Requirements Analysis and Specifications Document

ANDREA MAIOLI

The Given Request

- Design an application that is able to manage taxi requests.

- **Given constraints**:
  - Fair management of taxi queues.
  - Application must be accessible either through a web application or a mobile application.
  - The system must inform the passenger with the code of the incoming taxi and the waiting time.
  - Taxi driver can accept requests via mobile application.
  - Division of the city in zones, and for each zone one queue.
  - The system must provide an API to enable development of additional services.

Actual System: my suppositions

- Call Center Infrastructure available:
  - Each call center operator has a computer, a radio device and a telephone.
  - One server stores GPS informations of taxies (GPS information cames from a radio-antenna).
  - Each taxi vehicle is equipped with a radio device and a taximeter.

- The choice of the vehicle that will respond to a given request is made manually by the operator, and the location is provided to the driver using the radio device.

- More efficiency needed! There is enough space for improvements!

Goals of the new system

1. Simplify the access of passengers to the taxi service, allowing them to:
    - Register an account using the mobile application or the website.
    - Log into the system.
    - Log out from the system.
    - Request a taxi trip using the web application, the mobile application or the telephone.
    - Check the status of the current request (only if made using the application).

2. Guarantee a fair management of the taxi queue.

Goals of the new system

3. Simplify the communications between the driver and the call center infrastructure, allowing the call center operator to:
   - Log into the system with a special level of clearance.
   - Log out from the system at the end of the working day.
   - Insert a request coming from a call in the system, which will manage it.

4. Simplify the methods of access to the requests for the taxi drivers, allowing them to:
   - Log into the system with a special level of clearance (different from the call center operator)
   - Set his/her status in available or busy.
   - Log out from the system at the end of the working day.
   - Get on his/her tablet a request if his/her status is available.
   - Accept or decline an incoming request.
   - Report an exceptional event that will be manage from the system

Goals of the new system

5. Allow the system to:
   - Handle requests and their assignment.
   - Manage taxi queue.
   - Handle exceptional events.

Actors: who will use the new system?

- Guest: a non register user.
- Passenger: registered user who will use the system for requesting a ride.
- Call Center Operator: special registered user who will handle the request coming from the telephone and instert them into the system.
- Taxi Driver: special registered user who will bring to completion the requests coming from passengers.

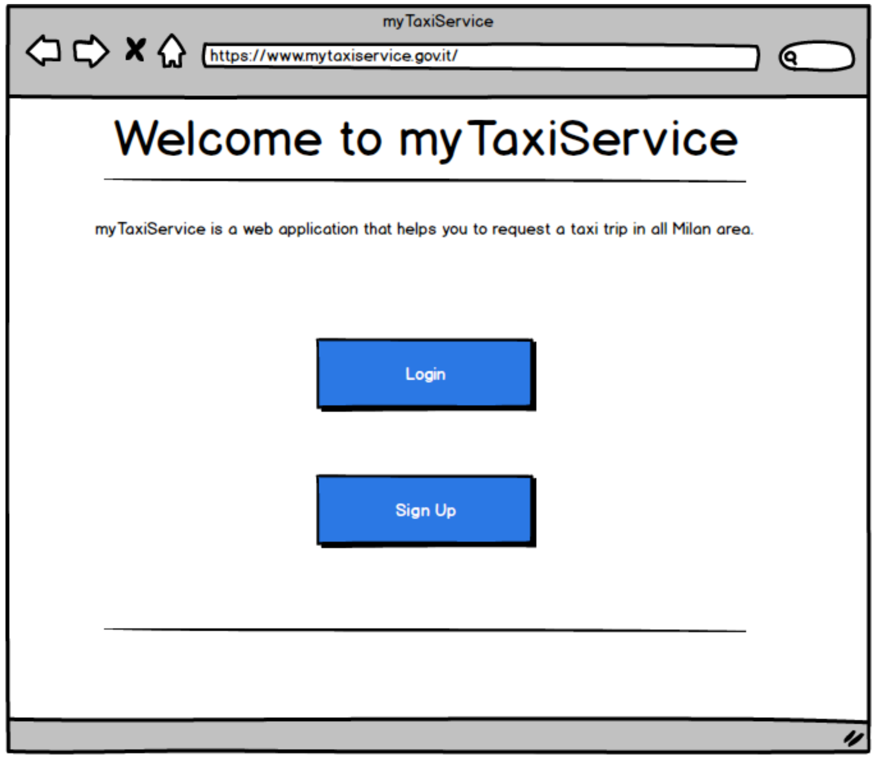
Assumptions

- City: Milan (3.2 Millions of people).
- Number of drivers: 5.000 (16 every 10.000 people).
- Each Taxi Driver can manage up to 4 request per hour (average time per request: 15 minutes).
- In the maximum service congestion, there are about 110 request per minute.
- The number of requests never saturare the available taxies.
- At the start of the new service, one fourth of the request will came from a call.
- There are 50 Call Center Operator and the average time for each call is 2 minutes.

Assumptions

- All the drivers and call center operator are directly hired from the government and are considered public employees (No private taxi company is considered for simplification).
- Each taxi can be assigned to only one queue.
- If a driver is not assigned to a request, he/she can pick up a passenger without a request made from the system.
- If a request arrives for an area with an empty queue, the request can be forwarded to a taxi in the nearest area’s queue available.
- The call center will not be discontinued.

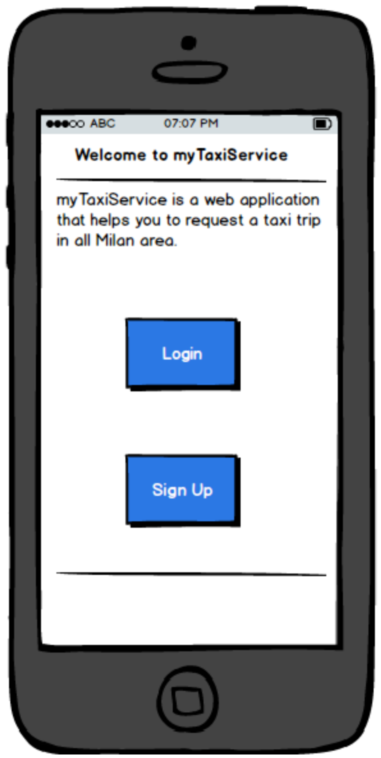
User Interface – Home Page


myTaxiService

https://www.mytaxiservice.gov.it/

Welcome to myTaxiService

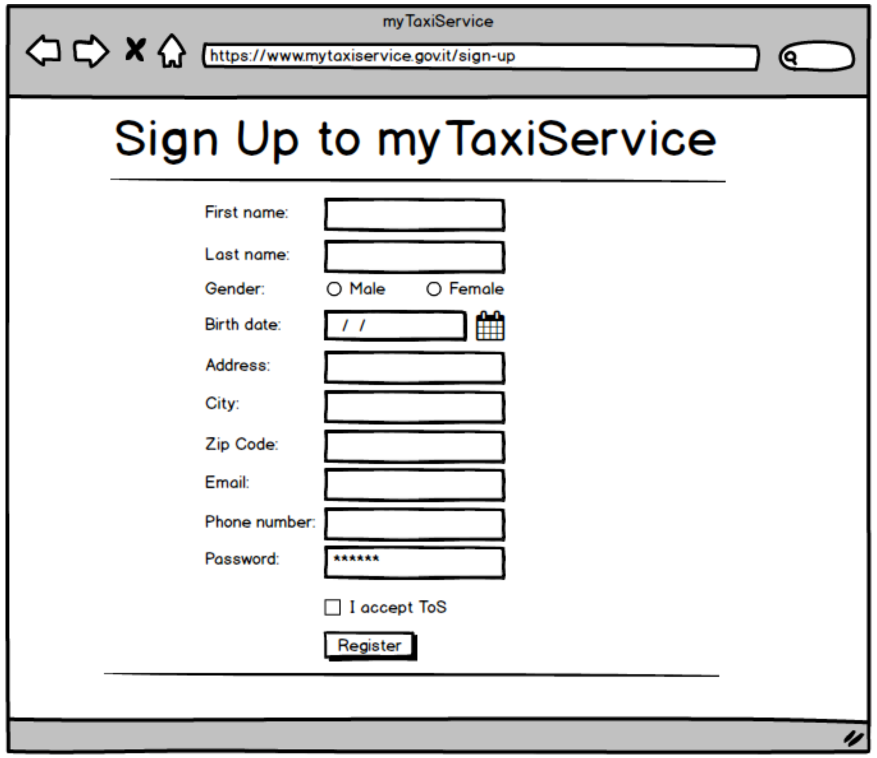
myTaxiService is a web application that helps you to request a taxi trip in all Milan area.

