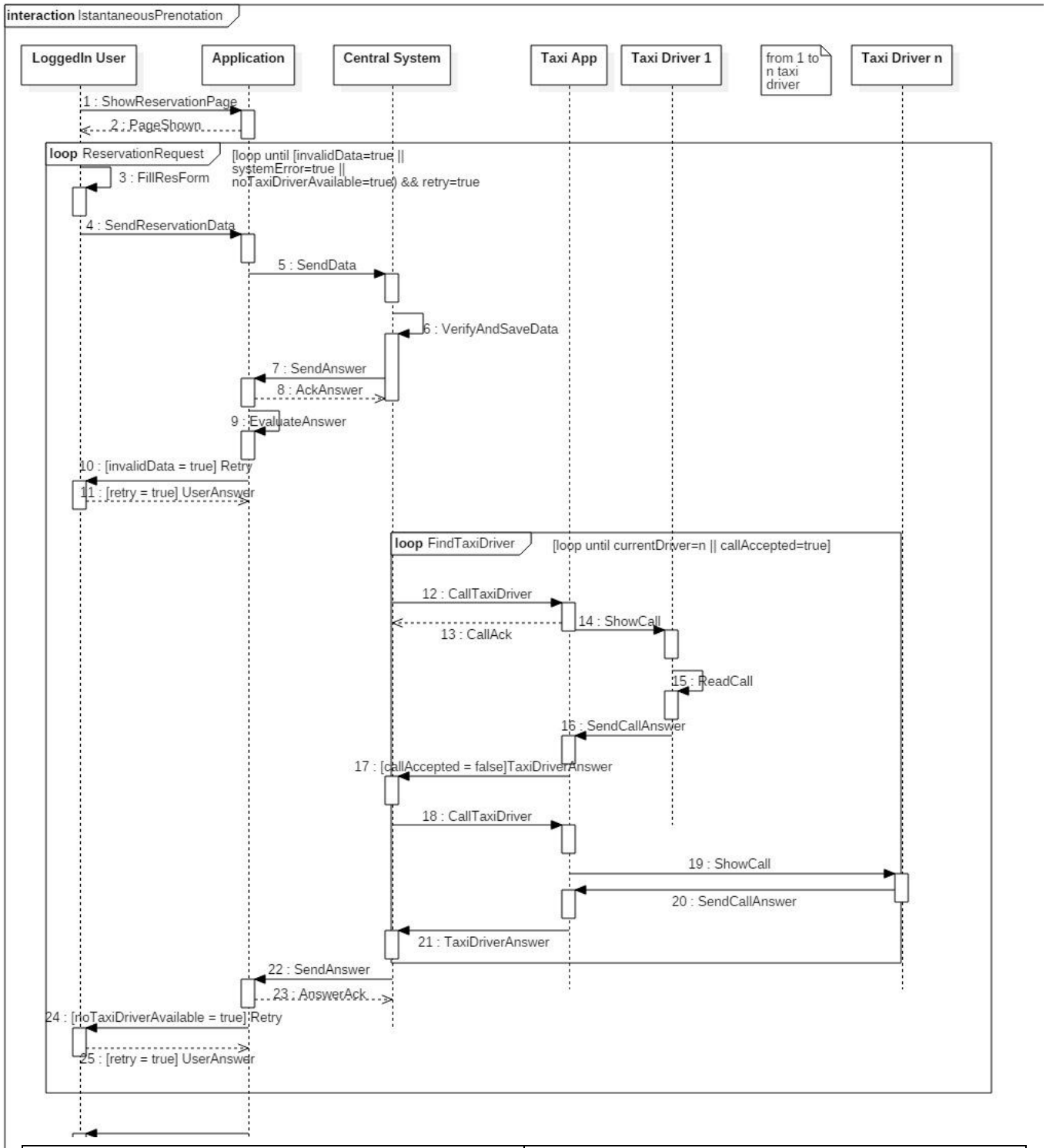

Central System
+participate
Verify credentials
Registered User
LogIn
«extend»
«include»
Invalid Credentials
Specify Credentials
+participate
User App


interaction Login
Registered User
Application
Central System
1 : Show Home
2 : Page Shown
loop Check credentials
3 : Login
4 : Send Credentials
5 : Verify Credentials

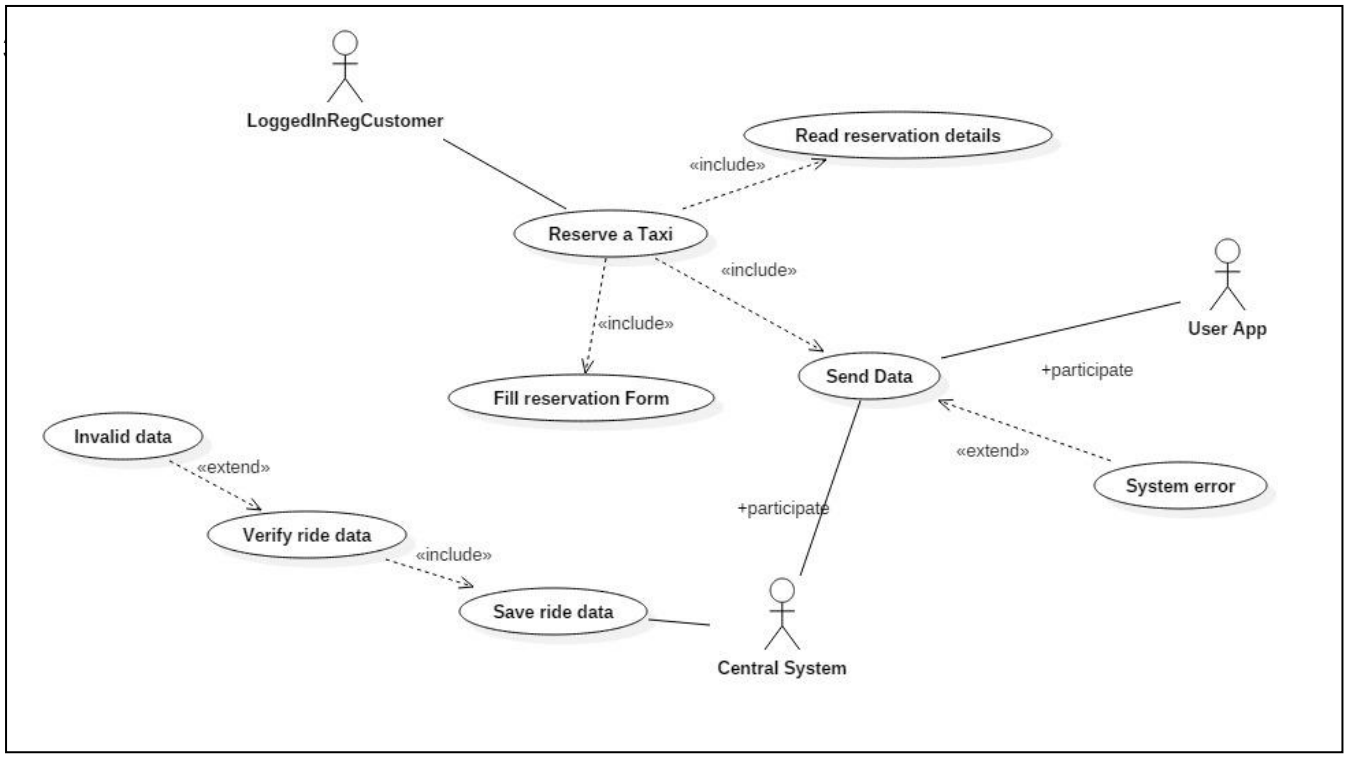
Instantaneous reservation

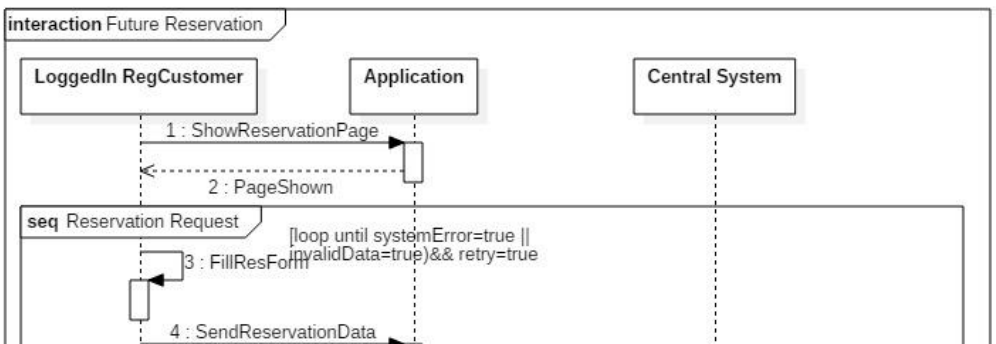
Actor	LoggedInRegCustomer, Central system,UserApp
Goal	CG3
Pre- Condition	The user is already logged in
Event Flow	1. The user opens the page for the taxi reservation 2. He fills the form (required data for the reservation) 3. He sends sends data to the central system 4.He reads the confirmation message
Post- Condition	1. The customer has successfully reserved a taxi for a future moment
Exception	1. System error: an unexpected error occurred 2.Invalid data: If the user insert invalid data (uncorrect data in any other field) 3. Taxi not available: There is no taxi available to satisfy the request





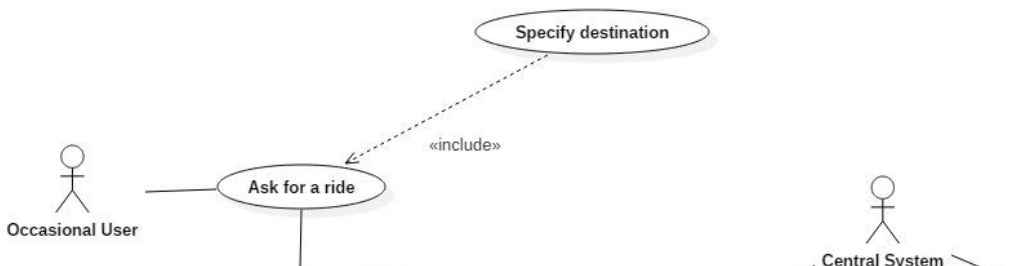
Actor	LoggedInRegisteredCustomer, User App, Central System
Goal	CG4
Pre- Condition	The user is already logged in
Event Flow	1. The user opens the page for the taxi

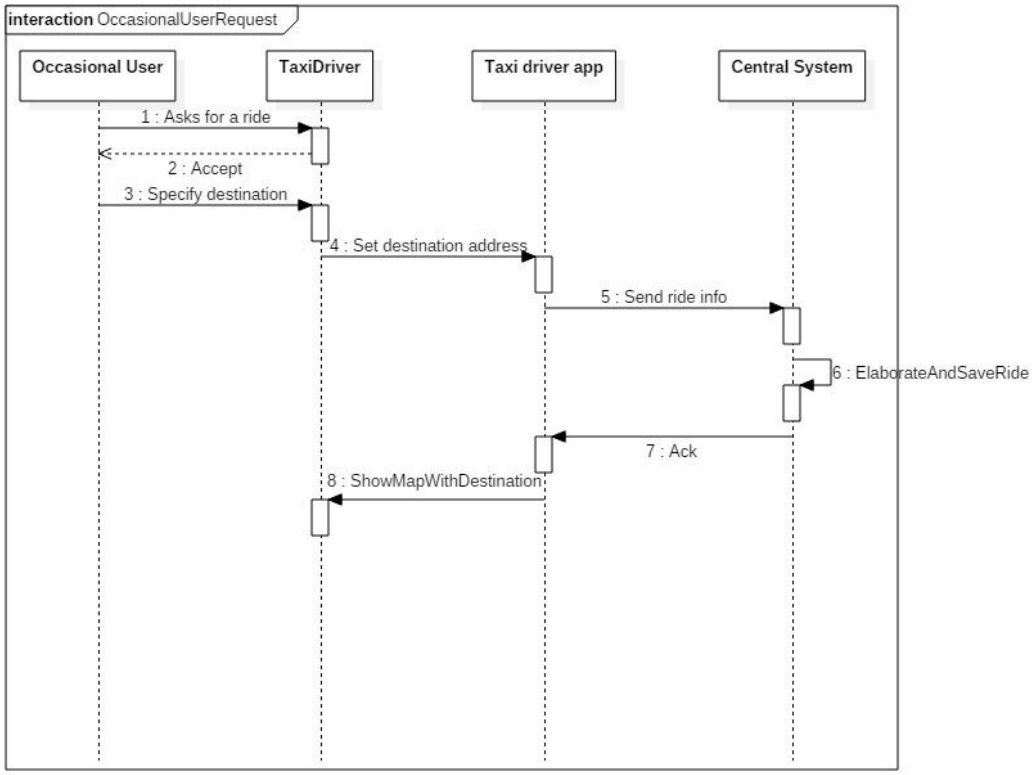

LoggedInRegCustomer
Read reservation details
«include»
Reserve a Taxi
«include»
«include»
User App
Fill reservation Form
Send Data
+participate
Invalid data
«extend»
«extend»
System error
+participate
Verify ride data
«include»
Save ride data
Central System


interaction Future Reservation
LoggedIn RegCustomer
Application
Central System
1 : ShowReservationPage
2 : PageShown
seq Reservation Request
[loop until systemError=true ||
invalidData=true)&& retry=true
3 : FillResForm
4 : SendReservationData

