Politecnico di Milano

Computer Science and Engineering

Project of Software Engineering 2

MyTaxiService

Requirement
Analysis and
Specification
Document

Authors:

Antonio Iannacci - 854157

Daniele Romanini - 854732

Federico Seri - 854032

Reference Professor: Mirandola Raffaela

SUMMARY

1. INTRODUCTION

1.1 Purpose of the document

This document is about the requirement analysis and specifications (RASD) of the application myTaxiService. The purpose of the document is to identify the goals of the program which has to be developed, analyzing the requirement and the domain of the application. In order to do this, we went beyond the specification provided by the customer so we made some assumptions. This document was written after interactions with our stakeholders and it will be necessary to develop the project.

1.2 Stakeholders

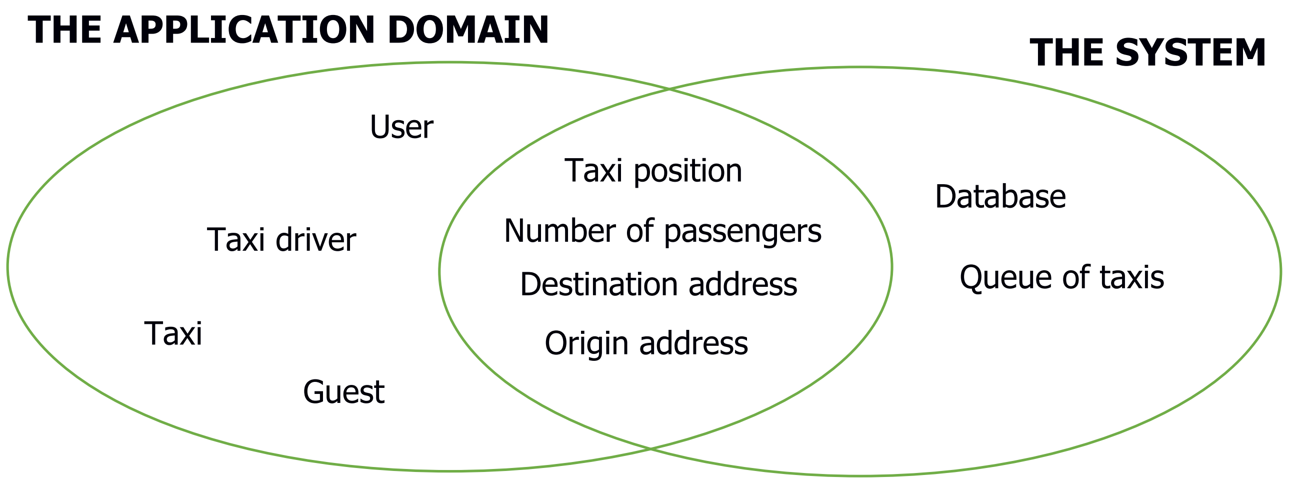
Our main stakeholder is our customer, the government of a city, wishing to optimize its taxi service.
It provided us the specification of the desired application, through which we can also identify the final users.
The application will be given to every taxi driver, employees of the city's government.
Moreover we suppose that the application will be free to download from the major mobile-application store and freely accessible through a web-site. Thus, the citizens of the city, or any other person who needs a taxi in the city, can access to it, simplifying the procedure to call a taxi.

1.3 Description of the problem

The project regarding myTaxiService application aims to optimize the taxi service of a large city, simplifying the access of passengers to the service.
This service will be based on either a website or a mobile application, through which passengers can request a taxi. The system answers to the request by informing the passenger about the code of the incoming taxi and the waiting time.
Also taxi drivers use a mobile application to inform the system about their availability and to confirm that they are going to take care of a certain call. In order to guarantee a fair management of taxi queues, the city is divided in taxi zones (approximately 2 km2 each).
Each zone is associated to a queue of taxis. The system automatically computes the distribution of taxis in the various zones based on the GPS information it receives from each taxi. When a taxi is available, its identifier is stored in the queue of taxis in the corresponding zone.
When a request arrives from a certain zone, the system forwards it to the first taxi queuing in that zone. If the taxi confirms, then the system will send a confirmation to the passenger. If not, then the system will forward the request to the second in the queue and will, at the same time, move the first taxi in the last position in the queue.
Besides the specific user interfaces for passengers and taxi drivers, the system offers also programmatic interfaces to enable the development of additional services on top of the basic one.

Moreover, a user can reserve a taxi by specifying the origin and the destination of the ride. The reservation has to occur at least two hours before the ride. In this case, the system confirms the reservation to the user and allocates a taxi to the request 10 minutes before the meeting time with him.
Another service (built on programmatic interfaces) that the system will be supported is "taxi-sharing". A user can enabled it if he/she is ready to share a taxi with others if possible, thus sharing the cost of the ride. In this case, the user is required to specify the destination of all rides, which he/she wants to share with others. If others are willing to start a shared ride from the same zone going in the same direction, then the system arranges the route for the taxi driver defines the fee for all persons sharing the taxi and informs the passengers and the taxi driver.

To reach to purpose of the RASD document, we tried to be the most complete and the most pertinent to the specification. We have identified goals, analyzed the requirements and the domain, locating the world and the machine, stressing on the shared phenomena.



1.4 Actors

The actors involved in our system are:

• Guest: a guest is a person who has not signed up to the system yet. The only operation that he is able to perform is signing up.

• User: a user is a person who has already signed up. He can access to his page after a login and perform all the functions implemented in our system.

• Taxi driver: taxi driver is a person who has logged into the system with a special code given to him from the district of the city. After the login, he is able only to accept or refuse ride requests.

1.5 Domain Properties

Analyzing the assignment, we suppose that the following properties characterize the application environment:

- If the taxi driver accepts the call made by the user, he will always turn up for the meeting.
- Rides can be required only by the mobile application and the website.
- Each taxi is equipped with a working GPS device integrated.
- Each taxi on duty is always visible from the system through the integrated GPS.
- Street names in the city are unambiguous.
- The number of passengers for each taxi is lower or equal to 4.
- The city can be divided into 2 km^2 zones.
- When the taxi driver accepts a call, he leaves immediately to reach the passenger.
- The taxi driver reaches the user that has made a call in at most 5 minutes.
- Taxi drivers use an application mobile to log into the system.
- Every taxi driver has a shift assigned by the government of the city in order to guarantee a fair management of the service.

1.6 Goals

Analyzing the text of the assignment, we have identified these main goals:

Users and taxi drivers should be able to:

- Sign up;
- Login;
- Logout.

Users should be able to:

- Call a taxi;
- Pay the ride;
- Book and manage a ride;
- Enable taxi-sharing option;

Taxi drivers should be able to:

- Accept or refuse a call.

Moreover, the system should provide the following options:

- Inform the client about the incoming taxi details;
- Offer programmatic interfaces (API) to enable the development of additional services;
- Calculate the path of the trip;
- Calculate an approximate fare for the ride.

Manage taxi queues:

- Organize a list of taxis for each covered zone;

Portability of the service:

- The application has to be used with both Mobile and Web;

1.7 Assumptions

In order to clarify some points not very clear in the assignment we have made the following assumptions:

• The taxi service works if and only if the departure place is inside the area covered by the service.

• Taxi drivers are available if and only if they are inside their working zone.

• The ride cost is calculated according to the number of kilometers travelled. If the taxi spends more than 60 seconds to travel 80 meters, 0,10€ are added to the fare. This condition is valid until the price of the fare is less than twice the minimum one. Once that the fare is more than the double of the minimum one, the user pays exactly the double of the minimum fare.

• The path is defined as a set of streets and a direction.

• If the option “taxi sharing” is selected the user can share the travel with other users that have selected the option “taxi sharing” if and only if the route of a user is included in the others’ ones and the selected schedules are compatible.

• Origin and destination addresses are identified using street name, a street number and the name of the city.

• A reservation is registered into the system if and only if on the credit card there is enough money to pay the maximum fare of the ride.

• In order to book a ride, a user should specify the destination of the desired trip.

• In case of a “booked ride”, if no taxi is available for that specific trip, the booking user will be informed 5 minutes before the scheduled departure time.

• A taxi ride booked with “myTaxiService” is “private” or, alternatively, “shared”.

• Users that chooses different types of ride (“private” or “shared”), should not be on the same taxi.

• Following the request of a ride in a certain zone, the system notifies all the taxi drivers in that zone (following the order of the queue), until one of them accepts the call. If no one of them accepts it, user that has requested the ride will be notify immediately and the order will be cancelled.

• The system assigns a number of taxis in each zone proportionally to the amount of traffic in that zone.

• A unique set of shared-taxi exists. When a taxi driver accepts a call with "taxi-sharing" option, he is added to the set of shared-taxi.

• Every taxi has an on-board computer where myTaxiService application is installed.

• Every taxi driver has an unambiguous code given to him by the government of the city.

• Each taxi driver drives only one taxi at time.

• In case of fast call, the taxi driver reaches the user in less than 5 minutes from the moment of the call.

• The system installed on the on board computer of the taxi can show maps and provide navigation option.

• If a user makes a call with Taxi Shared option enabled and the path of his desired ride is compatible with the path of an existing shared ride (already assigned to a taxi driver) the user will be automatically assigned to that taxi. Thus, the taxi driver will receive a notification about the presence of a new passenger and his map will automatically update showing the new path.

• The taxi driver waits the passenger at most 5 minutes after the meeting time. The passenger will not be refunded.

• The taxi driver communicates to the system information about the ride through some buttons:

  ◦ the passenger is on board;

  ◦ the passenger is not present in the meeting place at the specified time;

  ◦ the passenger has been brought to destination.

• The system is able to block the money on the user's credit card using apposite API.

• The taxi driver can accept or refuse a call only when he is available in his assigned zone.

• In case of a shared ride, the effective fare is divided into the number of passengers. Every time that a passenger joins the ride, a notification is sent to the other passengers. At the end of the ride, the system withdraws the money from the users' credit card and unlock the difference between the total fare (as if the user had to pay the entire ride) and the exact fare.

1.8 Glossary

We want to specify the definition of the main words that will often be used in this document:

• GUEST: a guest is a person who has not signed up for the service yet. He has less permissions than the other actors do and the only operation that he can perform is signing up.

• USER: a user is a person who has already signed up for the service. He has a private profile in which his personal information is registered such as:

  o Name;

  o Surname;

  o Email;

  o Password;

  o Date of birth;

  o Telephone number;

  o Details of its credit card;

He can manage his profile information, request for a ride, manage reservation that he has already requested.

• TAXI DRIVER: a taxi driver is the person who drives the taxi. He is identified by a unique code assigned to him by the government of the city. This code will allow him to sign in into the system. Once that he is logged into the system, the taxi driver will be assigned to a specific zone of the city covered by the system and he will be available in the taxi queue once he has reached the assigned zone. The taxi driver receives ride requests through the application and he will be able to accept or refuse them. Moreover, he can communicate the end of his shift.

•

• FAST CALL: a fast call is a request for booking a taxi immediately. It is asked to specify the destination address, the number of the passengers.

• BOOK A TAXI: the option book a taxi is, as the name specify, the possibility to make a reservation of a taxi at the time and day of interest. It is possible to choose this option only if the ride is two hours, or more, later than the booking time.

• TAXI SHARING: the option car sharing is the way with which a user can share a taxi with other passengers that have requested the same service, dividing the ride fare. It is possible to active this option both with a fast call and a booked ride.

• REQUEST: with request, we mean the notification that the taxi driver receives when a user has asked for a ride. The taxi driver can accept or refuse the request.

• COMPATIBLE PATH: Path “A” is compatible with path “B” if and only if the address of departure and arrival of “A” are inside the route of “B”.

• CONFIRMATION PAGE: The maximum and minimum fare of the ride are specified in this page. We have used it when a user modifies the details of a booked ride.

• MAXIMUM AND MINIMUM FARES: we mean that the cost of the ride is a value between the minimum and maximum. The cost depends on the traffic conditions. The maximum fare is at most twice the minimum one.

• FAST CALL FORM: it represents the page where the user can request a taxi with the fast call option.

• BOOKED CALL FORM: it represents the page where the user can book a taxi with the book a taxi option.

• BEST OR WORST TRAFFIC CONDITION: With the best traffic condition, we mean that there are no traffic and delays. With the worst traffic condition, we mean that there are traffic and delays in the city.

• BLOCK THE MONEY ON THE CREDIT CARD: the system makes unavailable the sum of the maximum fare on the credit card.

• TAXI DRIVER AVAILABLE: the taxi driver is available when he is in service, he has no calls and he is in his assigned zone.

• TAXI DRIVER IN SERVICE: the taxi driver is in service when he is logged into the system.

• PRIVATE RIDE: it is a ride that is not shared with other people.

2. Requirements

Assuming that the domain properties hold on we have identified the requirements in order to satisfy the goals.