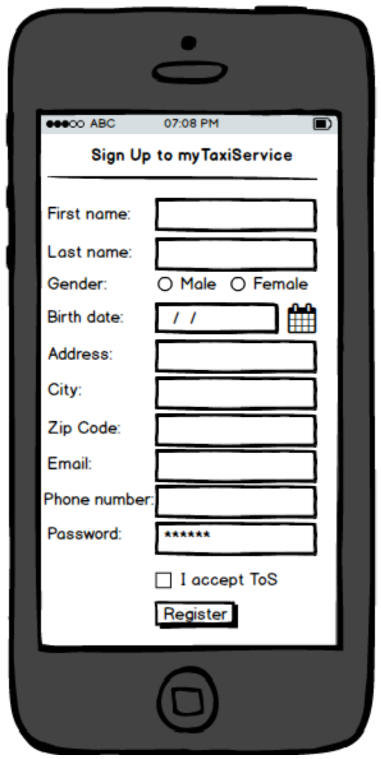


Welcome to myTaxiService

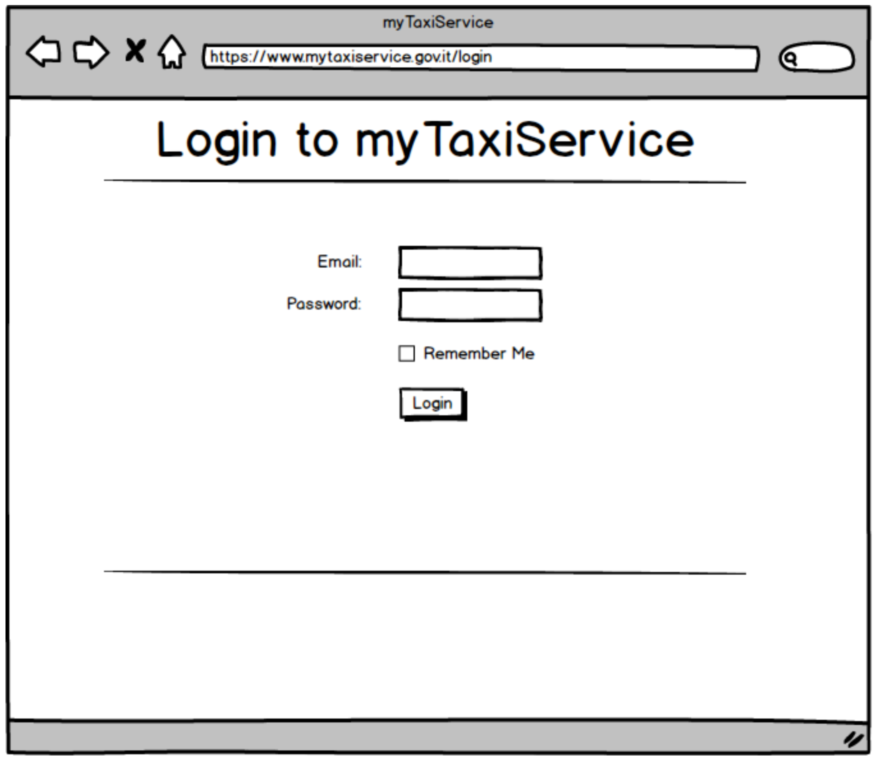
myTaxiService is a web application that helps you to request a taxi trip in all Milan area.

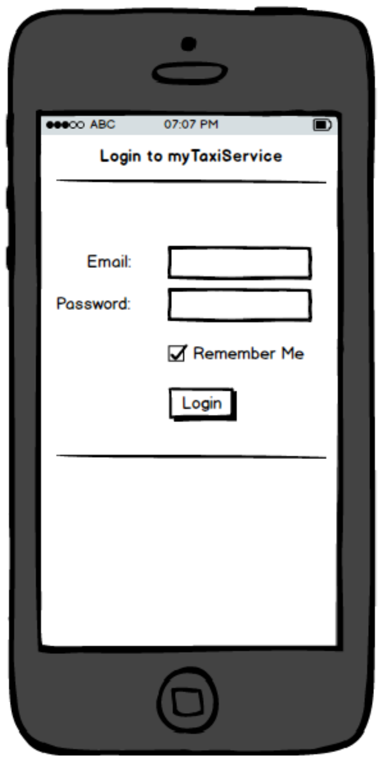
User Interface – Sign Up Page



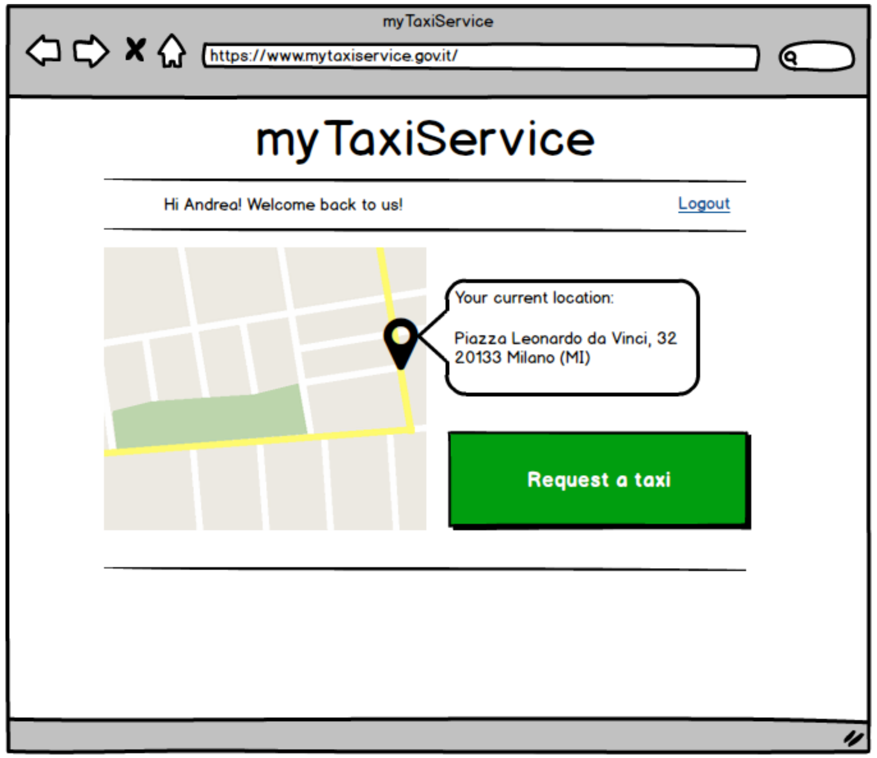


User Interface – Login Page





User Interface – Passenger’s Home Page


myTaxiService

https://www.mytaxiservice.gov.it/

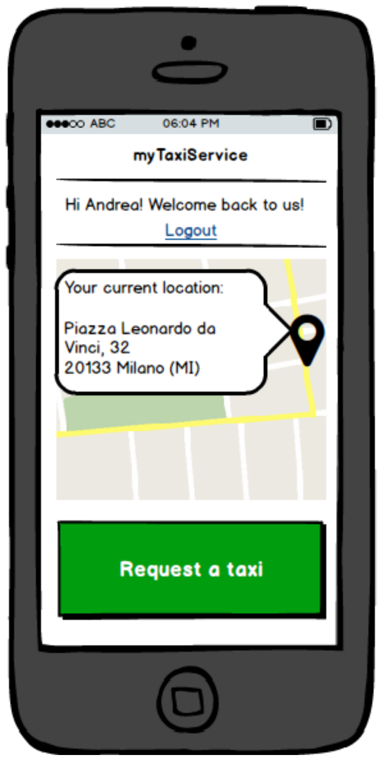
myTaxiService

Hi Andrea! Welcome back to us! Logout

Your current location:

Piazza Leonardo da Vinci, 32
20133 Milano (MI)

Request a taxi


●●●○○ ABC 06:04 PM

myTaxiService

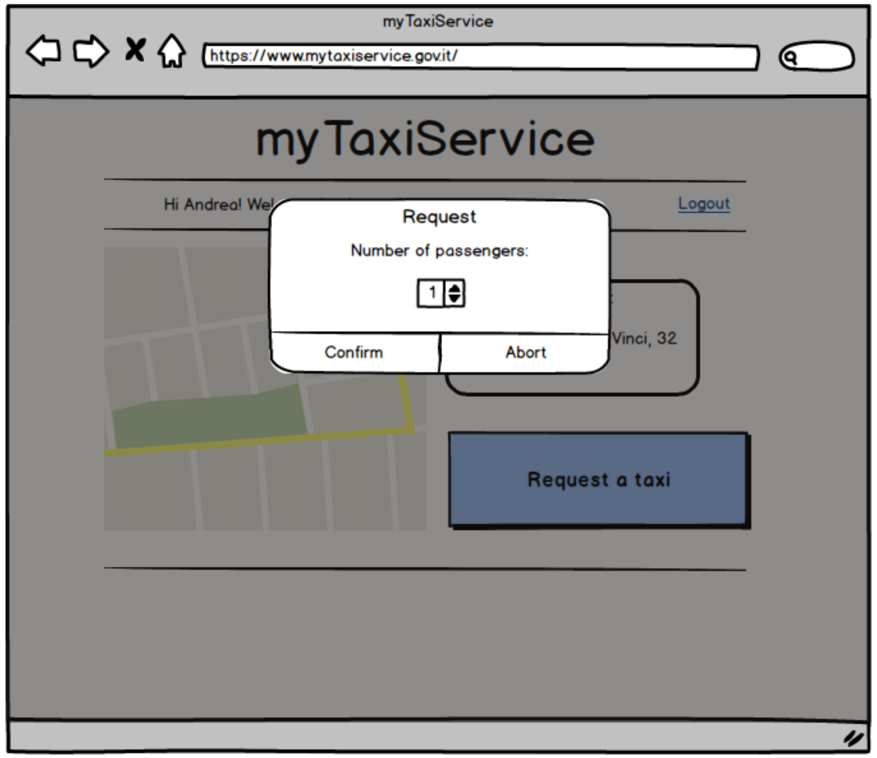
Hi Andrea! Welcome back to us!
Logout

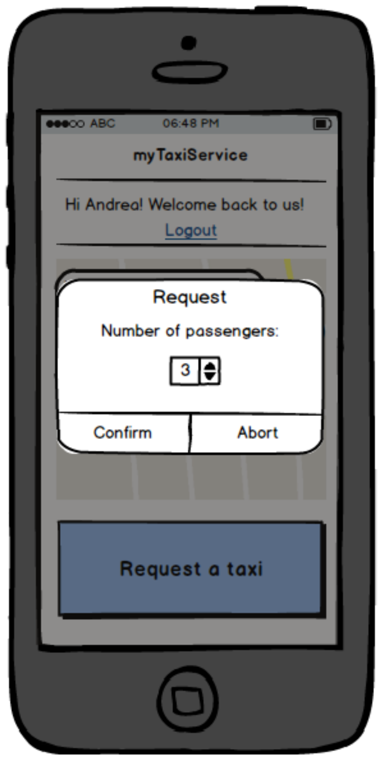
Your current location:

Piazza Leonardo da
Vinci, 32
20133 Milano (MI)

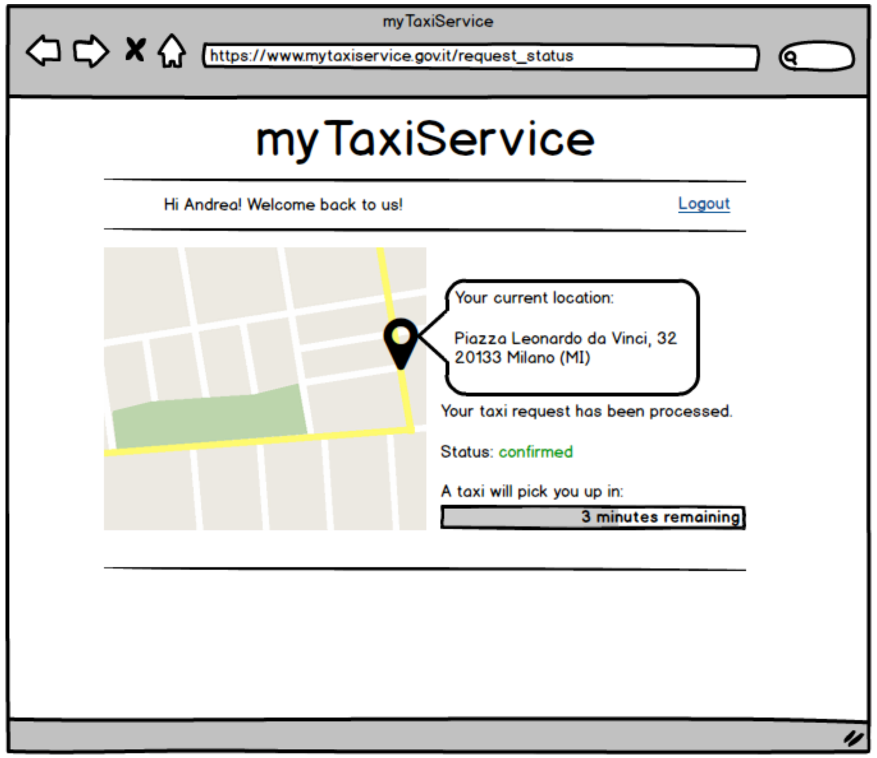
Request a taxi

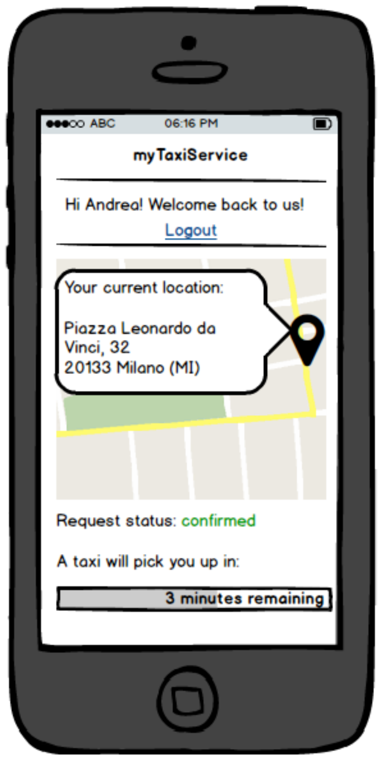
User Interface – Send Request



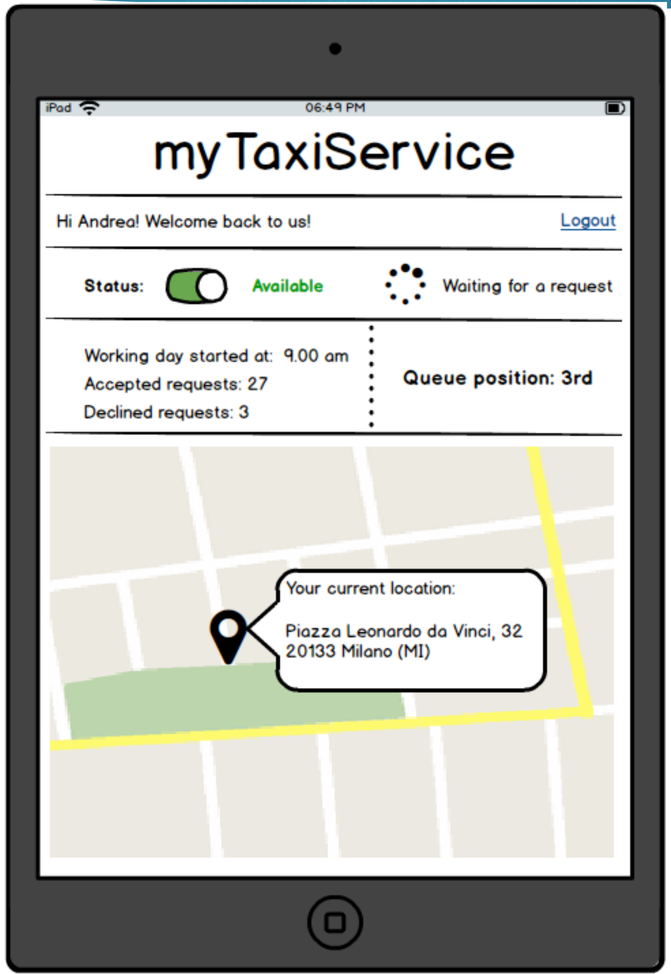


User Interface – Request Status Page


myTaxiService
https://www.mytaxiservice.gov.it/request_status
myTaxiService
Hi Andrea! Welcome back to us!
Logout
Your current location:
Piazza Leonardo da Vinci, 32
20133 Milano (MI)
Your taxi request has been processed.
Status: confirmed
A taxi will pick you up in:
3 minutes remaining


ABC
06:16 PM
myTaxiService
Hi Andrea! Welcome back to us!
Logout
Your current location:
Piazza Leonardo da Vinci, 32
20133 Milano (MI)
Request status: confirmed
A taxi will pick you up in:
3 minutes remaining

User Interface – Driver’s Home Page



Hi Andrea! Welcome back to us!

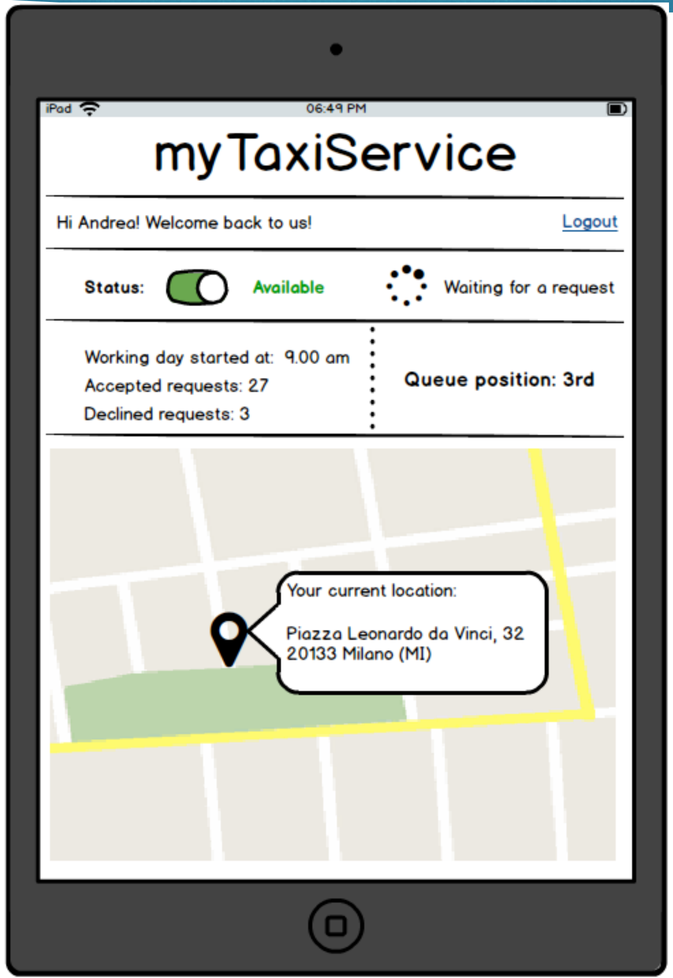
Working day started at: 9.00 am
Accepted requests: 27
Declined requests: 3

Queue position: 3rd

Your current location:

Piazza Leonardo da Vinci, 32
20133 Milano (MI)

User Interface – Incoming Request



Hi Andrea! Welcome back to us!