Occasional Ride

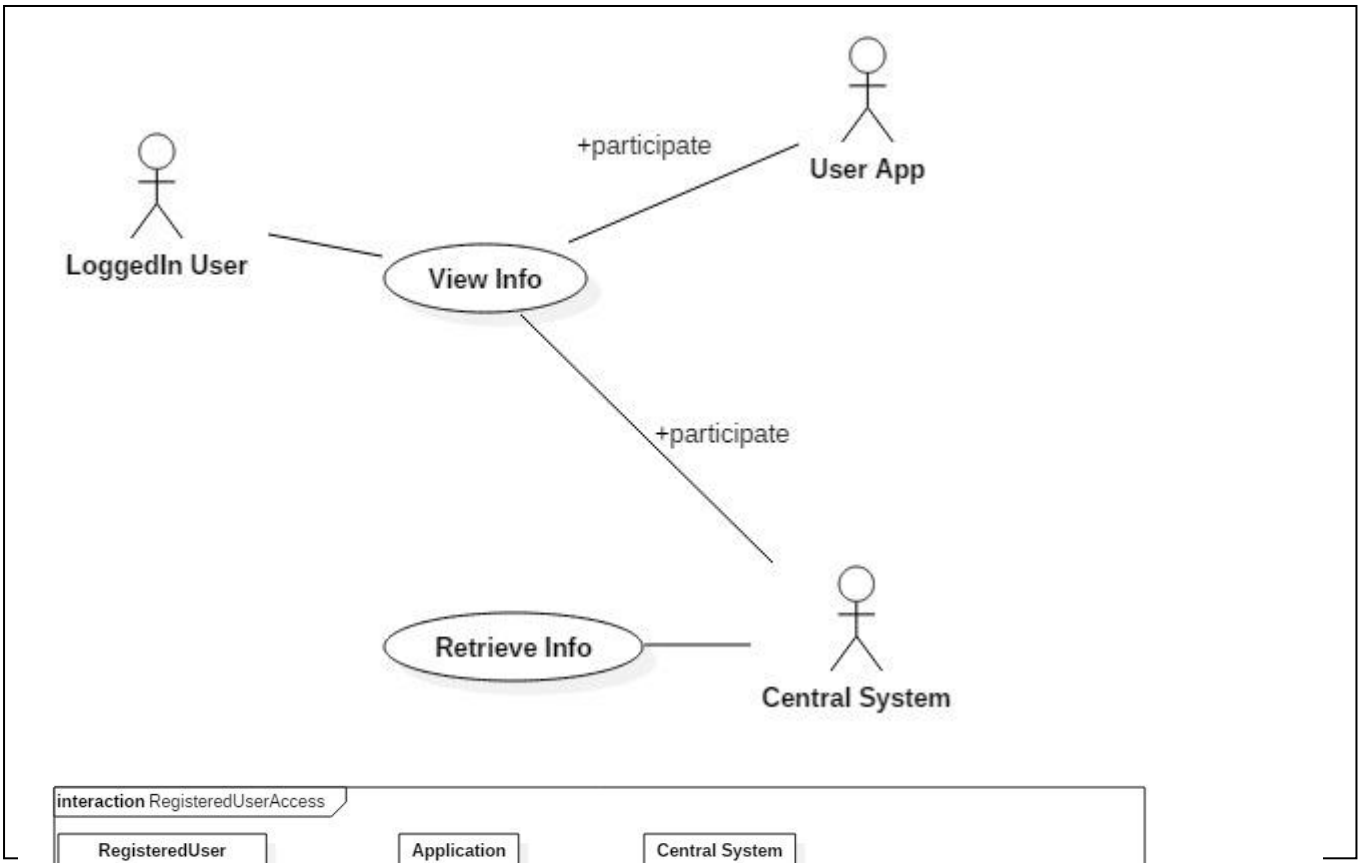
Actor	Customer, Central System, Taxi Driver, Taxi Driver App
Goal	MG9
Pre- Condition	NULL
Event Flow	1.The user sees a taxi and asks the taxi driver to pick him up. 2.The user specifies the destination to the taxi driver 3.The taxi driver specifies the destination in the mobile app. 4.A message is sent to the central system 5.The user pays the taxi driver at the end of the ride 6.The ride price is registered in the taximeter
Post- Condition	1.The customer has reached the destination.
Exception	NULL

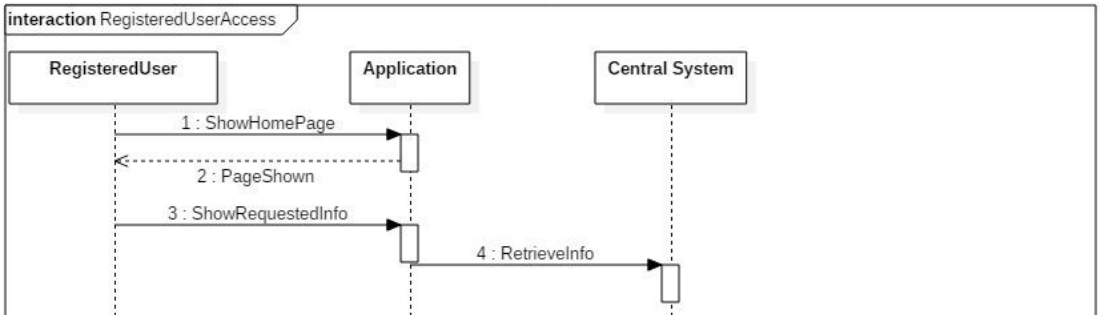




Access generic data

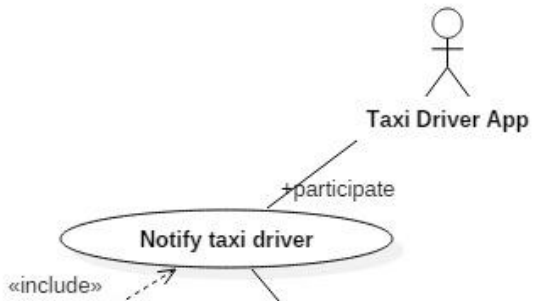
Actor	Customer, User app, Central System
Goal	CG10
Pre- Condition	The user is already logged in

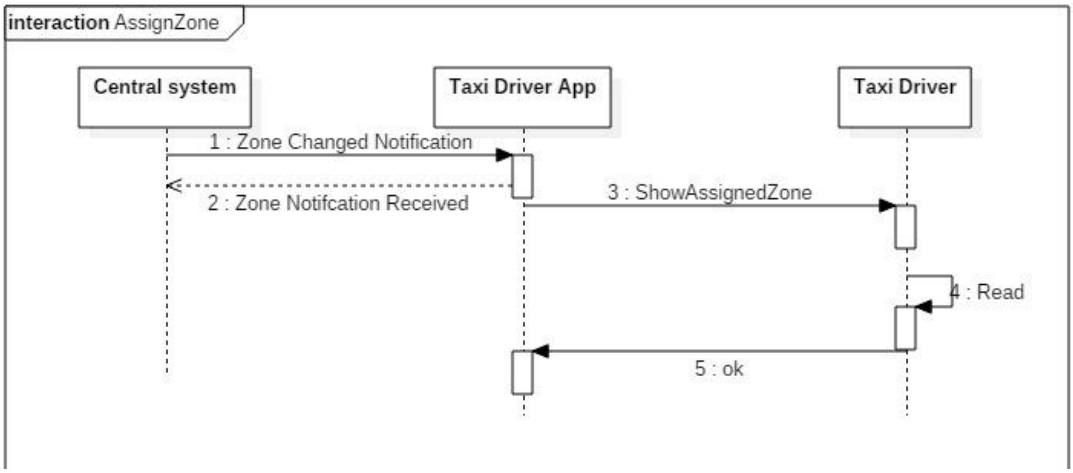

+participate
User App
LoggedIn User
View Info
+participate
Retrieve Info
Central System


interaction RegisteredUserAccess
RegisteredUser
Application
Central System
1 : ShowHomePage
2 : PageShown
3 : ShowRequestedInfo
4 : RetrieveInfo

Assign Zone

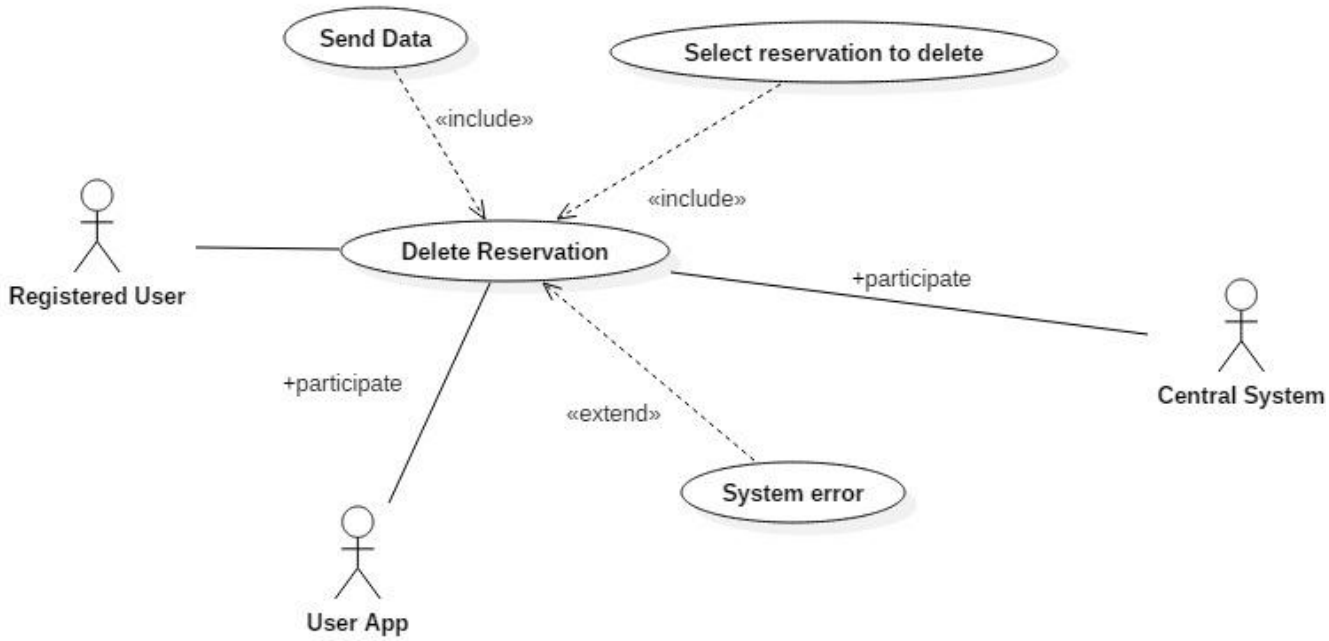
Actor	Taxi driver app, Taxi driver, Central System
Goal	MG8
Pre- Condition	The user is already logged in
Event Flow	1.Arrives a notification to the taxi driver in which is specified the zone to reach 2.The taxi driver reaches the zone 3.The taxi driver clicks on the app the button “ok” and informs the central system that he’s arrived to the zone.
Post- Condition	1.The central system has informed the taxi driver about his zone to reach
Exception	NULL