2.1 Functional Requirements

- Sign up: The system has to provide a sign up functionality.
- Login: The system has to provide a login functionality.
- Logout: The system has to provide a logout functionality.
- Call a taxi:
  - The system has to provide a function to call a taxi specifying: destination address and the number of passengers;
  - The system has to check if the origin address is covered by the service;
  - The system has to check if the number of passengers inserted by the user is between 1 and 4;
- Pay the ride: The system has to be able to connect to the API   of the user's credit card;
- Book and manage a reservation:
  - The system has to provide a function to book a taxi. In order to book a taxi, the user has to specify: origin address, destination address, the meeting time and the number of passengers;
  - The system has to provide a function for deleting a reservation created by the user until 11 minutes before the meeting time;
  - The system has to provide a function to modify a reservation. The user can change origin address, destination address, number of passengers until 11 minutes before the meeting time. The user can change the meeting time as long as the new meeting time is later than two hours from the modification time.
- Enable taxi-sharing option:
  - The system has to allow the user to share the ride with other passengers;
  - The system has a function for checking if a path of a ride is compatible with another one.
- Accept or refuse a call:
  - The system has to provide to the taxi driver a function for accepting or refusing a ride request.
- Inform the client about the incoming taxi details:
  - The system has to notify the client about the details of the taxi (taxi code and meeting time).
- Calculate the path of the trip:
  - The system has to connect to the API of a map service in order to calculate the path of the trip;
- Manage taxi queues:
  - The system has to assign a queue for each zone.
  - The system has to provide a function that manages the taxi queue.
- - Manage taxi queue:
  - Push at the end of the queue every available taxi driver that refuse a call;

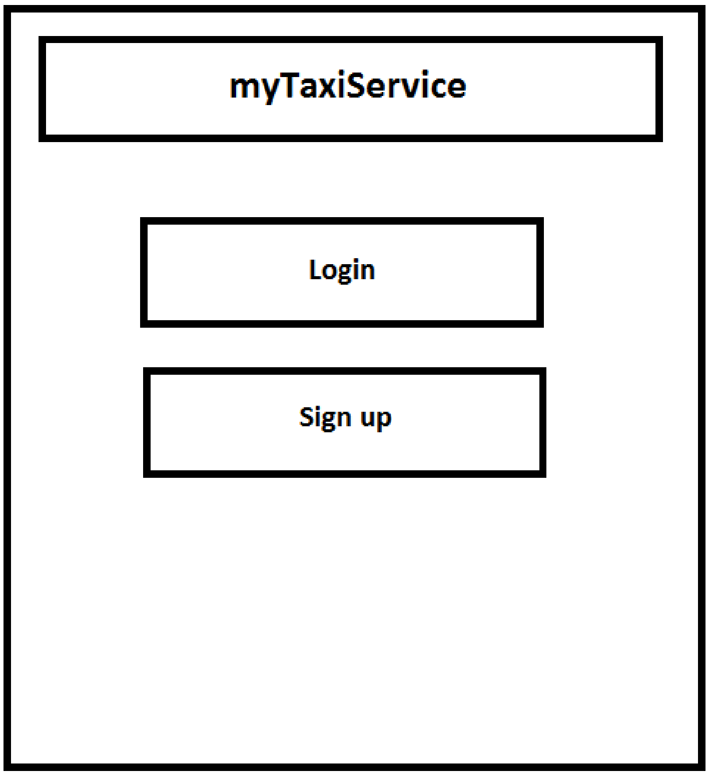
- Push at the end of the queue every taxi driver that comes back in his/her zone after have finished a ride;
- Remove from the queue of available taxis the ones who exit from their zone;
- Remove from the queue of available taxis the ones who accept a call;
- As soon as a taxi driver becomes available, the system put him at the end of the queue of the zone he was assigned to.

2.2 Non-functional Requirements

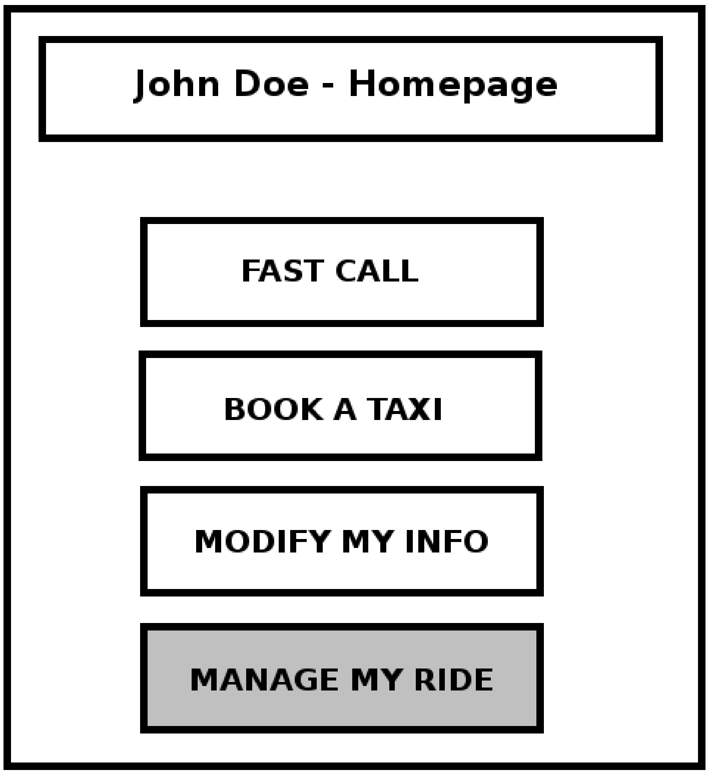
- In case of a booking-ride, the first available taxi driver is informed through a notification 10 minutes before the scheduled time.
- The system divides the city into zone of 2 km2 each;

User Interfaces

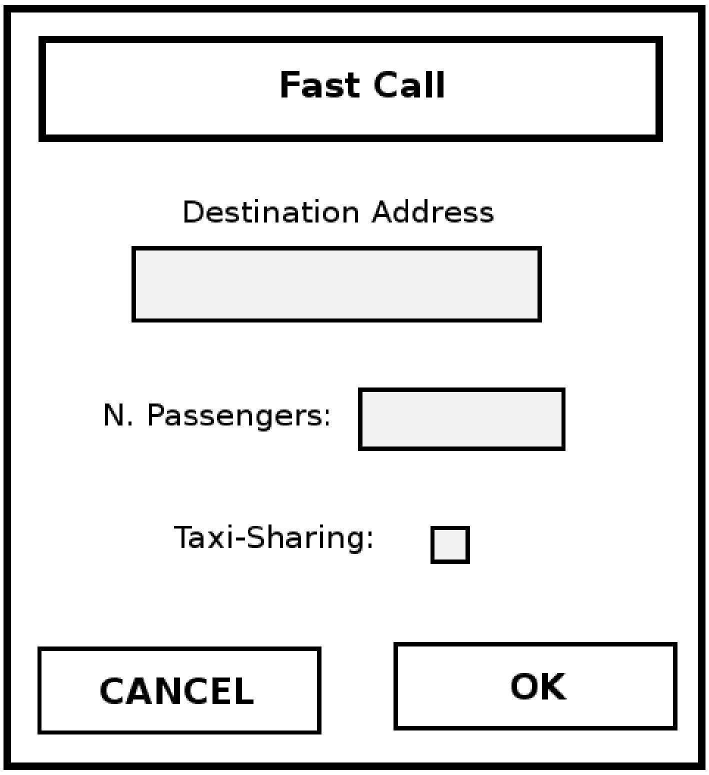
Our application is thought to be used both via web and via mobile application. Here we provide some sketches of the pages we consider the most important. This is the HOME page, where the user can log in or sign up for myTaxiService. There are only two button: LOGIN and SIGN UP.



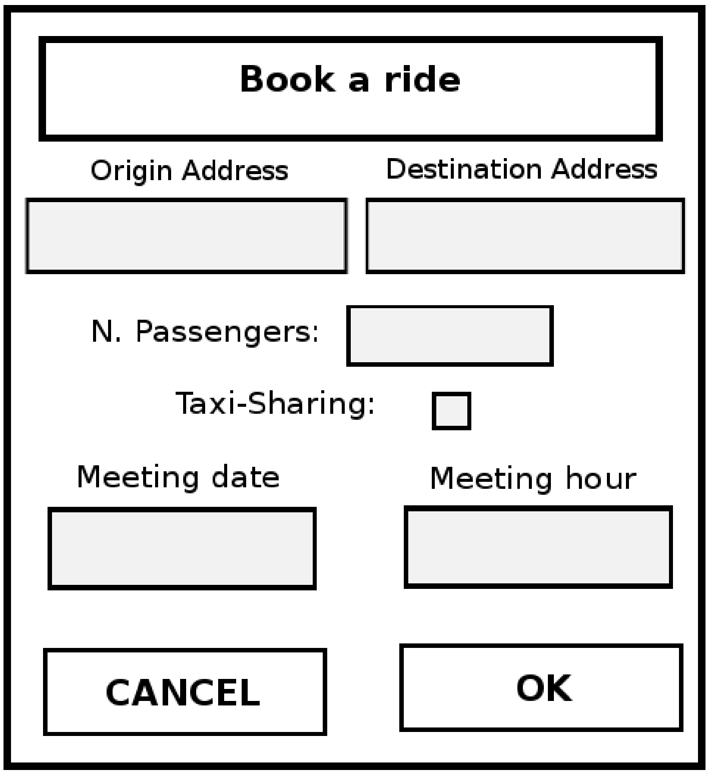
This image is the PERSONAL HOMEPAGE page, where the user can manage his infos, call or book a taxi and, if a previous taxi was booked, modify the the details about that ride.



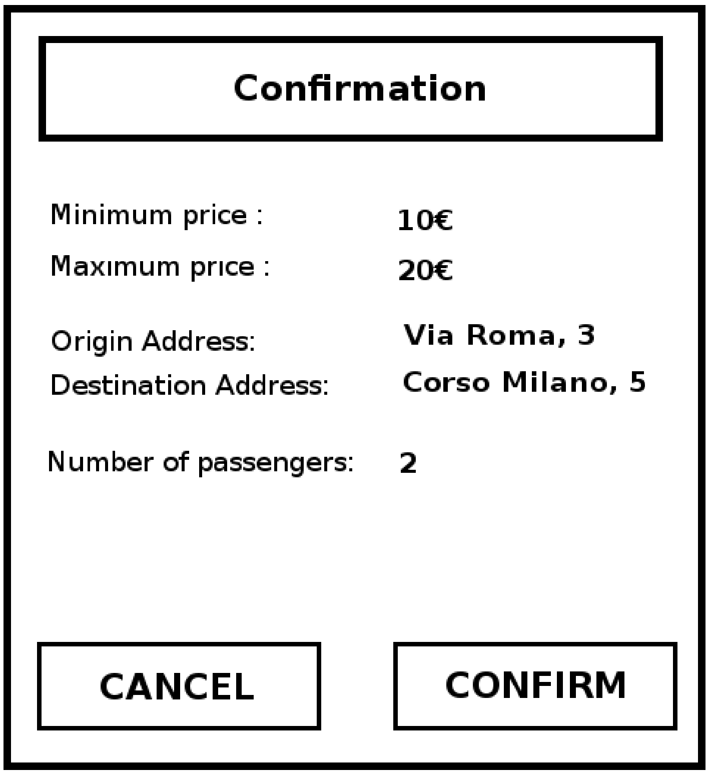
The next interface is the one related to “Fast Call” where a user has only to insert the destination address, the number of passenger and choose if share or not the trip with other people.



This is the “Book a ride” interface where the user insert origin and destination address, the number of passenger, the date and hour of the ride and so choose whether share or not the trip with other people.

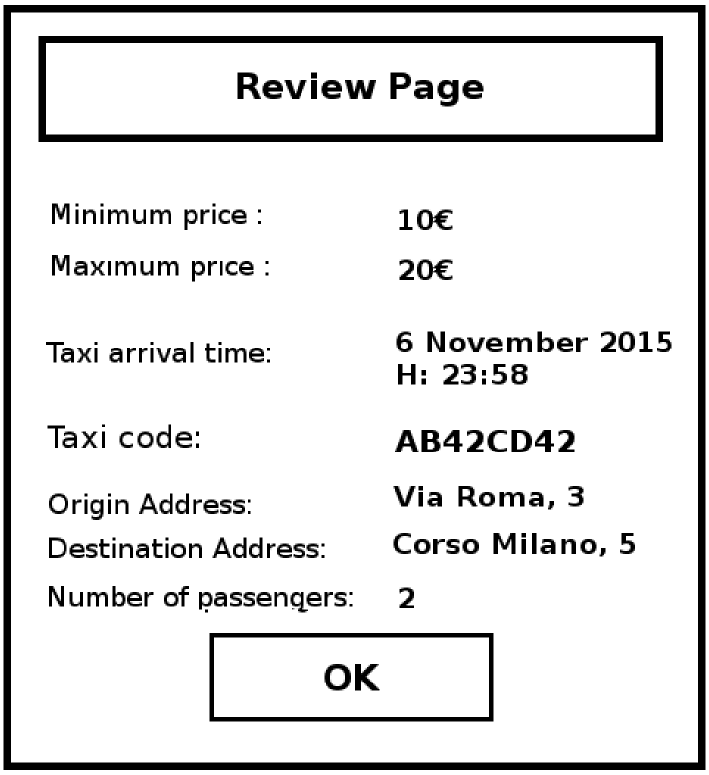


Before confirming a booking or a fast call it's presented a intermediate page where the user can chose to confirm or refuse after having seen the minimum and maximum price for the ride.



Minimum price :	10€
Maximum price :	20€
Origin Address:	Via Roma, 3
Destination Address:	Corso Milano, 5
Number of passengers:	2

After having confirmed the user can see all the information about his trip, including the taxi code and the estimated Taxi Arrival Time.



Review Page

Minimum price :	10€
Maximum price :	20€
Taxi arrival time:	6 November 2015 H: 23:58
Taxi code:	AB42CD42
Origin Address:	Via Roma, 3
Destination Address:	Corso Milano, 5
Number of passengers:	2

2.3 Constraints

- Users should have a credit card in order to sign up the system.

3. Scenarios

1) Al does not like the crowd in the public means of transportation; hence, he would like a smartphone application that allows him to move easily around the city. His friend John suggests him to download the application “myTaxiService”.
Thus, Al downloads this application from the Google Play Store and launch it. On the start-up screen, he clicks on “Sign-up” button in order to have access to all the functionalities. Afterwards, he fills the form with the following mandatory fields: name, surname, email, password and all the information about his credit card (name of the owner, number and security code).
Finally, the system accepts his registration and Jack can visualize his personal page.

2) Al decides to go to Cremona in order to visit his sister Anna. For this reason, he uses the application “myTaxiService” to call a taxi.
He opens the application and clicks on the login button, hence he inserts the email and password (chosen previously, at the moment of the registration) in the corresponding fields. Thus, the system asks him to choose between “Fast-call” and “Book a ride” options. Since he wants to leave the city immediately, he opts for the “Fast-call” option. In the next screen, he inserts the departure address (Via Roma, 4), the arrival address (Piazza Leonardo da Vinci, 32) and then he confirms through the apposite button. Afterwards, Al views a waiting-screen for 20 seconds after he pressed the confirm button. Finally, Al views on his smartphone a recap screen with all the information concerning his trip: date, time and the possible cost of the ride.
He leaves his house and wait for the taxi just outside the door of his apartment: the taxi arrives in two minutes; Al gets into the car, so he can start the trip towards Cremona, happy to have found in few minutes an available taxi.

