Requirements Analysis and Specification Document

myTaxiService

AA 2015/2016

POLITECNICO
MILANO 1863

Carlo Broggi

Nicola Ghio

Summary

1 Introduction

1.1 Overview

We will project myTaxiService: an online software that aims to optimize the taxi service in a large city. It will make easier for the customer to grab a taxi as fast as possible while providing a fair management of the taxi queues.

The system that has to be developed will divide the city in different given zones in which the taxis will queue. The first driver in the queue receive requests coming only from the customer in the same zone and they will be able to chose whether to accept or to refuse them.

The system will allow registration of new customer along with their personal information. Once they registered they will be able to request a taxi or to reserve one.

The system will provide different User interfaces for Customers, who can access to the service by a mobile application or by the online website, and drivers, who will access only through mobile application. It will also provide an API in order to enable the development of additional services.

1.2 Goals

The system should provide the following features.

Customers should be able to:

- *Sign up to the system.*
- *Log into the system by mobile application or website.*
- *Request a taxi*
- *Reserve a taxi.*
- *Being notified by the system of the estimated time of arrival of the taxi and the code of that taxi.*

Drivers should be able to:

- *Log into the system through mobile application.*
- *Queue for new requests according to the city zone they are currently in.*
- *Accept a request.*
- *Refuse a request.*

Other features are:

- *System can provide a taxi for a customer who reserved it 10 minutes before the meeting time.*
- *System will send a notification to the user to warn him that his reservation has been received*
- *Current positions of Driver and Customers is given by GPS coordinates*

1.3 Glossary

In this section we define some words that will be used often in the document.

Users: a physical person using the system, it can be either a driver or a customer. The system will be provided with the following information about users:

- *Name*
- *Surname*
- *Password*
- *Phone number*
- *Address*
- *Social Security Number (SSN)*

Drivers: the physical person who drives the taxi, he will be able to do what has been described in the Goals session. The system will be provided also with:

- *Profile picture*
- *ID*

Customer: the physical person that requires a taxi, he will be able to do what has been described in the Goals session.

Guest: User not yet registered. He can access only to the public area of the website or sign up.

System Administrator: a user who works directly for the government, he's in charge of managing the data of the database.

Request: we mean the possibility for the customer to ask for a taxi. The request will be forwarded by the system to the first taxi in the queue of the zone of the customer.

Reservation: a functionality that will assure that a taxi will be available for the customer ten minutes before the meeting time.

Ride: it's the moment in which the drivers brings the customer to the destination. Starts when a customers gets on a taxi and ends when he exit.

1.4 Domain properties

We suppose that the following properties hold in the analysed domain:

- *There are enough taxi to cover all requests/reservations.*
- *The drivers will automatically distribute themselves in order to cover any zone of the city.*
- *If a driver accepts a request, he will be present at the meeting.*
- *Location of Users is given by a reliable GPS system.*
- *Location of Taxi is given by a reliable GPS system.*
- *Customer's destination could be anywhere, even outside town.*
- *If the destination is outside the town, the driver will come back and will queue again.*
- *The driver is already in possess of a certified device that will calculate the cost of the ride.*
- *The driver is already in possess of a device that allows him to be paid with bancomat or cash.*
- *If a customer sends a request/reservation it will be accepted by the system.*

1.5 Assumptions

Since the assignment lacks some fundamental details we assume the following facts:

- *There is a System Administrator who is in charge of adding and removing taxi drivers from the database.*
- *The administrator will also have access to all the informations about Drivers and Customers.*
- *For security reasons, the data of the Users involved in every request/reservation and the data of each ride will be saved in the database according to the local privacy law.*
- *When requesting a taxi, if a customer does not specify it, the meeting point will be the actual position of the customer, provided by a GPS.*
- *The customer will be able to change his personal information from an apposite page.*
- *For security reasons, every change of personal information made by the User will be saved in the database according to the local privacy law.*
- *If a driver accept a request/reservation he will not be able to accept any other request until the end of the ride.*
- *If the taxi leaves the zone before having accepted a request it will be removed from the queue he was in.*
- *If a taxi does not reply to a new request within 1 minute the system will automatically consider it as a refusal.*
- *If the customer is not present the driver can use a functionality that will be unlocked 10 minutes after the meeting time to signal this to the system which will apply a fee of 5€ the next time the customer will reserve a taxi. The Customer will be also notified when this fee is applied.*
- *If the customer has more than 15€ worth in fees, he will be banned from the System and his account will be deactivated.*
- *If there is no internet connection, the User cannot log in.*
- *If suddenly the internet connection goes down, the User will be logged out.*

1.6 Stakeholders

We have identified the following stakeholders:

- *The government of the city. It is our main stakeholder and is interested in a reliable system that will, according to the local privacy law, store in the database all the information about the Users and the rides in order to maximize their security.*

- *Users: they will need a piece of software that will be easily understandable for every kind of person, from younger people to the older ones. They will also need a reliable mobile application that will save their data in case of the app crashes. The application should also be compatible with different operative systems (Android, IOS, Windows) and light enough to be supported by most smartphones.*

1.7 Actors identifying

We have identified the following Actors:

- *System Administrator: The person who is in charge of adding and eliminating drivers from the database. It also has access to all the information about users and rides.*
- *Customer: Person who will reserve/request a taxi, they will do it from a mobile application or from the website*
- *Drivers: the drivers of the taxi.*

2 Requirements

2.1 Functional Requirements

Given our assumptions and trusting that the domain properties will hold, we have identified the some requirements in order to achieve the goals:

Sign up:

- *the system must provide a sign in functionality for customers.*
- *In order to register the customer must accept the policy conditions and must guarantee. with a self-certification that all his personal data corresponds to truth.*
- *When accepting the use conditions the customer accepts to pay a fee of 5€ in his next ride if he is not present at the meeting.*

Log in: the system must provide log in functionality. User can decide to save credential on the device used to log in in order not to insert them every time.

- *If the user is logging through mobile application he must have GPS activated.*

Customer will be provided with the following functionalities:

- *PersonalPage: this functionality will provide to the customer a page with his personal information.*
  - *He can check them.*
  - *He can change the following features:*
    - *Passoword*
    - *Email*
    - *Address*
    - *Phone Number*
  - *If a user changes his informations, a copy of the previous one will be stored in the Database.*
- *Request a taxi: use this functionality to require a taxi.*
  - *This function can be used only if the customer is in a known zone of the city.*
  - *If the customer is logged via website he must insert the meeting location.*
  - *If the customer is logged via mobile application he can decide whether to insert the meeting location or not: if he doesn't, the meeting point will automatically be the GPS location of the customer.*
  - *The request will be seen only by the first taxi of the zone in which that request has been made.*
  - *If a taxi accepts the request the customer will be notified with the estimated arrival time and the code of the taxi.*
- *Reserve a taxi: the system must provide to customers a functionality to require a taxi.*
  - *Customer must insert the meeting time and the meeting point.*
  - *The meeting point must be a known zone of the city.*
  - *The meeting time must be at least 2 hour after the time the reservation has been made.*
  - *If details are consistent the system sends a confirmation to the user.*
  - *The reservation will be seen as a normal request by any taxi in the zone of the meeting 10 minutes before the meeting time.*
  - *When a taxi accepts this new request customer will be notified with the code of the taxi.*

- *The system will notify the customer when a taxi has been deployed, like a normal request.*

Driver will be provided with the following functionalities:

- *Queue: the system grants to the driver a functionality to get into the queue for the zone he is in. If he's not in a known zone, this functionality will be disabled.*
  - *Once activated, this functionality shows the position of the driver in the queue.*
  - *If the driver is the first of a queue, the system will notify only him when a new request arrives in the zone.*
  - *He will be removed from the queue if he leaves the zone in which he is currently queuing.*
  - *If the mobile application fails, the current queue position of the driver will be saved so that when he logs in again he will be still in the queue at the same position.*
  - *If the driver is the first in the queue and a request arrives while he is still logged out, the request will be dispatched to the second driver.*
- *Accept/Refuse requests: this functionalities manage the decisions of the driver and has the following properties:*
  - *If the driver is the first of a queue, he will be notified that a new request is waiting for him along with the address of the meeting point.*
  - *If he accepts he will be removed from the queue and he will not be able to queue. until he rides the passenger to his destination or until he fees the customer.*
  - *If he refuses a request he will be placed in the bottom of the queue.*
  - *If he does not respond within 1 minute it will be considered as a refusal.*
  - *At the end of the ride the driver can get available again though a button so that he will be able to queue again.*
- *CheckQueues: shows how many drivers are queued in which zone*
  - *The list appears as a simple table, updated every 10 seconds.*
  - *At the bottom of the table, the total number of drivers queuing is shown.*
- *FeeCustomer: This functionality allows the Taxi driver to fee a customer who is not present at the meeting*
  - *It becomes available only after 10 minutes of the estimated arrival time of the taxi.*
  - *Will apply a fee of 5€ that has to be paid by the Customer at his next ride.*
  - *If a Customer has more than 15€ fee he will be automatically banned from the system.*
  - *Once used, the driver can queue again.*

2.1 Non-functional Requirements

We have identified the following non-functional requirements:

- *Available 24/7: Since this is a taxi system, we think that the system should always be available at any time*
- *Robustness: The system should also be replicated, in order to be less likely to crash under unexpected conditions. This will also allow a periodic maintenance to be performed without interrupting the service. For this feature we thought to use: VMWARE ESXI a bare metal hypervisor with multiple physical machine in a cluster.*
- *Security: Since the database is filled with confidential information the system should grant the protection to these data.*
- *Since this is a large town, we have estimated that the system must support at least 3000 people connected at the same moment in order to satisfy every request.*
- *The system will grant access to customers both from mobile application as from a website.*
- *Drivers can access only though mobile application.*
- *The Users will have different user interfaces according to their kind (Customer, driver).*
- *The system administrator will access only though website and will also have a different user interface.*

3 Possible Scenarios

Here we list some possible scenarios of myTaxiService:

Bob want to become a Taxi driver in his town, In order to do so he goes to the Taxi Service office in the Town Hall and compiles the form. His request is accepted and the Administrator of the System insert him into the Database and will provide him with username, password and an ID which will be delivered to him through certified mail. Bob downloads the mobile application myTaxiService since he can finally login with the credentials he received.

Bob is a Taxi driver and wants to work. He jump on his taxi and logs into myTaxiService with the credentials provided to him. He wants to have a customer as soon as possibile therefore he check the taxi queues in the zone of the city. After seeing how the taxi are disposed he decide to go to a zone in which, by experience, he knows that he will begin a ride as soon as possible. Once arrived he presses the "Queue" button so that he can wait for a customer; while waiting he can check how many drivers are waiting before him in that zone just looking at the screen.

John is a customer and wants to request a taxi to go to the stadium from his home. He then logs into myTaxiService via website and presses on "request a taxi". The system asks him where the meeting point should be so John insert his address and presses "confirm". Bob is a Taxi Driver and he's the first of his queue. The app finally notify him that there is a pending request waiting for him. He reads the address of the meeting point, decides to accept the request and starts driving to the location. The system will automatically inform the customer with the ID and the estimated arrival time of Bob's Taxi. John wait for the taxi outside his house and, when Bob arrives, John recognises him by the ID code printed on his myTaxiService license. John jumps on the taxi and communicate to Bob the desired destination, Bob understands and start driving. At the end of the ride John pays Bob with an amount of cash given by a certified device already in possess of Bob. Bob confirms to the system that he is now free and can decide whether to queue in that zone or to change location.

John is a customer and wants to go his friend's house at 5 pm. Since it is only noon it is too soon to request a taxi so he logs into myTaxiService with his phone and press the "Reserve a Taxi" button. He insert the meeting point address at his house and the meeting time at 3 pm, he confirms the reservation and the system notifies him that his reservation ha been accepted. Bob is a taxi driver and he's the first of his queue. He receive a new request notification at 2:50 pm, he accepts it and he drive to the meeting point. After he accepts it, John receive a notification saying that a taxi has accepted his reservation along with the taxi ID. Like in the example before, John meets Bob at the established time at his house and gets on the taxi. Bob rides until the destination, John pays him and gets off the taxi. John would like to work again but this travel brought him outside the city. Therefore he drives to the city and queue again in the zone he wants, according to the information provided by the "CheckQueue" function.

Bob is a taxi driver and he's the first of his queue. He receives a request notification but he does not see it because he's away from his taxi. Jane is a taxi driver and she is second in the local queue. After the minute in which Bob does not accept the request she is automatically shifted in first position and immediately notified with the request that she accepts. After some minutes Bob gets back only to find out that now he is one of the last driver in the queue. He then realizes that a request has arrived while he was away and he has to wait again.

Bob is a taxi driver and he's the first of his queue. He receive a request notification but he suddenly needs to go away from the taxi for personal issues. He then refuses the request and becomes the last of his queue.

Bob is a taxi driver and he is queuing in a zone of the city. He thought that it was a good idea to wait for customers in this area but today it looks like no one here is requesting a taxi. By consulting the "CheckQueue" functionality he realizes that in another zone of the city, a lot of taxi are working therefore he decides to drive his taxi into this new zone.

As soon as he drives away from the initial zone he's removed from the queue of that zone so that every driver who was behind him shift one position ahead. He then arrives to the desired zone and starts queuing again.

John is a customer which already has 10€ fee on his account because he missed a taxi 2 times. He request a taxi though mobile phone and receives a notification that his taxi will arrive in 5 minutes to his current location. Before the taxi arrives John meets an old friend and start talking to him. Bob is the taxi driver who accepted John's request, when he arrives at the meeting he finds no one. He, so, waits for 10 minutes but since no one got on his taxi he decides to fee the customer so he can queue again. During this time John did not realize that the talk has been going on for more than 15 minutes when he receives a notification from the system saying that he's been feed again and his account has been blocked. In order to be able to use myTaxiService again John has to go to the office located in the Town Hall.

John has disappeared and the police needs a place to start the research. They asks to the Administrator of the system if John used myTaxiService before he disappeared, it turns out that he requested a taxi at a certain address but he never showed up. The police ask to the Administrator who was the driver who accepted the request and, after retrieving the name, will start the researches by looking by that address and by questioning the driver.