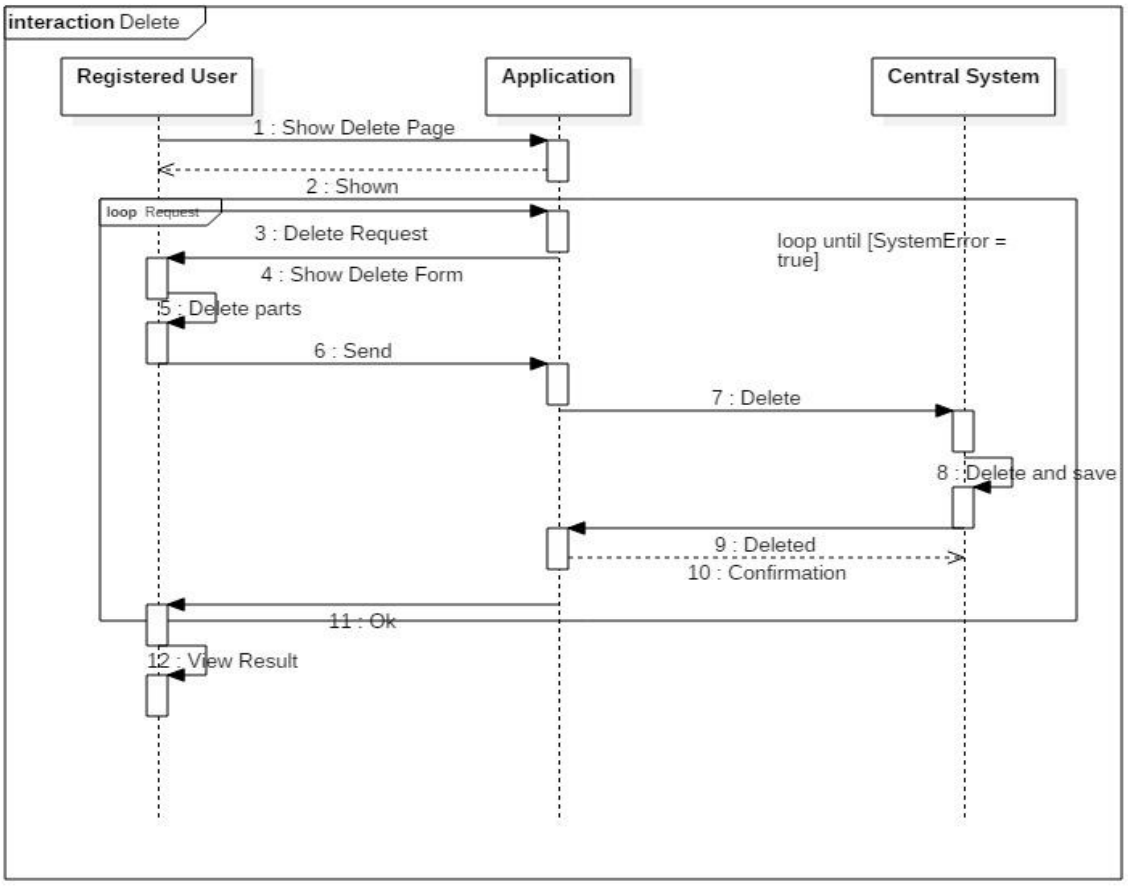




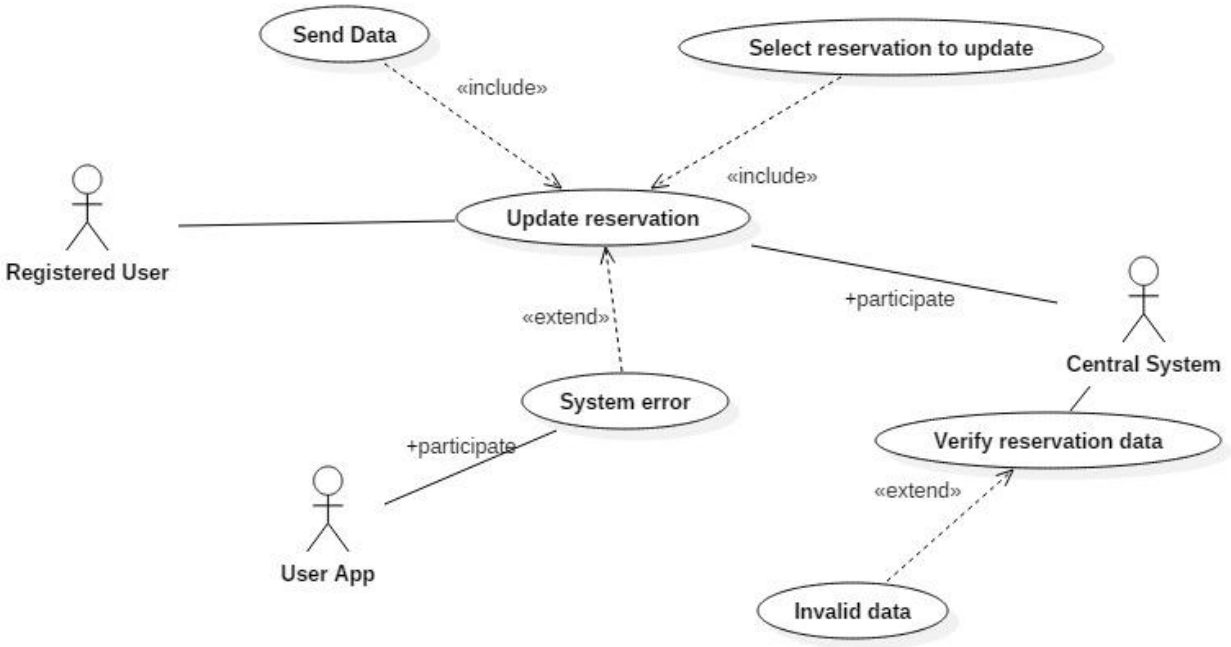
Delete Reservation

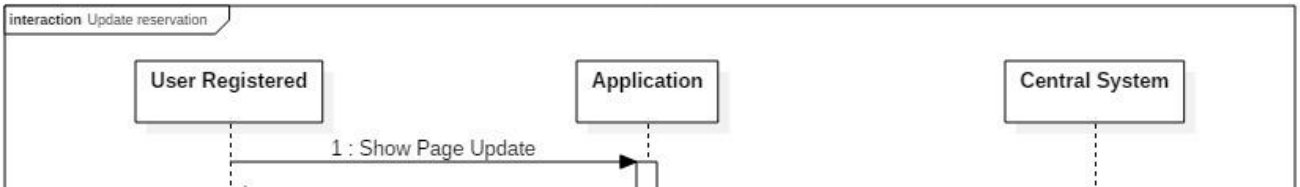
Actor	Registered User, Central System, User App
Goal	CG5
Pre- Condition	The user is already logged in
Event Flow	1.The user opens the past reservations' page. 2.The user select the reservation to delete 3.The user deletes the reservation
Post- Condition	1.The reservation is deleted from the system
Exception	2.System error: an unexpected error occurred





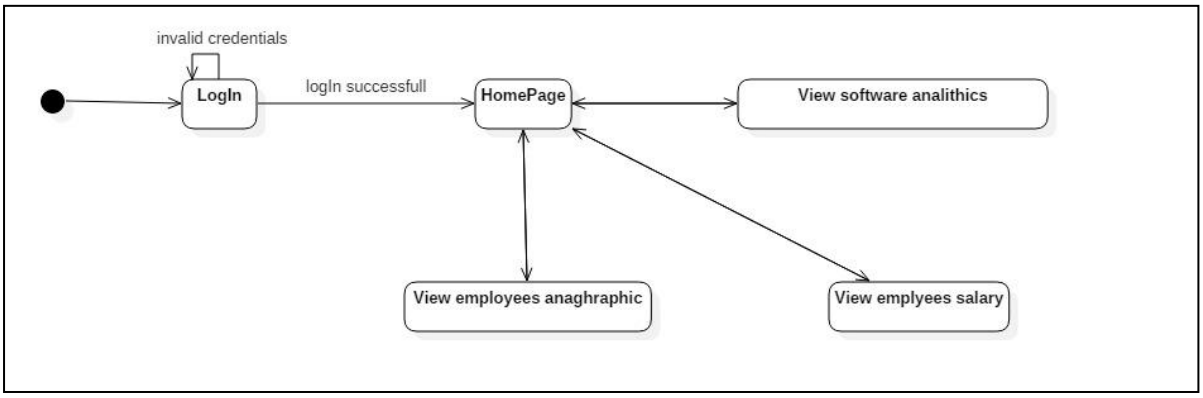
Actor	Registered User, Central System, User App
Goal	CG11
Pre- Condition	The user is already logged in
Event Flow	1 The user opens the past


Send Data
Select reservation to update
«include»
«include»
Registered User
Update reservation
+participate
Central System
«extend»
System error
Verify reservation data
+participate
«extend»
User App
Invalid data

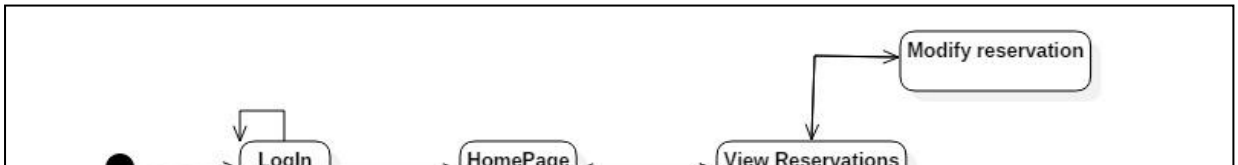

interaction Update reservation
User Registered
Application
Central System
1 : Show Page Update