3) John will leave for a business trip in two days. To reach the airport he decides to book a taxi using the booking option of “myTaxiService”.
Connecting to the homepage of the service, he logs in and select the option “Book a ride”. At this point John fills the form with departure address, arrival address, date and hour of the meeting leaving blank the check box “Taxi sharing” because he loves moving smoothly.
Immediately John is notified that his request was successful and the system estimate a price based on the kilometer to go and a maximum cap in case of heavy traffic.
The day of leaving, at the fixed time and place, John will find a taxi waiting for him.

4) Jack and Frankie would like to go to the disco tonight avoiding spending too much, they would like to use the service “Taxi sharing” provided by “myTaxiService”. Jack, from his mobile phone, using the mobile application, selects the option “Taxi sharing”; set the number of passengers to 2 and presses on the “Book a ride” button. He fills the form with departure address, arrival address, date and hour of the meeting and send the request. Immediately the system notify them that it has accepted the booking providing the details of the ride. The two friends were pleased to discover that the price for each person is one-

third of a normal ride because the system has found a passenger willing for the same route.
Just before the departure they receive a message form the system telling that a forth passenger has joint their ride so the actual price is a bit minor because he will travel a shorter route.
The taxi driver will turn up at the meeting point on time and will bring the guys to the destination address.

5) Al signs in to the application to call a taxi through "Fast call" option.
He inserts into the form the destination address and push the "Fast call" button. After few minutes, in the waiting screen, since any taxi is available, the system notifies him to retry later because the service is temporarily saturated.

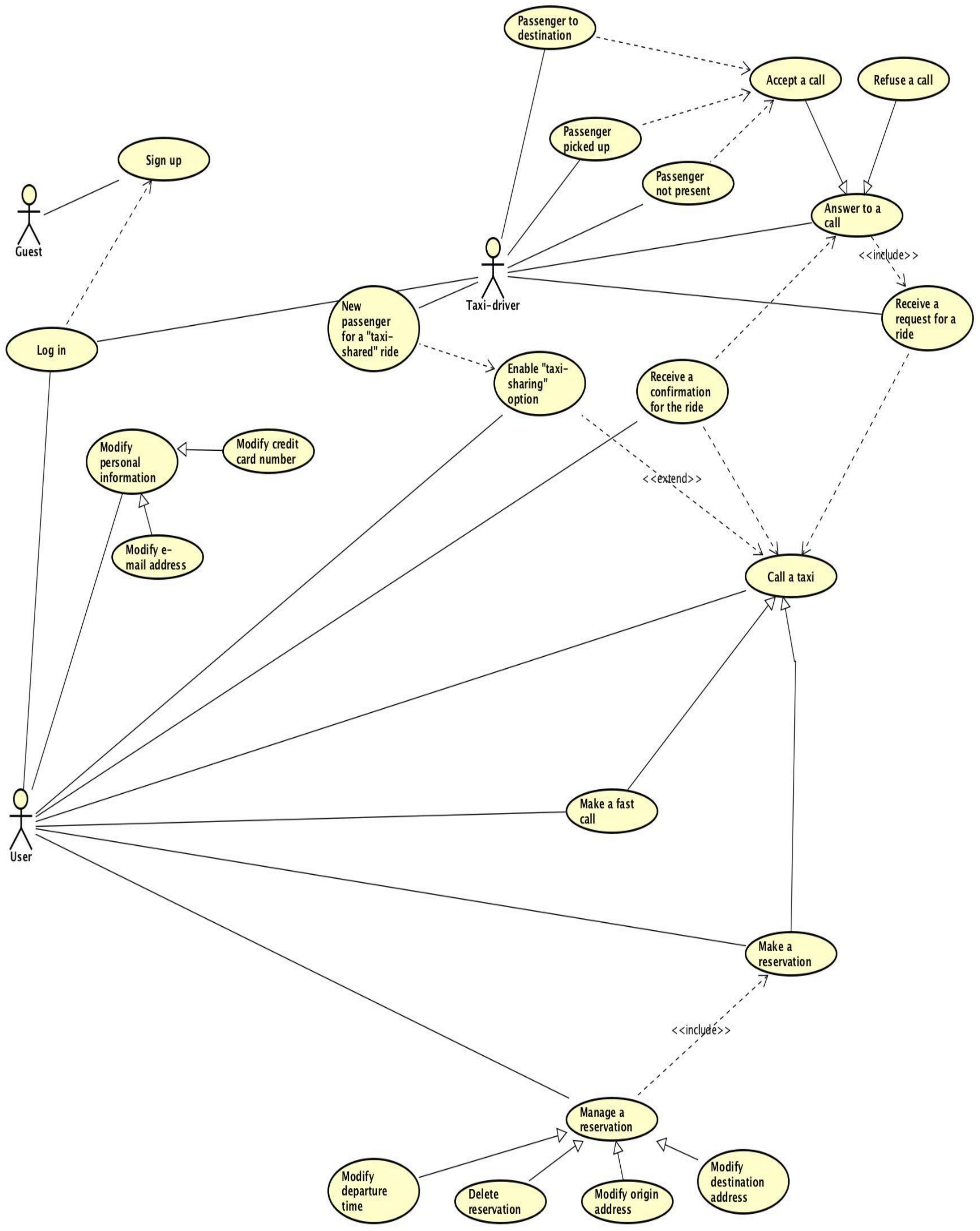
6) In few days, Rose has to take part in a conference at Polytechnic of Milan.
For reaching the conference, she books a taxi successfully from the web site of the service for the meeting date. Five minutes before the departure, the system notifies her that there are not available taxis for her predetermined ride. The system apologizes to her for the drawback and notifies her that her money, blocked at the time of the reservation, is available again.

7) Travis, a taxi driver of the group "myTaxiService", is waiting for a call in his taxi.
Five minutes before the end of his shift, the system notifies him that a request of ride is available. Travis reads the request information on his specific device installed into the taxi. He notices that the arrival address is far from the departure address so, due to his tiredness after a working day, he decides to refuse the request. The system will forward the request to the second taxi driver in the waiting queue and will insert Travis at the end of it.

8) It is Tuesday morning and Travis is waiting for some call inside the zone that the staff of "myTaxiService" assigned him. At ten o'clock, he receives a request from a user that has enabled the option "taxi sharing". Thus, he turns on his taxi and drives to the meeting point. Meanwhile, the system notifies Travis informing him about the presence of another passenger, who chooses an origin address during the route of the first trip. Once Travis has picked the first passenger up, the system will indicate him (through a map and a navigation system) the path to follow in order to reach the second client. Travis arrives at the indicated address, he picks the second passenger up, and the system updates the map, showing the route to the nearest destination, which is actually the destination of the first passenger. Arrived at the established place, the first passenger goes down, and then he leaves again through the destination address of the second one.
Once the client has been brought to destination, the ride is finished, and the on-board computer shows to Travis the shortest way to go back to his service zone. Moreover, it advices him that he will be "available" again as soon as he re-enters in the zone.

4. UML MODELS

4.1 Use Case Diagram



4.2 Use Case Description

Sign up

Name	Sign up
Actors	Guest
Entry Conditions	The person using myTaxiService is not registered into the system.
Flow of events	1. The guest enters the web site or open the mobile application; 2. The guest clicks on the "SIGN UP" button; 3. The guest compiles the form of the registration with: • Name • Surname • Email • Password • Date of birth • Telephone number • Details of its credit card • Compile the CAPTCHA field 4. The guest clicks the "CONFIRM" button; 5. The system sends him an email in order to confirm the registration; 6. The guest clicks on the link, valid for 24 hours, showed in the email so he confirms the registration.
Exit conditions	The system shows the new user's HOMEPAGE.
Exceptions	• Email already used. *Handling:* A message saying "Email already used" appears on the top of the registration form. The registration is considered as not done. • Some fields in the form are blank. *Handling:* A message saying "Complete all the requested fields" appears on the top of the registration form. The registration is considered as not done. • The credit card corresponding to the information inserted does not exists.

	Handling: A message saying "Wrong credit card information" appears on the top of the registration form. The registration is considered as not done. • The CAPTCHA inserted is wrong. *Handling:* A new CAPTCHA appears. The registration is considered as not done. • The user does not click on the link in 24 hour. *Handling:* A mail is send to the user informing that the link is no more valid and the registration is considered as not done.

Login

Name	Log in
Actors	User
Entry Conditions	User is already registered.
Flow of events	1. The user opens the application or enters the web site; 2. The user fills the fields of email and password requested in the application or in the web site home page; 3. The user clicks the "LOGIN" button.
Exit conditions	The system shows user's HOMEPAGE.
Exceptions	• The email or the password inserted by the user, does not match with any existing account. *Handling:* The system shows an error message and invites the user to insert his credential again.

Fast Call

Name	Call a taxi with 'Fast call'
Actors	User
Entry Conditions	User is already logged into the system.
Flow of events	1. The user clicks the "FAST CALL" button; 2. The user inserts the destination address and the number of passengers; 3. The user clicks on the "OK" button 4. The system shows the user an approximate fee for the ride. (Minimum and maximum price) 5. The user clicks on the "CONFIRM" button. 6. The system shows the WAITING PAGE.
Exit conditions	The request is on hold.
Exceptions	• Not enough money on the credit card to pay the maximum fee of the ride. *Handling:* A message saying that there is not enough credit on the card is send,the call is considered as not done and the system shows the HOMEPAGE to the user. • The position of the user is not covered by the service. *Handling:* A message says the position is not covered by the service, the call is considered as not done and the system shows the HOMEPAGE to the user. • In the field "Number of passengers" is not inserted a number between 1 and 4. *Handling:* A message says that a not valid number of passengers were inserted, the call is considered as not done and the user can write a new number or cancel the call. • At "Step 4" of the flow event the user press CANCEL button. *Handling:* The call is considered as not done and the system shows the HOMEPAGE

Booked Ride

Name	Call a taxi with "Booked Ride"
Actors	User
Entry Conditions	User is already logged into the system.
Flow of events	1. The user clicks the "BOOK A TAXI" button; 2. The user inserts: • The origin address • The destination address • The meeting date • The meeting time • Number of passengers; 3. The user clicks on the "OK" button. 4. The system shows the user an approximate fee for the ride. (Minimum and maximum price) 5. The user clicks on the "CONFIRM" button. 6. The system shows a review page.
Exit conditions	The request is on hold.
Exceptions	• The meeting time is already passed or is less than two hours before the beginning of the desired ride. *Handling:* Appears a message saying that an invalid time is inserted, the booking is considered as not done and the user can modify the time or exit from the Booked Ride procedure. • Not enough money on the credit card to pay the maximum fee of the ride. *Handling:* A message saying that there is not enough credit on the card is send,the booking is considered as not done and the system shows the HOMEPAGE to the user. • The address of the departure is not covered by the service. *Handling:* A message says the position is not covered by the service, the call is considered as not done and the system shows the HOMEPAGE to the user.

	• In the field "Number of passengers" is not inserted a number between 1 and 4. *Handling:* A message says that a not valid number of passengers were inserted, the call is considered as not done and the user can write a new number or cancel the call. • At "Step 4" of the flow event the user press CANCEL button. *Handling:* The call is considered as not done and the system shows the HOMEPAGE.

Confirmation for the ride

Name	Receive a confirmation for the ride
Actors	User
Entry Conditions	User must have done "Call a taxi with Fast Call" use case or "Call a taxi with Booked Ride" use case and the Taxi Driver must have selected "Accept Call" in the "Receive a request for a ride" use case.
Flow of events	1. The system blocks the maximum ride fee previously computed on the credit card. 2. The system notifies the user with the ride details: a. Number of passengers b. Meeting time c. Price per person d. Date e. Origin address f. Destination address
Exit conditions	The user clicks on the "OK" button.
Exceptions	• No taxi drivers available for the request. *Handling:* The system sends a notification of apologies to the user, the user will not pay any fee (the amount of money previously blocked on his credit card will be unlocked).

Request for a ride

Name	Receive a request for a ride
Actors	Taxi driver
Entry Conditions	Taxi driver must be logged in to the systems and he must be the first in the queue of his/her zone.
Flow of events	1. Taxi driver receives a notification saying "New request", with all the details for the ride: • Origin address • Destination address • Meeting time 2. Taxi driver accept or refuse a call. 3. If the Taxi driver accept the call then he goes to pick up the passenger. 4. When the passenger is on board the taxi driver press on the button PASSENGER ON BOARD. 5. The map is updated with the destination. 6. The taxi driver lead the passenger to the destination. 7. Money previously blocked from the credit card of the passenger is taken.
Exit conditions	At "Step 2" the taxi-driver refuse the call. At "Step 6" the taxi-driver press on the button PASSENGER AT DESTINATION
Exceptions	• At "Step 2" the taxi-driver do not answer in 15 seconds. *Handling:* The system considers the taxi-driver as he has answer pressing on REFUSE. • At "Step 3" the taxi-driver, after having waited for at least 5 minutes, press on the button PASSENGER ABSENT. *Handling:* The system considers the passenger lead to destination and money blocked is taken form the card.

Enable taxi-sharing

Name	Enable "taxi-sharing" option
Actors	User
Entry Conditions	This use case extends "Call a taxi with Fast Call" and "Book a taxi" use cases. The user must have started one of them: he must have filled the form with all the information regarding the ride.
Flow of events	1. The user clicks on the "Taxi-sharing" button. 2. In case of "Fast Call", go to "Step 3". In case of "Book a taxi", go to "Step 3" 10 minutes before the scheduled ride time. 3. The system searches in the list of shared-taxi if there is one of them with a compatible path, compatible time and a compatible number of passengers. 4. If there is a compatible taxi in the "shared-taxi" list, the user is assigned to that taxi and the taxi-driver receives a notification regarding the presence of a new passenger, with all the details of his ride (origin address, destination address, meeting time). 5. If there is not a compatible taxi in the "shared-taxi" list, the "Receive a request for a ride" use case starts.
Exit conditions	The request is on hold.
Exceptions	No exceptions

Modify Personal Information

Name	Modify personal information
Actors	User
Entry Conditions	User is already logged into the system.
Flow of events	1. The user clicks on the button "Modify personal information". 2. The user can modify: • Credit card information • Password 3. The user clicks on the "SAVE and EXIT" button;
Exit conditions	The personal information are successfully modified.
Exceptions	• The new password is blank. *Handling:* The system notify that the password wasn't modified, the modification is considered as not done and the user can insert a new password. • The new credit card information does not correspond to a valid card. *Handling:* The system notify that the information about the credit card wasn't modified, the modification is considered as not done and the user can insert a new credit card number.