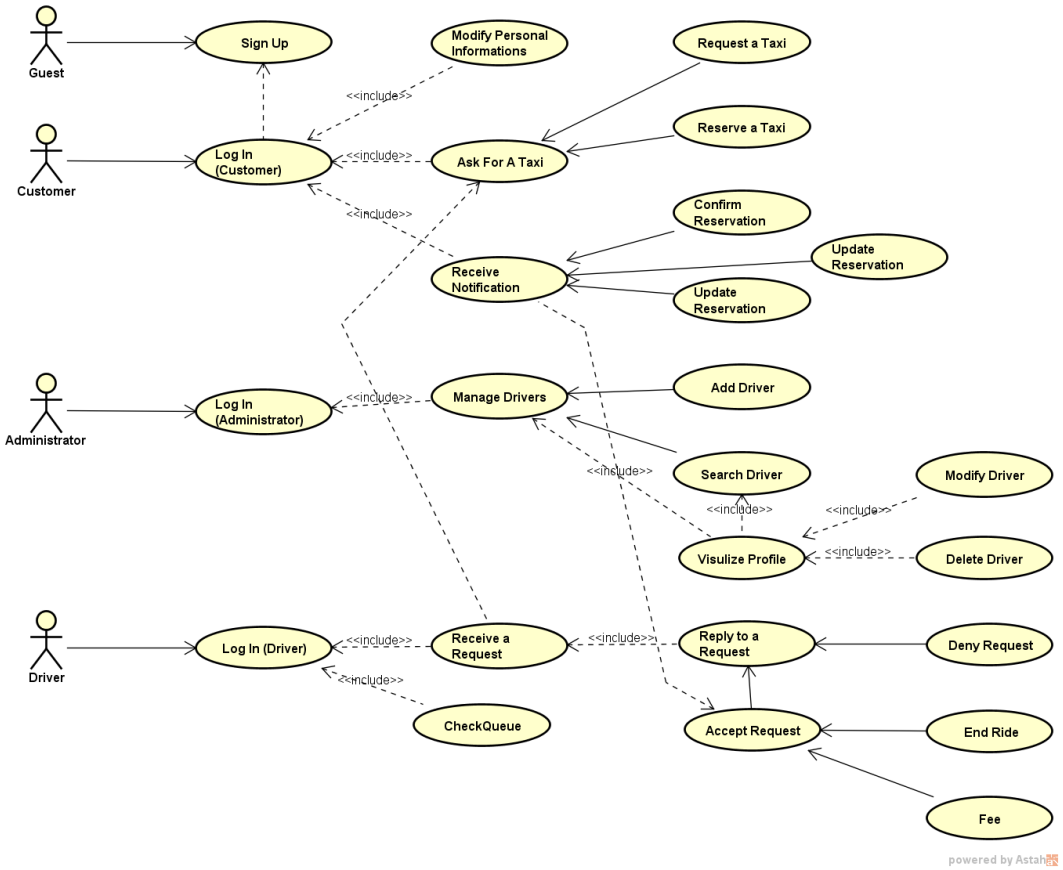
Bob is queuing in a zone of the city and he almost reached the first position. Suddenly the app crashes and is automatically killed by the Operative System of his phone. Since this is a forecast behaviour Bob can log in again with his credentials and he will find himself at the same position of the queue in which he was before. Jane is a taxi driver and she's just after Bob in the queue. When Bob's phone crashes a driver that is before John in the queue leaves the queue: since Bob hasn't logged in yet, Jane overpasses Bob in the queue.

4 UML Models

4.1 Use Case Diagram

We have derived some use cases from the scenarios described:

- *Sign Up*
- *Log In (Customer)*
- *Modify Personal Informations*
- *Ask For A Taxi*
- *Receive Notification*
- *Log In (Administrator)*
- *Manage Driver*
- *Add Driver*
- *Search Driver*
- *Visualize Profile*
- *Modify Driver*
- *Delete Driver*
- *Log In (Driver)*
- *Receive a Request*
- *Check Queue*



4.2 Use Case Description

In this paragraph we describe in detail the use cases listed above.

Name	*Sign Up*
Actors	*Guest*
Entry conditions	*The guest isn't registered and can only access public areas on the web site*
Flow of events	• *The guest enters the website/mobile application* • *The guest clicks on the "SIGN UP" button* • *The guest fills in the form where he has to write:* ➢ *SSN* ➢ *Name* ➢ *Surname* ➢ *Email* ➢ *Password* ➢ *Date of birth* *Optionally:* ➢ *Telephone number* ➢ *Address* ➢ *Picture* • *The guest clicks on the "DONE" button*
Exit conditions	*The system accepts the registration and shows the customer main page*
Exceptions	*The system rejects the registration due to fields not correctly formatted or filled*

Name	*Log In (Customer)*
Actors	*Customer*
Entry conditions	*Customer with an active account wants to access the private areas*
Flow of events	• *The customer enters the website/mobile application* • *The customer fills in the text fields in order to log in (email and password)* • *The customer clicks on the "LOG IN" button*
Exit conditions	*The system shows the customer main page*
Exceptions	*The system rejects the authentication due to wrong log in data*

Name	*Modify Personal Informations*
Actors	*Customer*
Entry conditions	*Customer must be logged in and clicks on "MODIFY PERSONAL INFORMATION"*

Flow of events	• *The system shows the "MODIFY PERSONAL INFORMATION" page*
	• *The customer can modify:*
	  ➢ *Email*
	  ➢ *Password*
	  ➢ *Telephone number*
	  ➢ *Address*
	  ➢ *Picture*
	• *The customer clicks on the "DONE" button*

Exit conditions	*The system accepts the modification and shows the customer main page*
Exceptions	*The system rejects the modification due to fields not correctly formatted or filled and remains on the current page*

Name	*Ask For A Taxi*
Actors	*Customer*
Entry conditions	*Customer must be logged in and clicks on "ASK FOR A TAXI"*

Flow of events	• *The system shows the "ASK FOR A TAXI" page*
	• *The customer choose to request or reserve a taxi:*
	  ➢ *If the customer wants to request a taxi, must enter the starting point manually (website/mobile application) or by GPS (mobile application)*
	  ➢ *If the customer wants to reserve a taxi, must enter the starting point manually and the final destination*
	• *The customer clicks on the "DONE" button*

Exit conditions	*The system accepts the request/reservation and shows the customer main page (a notification of confirmation will be send)*
Exceptions	*The system rejects the request/reservation if field are not correctly filled and remains on the current page*

Name	*Receive Notification*
Actors	*Customer*
Entry conditions	*Customer must be logged in*
Flow of events	• *When there is a new notification of confirmation, the system notifies it at the customer* • *The system sends to the customer one of the following notification types:* ➢ *Request accepted: the request of a taxi is accepted and is notified the arrival time of the taxi and the taxi number* ➢ *Reservation accepted: the reservation of a taxi is accepted (more info will be notificated later)* ➢ *Reservation update: the system tells the taxi number of a reservation*
Exit conditions	*After the customer reads the notification, the system shows the customer main page*
Exceptions	-

Name	*Log In (Administrator)*
Actors	*Administrator*
Entry conditions	*The administrator wants to access the private areas*
Flow of events	• *The administrator enters the website/mobile application* • *The administrator fills in the text fields in order to log in (email and password)* • *The administrator clicks on the "LOG IN" button*
Exit conditions	*The system shows the administrator main page*
Exceptions	*The system rejects the authentication due to wrong log in data*

Name	*Manage Driver*
Actors	*Administrator*
Entry conditions	*Administrator must be logged in and clicks on "MANAGE DRIVERS"*
Flow of events	• *The system shows the "MANAGE DRIVERS" page* • *All drivers are presented in a table*
Exit conditions	*The page is fully loaded by the system*
Exceptions	-

Name	*Add Driver*
Actors	*Administrator*
Entry conditions	*Administrator must be logged in, is in the "MANAGE DRIVERS" page and clicks on "ADD DRIVER"*

Flow of events	• *The administrator fills in the form where he has to write:*
	  ➢ *SSN*
	  ➢ *Name*
	  ➢ *Surname*
	  ➢ *Email*
	  ➢ *Password*
	  ➢ *Date of birth*
	  ➢ *Telephone number*
	  ➢ *Address*
	  ➢ *Picture*
	• *The administrator clicks on the "DONE" button*

Exit conditions	*The system accepts the new driver and shows the "MANAGE DRIVERS" page*
Exceptions	*The system rejects the new driver due to fields not correctly formatted or filled*

Name	*Search Driver*
Actors	*Administrator*
Entry conditions	*Administrator must be logged in, is in the "MANAGE DRIVERS" page and clicks on "SEARCH DRIVER"*
Flow of events	• *The administrator searches for a specific driver using his name (filling a text box)* • *Then clicks on the "EXECUTE" button*
Exit conditions	*The system shows the administrator a table where he can select the driver*
Exceptions	*The system shows an empty table if no drivers are found*

Name	*Visualize Profile*
Actors	*Administrator*
Entry conditions	*Administrator must be logged in and select a driver from a table (from the "MANAGE DRIVER" or "SEARCH DRIVER" page)*
Flow of events	• *The system show the profile of the selected driver*
Exit conditions	*The system loads the page*
Exceptions	-

Name	*Modify Driver*
Actors	*Administrator*
Entry conditions	*Administrator must be logged in and is visualizing the profile of a driver, then clicks on "MODIFY DRIVER"*