STATE DIAGRAMS

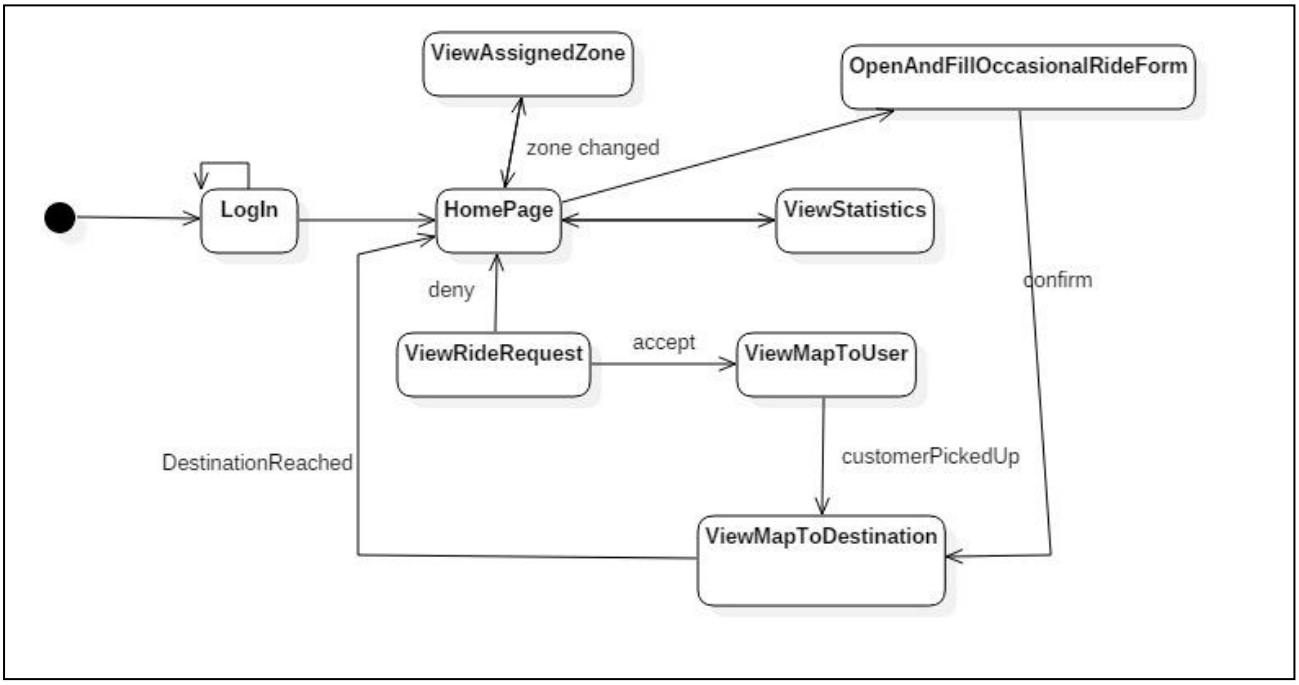
Government employee side



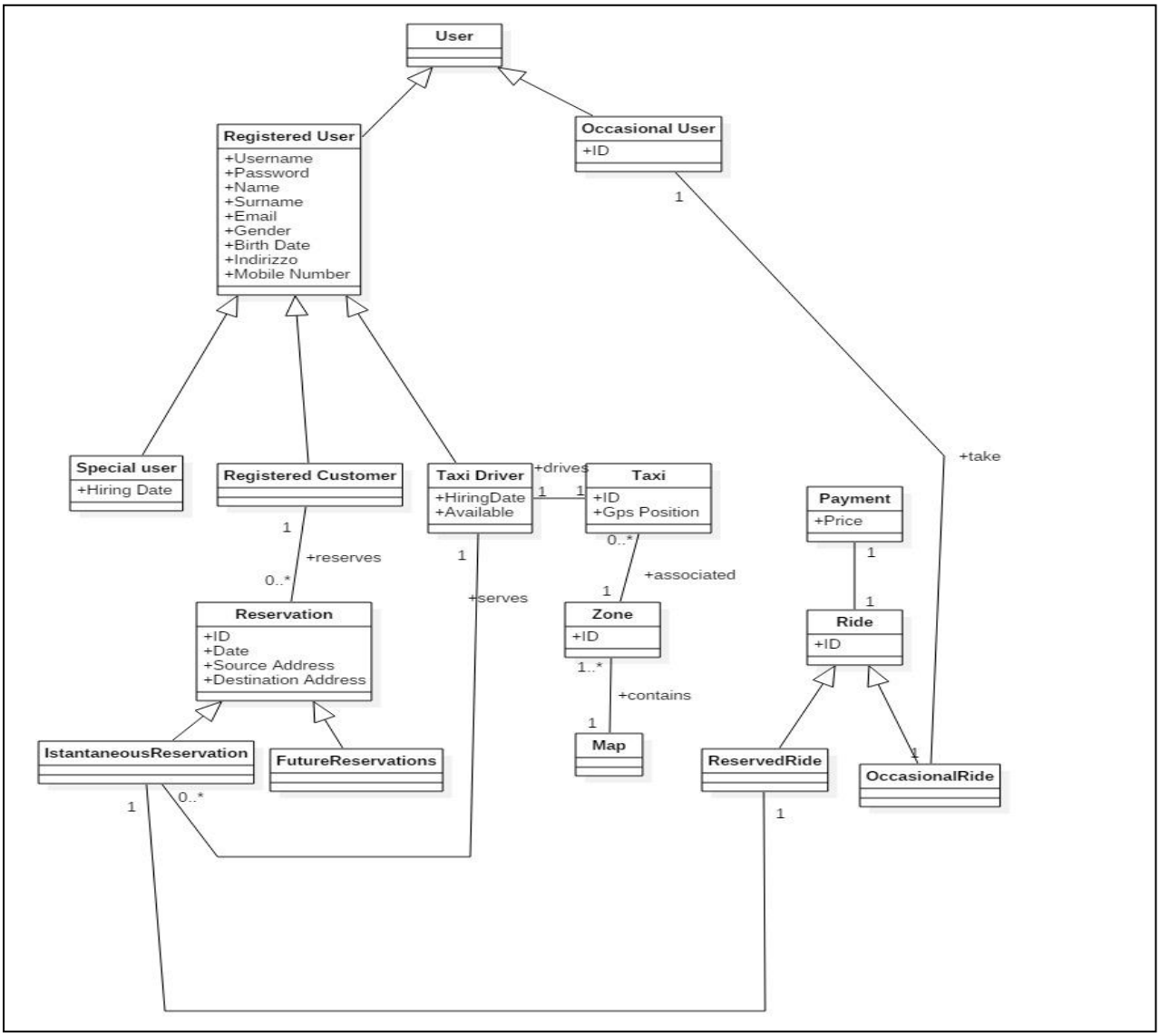
Client side



Taxi Driver Side



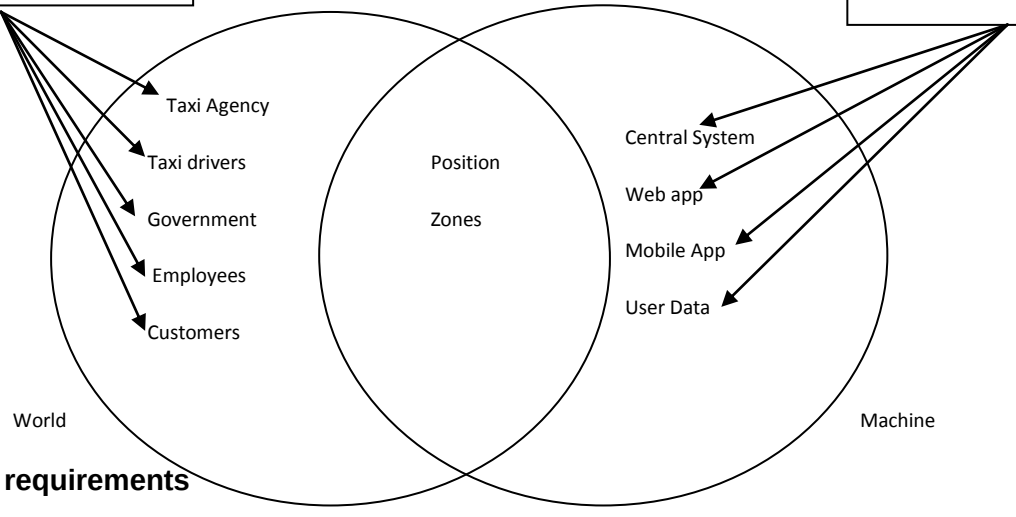
Class diagram



The world and the machine

Derived from actor,stake holders and actual system sections

Derived from the description of the system to be (see Scope section)



Client app requirements

-CR: Common requirement (for web app and mobile app)

-MR: Mobile requirement

-WR: Web app requirement

[CG1] A user can login in the system by writing his credentials.(username and password)

- [R1] A user must remember his username and password.
- [R2] The user must insert his credentials
- [R3] Wrong credentials will not allow the user to access the app.
- [R4] Application will implement the retrieve password mechanism. The retrieve mechanism is done in this way: The user clicks the button to retrieve the password and the app will send an email with a temporary password to access his profile and reset a new password.

[CG2] A user can register himself to the system if he hasn't an account.

- [R1] The user will have to insert a username that doesn't already exists in the system.
- [R2] After a successful registration process the app must show the login page
- [R3] The email address specified in the registration process must be valid. So the app must verify first if the syntax of the mail is correct,after that (if the checks is positive) a verification message (whith a confirmation link) is sent to the specified email address and the account will be active only after that the user has clicked on the confirmation link.
- [R4] The password must have at least 8 of length and must contain at least a special character,a capital letter and a number

[CG3] A user can book an instantaneous taxi ride

- [CR1] The user must be logged in (see **G1**)
- [WR2] The user must specify the start address and the destination address
- [MR2] The user can only specify the destination address if the GPS service is available and can automatically set the source of the ride.
- [CR1] The app must send the specified data to the central system and display a confirmation dialog to advertise the user if all went fine or some errors occurred
- [CR2] When the central system has selected the taxi driver for the user's taxi ride (or no taxi driver was available) the app must display a notification icon so that the user can see the notification message
- [CR3] The user must be able to view the information in the notification to see: if an error happened what kind of error was and if was successfull the incoming taxi code and the waiting time
- [MR3] The user must be able to view the information in the notification to see: if an error happened what kind of error was and if was successfull the incoming taxi code and the waiting time and the position of the incoming taxi on a map

[CG4] A user can book a future taxi ride

- [CR1] The user must be logged in (see **G1**)
- [WR2] The user must specify the start address and the destination address and the date and time that he wants he taxi. The app must check if the actual time is at least two hours before than the date and time specified for the ride.

- [MR2] The user must only specify the destination address if the GPS service is available and can automatically set the source of the ride,otherwise he will have to specify both the source address and destination address manually. The app must check if the actual time is at least two hours before than the date and time specified for the ride.
- [CR1] The app must send the specified data to the central system and display a confirmation dialog to advertise the user if all went fine or some errors occurred
- [WR2] When the central system has selected the taxi driver for the user's taxi ride (or no taxi driver was available) the app must display a notification icon so that the user can see the notification message
- [MR2] When the central system has selected the taxi driver for the user's taxi ride (or no taxi driver was available) the app must show a notification so that the user can open the app and go directly to the notification information page (see CR3)
- [CR3] The user must be able to view the information in the notification to see: if an error happened what kind of error was and if was successful the incoming taxi code and the waiting time and the position of the incoming taxi on a map

[CG5] A user can cancel a previous reserved taxi ride.

- [CR1] The user must be logged in (see **G1**)
- [CR2] The user can open the page to see all the reserved taxi ride from the moment he registered to the service. In this list the user selects the ride that will be marked graphically as "FUTURE" and will click on the cancel button. A confirmation dialog will pop up and will ask to the user if he really wants to cancel the reservation. If the answer is positive,then the app sends the request to the central system and the app will show the success or not of the operation.
- [CR3] After the cancellation the app will automatically refresh the reservation,so that a specific reservation has been cancelled it won't be shown anymore in the list.

[CG6] A user can view his previous reservations

- [CR1] The user must be logged in (see **G1**)
- [CR2] The user can view his taxi reservations in the past reservation page by clicking the button to open that page
- [CR3] The list of reservations must be sorted by date in a descending way so that the future reservations will be the first in the list and to on to the latest reservations to the oldest ones.
- [CR4] The future reservations must be graphically marked as "FUTURE"
- [CR5] The reservation with a taxi arriving to pick up the user must be marked as "ARRIVING"
- [CR6] The past reservations must be graphically marked as "PAST"

[CG7] A user can view his position and the incoming taxi driver position.

- [CR1] The user must be logged in (see **G1**)
- [CR2] The user must be able to access a map that shows his location and the location of the taxi driver

[MG8] When the system has assigned to the customer a taxi the mobile app must notify this fact to the user so he can view how much time he will have to wait for the taxi.

- [MR1] The mobile app shows the notification.

- [MR2] The user opens the app and he's already logged in.
- [MR3] The user must be able to see the incoming taxi details.

[CG9]A user can see his personal info

- [CR1] The user must be logged in.
- [CR2] The user must be able to open his personal page.

[CG10] A user can update a previous made reservation

- [CR1] The user must be logged in.
- [CR2] The user opens the past reservations' page.
- [CR3] The user selects the reservation to update.
- [CR4] The user must be able to update the reservation.

[WG11] A user sees the notifications of incoming taxi when he opens the web page

- [WR1] The user must be logged in.
- [WR2] The user must be able to see all the notifications
- [WR3] The user must be able to click on each notification and to see the details.

Taxi driver app requirement

[MG1] The taxi driver can login himself to the system

- [MR1] The taxi driver must enter his username and the password.
- [MR2] The logIn process must verify that the username and password are correct. If they are then the home page is shown,otherwise a "wrong credentials" error is shown.
- [MR3] If any remote error occurs during the logIn process the user must be notified with a pop up message.

[MG2] The taxi driver can view his past served rides

- [MR1] The taxi driver must be already logged in (see **G1**)
- [MR2] The taxi driver can view his past served rides ordered by date, price paid by the customer and the Kilometers of the ride length.

[MG3] The taxi driver can accept or deny the customer requests

- [MR1] The taxi driver must be already logged in (see G1)

- [MR2] When a request is dispatched by the central system to the app,the accept/deny interface must be shown.
- [MR3] If the taxi drivers refuses the request the app returns to the home page. If the taxi drivers accept a map is shown with the starting position of the taxi and the source address specified by the customer needed to be picked up,also a timer starts to track the time
- [MR4] When the taxi driver has picked up the user he must click on the button “Customer picked up” and the map must be refreshed with a new map with the starting point the pick up point and as destination point the destination specified by the user. Also a message is sent to the central system and notifies the time that it took to go to the user.
- [MR5] When the taxi driver arrived to the destination the taxi driver clicks on the “arrived” button and a message is sent to the central system to say that the ride is over. After that the central system can propose another customer request or decide the zone in which the taxi driver will have to go.

[MG4] The application must periodically communicate the taxi driver position.

- [MR1] The taxi driver must be already logged in (see **G1**)
- [MR2] As soon as the taxi changes its position (based on gps events) the app must send the coordinates of the taxi in the map to the central system.
- [MR3] The update must be done in an automatic way and without taxi driver assistance.

[MG5] The taxi driver can set the occupied/available state

- [MR1] The taxi driver must be already logged in.
- [MR2] The taxi driver opens the home page.
- [MR3] The taxi driver must be able to set the state.

[MG6] The taxi driver can view how much he has already earned this month as variable month salary

- [MR1] The taxi driver must be already logged in (see **G1**)
- [MR2] The taxi driver opens the “past served rides” page (see **G2**) where each item is associated to the earned money for each ride (see assumptions of the variable month salary)

[MG7] The taxi driver can view the assigned zone by the system where he will have to go once a ride is over.

- [MR1] The taxi driver must be already logged in (see **G1**)
- [MR2] The zone chosen by the system is shown in a dialog that shows app when the message from the central system arrives after a ride is over (see G4).
- [MR3] When the taxi driver arrives to the assigned zone will click the button and set the free state. This state must be notified to the central system with a message.
- [MR4] The app must show the specified zone in the home page and keep it updated as soon as it changes

[MG8] The taxi driver can accept an occasional user ride

- [MR1] The taxi driver must be already logged in (see **G1)**

- [MR2] The taxi must be able to introduce the destination of the occasional user and to send the data to the system

Government app requirements

[WG1] The employee of the government can access as a special user to the central system

- [WR1] The government employee specifies his username and password
- [WR2] After the user is logged in the “special home page” is shown
- [WG2] The employee of the government can view the agency employees and their personal data
- [WR1] The employee must be logged in (see **G6**)
- [WR2] The employee can click on the “view employees data”.
- [WR3] The list in the “view employees data” shows the name,CF (and all relevant data) of all employees. This information can be sorted by name,CF,birth date or the hiring date

[WG3] The employee of the government can view the service statistics

- [WR1] The employee must be logged in (see **G6**)
- [WR2] The employee can click on the “view service statistics data”.
- [WR3] The employee can view the number of registered users in the service,the number of rides/user,the number of users served by each taxi driver and this list can be sorted with some criteria(number of users served by each taxi driver)

[WG4] The employee of the government can view the worker statistics

- [WR1] The employee must be logged in (see **G6**)
- [WR2] The employee can click on the “view the worker statistics”.
- [WR3] The employee can view the hours worked by each worker,the variable part of the salary per month already earned and other relevant data and this list can be sorted with some criteria(for example the descending working hours value)

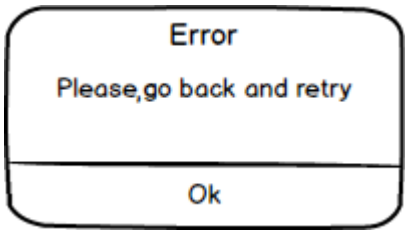
Non-functional requirements

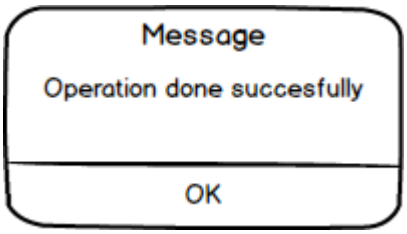
The response time of the application for every remote operation the user (with perfect and ideal internet connection) must not wait more than 2 seconds.