Manage a Booked Ride

Name	Manage a Booked Ride
Actors	User
Entry Conditions	The "Make a Booked Ride" use case must be done and the booked ride has not to be already happened.
Flow of events	The system shows a page in which there are the following modifiable fields: • Origin address • Destination address • Departure time • Number of passengers. There are the following buttons: • CONFIRM • CANCEL • DELETE
Exit conditions	The booked ride is modified. The user visualizes the HOMEPAGE.
Exceptions (common)	• The modification or the deletion happens less than 11 minutes than the established departure time. *Handling:* The modification does not take place.

Manage a Booked Ride – *Delete*

Name	Manage a Booked Ride – Delete Booked Ride
Flow of events	1. The user clicks on the button DELETE BOOKED RIDE. 2. The user will be paid back with the ride fee previously blocked on his credit card.
Exceptions	No exception

Manage a Booked Ride – *Modify number of passenger*

Name	Manage a Booked Ride – Modify number of passengers
Flow of events	1. The user inserts the new number of passengers in the field "Number of passengers". 2. The user clicks on the CONFIRM button.
Exceptions	In the field "Number of passengers" is not inserted a number between 1 and 4. *Handling:* A message says that a not valid number of passengers were inserted, the modification is considered as not done and the user can write a new number.

Manage a Booked Ride – *Modify departure time*

Name	Manage a Booked Ride – Modify departure time
Flow of events	1. The user inserts the new departure time in the field "Departure time". 2. The user clicks on the CONFIRM button.
Exceptions	The new time inserted is less than 2 hours later than the modification time. *Handling:* Appears a message saying that the departure time is invalid. The modification is considered as not done.

Manage a Booked Ride – *Modify origin address*

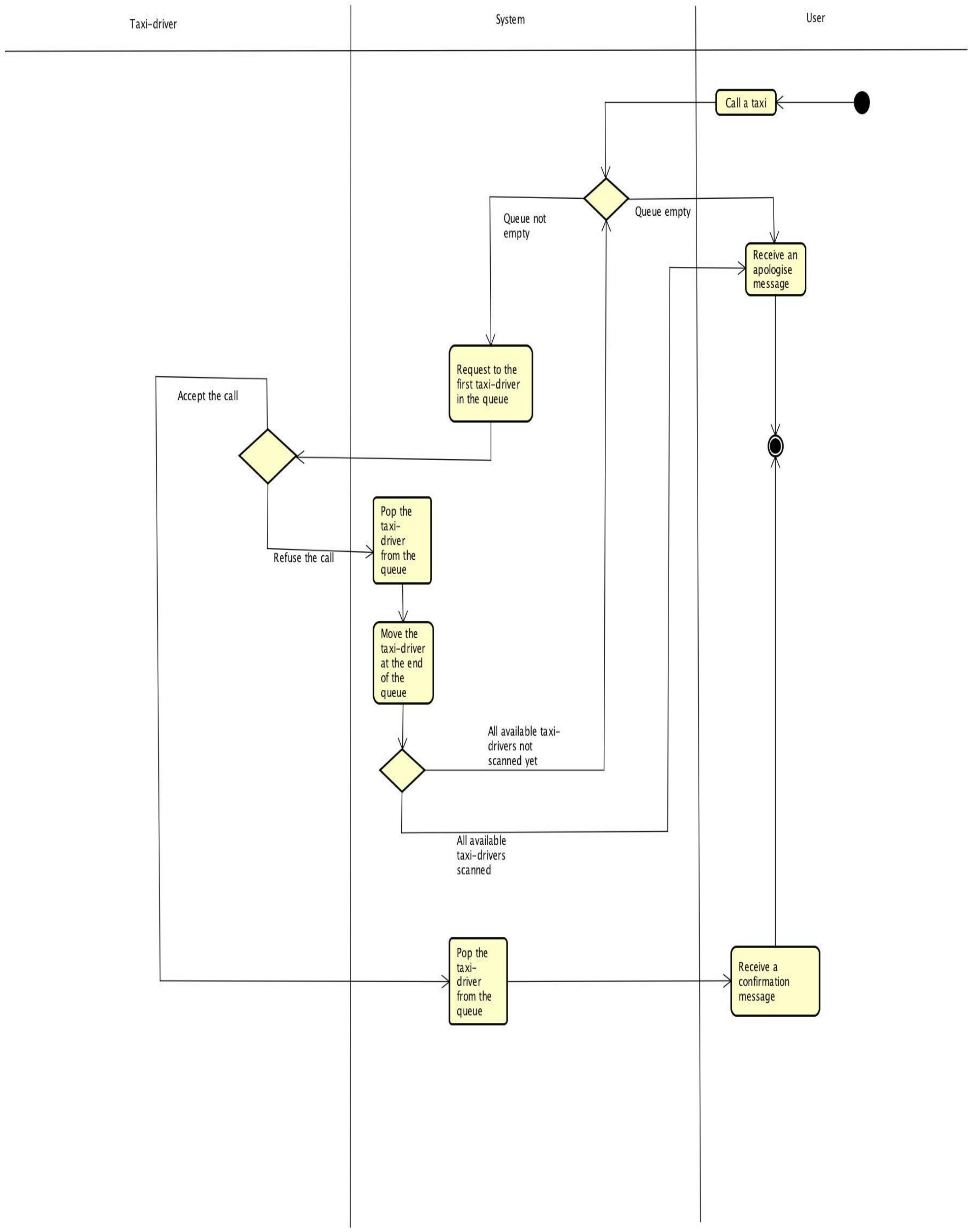
Name	Manage a Booked Ride – Modify origin address
Flow of events	1. The user inserts the new origin address in the field "Origin address". 2. The user clicks on the CONFIRM button.
Exceptions	The origin address is not valid (it does not exists or is not covered by the service). *Handling:* Appears a message on the screen saying that the origin address is invalid and the modification is considered as not done.

Manage a Booked Ride – *Modify destination address*

Name	Manage a Booked Ride – Modify destination address
Flow of events	1. The user inserts the new destination address in the field "Destination address". 2. The user clicks on the CONFIRM button.
Exceptions	The destination address is not valid (it does not exists). *Handling:* Appears a message on the screen saying that the destination address is invalid and the modification is considered as not done.

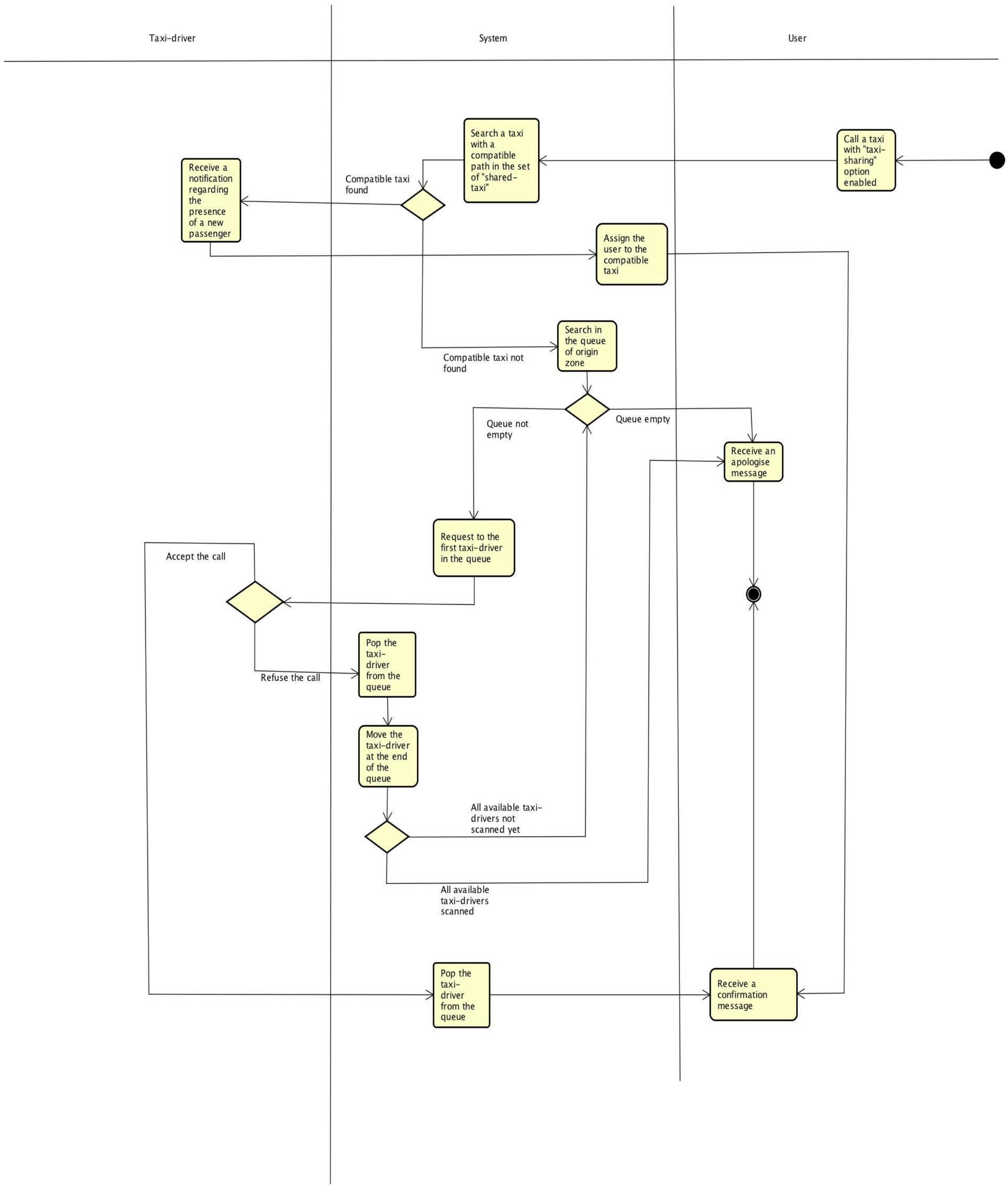
4.3 Activity Diagrams

4.3.1 Management of a call



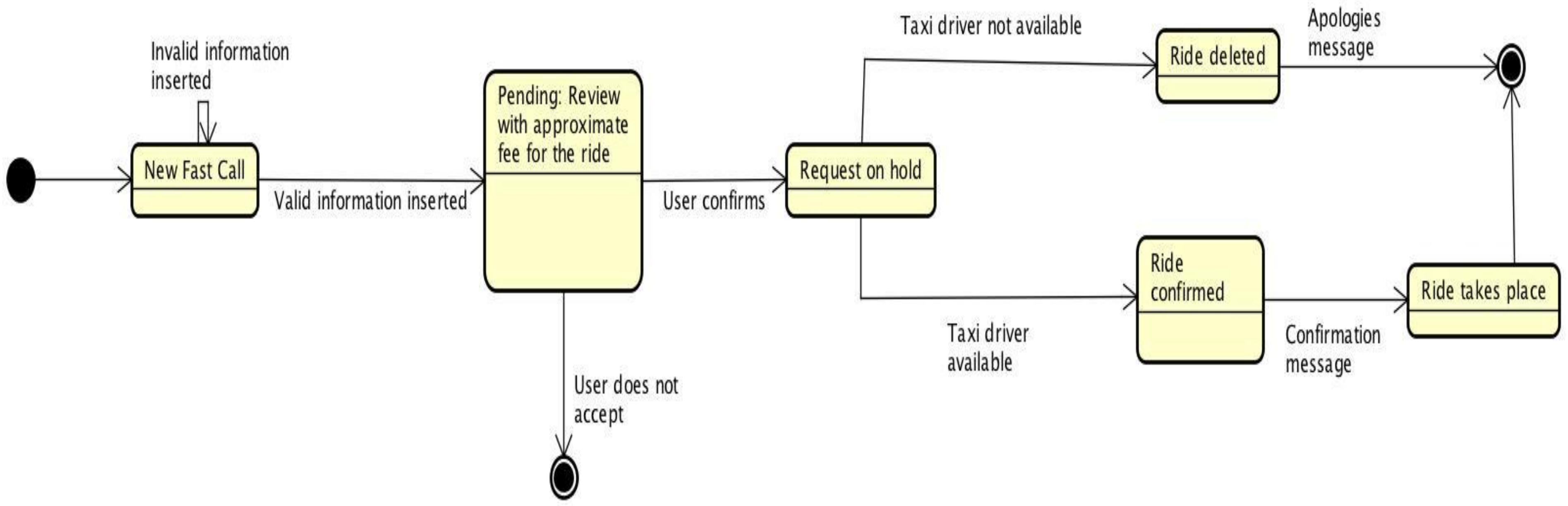
Taxi-driver
System
User
Call a taxi
Queue not empty
Queue empty
Receive an apologise message
Request to the first taxi-driver in the queue
Accept the call
Refuse the call
Pop the taxi-driver from the queue
Move the taxi-driver at the end of the queue
All available taxi-drivers not scanned yet
All available taxi-drivers scanned
Pop the taxi-driver from the queue
Receive a confirmation message