Flow of events	• *The system shows the "MODIFY DRIVER" page*
	• *The administrator can modify:*
	  ➢ *SSN*
	  ➢ *Name*
	  ➢ *Surname*
	  ➢ *Email*
	  ➢ *Password*
	  ➢ *Date of birth*
	  ➢ *Telephone number*
	  ➢ *Address*
	  ➢ *Picture*
	• *The administrator clicks on the "DONE" button*

Exit conditions	*The system accepts the modification and shows the "VISUALIZE PROFILE" page*
Exceptions	*The system rejects the modification due to fields not correctly formatted or filled and remains on the current page*

Name	*Delete Driver*
Actors	*Administrator*
Entry conditions	*Administrator must be logged in and is visualizing the profile of a driver, then clicks on "DELETE DRIVER"*
Flow of events	• *When the administrator want to delete a driver, select the "DELETE DRIVER" button*
Exit conditions	*The system deletes the driver and shows the "SEARCH DRIVER" page with the last search*
Exceptions	-

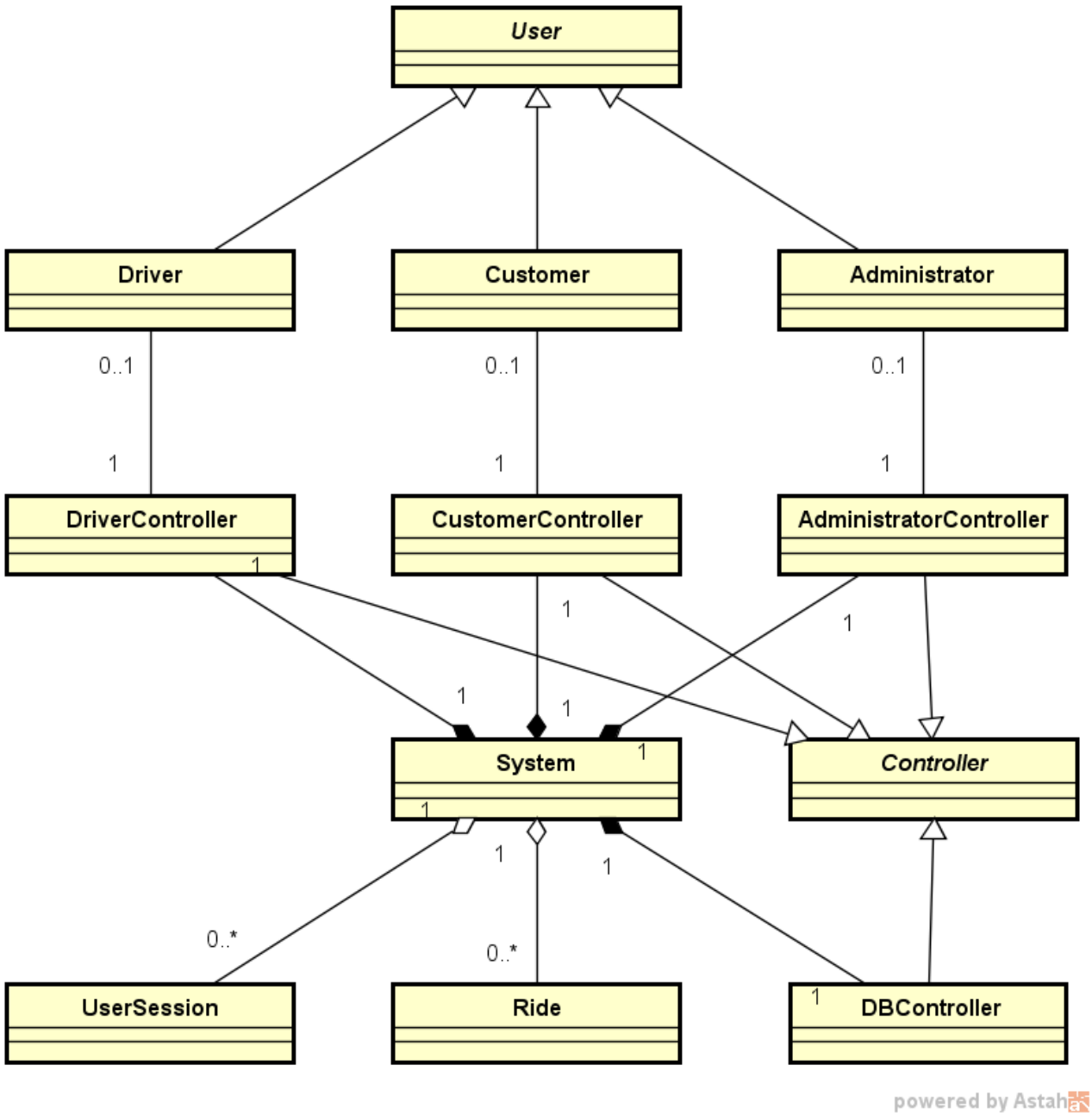
Name	*Log In (Driver)*
Actors	*Driver*
Entry conditions	*Driver with an active account wants to access the private areas*
Flow of events	• *The driver enters the website/mobile application* • *The driver fills in the text fields in order to log in (email and password)* • *The driver clicks on the "LOG IN" button*
Exit conditions	*The system shows the driver main page and makes the taxi online*
Exceptions	*The system rejects the authentication due to wrong log in data*

Name	*Receive a Request*
Actors	*Driver*
Entry conditions	*The driver must be logged in*
Flow of events	• *The system shows the driver a message asking to accept o negate a request* • *The use case REPLY TO AN REQUEST starts*
Exit conditions	*The driver accepts the request and the taxi is now busy*
Exceptions	*The driver deny the request and waits for another request*

Name	*Check Queue*
Actors	*Driver*
Entry conditions	*The driver must be logged in and clicks on "CHECK QUEUE"*
Flow of events	• *The system shows the driver the current status of all queues*
Exit conditions	*The system complete the load of the page*
Exceptions	-