Waiting for a request

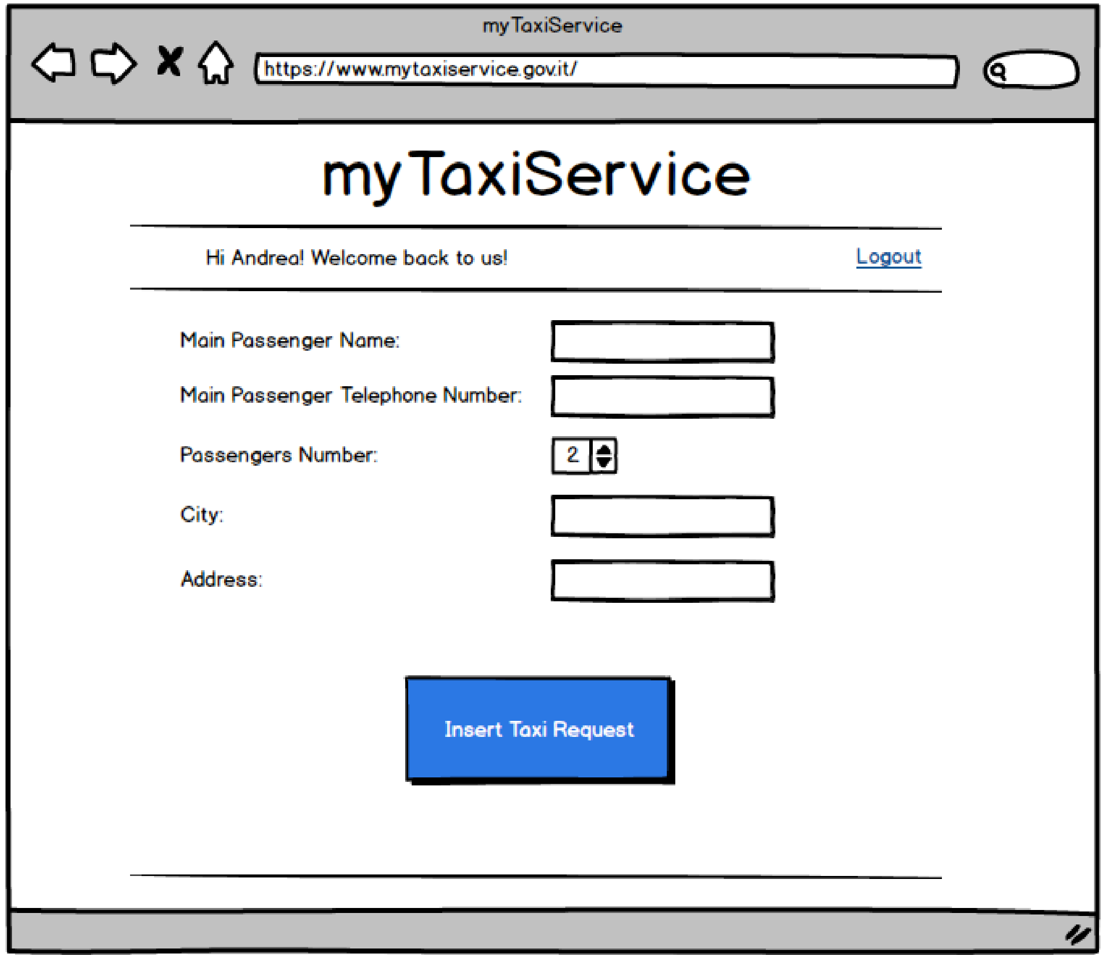
Working day started at: 9.00 am
Accepted requests: 27
Declined requests: 3

Queue position: 3rd

Your current location:

Piazza Leonardo da Vinci, 32
20133 Milano (MI)

User Interface – Operator’s Home Page



myTaxiService

https://www.mytaxiservice.gov.it/

myTaxiService

Hi Andrea! Welcome back to us! Logout

Main Passenger Name:	
Main Passenger Telephone Number:	
Passengers Number:	2
City:	
Address:	

Software Interfaces

- Database Management System:
  - MySQL Server 5.7
- Application Server and Web Server:
  - Glassfish 4.1.1
- Operative System:
  - Each operative system that can run MySql and Glassfish without any compatibility issues.

Communication Interfaces

- The Web Server needs:
  - TCP port number 80 for HTTP
  - TCP port number 443 for HTTPS

- The DBMS needs:
  - TCP port number 3306 for serving the connection with the database

Functional Requirements

- Guest can:
  - Sign Up

- Passenger can:
  - Login both on web application and mobile application
  - Modify his/her profile information
  - Request a taxi
  - Get the status of an active request
  - Get the ETA of an active request
  - Logout from the application

Functional Requirements

- Driver can:
  - Login on the mobile application
  - Set his/her status as Available or Busy
  - See his/her queue position
  - See his/her workday statistics
  - Receive taxi request
  - Accept or Decline a taxi request
  - Report if a passenger is found or not, after getting to the pick-up point
  - Report an exceptional event which prevent him/her to get to the pick-up point
  - Get the estimated arrival time to the pick-up point
  - Get indications on how to get to the pick-up point
  - Logout from the application

Functional Requirements

- Call Center Operator can:
  - Login on the web application
  - Receive calls for taxi requests
  - Insert inside the system a taxi request
  - Get the status of the inserted request
  - Get the estimated ETA of the inserted request
  - Communicate request information to the client through the call
  - Logout from the application

Functional Requirements

- **The system** is not considered as an actor, but it must be able to:
  - Assign a request for an area to the first taxi in the queue of this area.
  - Organize drivers in queue.
  - Assign exactly one queue for each area.
  - Automatically forward a request to the second driver in the queue if the first driver decline it.
  - Automatically forward a request to the second driver in the queue if the first driver doesn’t accept or decline it within 30 seconds.
  - Assign a new driver to an accepted request if the assigned driver report an exceptional events that prevent him/her to get to the passenger.
  - If no driver is available in a queue, the system must be able to find the driver that will arrive to a fixed pick-up point in the less possible time. The driver can be one from another queue, or one that will finish a ride in the request’s area.

Performance Requirements

- Each passenger’s request must be confirmed in no more than 2 minutes, and his/her taxi must arrive in no more than 10 minutes, if any available.
- Each request must be accepted or declined from a driver in less than 30 seconds. After this period, the request must be automatically declined and assigned to the next taxi in the queue, and the driver must be moved in the last position of the queue.
- The system must be able to compute the response to each request in no more than 500ms.
- The system must be able to serve simultaneously 1600 taxi drivers, 50 call center operators and 110 passengers with a response time for each action lower than 3 seconds.
- Given the analysis made in the assumptions, the total number of taxi requests per minutes can be up to 110. The application must be able to handle 200 taxi requests per minutes.

Software System Attributes

- **Availability**: The application must be accessible 24h per day and 7 days per week and must be always possible to use the service. All the software updates to the main infrastructure must be applied during the night, when the requests are minimum, and the call center must be available.
- **Security**:
  - User's credential must be stored using a hashing function.
  - Users must be able to login only by entering the correct credentials.
  - User's password must be at least 8 characters long and a composition of letters and numbers.
  - The application server must be the only machine that can communicate with the DBMS.
  - All the user inputs must be filtered in order to prevent SQL-Injection, Cross Site Scripting and other type of attacks.
  - HTTPS must be enable on the web server and all the connection coming from the HTTP protocol must be redirected to the HTTPS one.

Software System Attributes

- **Maintainability**:
  - Each function and class of the application must be well commented in order to allow future developers to understand and modify the code.
  - Each update or modification to the application or the infrastructure must be traceable and well documented.
  - Each solution to every future problem must be well documented, in order to know how it has solved.
  - Each method provided from the API will be documented and some example will be provided.

Scenarios

Taxi request using mobile application.

- Andrea needs a taxi to return his home, so he open the myTaxiService application on his smartphone and login. After that, he see his current location and click the “Request a taxi” button. The application ask him to insert the number of passengers for this ride, and since he is with her girlfriend, he select two and confirm the request. Now the application shows him that the request is in the processing phase and after one minutes he sees that the request status is “confirmed” and a counter says that a taxi will pick they up in no more than 5 minutes. The taxi arrives in only 3 minutes and brings them home.

Scenarios

No taxi available in a queue.

- Giovanni want to go home, so he requests a taxi using the myTaxiService mobile application. Unfortunately in this area there is no taxi available, so the system search for all the busy taxies that will end the ride in the request's area and compute the estimated time of the end of the request, considering the taxi of John, that has the less timing of arrival. The system also compute the ETA for the arrival of the taxi that is in the first position of the nearest area queue and compare with the ETA of John. John will end the request in less than one minutes, so the system forward the request to him.