4.3.2 Management of a call with taxi sharing option enabled

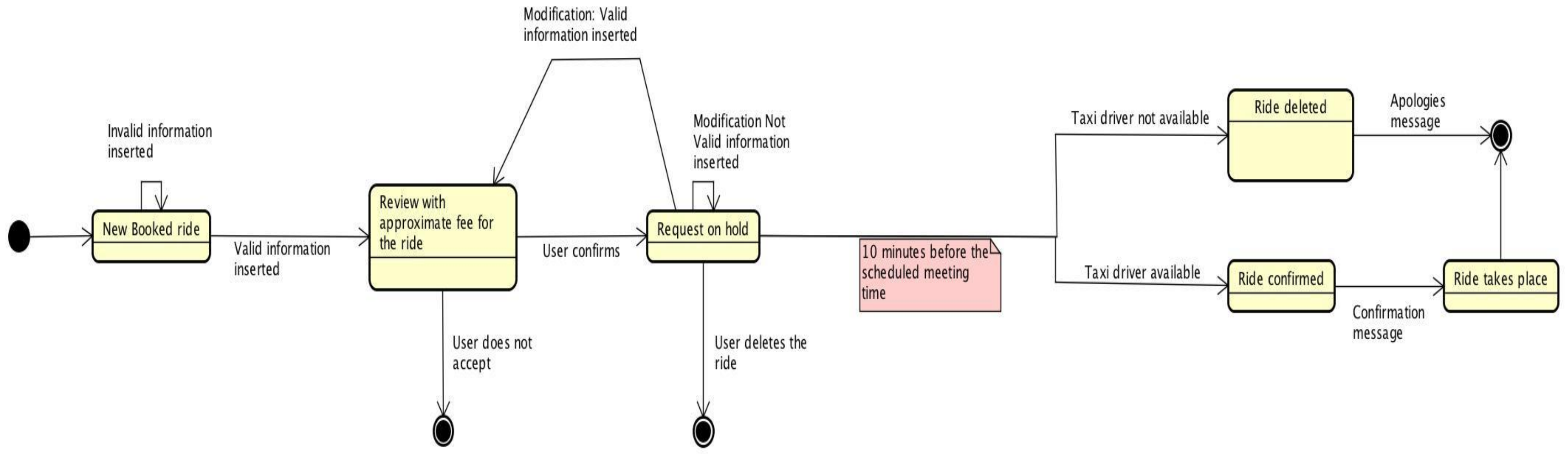

Taxi-driver
System
User
Call a taxi with "taxi-sharing" option enabled
Search a taxi with a compatible path in the set of "shared-taxi"
Compatible taxi found
Receive a notification regarding the presence of a new passenger
Assign the user to the compatible taxi
Compatible taxi not found
Search in the queue of origin zone
Queue not empty
Queue empty
Receive an apologise message
Request to the first taxi-driver in the queue
Accept the call
Refuse the call
Pop the taxi-driver from the queue
Move the taxi-driver at the end of the queue
All available taxi-drivers not scanned yet
All available taxi-drivers scanned
Pop the taxi-driver from the queue
Receive a confirmation message

4.4 State Chart Diagrams

4.4.1 Fast call process from the user's point of view



4.4.2 Booked ride process from the user’s point of view

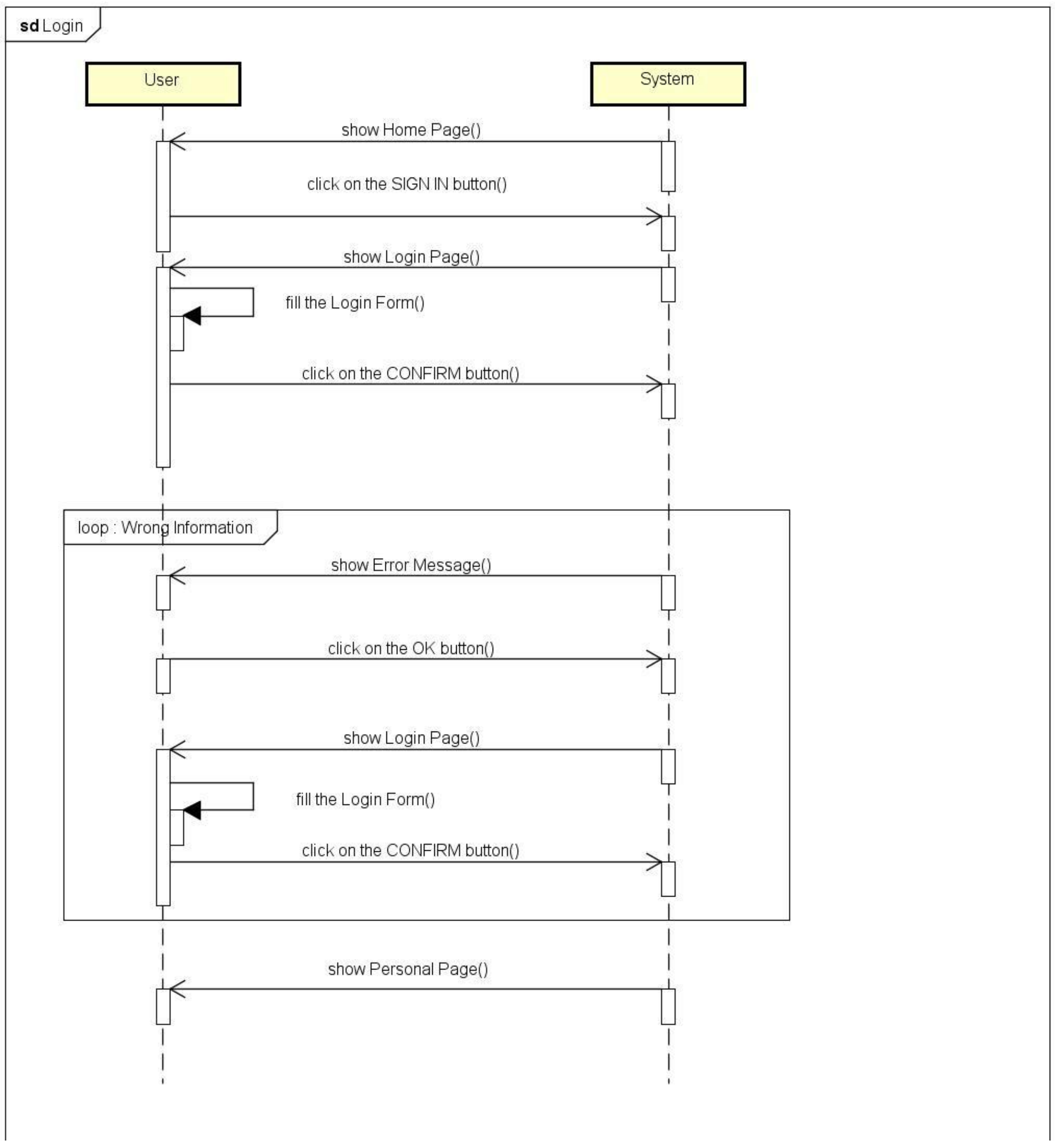


Invalid information inserted
New Booked ride
Valid information inserted
Modification: Valid information inserted
Review with approximate fee for the ride
User does not accept
User confirms
Modification Not Valid information inserted
Request on hold
User deletes the ride
10 minutes before the scheduled meeting time
Taxi driver not available
Ride deleted
Apologies message
Taxi driver available
Ride confirmed
Confirmation message
Ride takes place

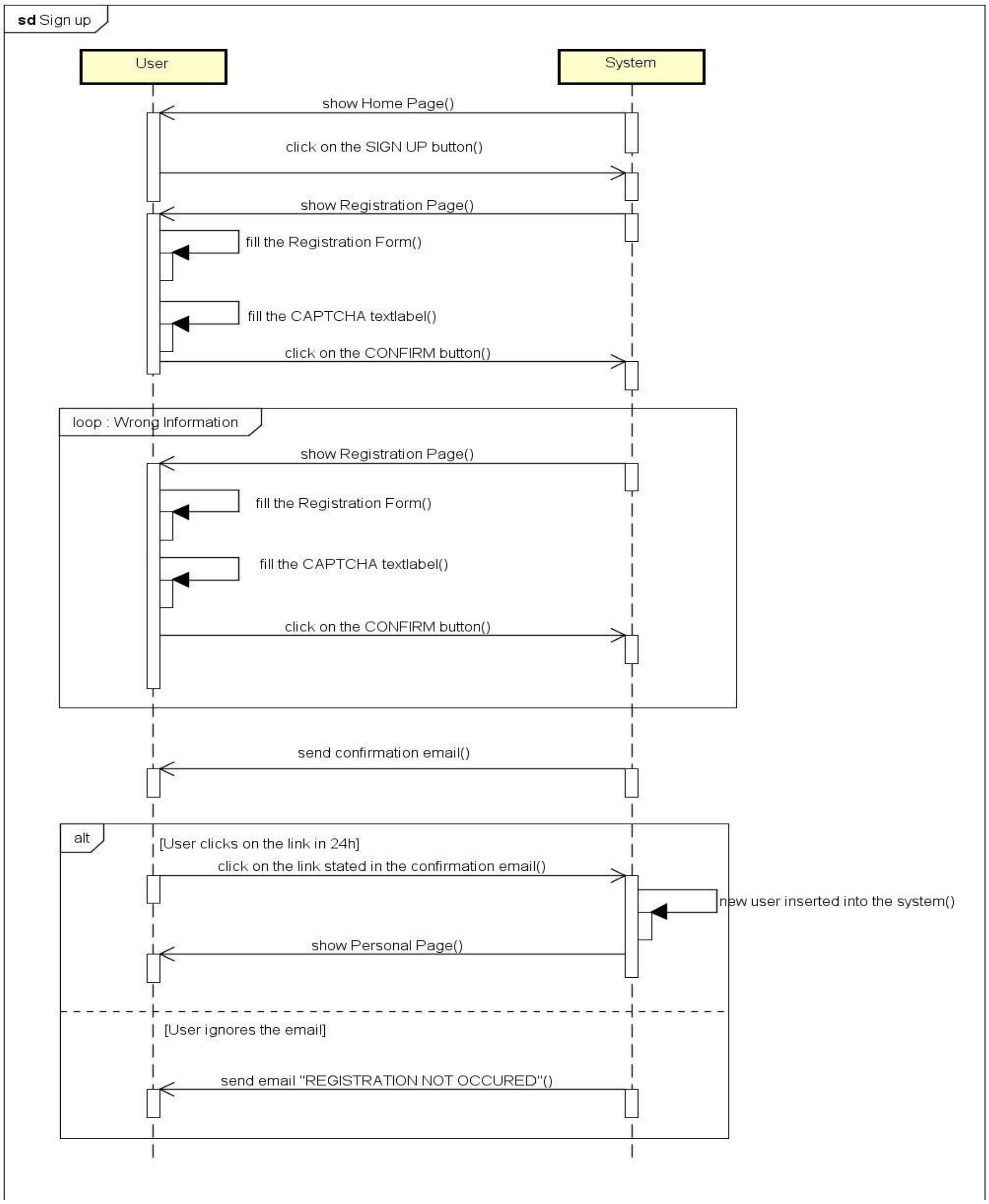
4.5 Sequence Diagram

Through the sequence diagrams, we want to specify some interactions between the actors and the system