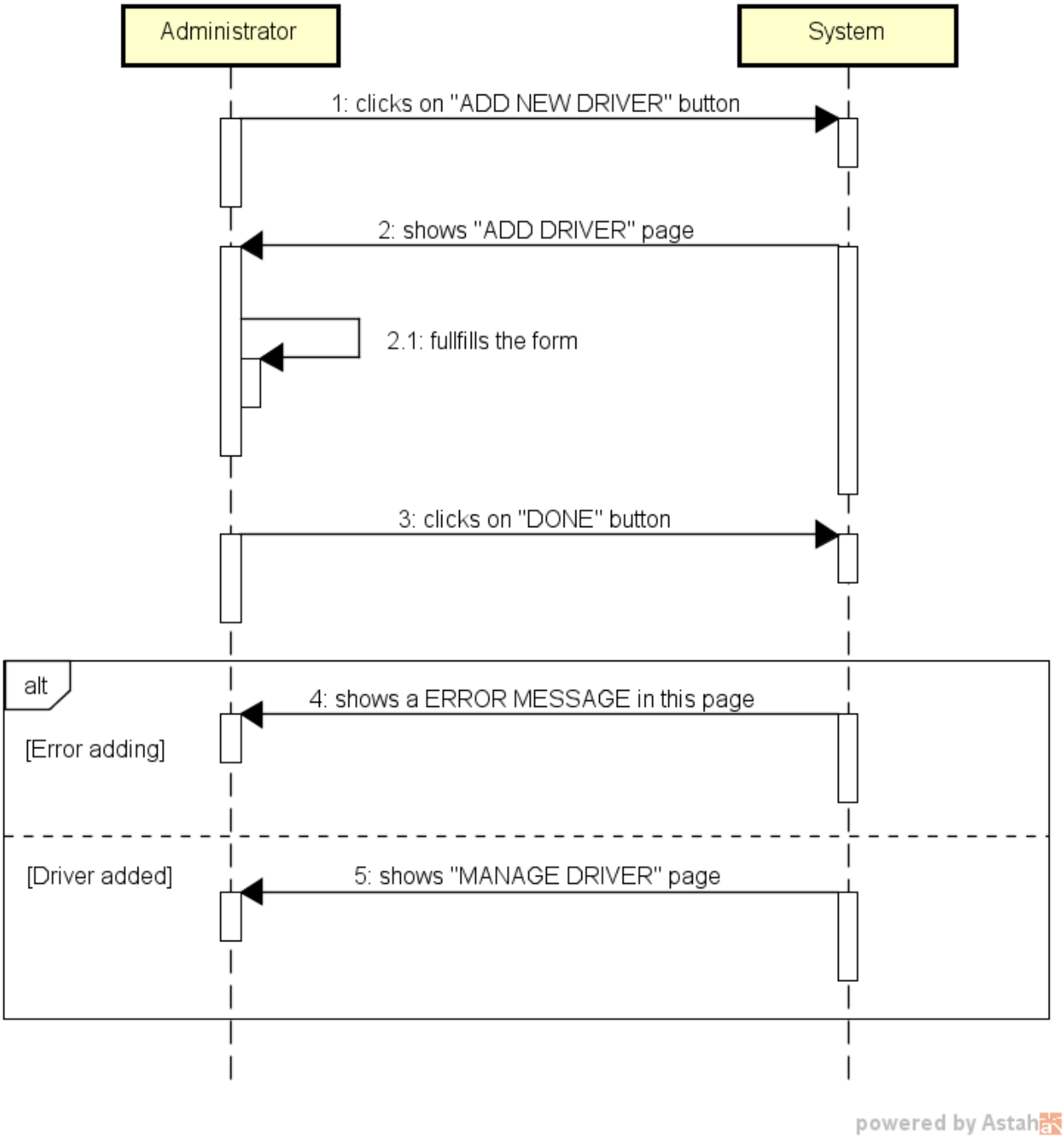
4.3 Class Diagram

Here we have the class diagram of our system:

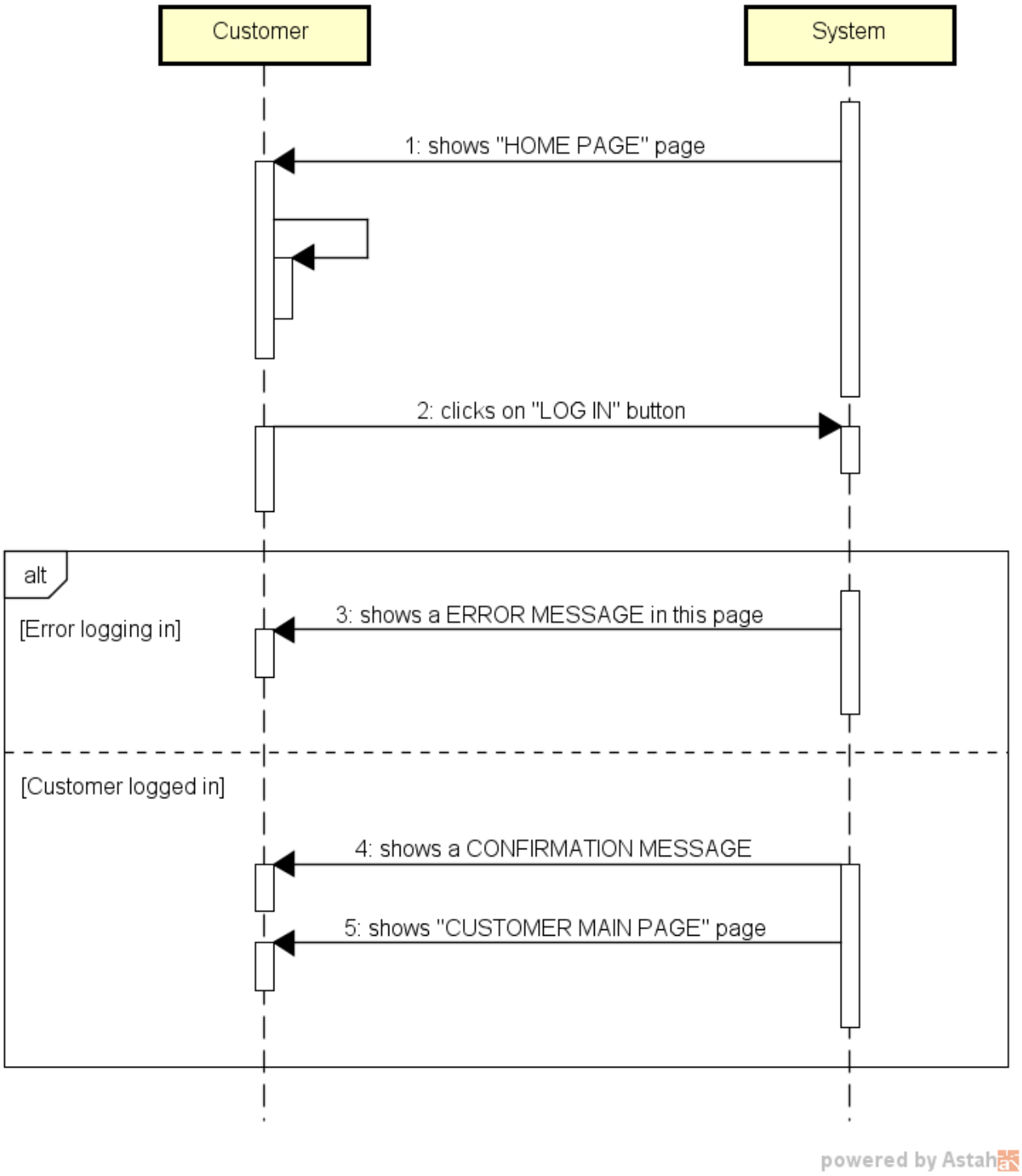


4.4 Sequence Diagram

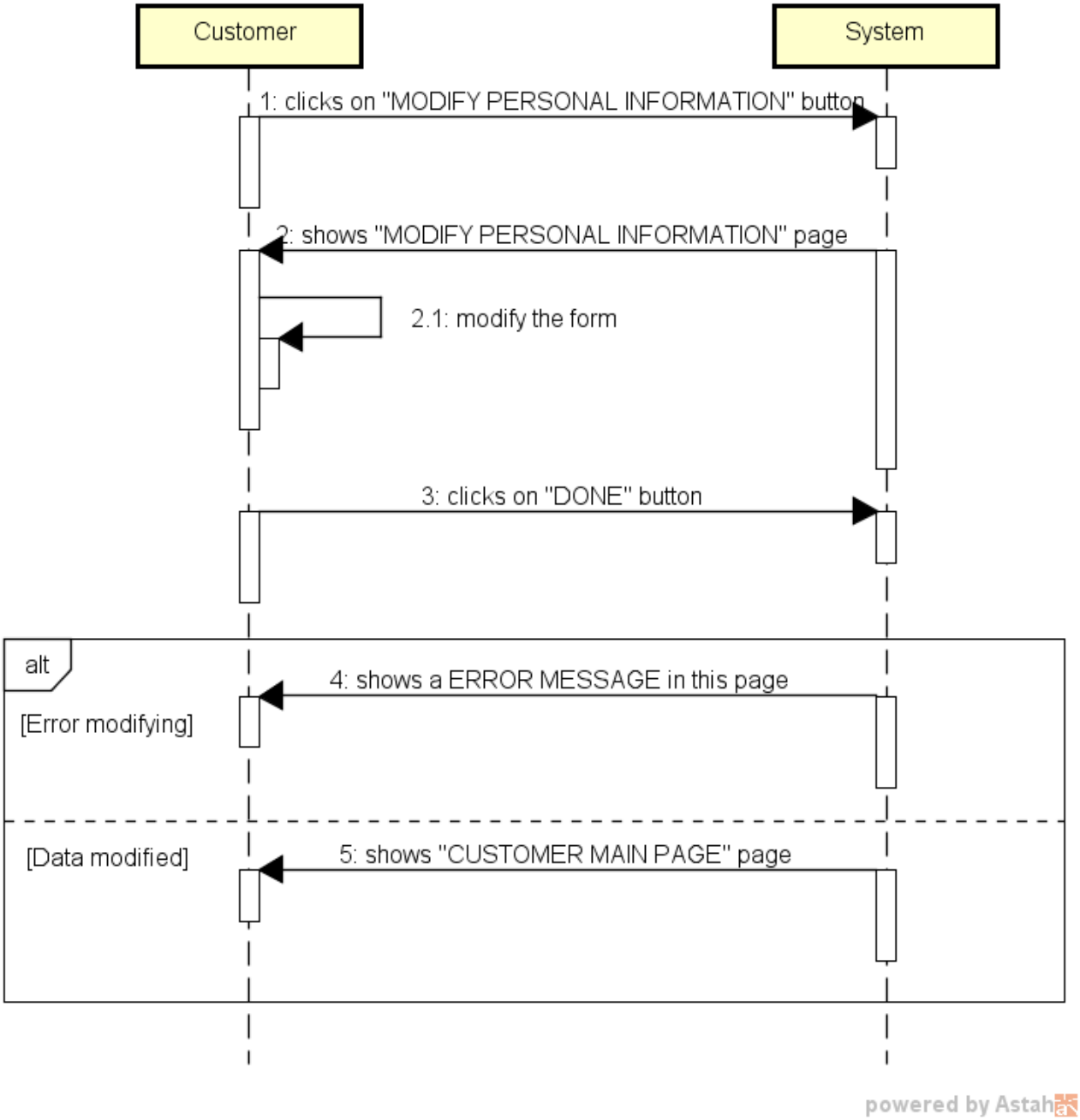
4.4.1 Add New Driver



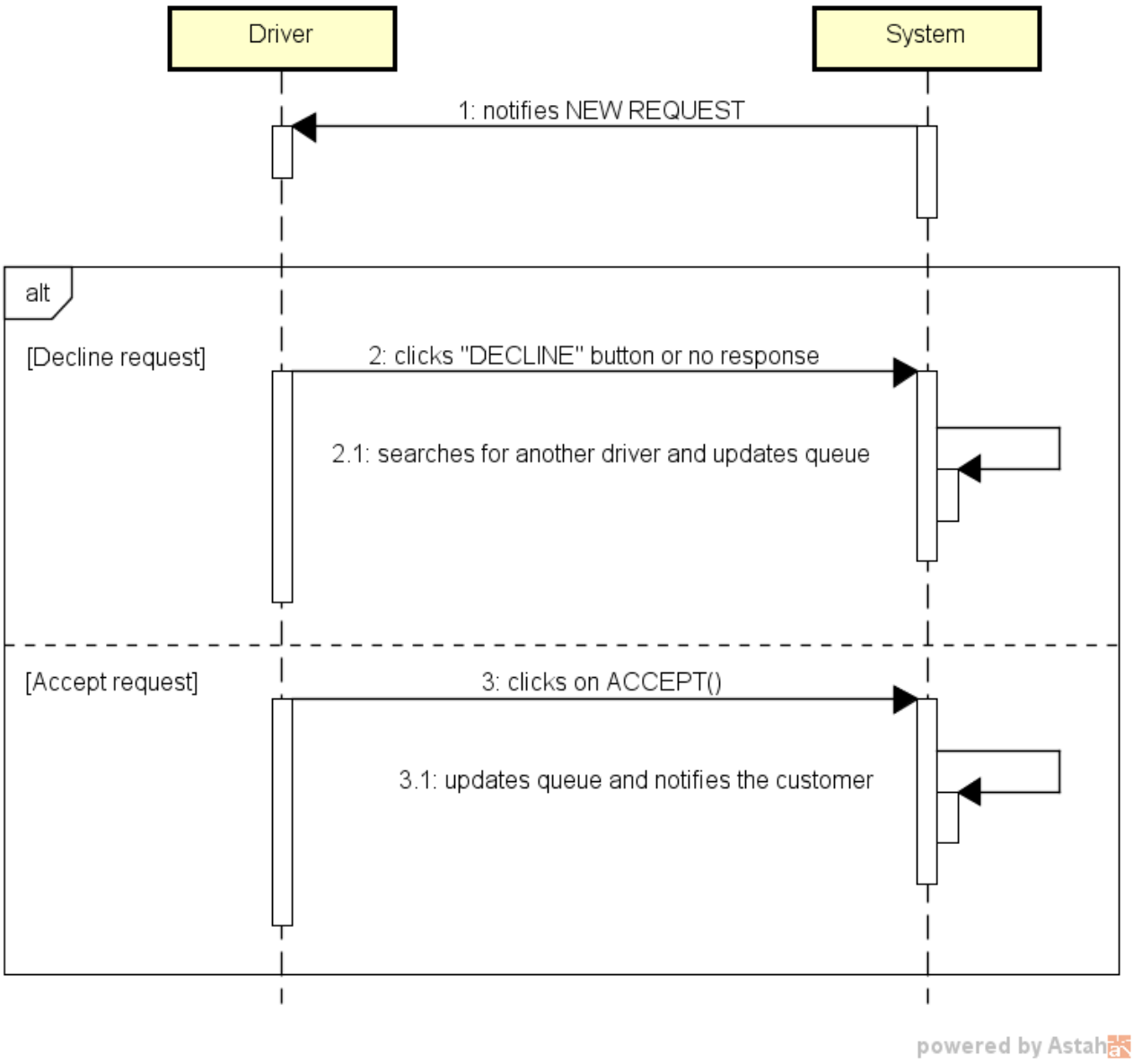
4.4.2 Log In (Customer)