Use Case

Name	Taxi Request Using Application
Actors	Passenger, Driver
Entry Conditions	The passenger is logged into the system and want to request a taxi for a ride
Flow of Events	1. The user clicks on the “Request a taxi” button. 2. The user enters the number of passengers for this ride. 3. The application sends to the system the current location of the user. 4. The user send the request for a ride. 5. The system assign the request to the first driver in the area’s queue and move him/her to the last position of the queue. 6. If the driver doesn’t accept the request, restart from 4 until the request is confirmed from a driver. 7. The status of the driver who accepted the request is removed from the queue.
Exit Conditions	A taxi is assigned to the passenger and the application shows the ETA of the taxi arrival.
Possible Exceptions	• None

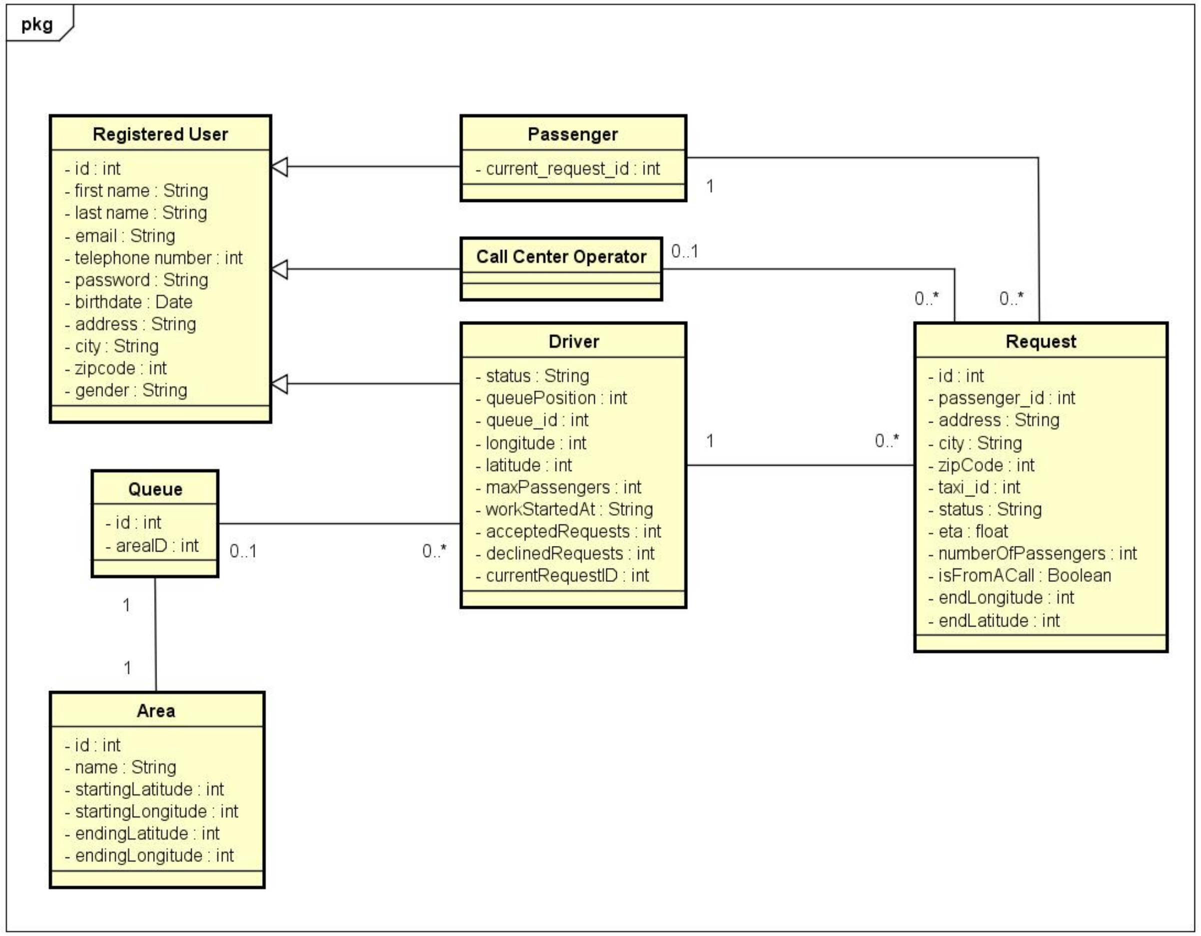
Use Case

Name	Accepted incoming taxi request
Actors	Driver
Entry Conditions	The driver has accepted an incoming request.
Flow of Events	1. The system remove the driver from the queue. 2. The application display to the driver the location of the passenger. 3. The driver start the ride to the pick-up point.
Exit Conditions	a. The driver arrives to the pick-up point. OR a. The driver has an inconvenient and click on the “Report Exceptional Event” button.
Possible Exceptions	• None

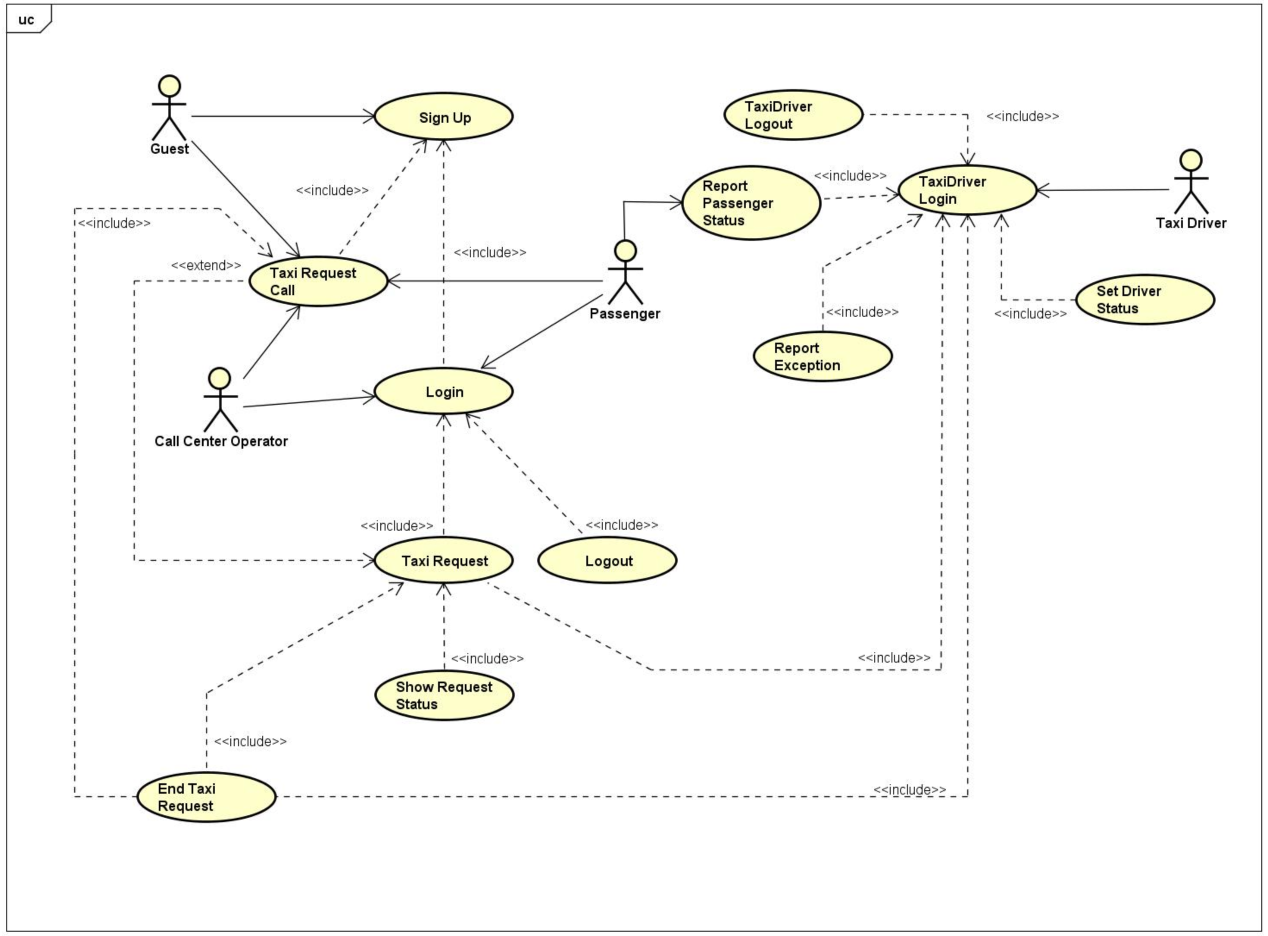
Use Case

Name	Passenger not found for a request
Actors	Driver, Passenger
Entry Conditions	The driver has arrived at the pick-up point and the passenger is not present.
Flow of Events	1. The driver click on “Report passengers not found” button. 2. The system sends a text message to the passenger. 3. The driver wait 5 minutes for the passenger.
Exit Conditions	a. The passenger is found and the new situation is described from the Use Case “Passenger Found”. OR a. The passenger is not found, the request is deleted from the system and the driver is inserted in the last position of the queue.
Possible Exceptions	• None