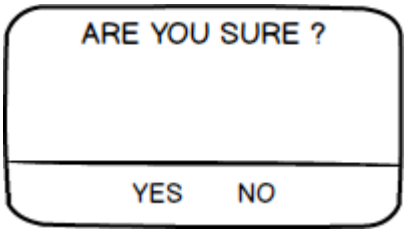
User Interfaces

Common dialogs

These dialogs are common for mobile and web app



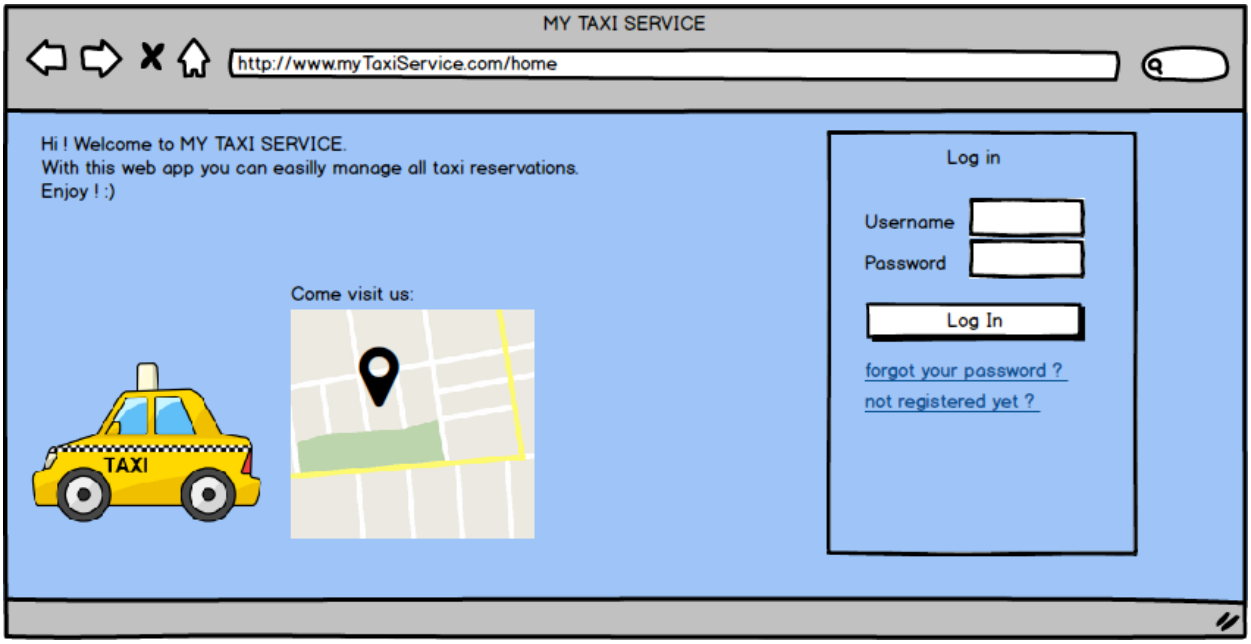




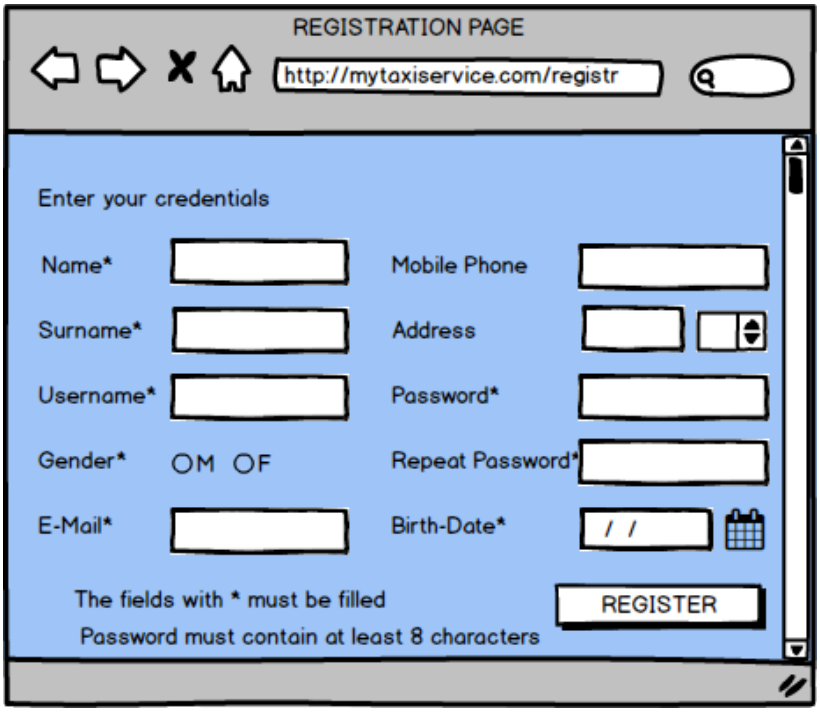
When the user taps on the delete button this message pops up and asks a confirmation before deleting. If he chooses YES,than the deleting message is sent to the central system and after the operation has completed one of the two confirmation result is displayed.

Web app

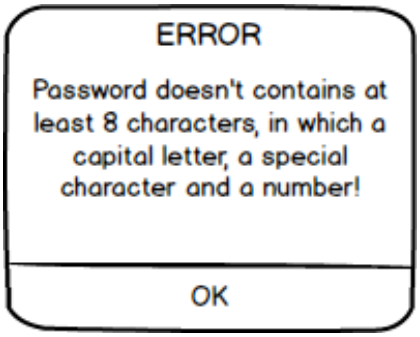
Home Page



Registration page

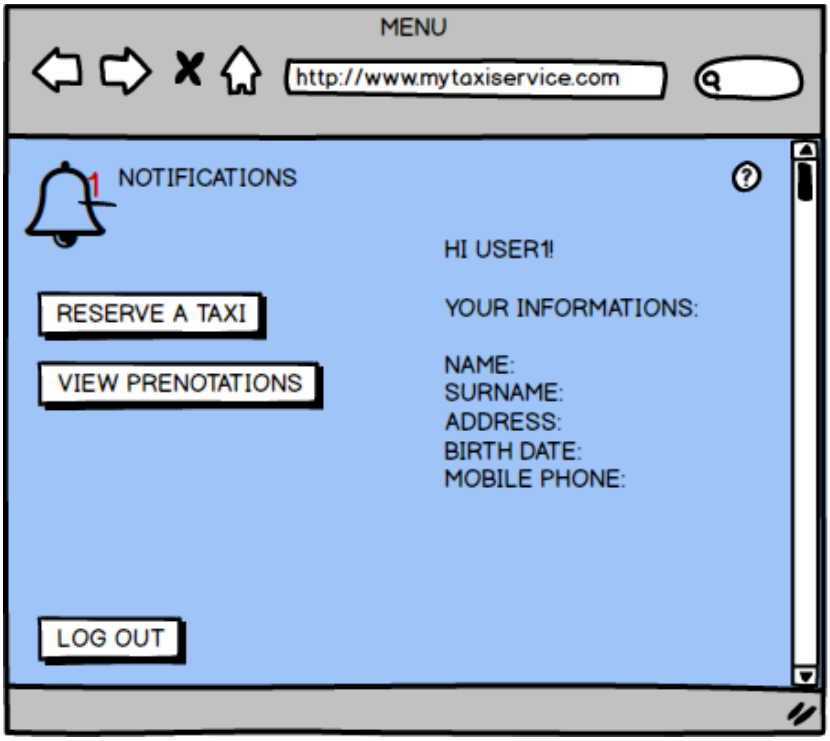


If the password isn't conform to the security policies this message is shown (after the user clicks on register):



With this page a client can register himself.

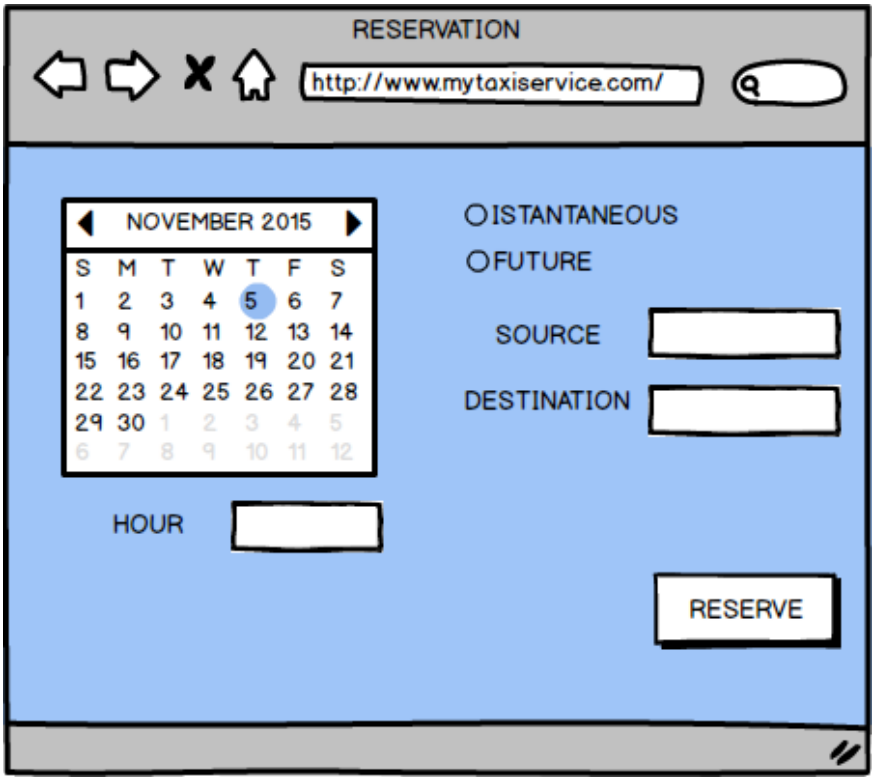
Personal registered customer page



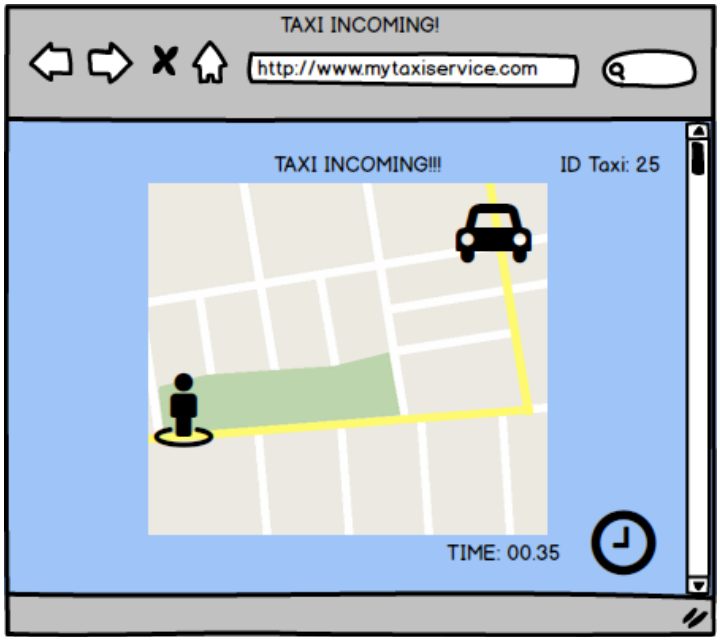
This is the page opened when a registered customer correctly log in into the system

Now all the functionalities given to the registered customer are shown:

Registered customer-Taxi reservation

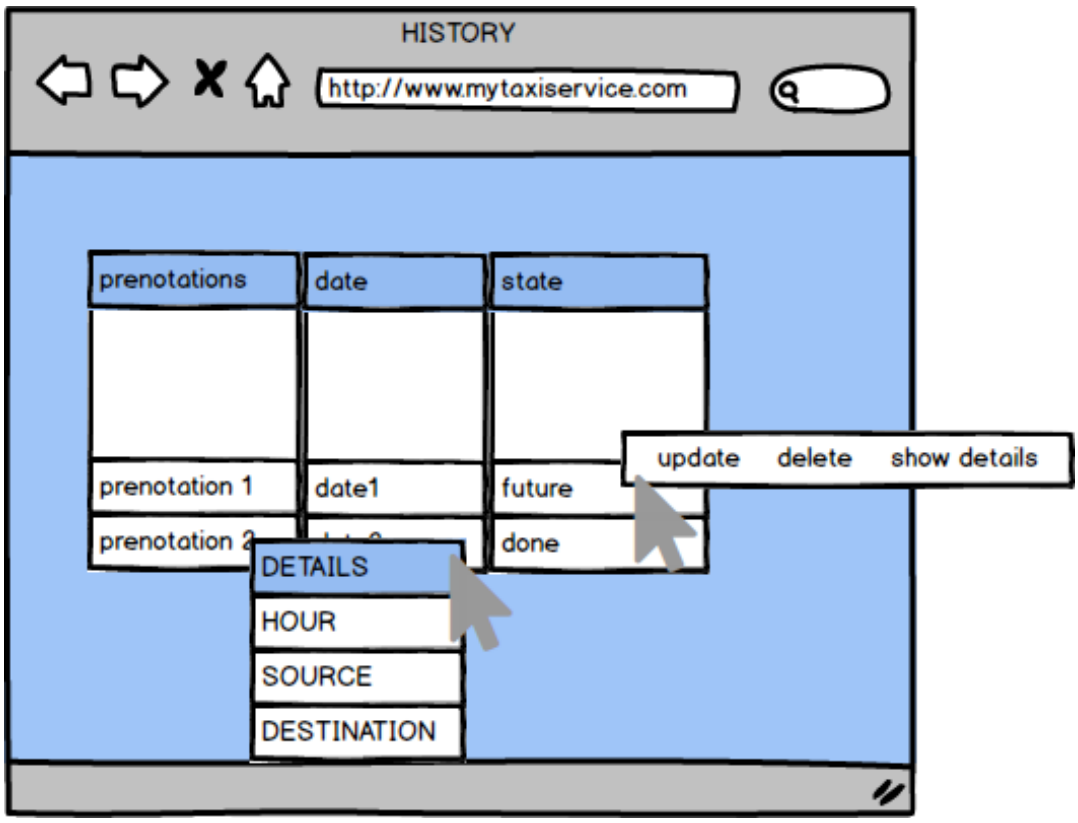


If the reservation is instantaneous ,the position (updated as soon as the taxi changes his position) of the incoming taxi is shown:



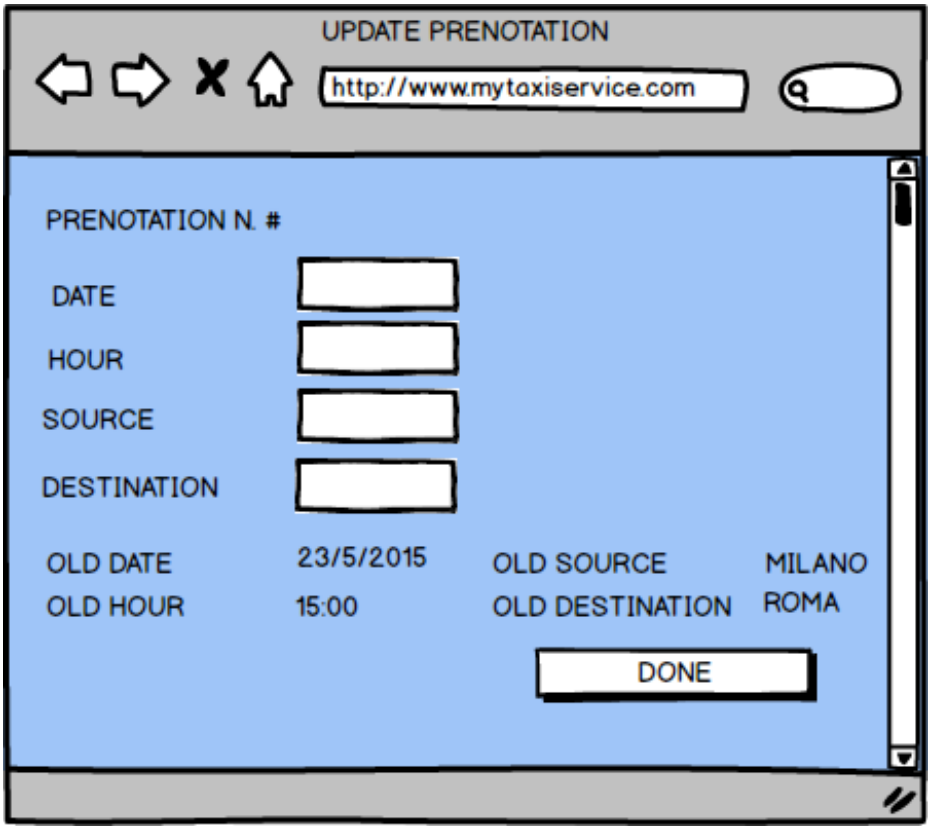
With this form the user can also know the estimated time that takes to the taxi to arrive to the client.

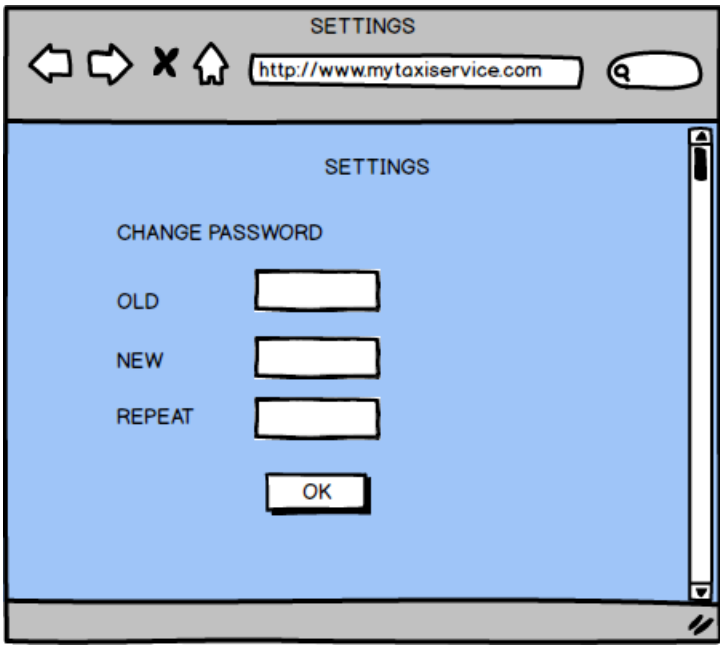
If the reservation is future,a confirmation dialog (error or done successfully) is shown and 10 minutes before the specified time it is converted to an instantaneous reservation and the logic and interfaces are the same as the instantaneous one shown before.



In this page the registered customer can see all the reservations done.

Registered customer page – view reservations – Update



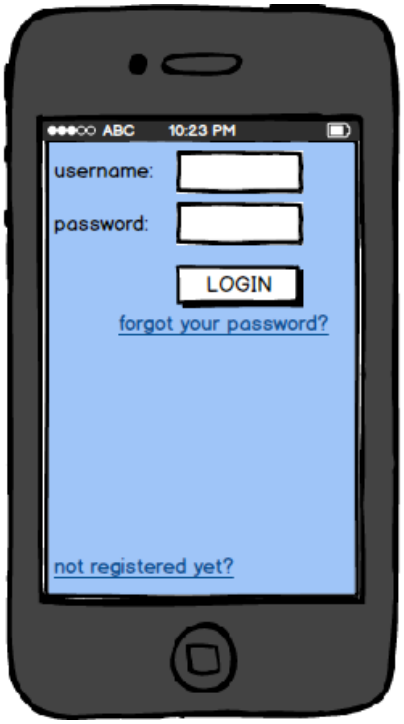


Client mobile app

Registration

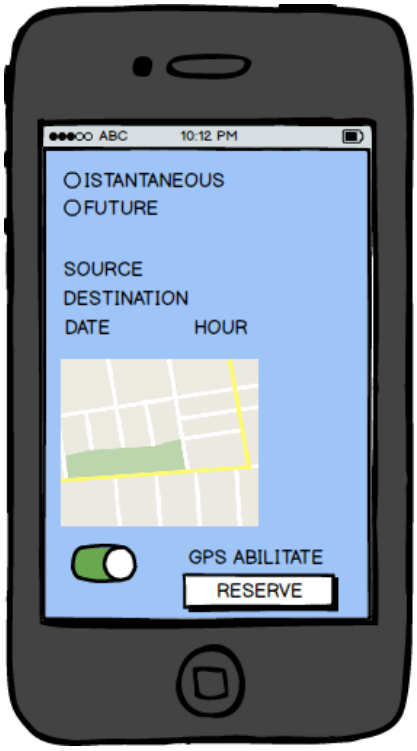
The registration is done by opening the corresponding web page.This is because if in a second moment the required data changes the web app is changed more quickly than changing the client mobile app.