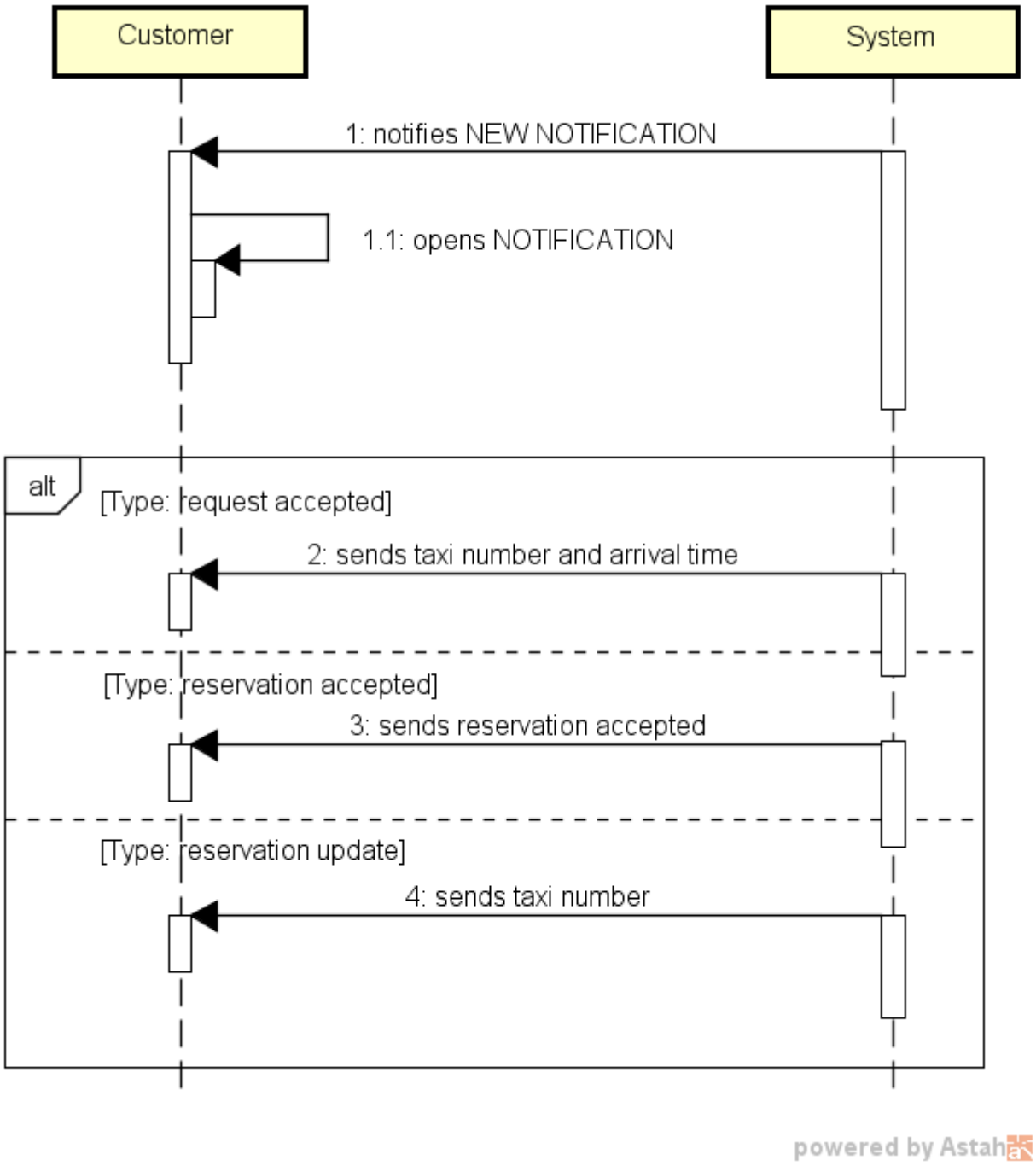
4.4.3 Modify Personal Information



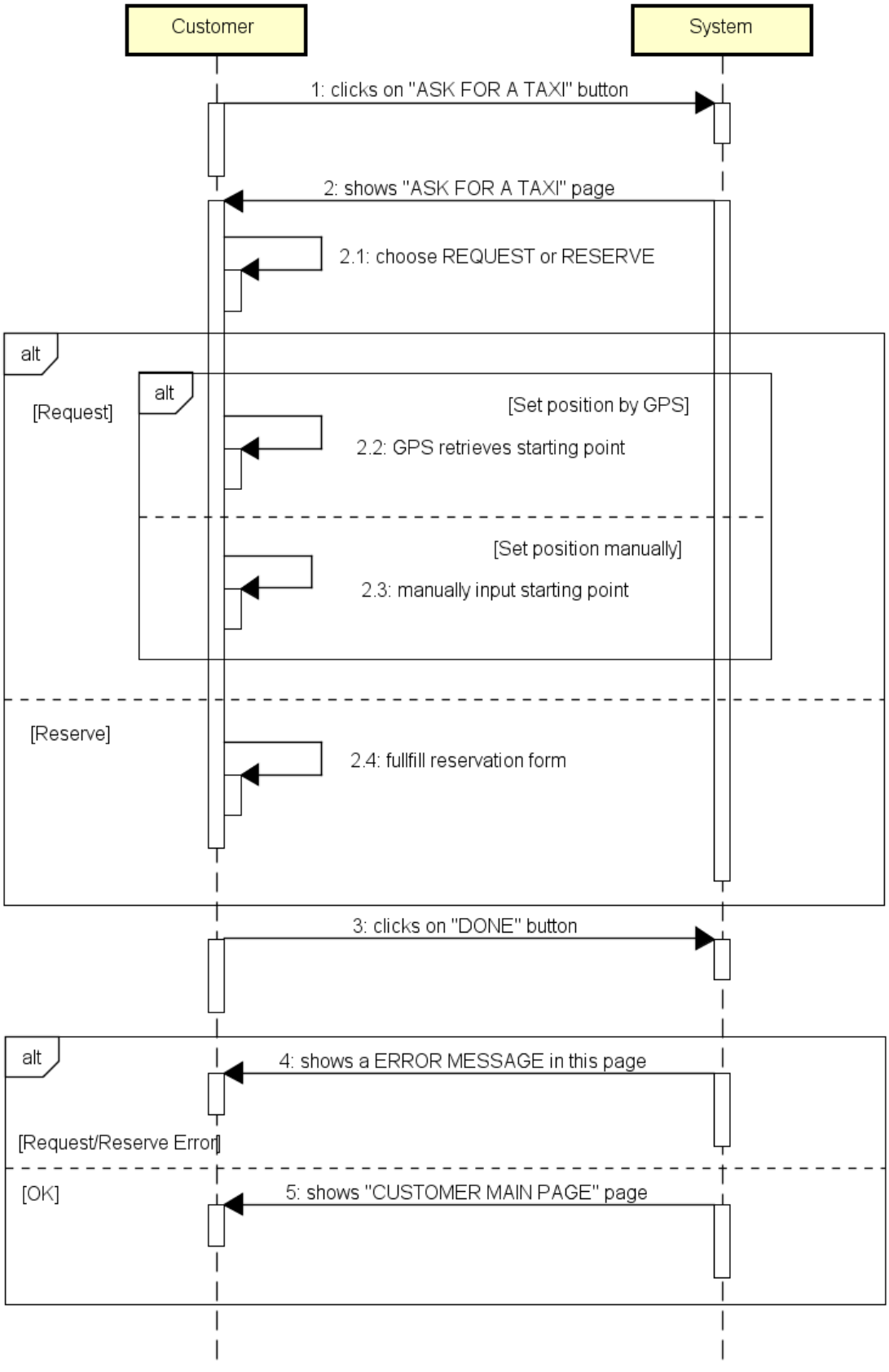
4.4.4 Reply to a Request



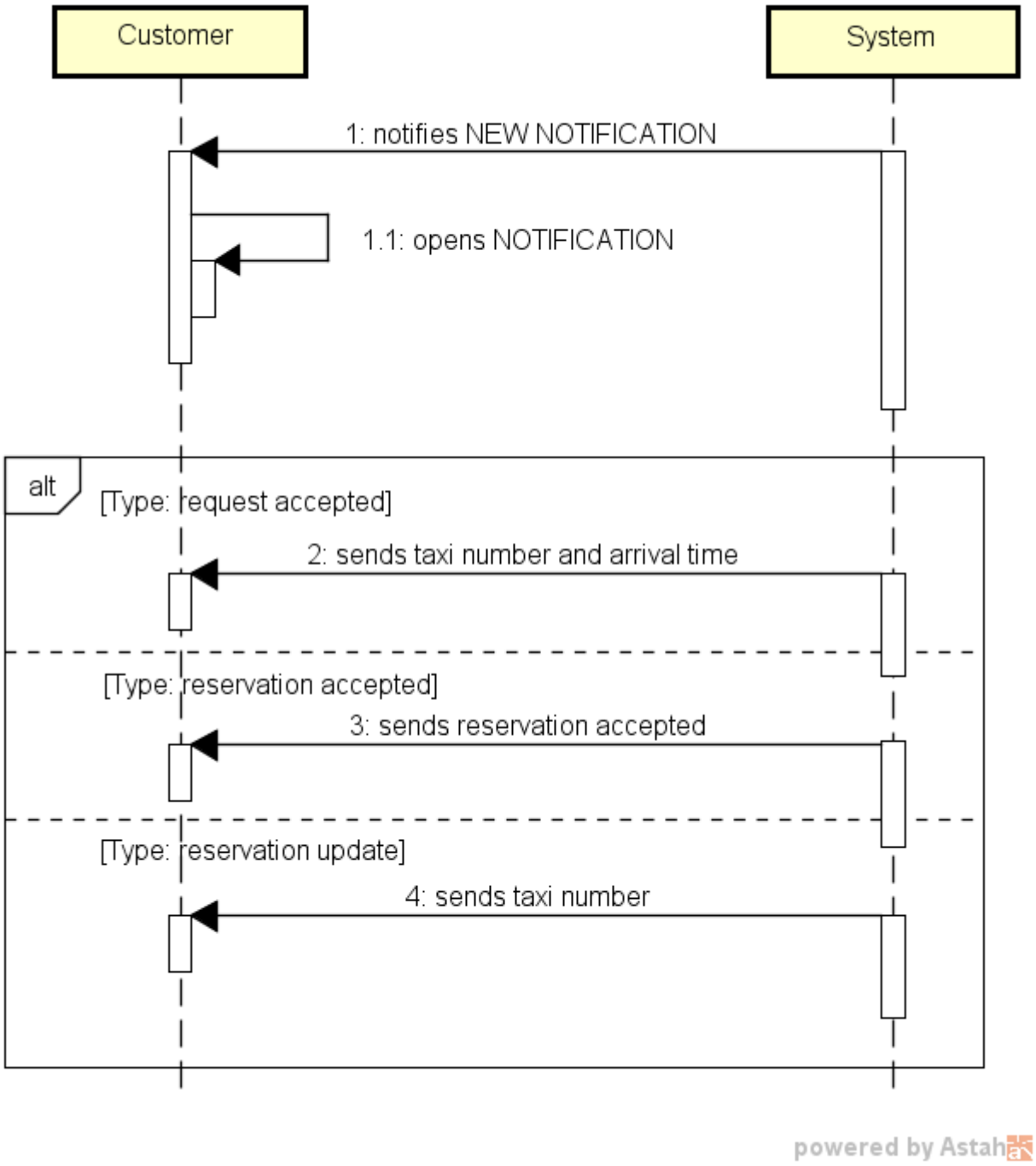
4.4.5 Receive Notification



4.4.6 Ask For A Taxi



4.4.7 Receive Notification



5 Alloy Modelling

5.1 Alloy Code

In this paragraph we report a simple alloy code, used to create a possible scenario of our system.

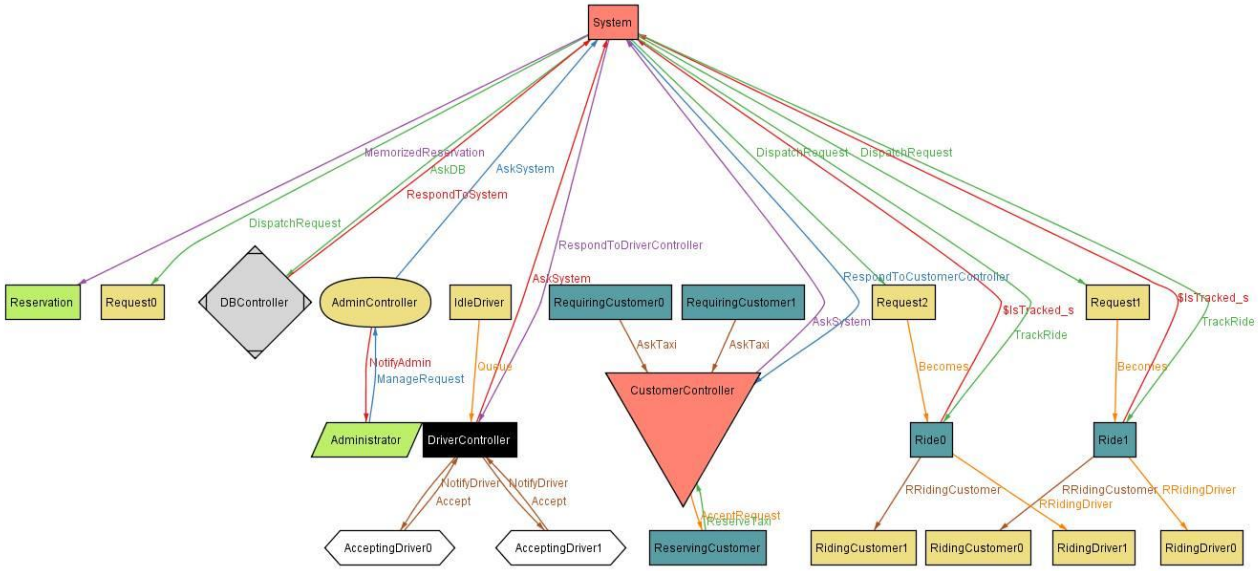
```
//SIGNATURES
abstract sig Customer{}
abstract sig Driver{}
sig ReservingCustomer extends Customer{
        ReserveTaxi: CustomerController
}
sig RequiringCustomer extends Customer{
        AskTaxi:  CustomerController
}
sig IdleDriver{
        Queue: one DriverController,
}
sig AcceptingDriver extends Driver{
        Accept: one DriverController
}
sig Administrator{
        ManageRequest: lone AdminController
}
sig AdminController{
        NotifyAdmin: lone Administrator,
        AskSystem: lone System
}
sig RidingCustomer{}
sig RidingDriver{}
sig CustomerController{
        NotifyCustomer:  lone Customer,
        AcceptRequest: ReservingCustomer,
        AskSystem: lone System
}
sig DriverController{
        NotifyDriver: set AcceptingDriver,
        AskSystem : lone System
}
sig DBController{
        RespondToSystem: lone System
}
sig System{
        RespondToCustomerController: lone CustomerController,
        RespondToDriverController: lone DriverController,
        RespondToAdminController: lone AdminController,
        AskDB: lone DBController,
```

```
        DispatchRequest: set Request,
        MemorizedReservation: set Reservation,
        TrackRide: set Ride
}
sig Ride{
        RRidingCustomer: one RidingCustomer,
        RRidingDriver: one RidingDriver
}
sig Request{
        Becomes: lone Ride
}
sig Reservation{}
//FACTS
fact RequestBecomesOnlyOneRide{
        all disj r,r1:Request | r.Becomes != r1.Becomes
}
fact RideAreUnique{
        all disj r,r1:Ride | r.RRidingCustomer != r1.RRidingCustomer