Class Diagram

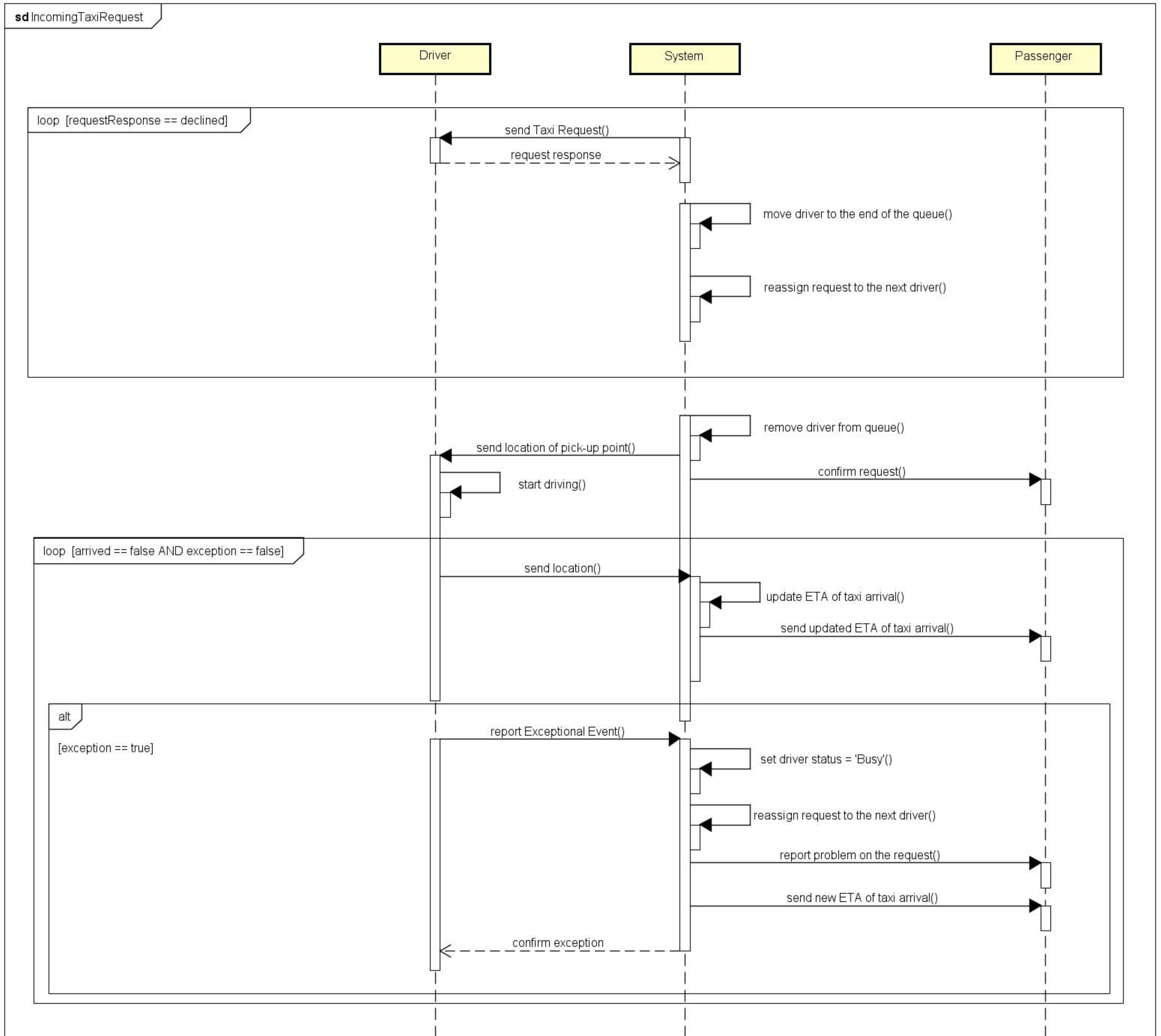


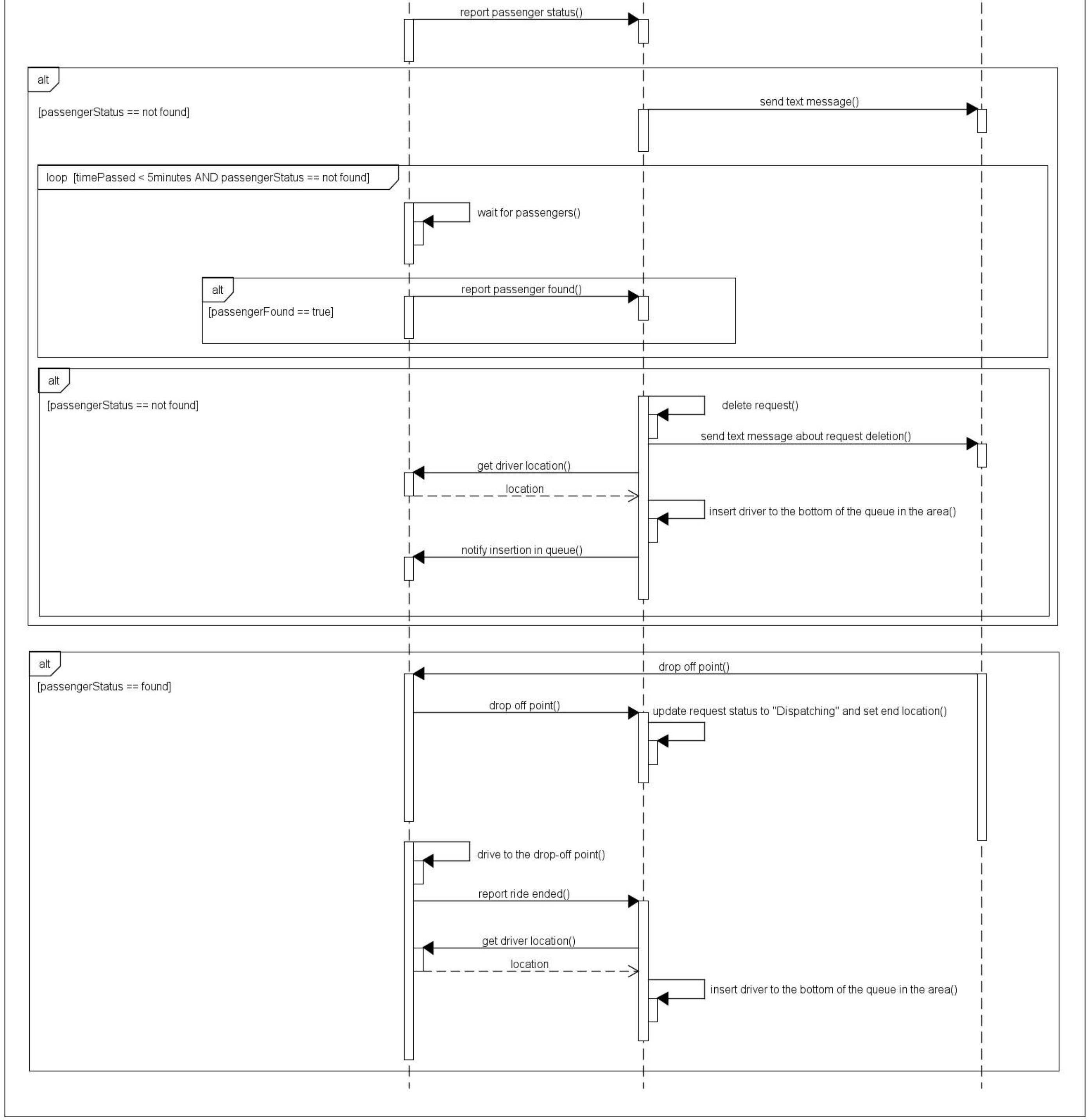
UML


uc
Guest
Sign Up
<<include>>
<<include>>
<<extend>>
Taxi Request Call
<<include>>
Passenger
Call Center Operator
Login
<<include>>
<<include>>
Taxi Request
Logout
Show Request Status
<<include>>
<<include>>
End Taxi Request
<<include>>
<<include>>
TaxiDriver Logout
<<include>>
Report Passenger Status
<<include>>
TaxiDriver Login
Taxi Driver
Report Exception
<<include>>
Set Driver Status
<<include>>