Log In



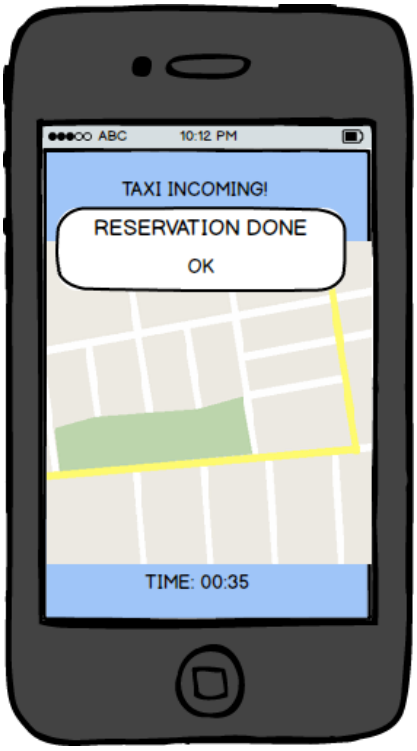
Home

Make a reservation

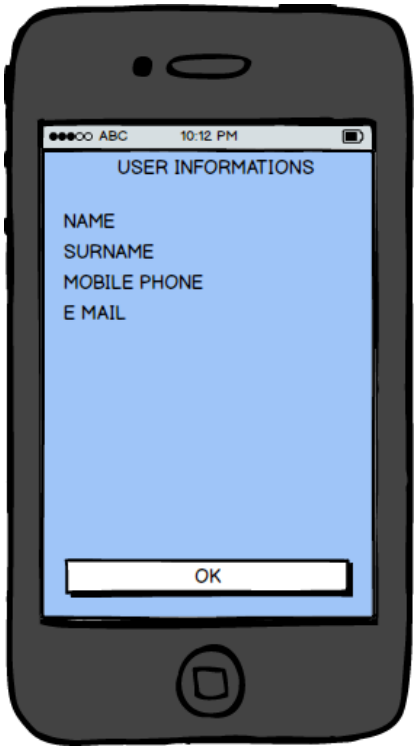


istantaneous

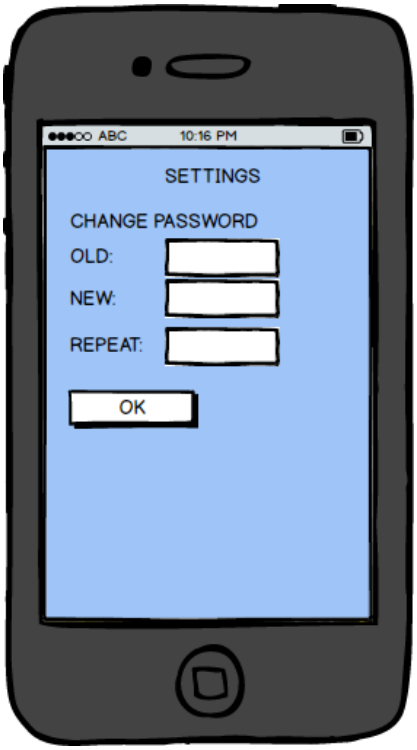
Show incoming taxi



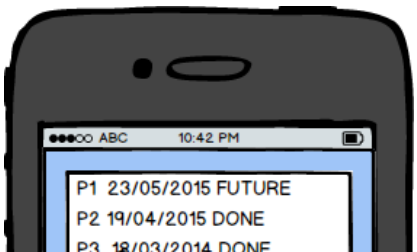
Profile information



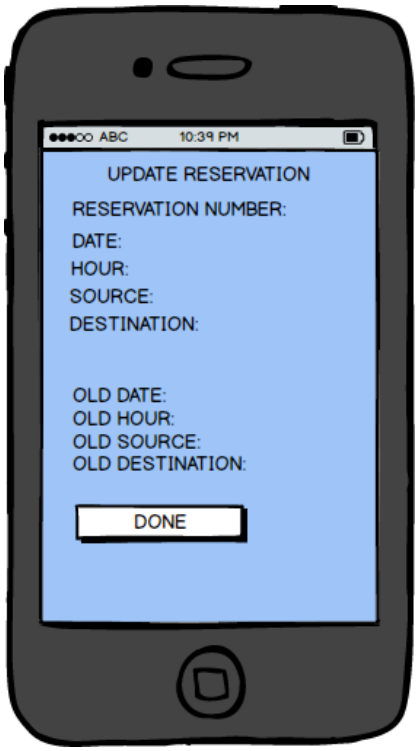
Settings/ChangePassword



Past reservations

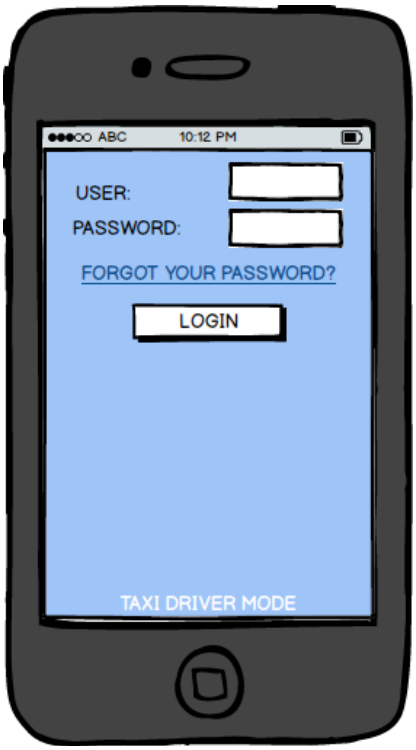


Update reservation



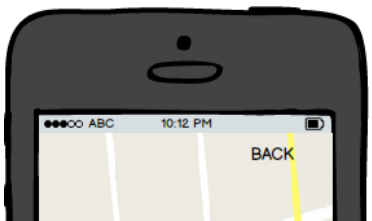
Taxi driver mobile app

Log In

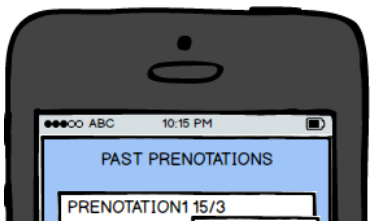


Home Page

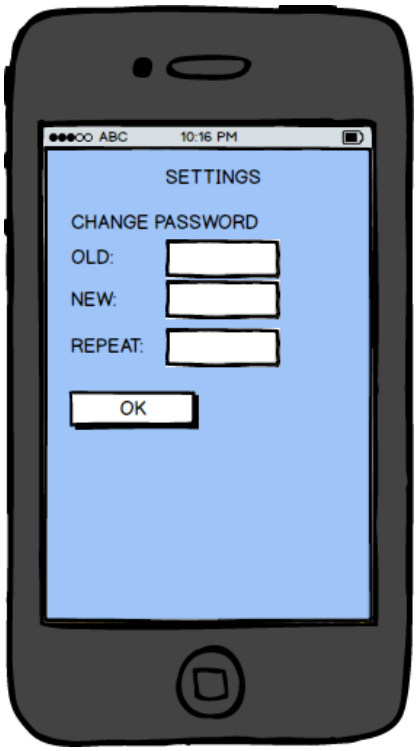
Where I Am



View past reservation



Settings



Accept/Deny reservation

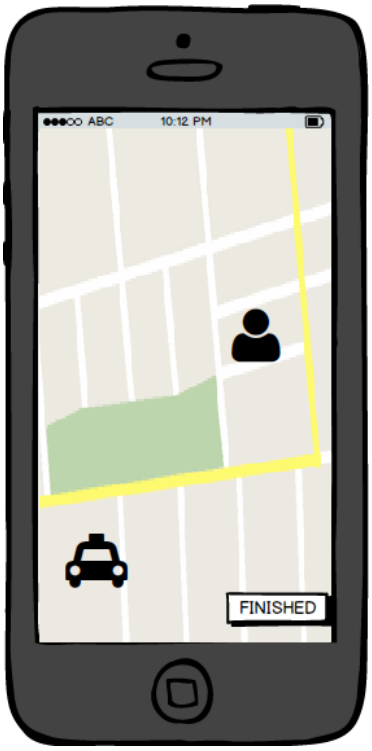
After a taxi driver accepts there are these 2 different phases:

The taxi driver reaches the destionation specified by the client:

The taxi driver goes to pick up the client:

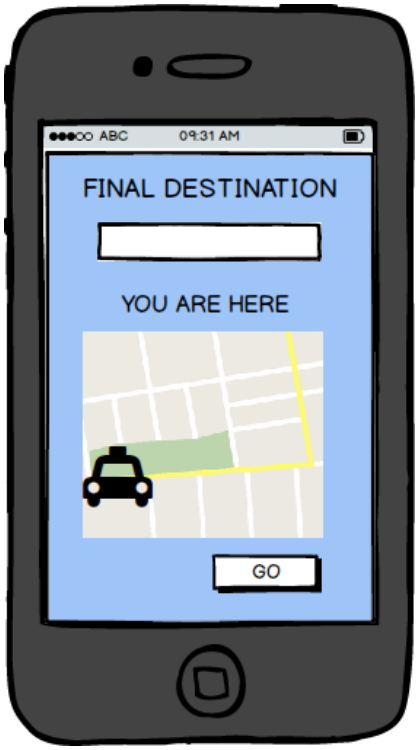
Picking up the client

Reaching the destination



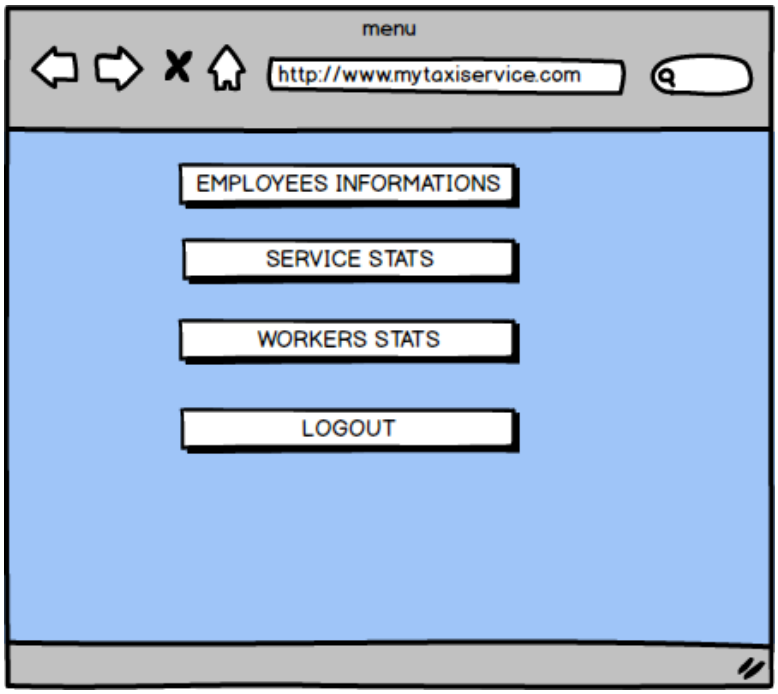
Go to specified zone

Occasional ride data



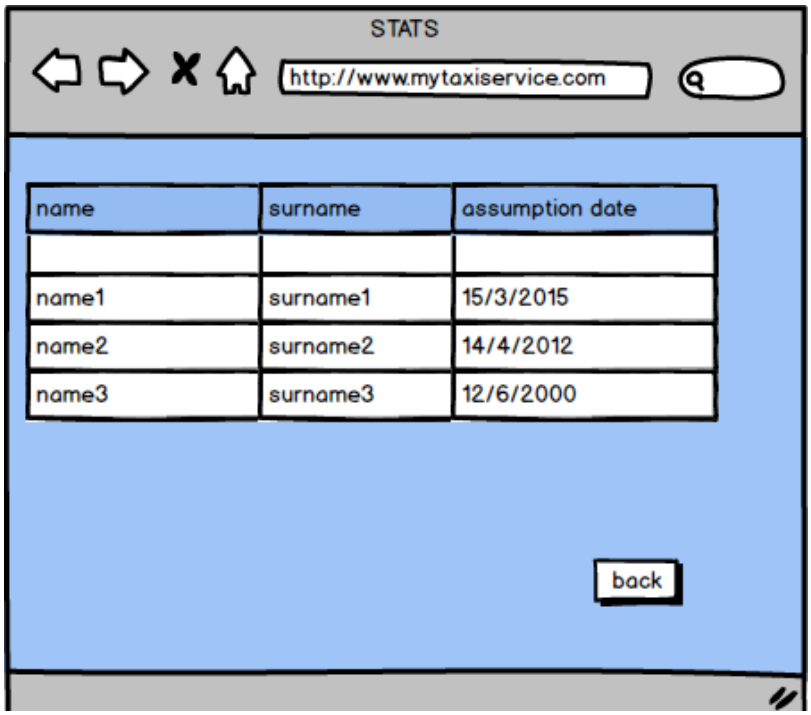
Government employee app

Personal special user page - Home



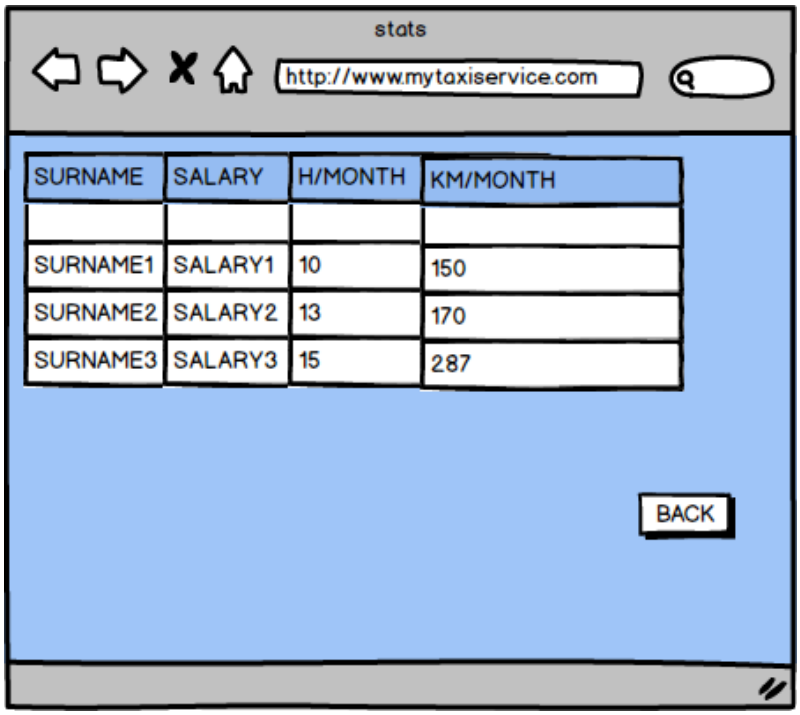
This is the personal page opened once a government employee (special user) log into the system. Now each functionality will be shown in details.

Personal special user page – Employees information



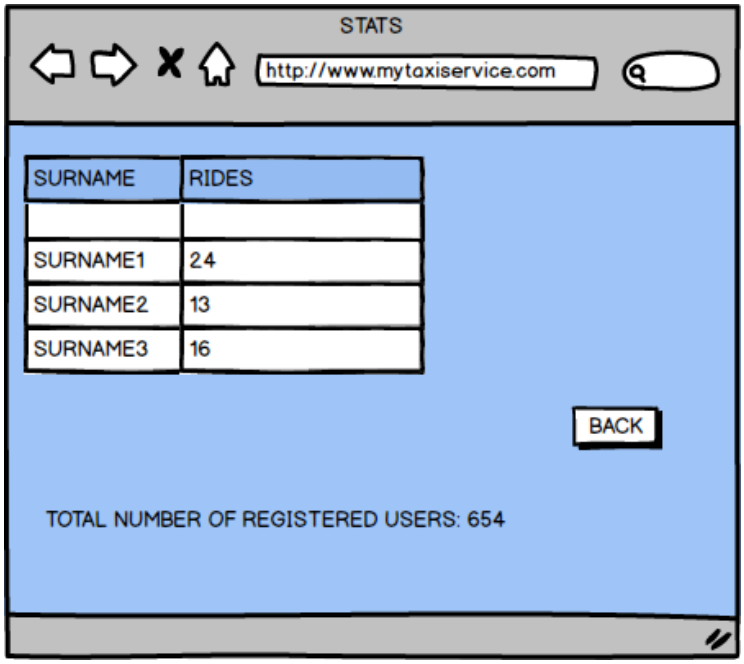
In this form information about taxi drivers is shown.

Personal special user page –Worker Stats



This form shows some important values to mesure efficiency of taxi drivers and consequently to the service and the agency itself (based on salary of taxi drivers)

Personal special user page – Service Stats



This form helps to evaluate the usage of the system by the clients

Software system attributes

Maintainability

The maintainability of the system is granted by the technical administrator of the system and the future extension is granted by the API's which we used to extend the system and that will be used to implement the functionalities in the future implementations

Security

The login authentication is used to guarantee that the users of the system are certified users and not anyone can access to the system. The hash of the password is stored for better security.
Our systems also have the concept of strong password which requires at least 8 characters a capital letter,a number and a special character. There is also the password retrival and password change.