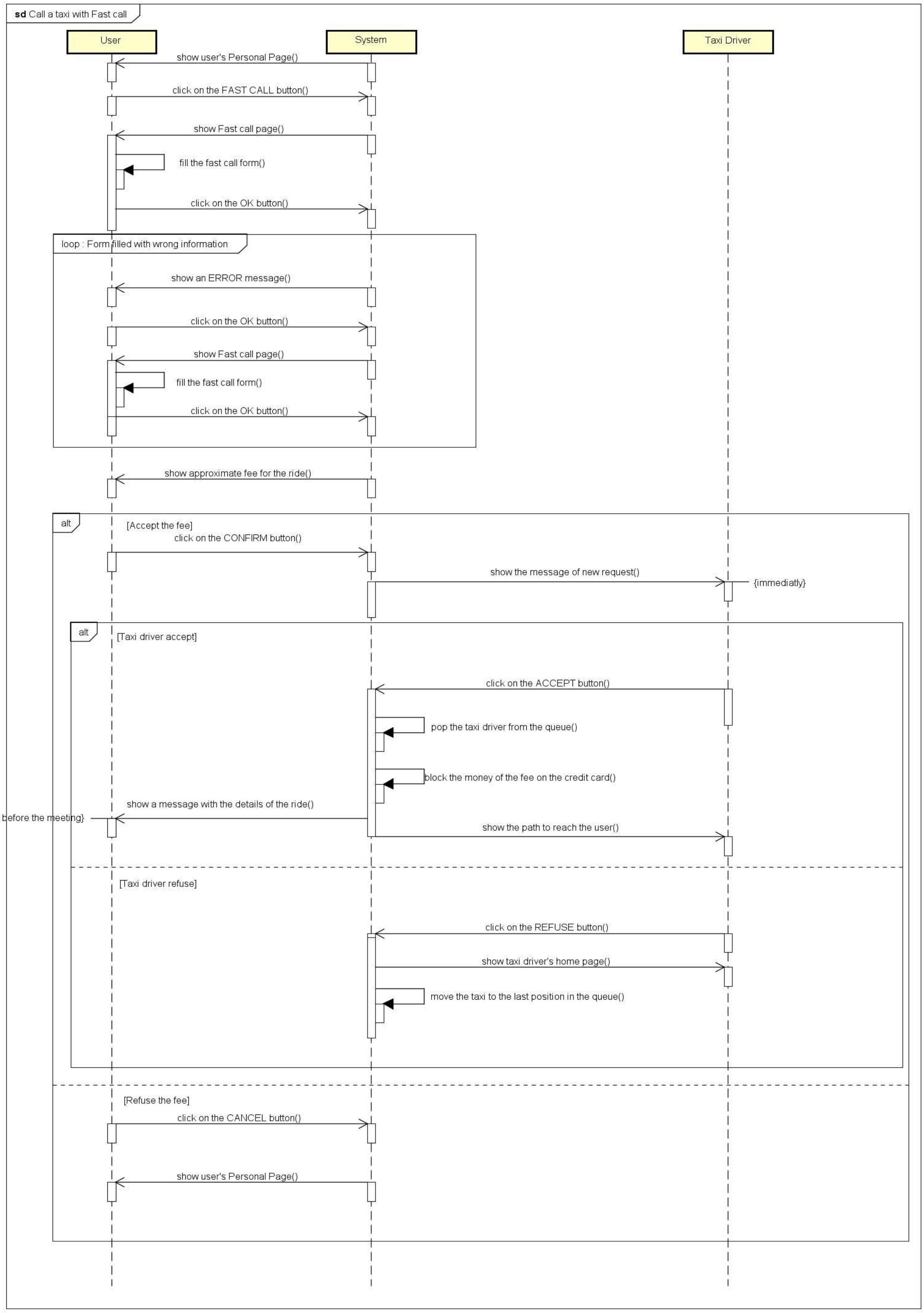
4.5.1 Login



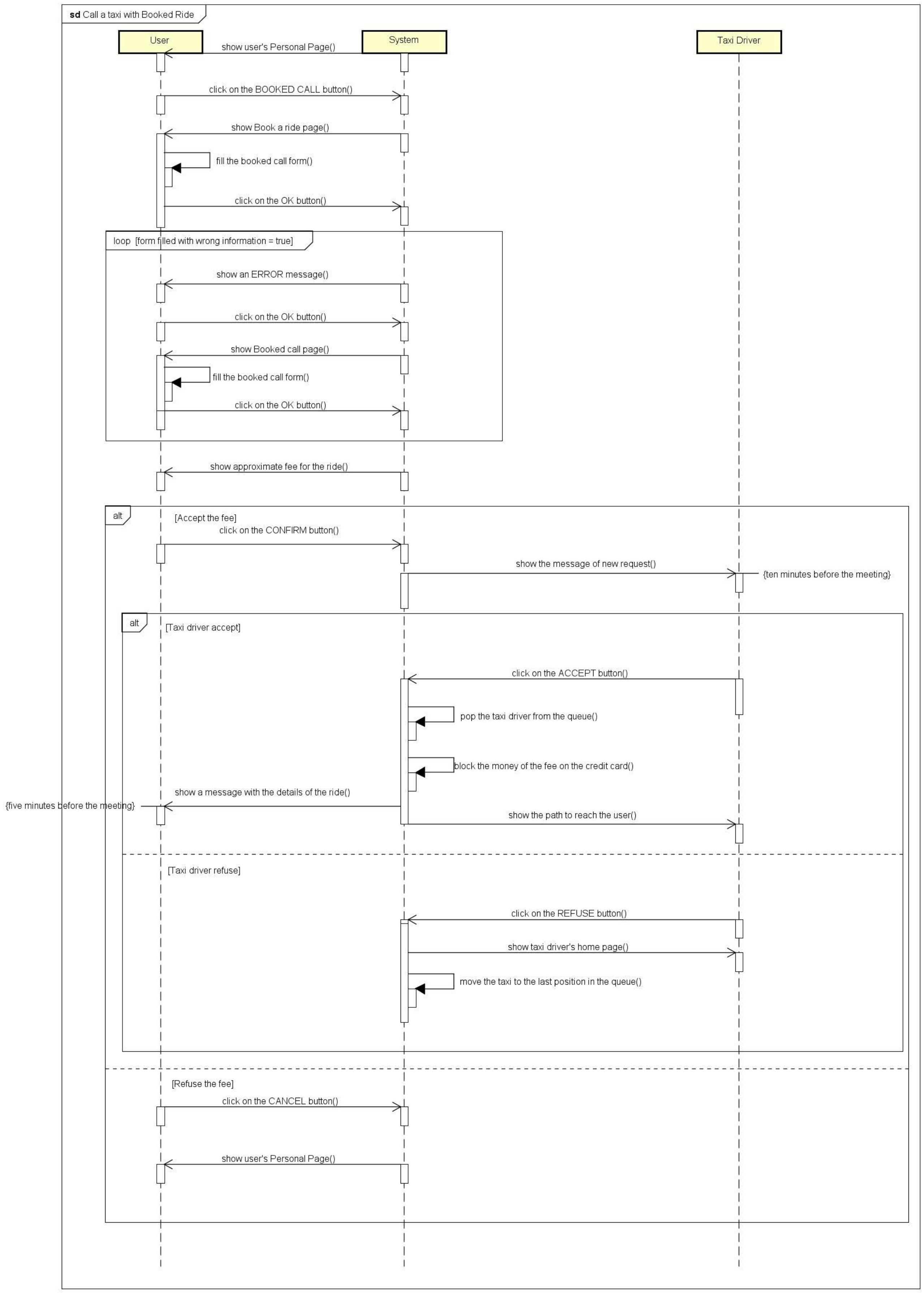
4.5.2 Sign up


sd Sign up
User
System
show Home Page()
click on the SIGN UP button()
show Registration Page()
fill the Registration Form()
fill the CAPTCHA textlabel()
click on the CONFIRM button()
loop : Wrong Information
show Registration Page()
fill the Registration Form()
fill the CAPTCHA textlabel()
click on the CONFIRM button()
send confirmation email()
alt
[User clicks on the link in 24h]
click on the link stated in the confirmation email()
new user inserted into the system()
show Personal Page()
[User ignores the email]
send email "REGISTRATION NOT OCCURED"()

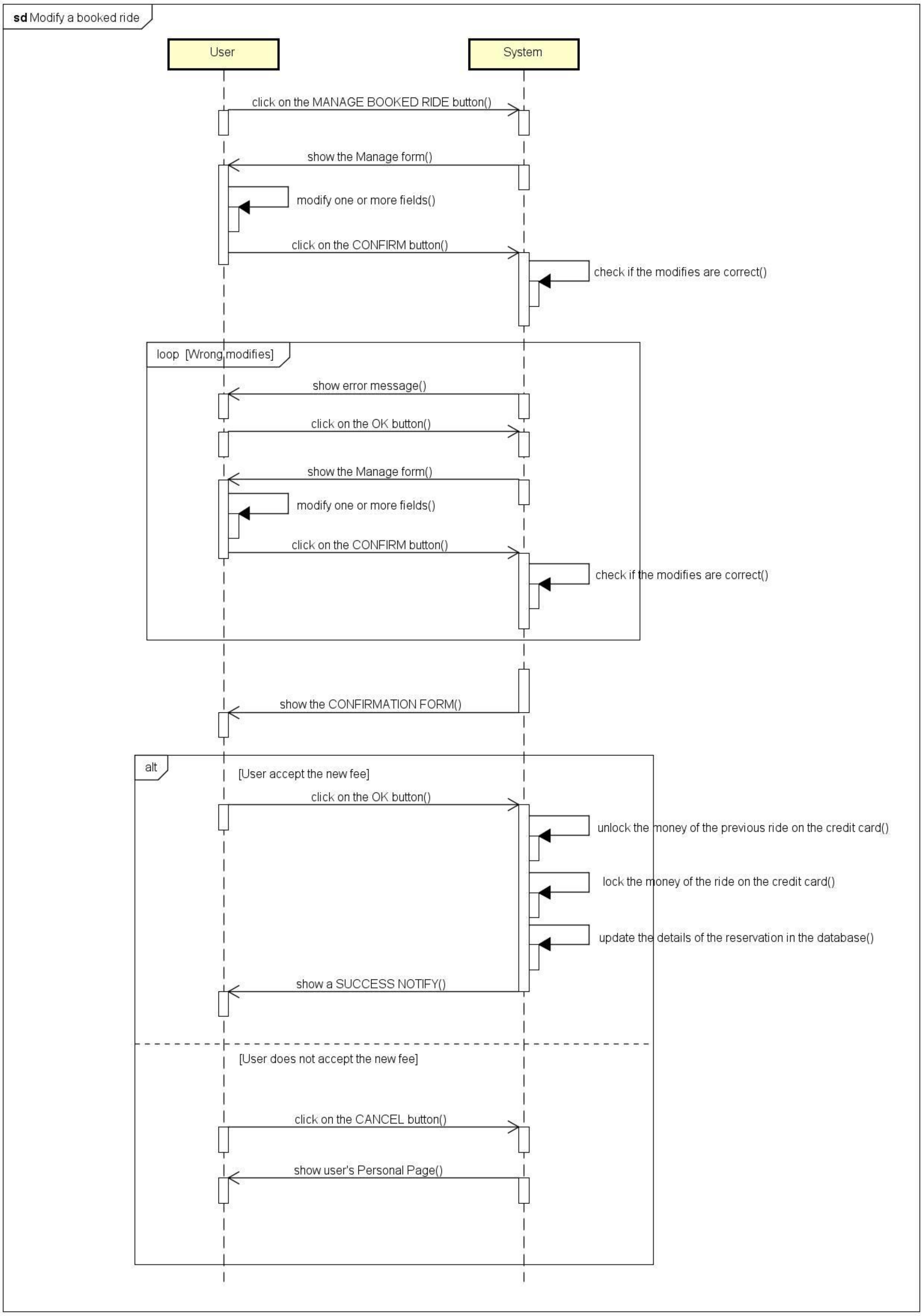
4.5.3 Fast call


sd Call a taxi with Fast call
User
System
Taxi Driver
show user's Personal Page()
click on the FAST CALL button()
show Fast call page()
fill the fast call form()
click on the OK button()
loop : Form filled with wrong information
show an ERROR message()
click on the OK button()
show Fast call page()
fill the fast call form()
click on the OK button()
show approximate fee for the ride()
alt
[Accept the fee]
click on the CONFIRM button()
show the message of new request()
{immediatly}
alt
[Taxi driver accept]
click on the ACCEPT button()
pop the taxi driver from the queue()
block the money of the fee on the credit card()
show a message with the details of the ride()
{five minutes before the meeting}
show the path to reach the user()
[Taxi driver refuse]
click on the REFUSE button()
show taxi driver's home page()
move the taxi to the last position in the queue()
[Refuse the fee]
click on the CANCEL button()
show user's Personal Page()

4.5.4 Booked call

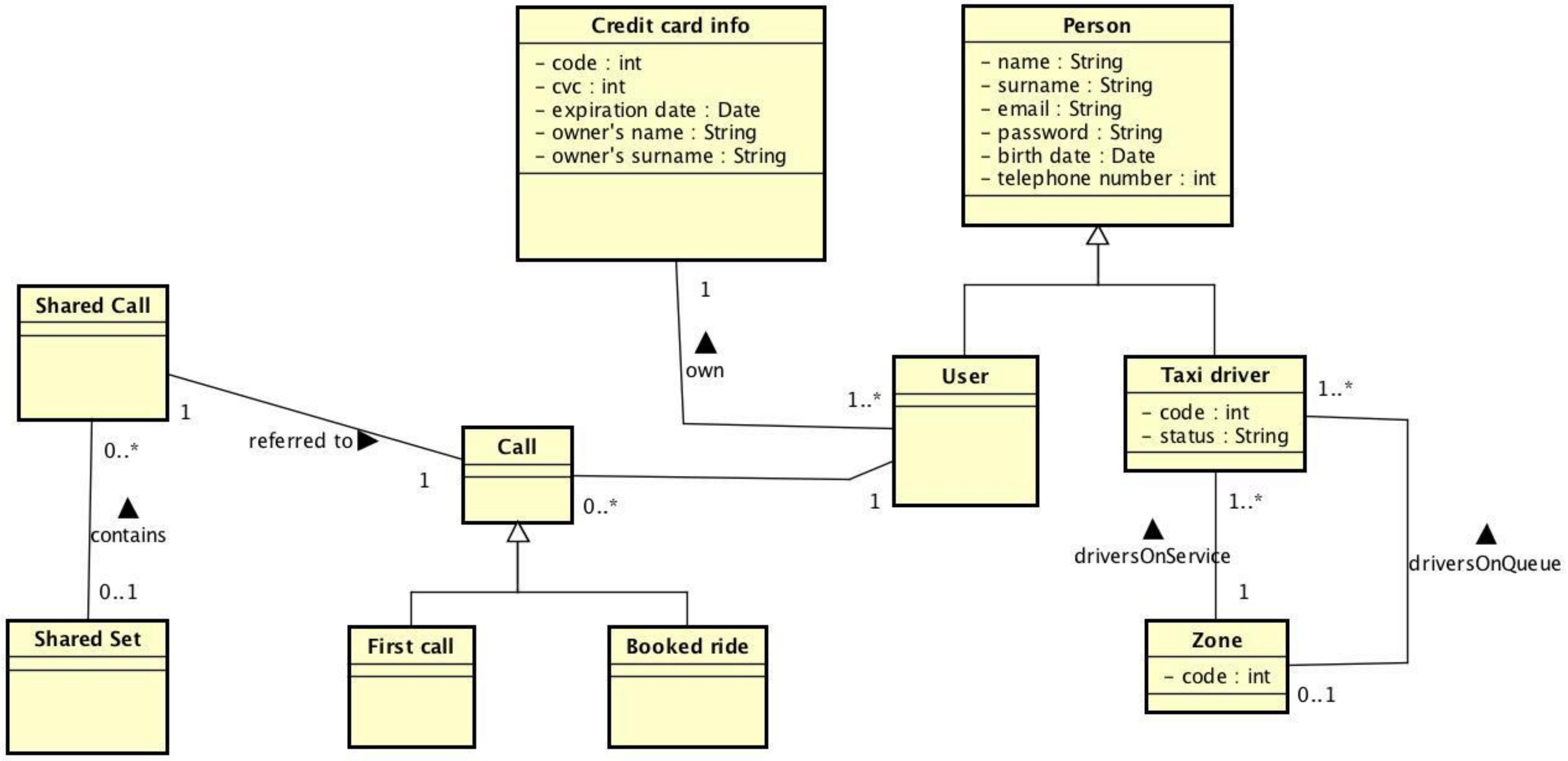

sd Call a taxi with Booked Ride
User
System
Taxi Driver
show user's Personal Page()
click on the BOOKED CALL button()
show Book a ride page()
fill the booked call form()
click on the OK button()
loop [form filled with wrong information = true]
show an ERROR message()
click on the OK button()
show Booked call page()
fill the booked call form()
click on the OK button()
show approximate fee for the ride()
alt
[Accept the fee]
click on the CONFIRM button()
show the message of new request()
{ten minutes before the meeting}
alt
[Taxi driver accept]
click on the ACCEPT button()
pop the taxi driver from the queue()
block the money of the fee on the credit card()
show a message with the details of the ride()
{five minutes before the meeting}
show the path to reach the user()
[Taxi driver refuse]
click on the REFUSE button()
show taxi driver's home page()
move the taxi to the last position in the queue()
[Refuse the fee]
click on the CANCEL button()
show user's Personal Page()

4.5.5 Modify a booked ride


sd Modify a booked ride
User
System
click on the MANAGE BOOKED RIDE button()
show the Manage form()
modify one or more fields()
click on the CONFIRM button()
check if the modifies are correct()
loop [Wrong modifies]
show error message()
click on the OK button()
show the Manage form()
modify one or more fields()
click on the CONFIRM button()
check if the modifies are correct()
show the CONFIRMATION FORM()
alt
[User accept the new fee]
click on the OK button()
unlock the money of the previous ride on the credit card()
lock the money of the ride on the credit card()
update the details of the reservation in the database()
show a SUCCESS NOTIFY()
[User does not accept the new fee]
click on the CANCEL button()
show user's Personal Page()

4.6 Class Diagram

Now that we have analyzed every use case and every interaction between actors, we can draw a class diagram of our system:



5. Alloy Modelling

Starting from the representation of the Class Diagram, we have used Alloy Analyzer to specify the properties of our application.