Sequence Diagram

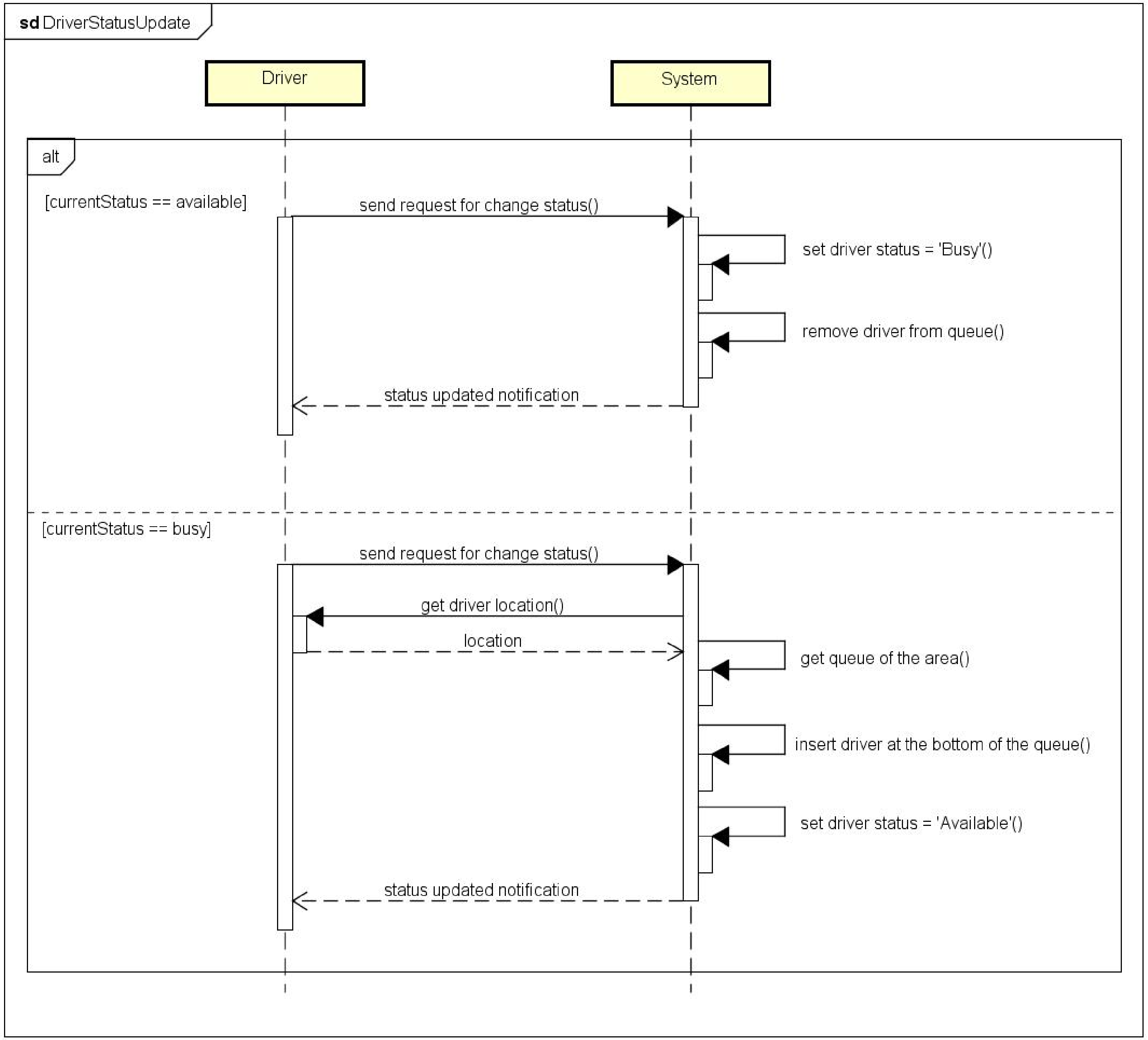
INCOMING TAXI REQUEST


sd IncomingTaxiRequest
Driver
System
Passenger
loop [requestResponse == declined]
send Taxi Request()
request response
move driver to the end of the queue()
reassign request to the next driver()
remove driver from queue()
send location of pick-up point()
confirm request()
start driving()
loop [arrived == false AND exception == false]
send location()
update ETA of taxi arrival()
send updated ETA of taxi arrival()
alt
[exception == true]
report Exceptional Event()
set driver status = 'Busy'()
reassign request to the next driver()
report problem on the request()
send new ETA of taxi arrival()
confirm exception

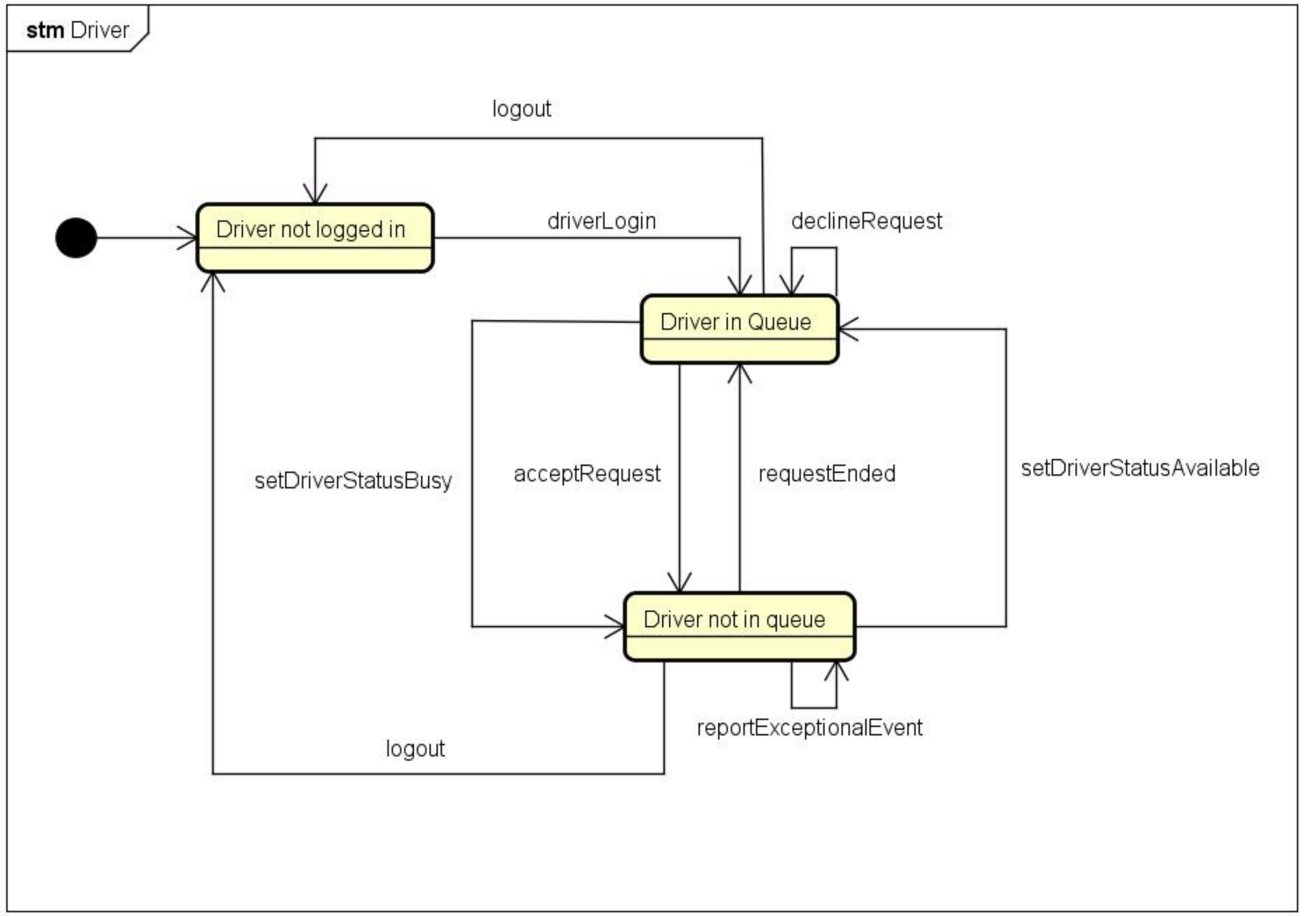

report passenger status()
alt
[passengerStatus == not found]
send text message()
loop [timePassed < 5minutes AND passengerStatus == not found]
wait for passengers()
alt
[passengerFound == true]
report passenger found()
alt
[passengerStatus == not found]
delete request()
send text message about request deletion()
get driver location()
location
insert driver to the bottom of the queue in the area()
notify insertion in queue()
alt
[passengerStatus == found]
drop off point()
drop off point()
update request status to "Dispatching" and set end location()
drive to the drop-off point()
report ride ended()
get driver location()
location
insert driver to the bottom of the queue in the area()

Sequence Diagram

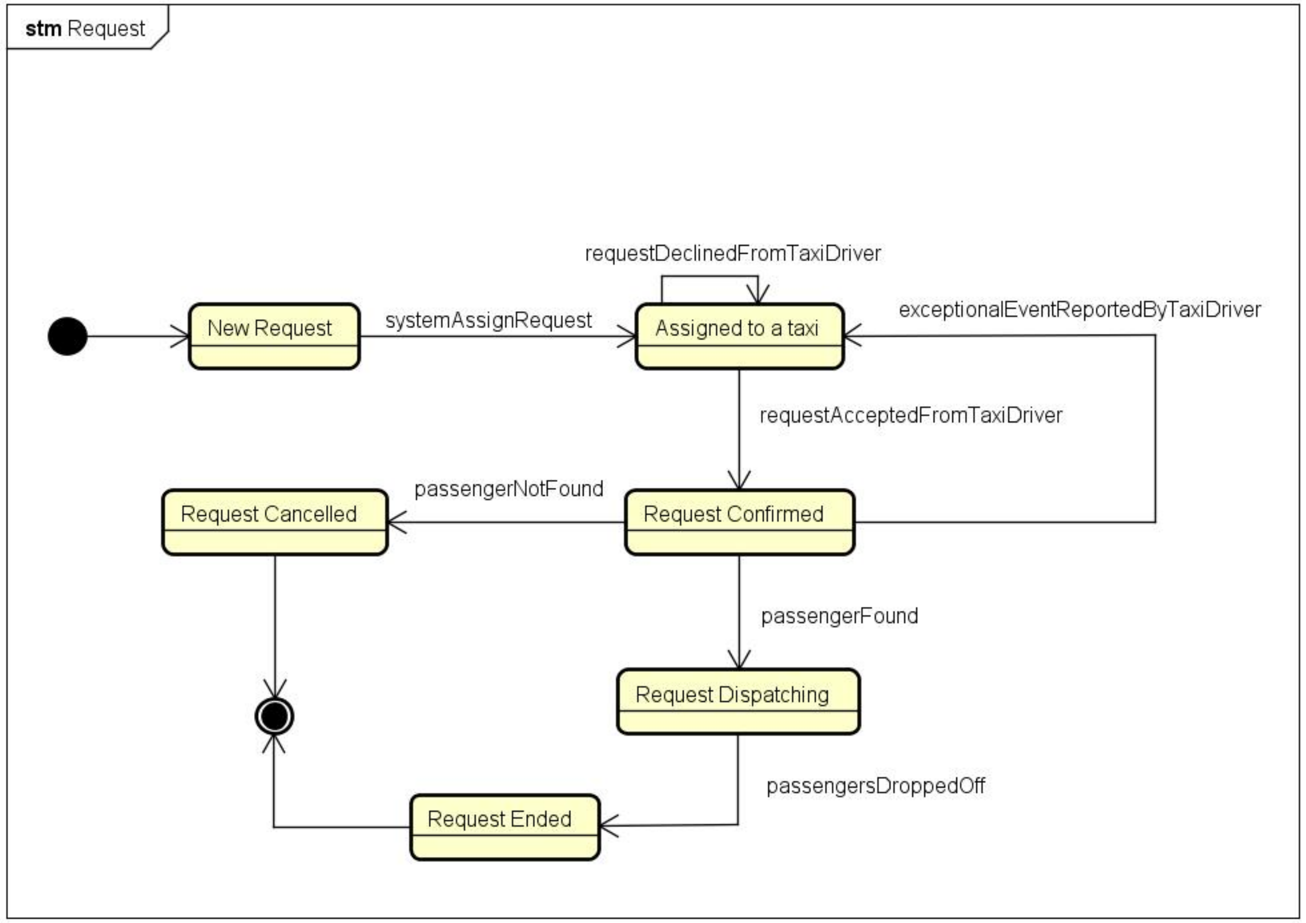
DRIVER STATUS UPDATE


sd DriverStatusUpdate
Driver
System
alt
[currentStatus == available]
send request for change status()
set driver status = 'Busy'()
remove driver from queue()
status updated notification
[currentStatus == busy]
send request for change status()
get driver location()
location
get queue of the area()
insert driver at the bottom of the queue()
set driver status = 'Available'()
status updated notification