        all disj r,r1:Ride | r.RRidingDriver != r1.RRidingDriver
}
fact RequiresOrIsNotified{
        all c:ReservingCustomer | IsNotified[c] implies #c.ReserveTaxi=0
        all c:RequiringCustomer | IsNotified[c] implies #c.AskTaxi=0
}
fact allRidingCustomerHaveARide{
        all r:RidingCustomer, disj ri,ri1:Ride | r in (ri.RRidingCustomer)+(ri1.RRidingCustomer)
}
fact allRidingDriverHaveARide{
        all r:RidingDriver, disj ri,ri1:Ride | r in (ri.RRidingDriver)+(ri1.RRidingDriver)
}
fact AllRidesAreTracked{
        all r:Ride | IsTracked[r]
}
fact AllReqAreDispatchedBySystem{
        all r:Request,s:System | r in s.DispatchRequest
}
fact AllReservationAreMemorized{
        all r:Reservation, s:System | r in s.MemorizedReservation
}
fact Singletons{
        #DriverController=1
        #CustomerController=1
        #System=1
        #RequiringCustomer = # AcceptingDriver
        #RidingDriver = #Ride
        #AdminController=1
        #DBController =1
```

```
}
//PREDICATES
pred IsTracked[r:Ride]{
        some s:System | r in s.TrackRide
}
pred IsOnRide[d:RidingDriver]{
        some r:Ride | d in r.RRidingDriver
}
pred IsNotified[c:Customer] {
        some con:CustomerController | c in con.NotifyCustomer
}
assert AllRidingDriverRides{
        #Ride = #RidingDriver
}
check AllRidingDriverRides
//ASSERTIONS
assert AllRidingCustomerRides{
        #Ride = #RidingCustomer
}
check AllRidingCustomerRides
pred show{
        #Administrator =1
        #ReservingCustomer>0
        #RequiringCustomer>0
        #RidingCustomer>0
        #Driver>1
}
run show
```

5.2 General World

Here is a possible general world generated.



6 Tools Used

The tools we used to create the RASD document are:

- *Microsoft Office Word 2007: to redact this document*
- *Astah: to create the Class Diagram, the Sequence Diagrams and the Use Case Diagram;*
- *Alloy Analizer: to prove the consistency of our model.*

For redacting and writing this document we spent 28 hours per person