5.1 Alloy Code

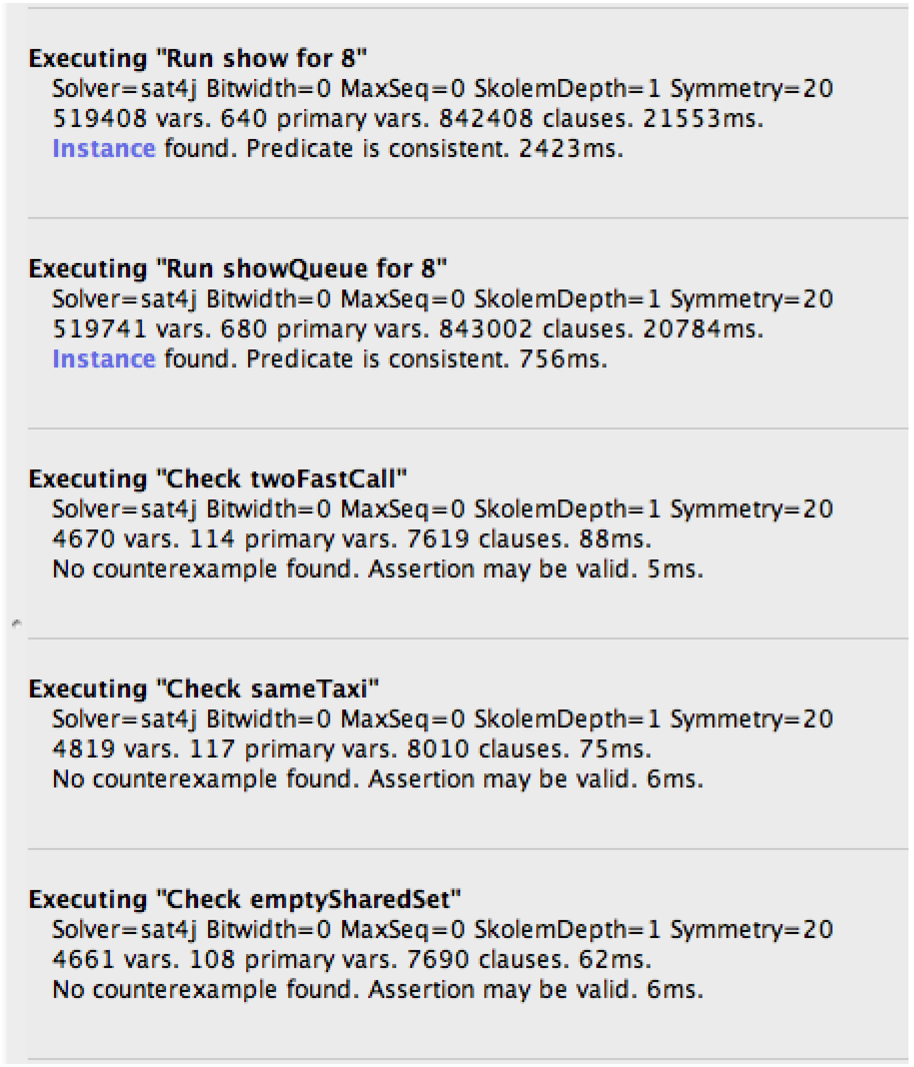
```
//General Person
abstract sig Person{
        }

//User
sig User extends Person{
        infoCard: one CreditCard,
        call: set Call
        }

//Credit card
sig CreditCard{
    }

//Zone
sig Zone {
    taxiOnService: some TaxiDriver, //A zone contains at least one taxi-driver, associated to that zone in that moment.
    taxiOnQueue: set TaxiDriver //A zone contains a queue of taxi. It can be empty.
    }


//Taxi-driver
sig TaxiDriver extends Person{
        }

//Call
abstract sig Call{
        user: one User, //User that has done the call
        driver: lone TaxiDriver, //Taxi-driver who accepted the request for the ride
        originZone: one Zone //Origin zone for the ride correspondent to the call
        }

//Fast call
sig FastCall extends Call{
        }

//Booked-call
sig BookedCall extends Call{
        }


//Shared-call - Call with "taxi-sharing" option enabled
sig SharedCall {
        call: one Call //A Shared-call refers to one and only one call
}
```

```
//Shared-call - Call with "taxi-sharing" option enabled
sig SharedCall {
	call: one Call //A Shared-call refers to one and only one call
}

//Shared-set: a set of shared-calls assigned to taxi-drivers.
sig SharedSet {
	callsAssigned:  set SharedCall
}

{
   #SharedSet <= 1 //Just one set of shared-calls can exist in the world
}

fact driverInZone{
   //A taxi driver should always have one and only one zone assigned
   all d:TaxiDriver | one z:Zone| d in z.taxiOnService
   all d:TaxiDriver | no disj z1,z2:Zone | ((d in z1.taxiOnService) && (d in z2.taxiOnService))
}

fact driverStatus{
   //If a taxi-driver is on queue on a certain zone, he cannot have a call associated
   all d:TaxiDriver, z:Zone | ((d in z.taxiOnQueue) iff (no c:Call | c.driver = d))
}


fact driverOnQueue{
   //If a taxi driver is on queue on a certain zone, he is also on service on that zone
   all d:TaxiDriver, z:Zone | (d in z.taxiOnQueue) implies (d in z.taxiOnService)
}


fact SameCallUser{
	//A call belongs to one and only one user
	all c:Call | no disj u1,u2:User | ((c in u1.call) && (c in u2.call))
	all c:Call, u:User | (c in u.call) iff (c.user=u)
}
```

```
//Constraints on calls per user
fact callPerUser{
    //A user can make just one fast call per time
    (no disj f1, f2: FastCall | f1.user = f2.user)
    //If a taxi driver is serving a user because of a call, no other taxi-driver serving this user can exist.
    //(A user cannot be served two times contemporarely)l
    all u:User | no disj c1, c2:Call | (c1 in u.call && c2 in u.call) && (one d1, d2:TaxiDriver | c1.driver = d1 && c2.driver=d2)
    //If a taxi-driver is serving a user that has booked the ride, no fast calls associated to this user can exists
    all u:User, b:BookedCall, d:TaxiDriver | (((b in u.call) && (d in b.driver)) implies (no f:FastCall | f in u.call))
    //If a user has a fast call associated to, no booked call served can exist
    all u:User, f:FastCall, d:TaxiDriver  | ((f in u.call) implies (no b:BookedCall | (b in u.call) && b.driver =d))
}

//All calls belong to user
fact allCallsToUser{
    Call in User.*call
}


fact OneCreditCardPerUser{
        //Each user has one and only one credit card per time
        all u:User | one c:CreditCard | u.infoCard = c
}

fact allCreditCardsHaveUser{
        //All credit card has a user associated
        CreditCard in User.^*infoCard
}


fact sharedTaxiForTwoCalls{
    //if two calls have the same taxi-driver, they are shared calls. It means that the correspondent users will be on the same taxi
    all disj c1, c2: Call | (c1.driver = c2.driver) implies ((one s1, s2:SharedCall, ss:SharedSet | s1.call = c1 && s2.call = c2 && s1 in ss.callsAssigned && s2 in ss.callsAssigned))
}

fact differentSharedCall{
    //Two different shared calls refer to two different calls
    no disj s1,s2: SharedCall | (s1.call = s2.call)
}

fact sharedCallInSharedSet{
    //A shared-call can be in the shared-set if and only if there is one taxi-driver that accepted that call
    all s:SharedCall | one ss: SharedSet | ((s in ss.callsAssigned) iff (one d:TaxiDriver | s.call.driver = d))
}
```

```
fact callPerDriver{
    //A taxi-driver can serve just one call per time, unless the calls are and in the shared-set
    all d:TaxiDriver, ss:SharedSet, s1,s2:SharedCall, c1, c2:Call | (c1.driver = d && c2.driver = d) implies ((s1.call=c1 && s2.call=c2) && (s1 in ss.callsAssigned && s2 in ss.callsAssigned))
    //If a call is served by a taxi-driver, it is in the shared-set
    all d:TaxiDriver, s:SharedCall, ss:SharedSet | (s.call.driver = d) iff (s in ss.callsAssigned)
    //A taxi-driver serving some shared-calls, can serve maximum 4 calls
    all d:TaxiDriver | no disj s1,s2,s3,s4,s5:SharedCall | ((s1.call.driver = d) && (s2.call.driver = d) && (s3.call.driver = d) && (s4.call.driver = d) && (s5.call.driver = d))
    //A call associated to a taxi driver comes from the zone of the taxi-driver, unless it is shared
    all d:TaxiDriver, c:Call, z:Zone | (d in c.driver && d in z.taxiOnService) implies ((c.originZone = z) or (one s:SharedCall, ss:SharedSet | c in s.call && s in ss.callsAssigned))
}

//A user cannot have two fast calls per time
assert twoFastCall {
    all u:User, disj f1,f2:FastCall | (f1 in u.call) implies !(f2 in u.call)
}
check twoFastCall


//If two users are on the same taxi, they should have done a shared-call
assert sameTaxi {
    all disj u1, u2:User, disj c1, c2:Call | ((c1 in u1.call && c2 in u2.call) && (c1.driver =c2.driver)) implies (one disj s1,s2:SharedCall | s1.call = c1 && s2.call = c2)
}
check sameTaxi

//If no shared call are served, shared set is empty
assert emptySharedSet {
     all s:SharedCall | one ss:SharedSet | !(s in ss.callsAssigned) implies (no d:TaxiDriver | s.call.driver = d)
}
check emptySharedSet


pred show {

}

run show for 8


pred showQueue {
    some z1,z2:Zone, c:Call, disj d1,d2:TaxiDriver | (d1 in z1.taxiOnQueue) && (d2 in z2.taxiOnService) && (d2 in c.driver)
}
run showQueue for 8
```

This is the result of the executions of our Alloy code.

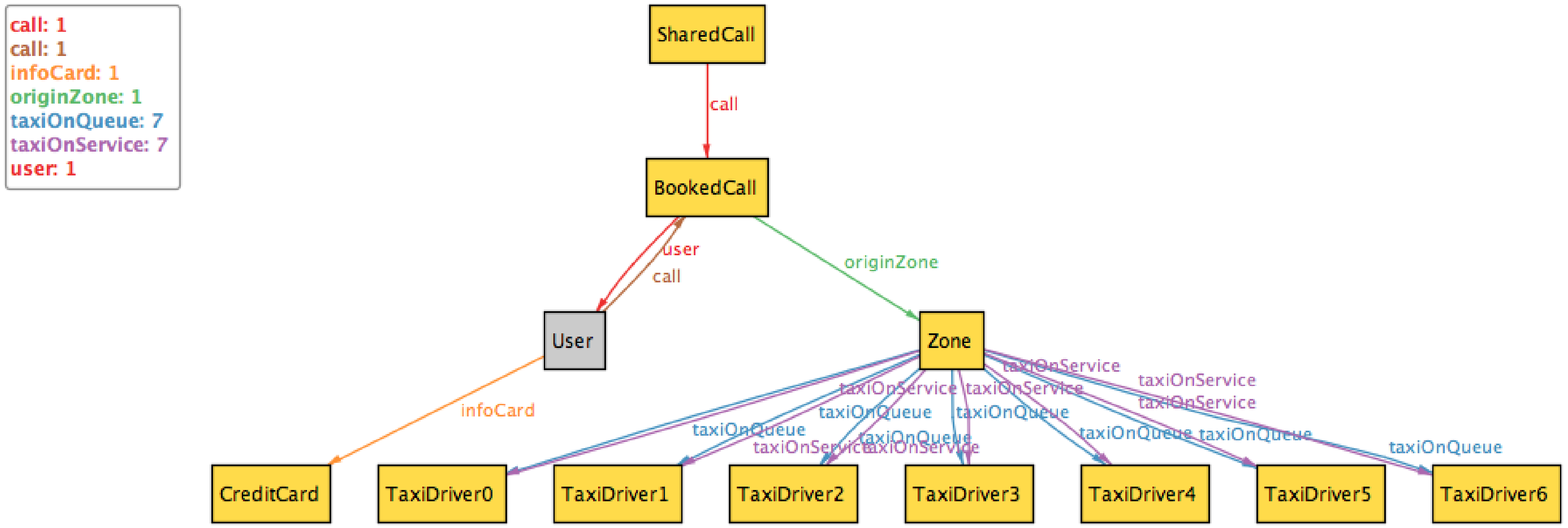


```
Executing "Run show for 8"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   519408 vars. 640 primary vars. 842408 clauses. 21553ms.
   Instance found. Predicate is consistent. 2423ms.

Executing "Run showQueue for 8"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   519741 vars. 680 primary vars. 843002 clauses. 20784ms.
   Instance found. Predicate is consistent. 756ms.

Executing "Check twoFastCall"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   4670 vars. 114 primary vars. 7619 clauses. 88ms.
   No counterexample found. Assertion may be valid. 5ms.

Executing "Check sameTaxi"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   4819 vars. 117 primary vars. 8010 clauses. 75ms.
   No counterexample found. Assertion may be valid. 6ms.

Executing "Check emptySharedSet"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   4661 vars. 108 primary vars. 7690 clauses. 62ms.
   No counterexample found. Assertion may be valid. 6ms.
```