Alloy

Complete Code

```
abstract sig User{}

enum Gender{M,F}


sig string,boolean{}


sig Date{
   Day: one Int,
   Month: one Int,
   Year: one Int
}


sig RegisteredUser extends User{
  Username: one string,
  Password: one string,
  Name: one string,
  Surname: one string,
```

```
  Email: one string,
  Gender: one Gender,
  BirthDate: one Date,
  HouseAddress: lone string,
  MobileNumber: one string
}

sig OccasionalUser extends User{
   ID: one Int
}

sig Taxi {
  ID: one Int,
  Position : set string,
  TaxiZone: one Zone
}

sig TaxiDriver extends  RegisteredUser{
      taxi: one Taxi,
      HiringDate: one Date,
      Available: one boolean,
      reservations: some IstantaneousReservation
}

sig Customer extends RegisteredUser{
      reservations: some Reservation,
}

sig SpecialUser extends RegisteredUser{
```

```
 HiringDate: one Date
}


abstract sig Reservation {
   ID: one Int,
   ResDate: one Date,
   SourceAddress: one string,
   DestinationAddress: one string,
}

sig IstantaneousReservation extends Reservation{
}

sig FutureReservation extends Reservation{
}

sig Payment{
  Price: one Int
}

abstract sig Ride{
  ID: one Int,
  paymentInfo: one Payment
}

sig ReservedRide extends Ride
{
    associatedReservation: one IstantaneousReservation,
```

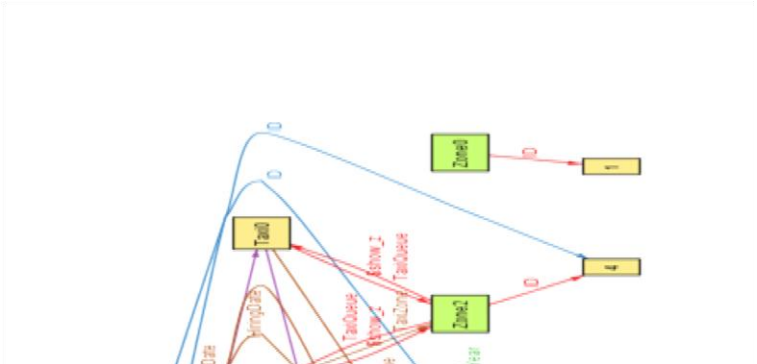
```
}

sig OccasionalRide extends Ride
{
   takenBy: one OccasionalUser
}

//Vincoli

//User
fact uniqueUserID{
no ru1,ru2: RegisteredUser | (ru1!=ru2 and ru1.Username=ru2.Username)
}

//Taxi Driver
fact oneTaxiToOneTaxiDriver{
no t1,t2: TaxiDriver | (t1!=t2 and t1.taxi=t2.taxi)
}


//Taxi - Taxi Driver
fact taxiAssociation{
#TaxiDriver = #Taxi
}


//Taxi

//unique ID
```

```
fact uniqueTaxiID{
no t1,t2: Taxi | (t1!=t2 and t1.ID=t2.ID)
}

fact postiveTaxiID{
no t1: Taxi | t1.ID<0
}


fact existsZoneForTaxi{
no t: Taxi,z1,z2:Zone | (z1!=z2) and (t in z1.TaxiQueue and t in z2.TaxiQueue)

}

fact uniqueZoneID{
no z1,z2: Zone | (z1!=z2 and z1.ID=z2.ID)
}

fact postiveZoneID{
no z1: Zone | z1.ID<0
}

//Occasional User
fact uniqueOccasionalUserID{
no ou1,ou2: OccasionalUser | (ou1!=ou2 and ou1.ID=ou2.ID)
}

fact postiveOccUserID{
no ou1: OccasionalUser | ou1.ID<0
```

```
}

//Ride

fact uniqueRideID{
no r1,r2: Ride| (r1!=r2 and r1.ID=r2.ID)
}

fact postiveRideID{
no r1: Ride | r1.ID<0
}

//Reservation

fact uniqueReservationID{
no r1,r2: Reservation| (r1!=r2 and r1.ID=r2.ID)
}

fact differentAddresses{
no r1: Reservation | r1.SourceAddress=r1.DestinationAddress
}

fact postiveReservationID{
no r1: Reservation| r1.ID<0
}


//Payment
fact PositivePrice{
```

```
no p1: Payment | p1.Price<0

}


//Date


//we suppose that february has always 28 days

fact validDate{

   no d1: Date | ( (d1.Month = 1 or d1.Month = 3 or d1.Month = 5 or d1.Month = 7 or
d1.Month=10 or d1.Month=12) =>

      (d1.Day >0 and d1.Day <=31)) and ((d1.Month=4 or d1.Month=6 or d1.Month = 9 or
d1.Month=11) => (d1.Day >0 and d1.Day<=30))

     and ((d1.Month=2) => (d1.Day>0 and d1.Day <=28)) and d1.Month>=1 and d1.Month<=12
and d1.Year>0 and d1.Day>0




}


fact noRideBeforeAssumption{

no t1: TaxiDriver, ir1: IstantaneousReservation | (ir1.ResDate.Year>t1.HiringDate.Year) or
((ir1.ResDate.Year = t1.HiringDate.Year) and (ir1.ResDate.Month>t1.HiringDate.Month))

or((ir1.ResDate.Year)=(t1.HiringDate.Year) and (ir1.ResDate.Month = t1.HiringDate.Month)
and (ir1.ResDate.Day>t1.HiringDate.Day))


}




fact OneAtLeastZone{

no t: Taxi | (all z: Zone |( t not in z.TaxiQueue))

}
```

```
pred show(){
#TaxiDriver>1
#User>1
#Reservation >1
#Ride>1
}

run show for 3 but 2 TaxiDriver, 2 User
```



Simplified Code

This is the code without some attributes that were irrelevant for the analysis and it was done to make a more readable alloy graphic

```
abstract sig User{}

enum Gender{M,F}


sig string,boolean{}


sig Date{
   Day: one Int,
   Month: one Int,
   Year: one Int
}


sig RegisteredUser extends User{
  Username: one string,
}


sig OccasionalUser extends User{
   ID: one Int
}


sig Taxi {
  ID: one Int,
  Position : set string,
  TaxiZone: one Zone
}


sig TaxiDriver extends  RegisteredUser{
```

```
    taxi: one Taxi,
    HiringDate: one Date,
    Available: one boolean,
    reservations: some IstantaneousReservation
}

sig Customer extends RegisteredUser{
    reservations: some Reservation,
}

sig SpecialUser extends RegisteredUser{
 HiringDate: one Date
}


abstract sig Reservation {
  ID: one Int,
  ResDate: one Date,
  SourceAddress: one string,
  DestinationAddress: one string,
}

sig IstantaneousReservation extends Reservation{
}

sig FutureReservation extends Reservation{
}

sig Payment{
```

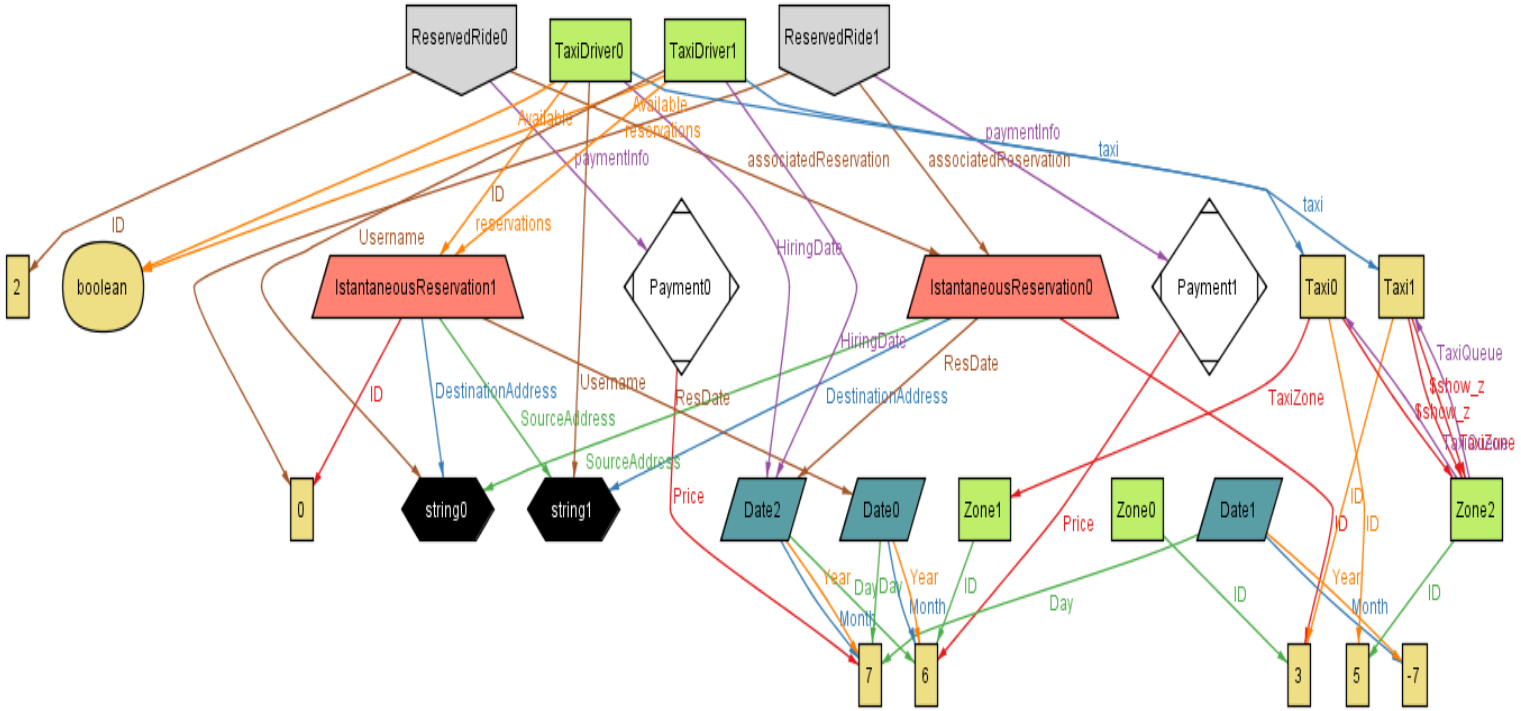
```
  Price: one Int
}

abstract sig Ride{
  ID: one Int,
  paymentInfo: one Payment
}

sig ReservedRide extends Ride
{
     associatedReservation: one IstantaneousReservation,
}

sig OccasionalRide extends Ride
{
   takenBy: one OccasionalUser
}

//Vincoli

//User
fact uniqueUserID{
no ru1,ru2: RegisteredUser | (ru1!=ru2 and ru1.Username=ru2.Username)
}

//Taxi Driver
fact oneTaxiToOneTaxiDriver{
no t1,t2: TaxiDriver | (t1!=t2 and t1.taxi=t2.taxi)
}
```

```
//Taxi - Taxi Driver
fact taxiAssociation{
#TaxiDriver = #Taxi
}


//Taxi

//unique ID
fact uniqueTaxiID{
no t1,t2: Taxi | (t1!=t2 and t1.ID=t2.ID)
}

fact postiveTaxiID{
no t1: Taxi | t1.ID<0
}


fact existsZoneForTaxi{
no t: Taxi,z1,z2:Zone | (z1!=z2) and (t in z1.TaxiQueue and t in z2.TaxiQueue)

}

fact uniqueZoneID{
no z1,z2: Zone | (z1!=z2 and z1.ID=z2.ID)
}
```

```
fact postiveZoneID{
no z1: Zone | z1.ID<0
}

//Occasional User
fact uniqueOccasionalUserID{
no ou1,ou2: OccasionalUser | (ou1!=ou2 and ou1.ID=ou2.ID)
}

fact postiveOccUserID{
no ou1: OccasionalUser | ou1.ID<0
}

//Ride

fact uniqueRideID{
no r1,r2: Ride| (r1!=r2 and r1.ID=r2.ID)
}

fact postiveRideID{
no r1: Ride | r1.ID<0
}

//Reservation

fact uniqueReservationID{
no r1,r2: Reservation| (r1!=r2 and r1.ID=r2.ID)
}
```

```
fact differentAddresses{
no r1: Reservation | r1.SourceAddress=r1.DestinationAddress
}

fact postiveReservationID{
no r1: Reservation| r1.ID<0
}

//Payment
fact PositivePrice{
no p1: Payment | p1.Price<0
}

//Date

//we suppose that february has always 28 days
fact validDate{
  no d1: Date | ( (d1.Month = 1 or d1.Month = 3 or d1.Month = 5 or d1.Month = 7 or
d1.Month=10 or d1.Month=12) =>
     (d1.Day >0 and d1.Day <=31)) and ((d1.Month=4 or d1.Month=6 or d1.Month = 9 or
d1.Month=11) => (d1.Day >0 and d1.Day<=30))
    and ((d1.Month=2) => (d1.Day>0 and d1.Day <=28)) and d1.Month>=1 and d1.Month<=12
and d1.Year>0 and d1.Day>0

}

fact noRideBeforeAssumption{
no t1: TaxiDriver, ir1: IstantaneousReservation | (ir1.ResDate.Year>t1.HiringDate.Year) or
((ir1.ResDate.Year = t1.HiringDate.Year) and (ir1.ResDate.Month>t1.HiringDate.Month))
```

```
or((ir1.ResDate.Year)=(t1.HiringDate.Year) and (ir1.ResDate.Month = t1.HiringDate.Month)
and (ir1.ResDate.Day>t1.HiringDate.Day))


}




fact OneAtLeastZone{

no t: Taxi | (all z: Zone |( t not in z.TaxiQueue))

}


pred show(){

#TaxiDriver>1

#User>1

#Reservation >1

#Ride>1

}


run show for 3 but 2 TaxiDriver, 2 User
```


ReservedRide0
TaxiDriver0
TaxiDriver1
ReservedRide1
Available
reservations
paymentInfo
associatedReservation
associatedReservation
taxi
taxi
ID
Username
reservations
HiringDate
2
boolean
IstantaneousReservation1
Payment0
IstantaneousReservation0
Payment1
Taxi0
Taxi1
TaxiQueue
$show_z
$show_z
TaxiZone
ID
DestinationAddress
Username
ResDate
SourceAddress
HiringDate
ResDate
DestinationAddress
SourceAddress
0
string0
string1
Price
Date2
Date0
Zone1
Price
Zone0
Date1
ID
ID
ID
Zone2
Year
Day
Day
Year
Month
ID
Day
ID
Year
Month
ID
Month
7
6
3
5
-7