State Chart Diagram - Driver

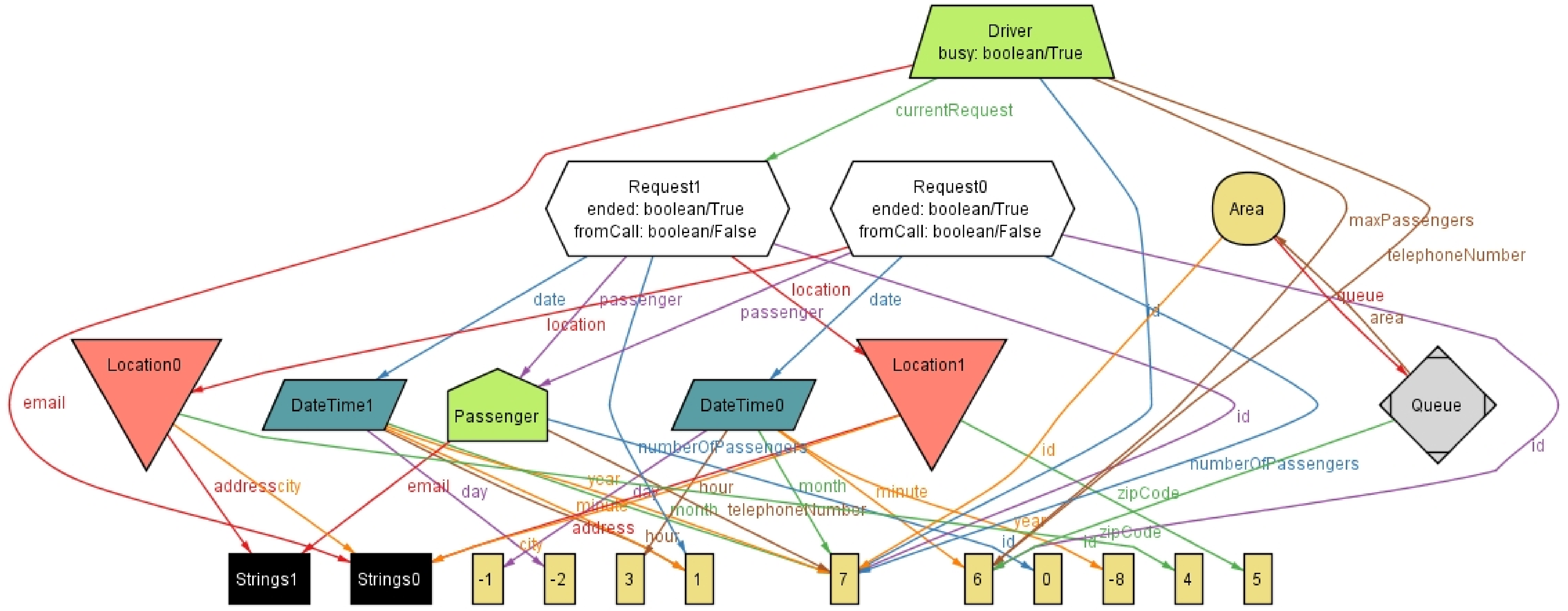


State Chart Diagram - Request

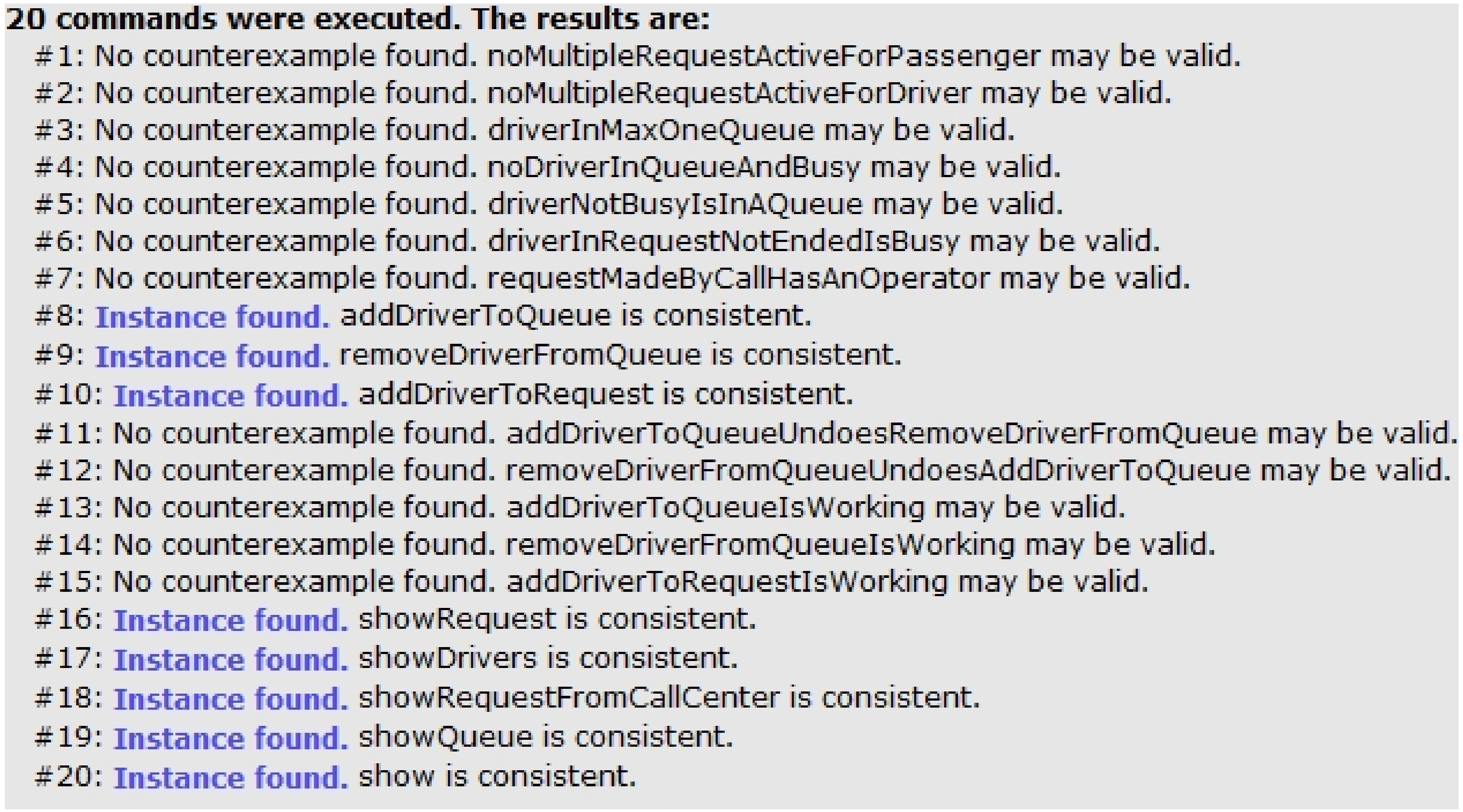


Alloy – Generated World

address: 2
area: 1
city: 2
currentRequest: 1
date: 2
day: 2
email: 2
hour: 2
id: 1
id: 1
id: 2
id: 2
location: 2
maxPassengers: 1
minute: 2
month: 2
numberOfPassengers: 2
passenger: 2
queue: 1
telephoneNumber: 2
year: 2
zipCode: 2



Alloy – Result of Analysis



```
20 commands were executed. The results are:
  #1: No counterexample found. noMultipleRequestActiveForPassenger may be valid.
  #2: No counterexample found. noMultipleRequestActiveForDriver may be valid.
  #3: No counterexample found. driverInMaxOneQueue may be valid.
  #4: No counterexample found. noDriverInQueueAndBusy may be valid.
  #5: No counterexample found. driverNotBusyIsInAQueue may be valid.
  #6: No counterexample found. driverInRequestNotEndedIsBusy may be valid.
  #7: No counterexample found. requestMadeByCallHasAnOperator may be valid.
  #8: Instance found. addDriverToQueue is consistent.
  #9: Instance found. removeDriverFromQueue is consistent.
  #10: Instance found. addDriverToRequest is consistent.
  #11: No counterexample found. addDriverToQueueUndoesRemoveDriverFromQueue may be valid.
  #12: No counterexample found. removeDriverFromQueueUndoesAddDriverToQueue may be valid.
  #13: No counterexample found. addDriverToQueueIsWorking may be valid.
  #14: No counterexample found. removeDriverFromQueueIsWorking may be valid.
  #15: No counterexample found. addDriverToRequestIsWorking may be valid.
  #16: Instance found. showRequest is consistent.
  #17: Instance found. showDrivers is consistent.
  #18: Instance found. showRequestFromCallCenter is consistent.
  #19: Instance found. showQueue is consistent.
  #20: Instance found. show is consistent.
```