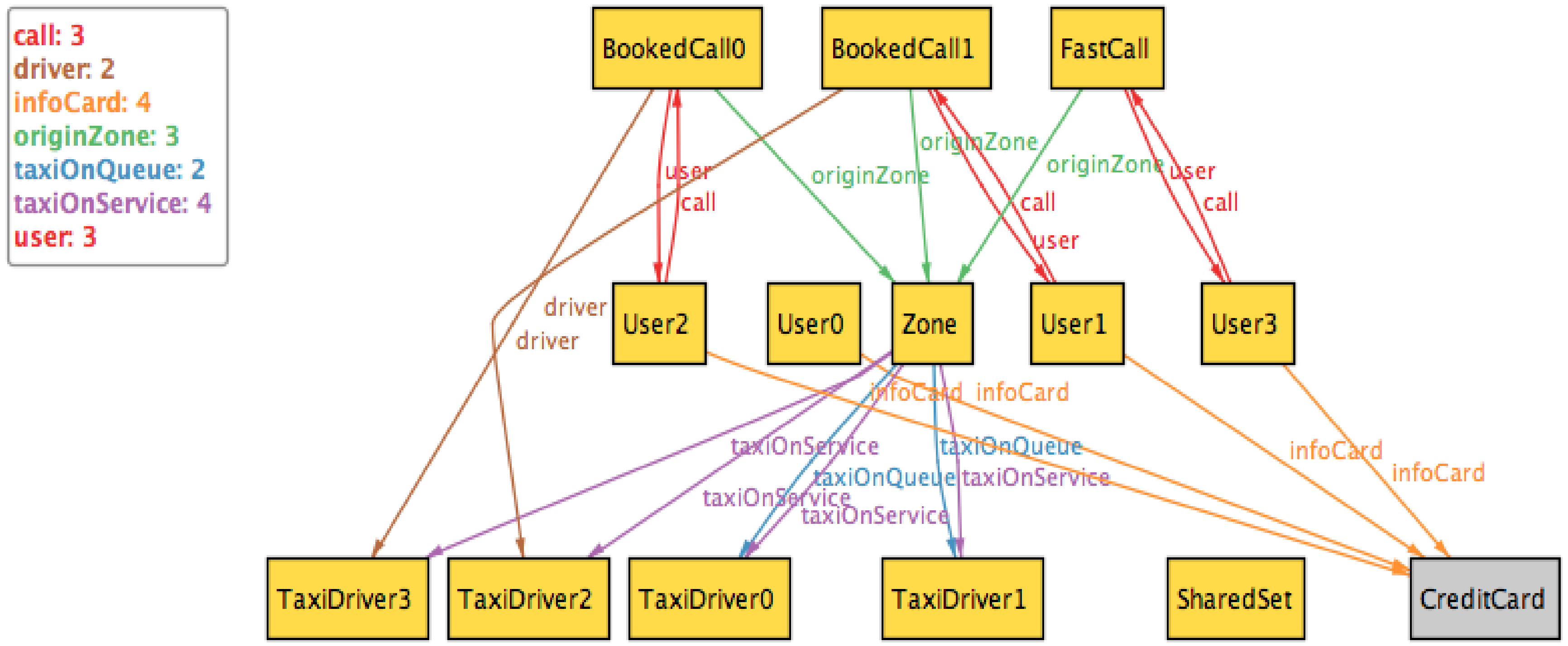
5.2 Alloy Worlds

Some iterations of the predicate *show*:

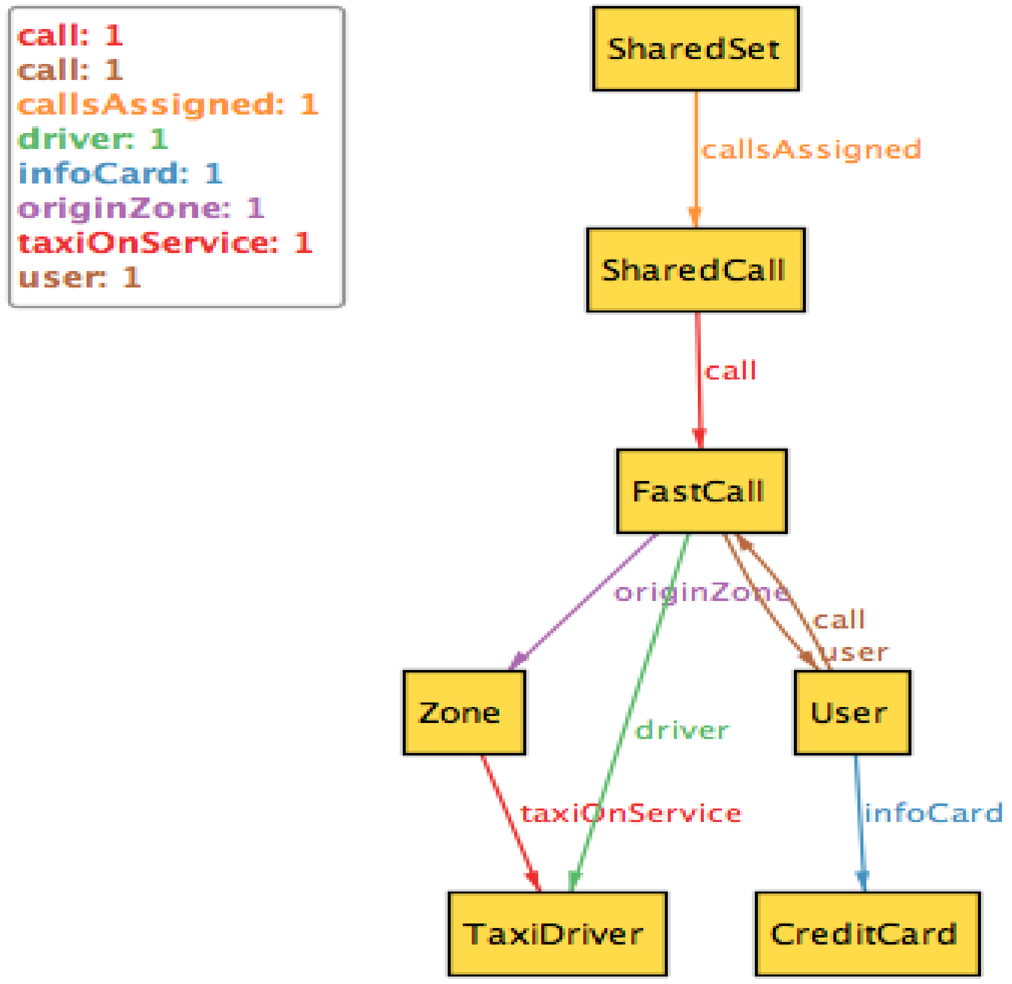
Some taxi drivers are in queue on the same zone; a user has made a reservation with "taxi-sharing" option, but no taxi has answered yet.



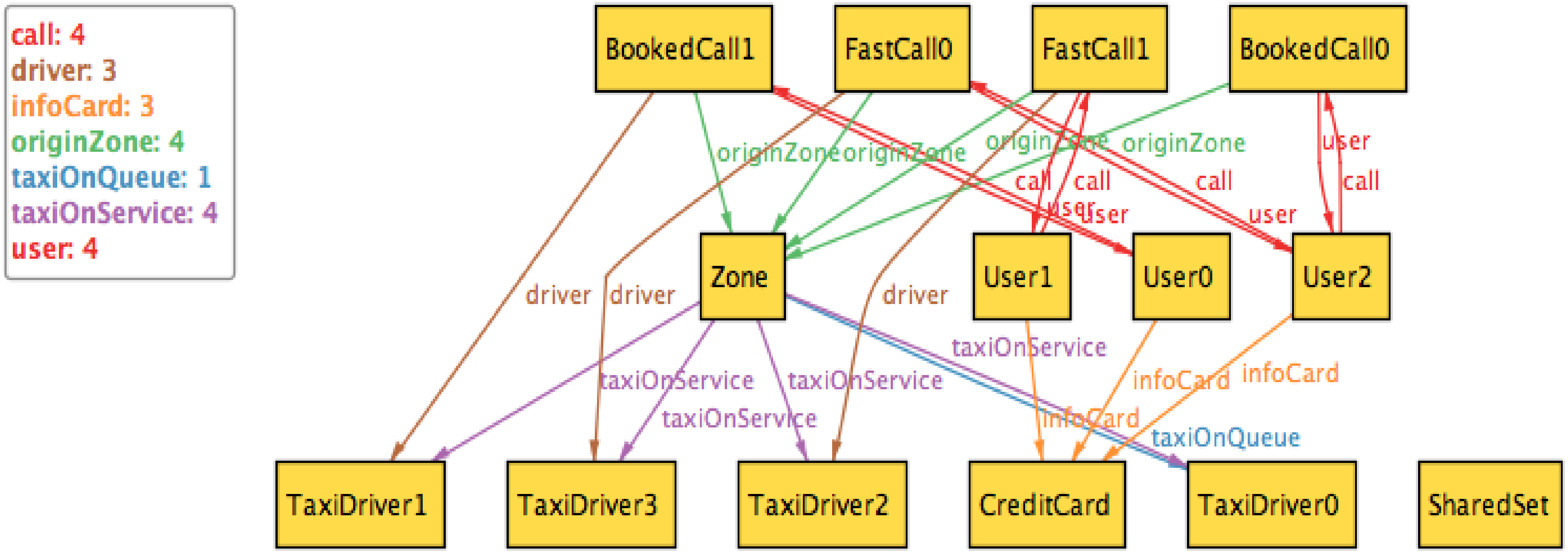
In this execution, there are four users; three of them make a private call and two different taxi drivers answer to them.



In this case, a user has made a fast call with "taxi-sharing" option enabled and the call has been accepted; thus, the call has been added to the shared-set.

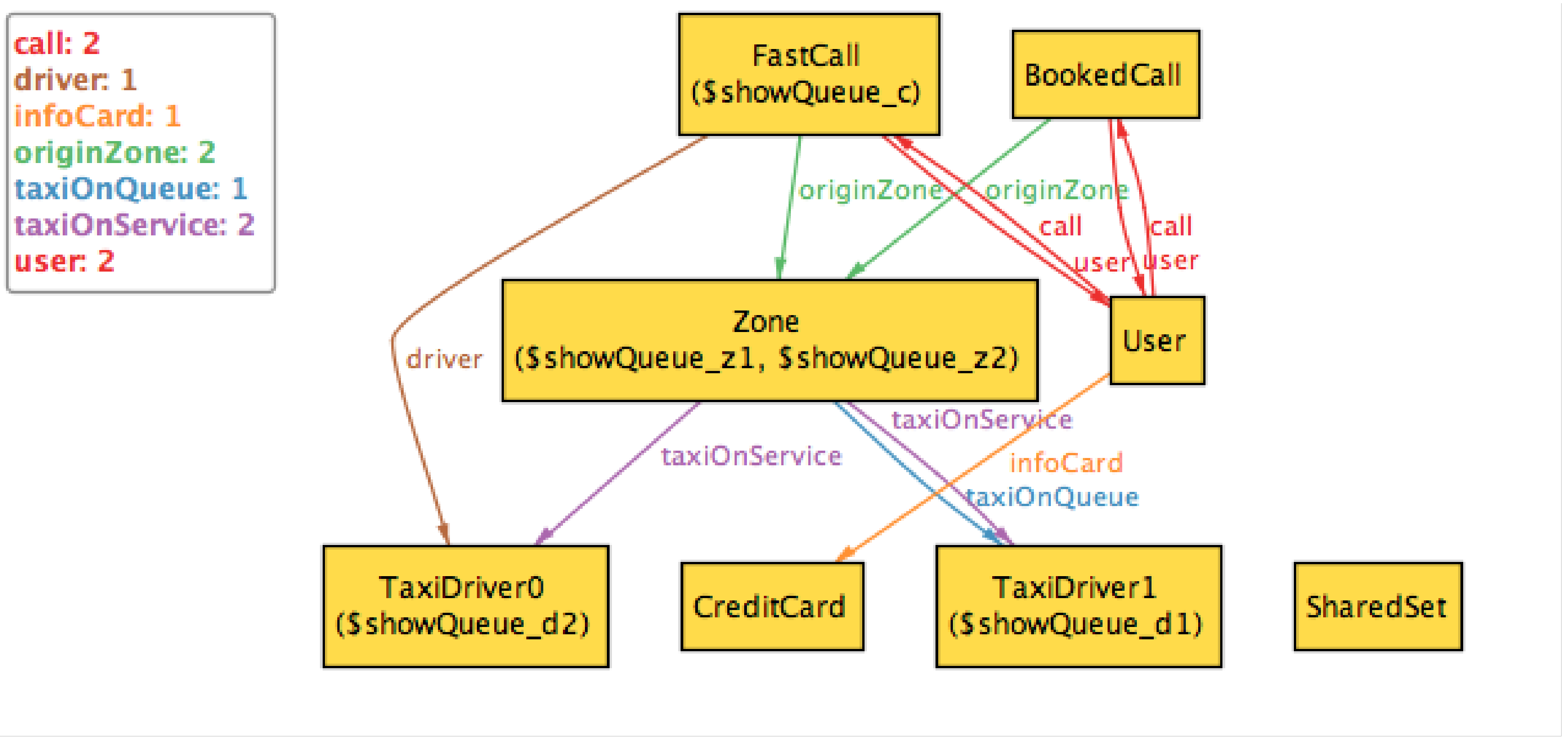


A user has made two calls, one of them is a fast call and it has been accepted by a taxi driver



One interaction of the predicate *showQueue*:

A user has made two calls from the same zone and one of them has been accepted by a driver. There is also another taxi driver in the queue of the zone.



Used tools

The tools we used to create the RASD document are:

- Microsoft Office Word 2013: to write and to format this document;
- Astah: to create the State Charts, the Activity Diagrams, the Class Diagrams, the Sequence Diagrams and the Use Case Diagrams;
- Alloy Analizer 4.2: to prove the consistency of our model.