SE2 Requirement Analysis and Specification Document

Edoardo Giacomello Mattia Fontana

November 6, 2015

Contents

1 Introduction

1.1 Purpose

This document represents the Requirement Analysis and Specification Document (RASD). The main goal of this document is to completely describe the system in terms of functional and non-functional requirements, analyse the real need of the customer to modelling the system, show the constraints and the limit of the software and simulate the typical use cases that will occur after the development. This document is intended to all developer and programmer who have to implement the requirements, to system analyst who want to integrate other system with this one, and could be used as a contractual basis between the customer and the developer.

1.2 Scope

The scope of MyTaxiService is to manage a taxi service in a more efficient way, along with an emprovement in the interaction between the customers and the service providers.
In particular, the following goals have be highlighted by the stakeholders:

1.2.1 Goals

1. Providing the passenger a ***simplified access*** for making real-time **requests** for the taxi service, according to appropriate ***usability metrics***.
2. Providing the passenger the possibility to **reserve** a taxi for a later moment.
3. Providing the passenger a clear confirmation of the taxi ride hes requesting for.
4. Providing the passenger an estimation of the **waiting time** for a taxi ride.
5. Providing the taxi drivers a ***simplified access*** for receiving and handling transportation requests from the passengers.
6. Guaranteeing a fair management of the taxi queue, in term of minimizing the passenger ***waiting time***.
7. Providing ***system developers*** a programmatic interface to further extend the system.

1.2.2 Actors and Stakeholders

Customer The person(s) who requested the developing of the system. Also referred as Government in the assignment text.

Passenger The person who avails of the taxi service, in particular those who make a request for a Taxi ride.

Taxi Driver The employee which is in charge to meet the passenger and take him to the desired location

User General term for describing whoever interacts with the system interfaces. It can be a passenger, a taxi driver, a system developer, a system administrator, etc.

System Developers The persons which are qualified and in charge to extend the present system with additional features or services

System Administrator : The persons in charge to manage the relationship between the transportation service and the system. For example a system administrator could register new taxi drivers profiles, manage the passenger profiles or manage data backups.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Simplified Access Human-to-Machine interaction that satisfy some usability metrics

Usability Metric Precise measurable requirements that an interface has to satisfy in order to provide an efficient and easy experience to the user.

Waiting time Time in minutes between the submission of a request and the arrival of a taxi.

Request ***(when made by the passenger to the taxi service)*** In this document we refer to a *request* when the passenger's intent is to meet the taxi as soon as possible.

Reservation In this document we refer to a *reservation* when the passenger's intent is to meet the taxi at a precise moment in the future

1.3.2 Acronyms

1.3.3 Abbreviations

1.4 Reference Documents

- Specification Document: Assignments 1 and 2 (RASD and DD).pdf.
- Specification Document:Project Description And Rules.pdf
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.

1.5 Document Overview

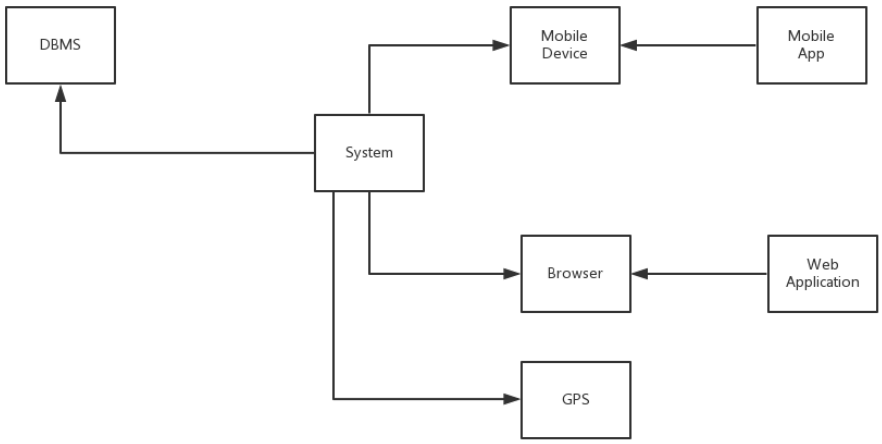
The following part of this document will focus on a deeper analysis of the goals highlighted by the stakeholders in addition to some assumptions that must hold in order to keep the system properties valid. The last part of the document will exhibit the software requirements that resulted from the goals examination along with some scenario and use cases that will offer the stakeholders a better comprehension of the final system.

2 Overall description

This section will cover the overall description of the product. In particular it will put this product into the perspective of other products or systems.

2.1 Product Perspective

The MyTaxiService system will partially replace any former traditional system that is currently based on phone calls with a new interactive system based on mobile and web interfaces.
The decision of completely dismissing the former phone call system in favor to the new one will be left to the customer, since MyTaxiService requires the passengers to access a web interface or a mobile application, which is still generally not accessible by some users.
The project will therefore consists in releasing a web application and a mobile app which are not integrated with any other existing taxi management service. The system is created to improve the possibility to connect with the taxi service for requesting a taxi, that is based on phone calls. With this system, the taxi service can provide a fast and innovative system that allow to manage the user request rapidly, without any loss of time. Both the mobile application and the web interface are connected to the central system and allow the user to know in real time the waiting time, the taxi occupation and their location, and allows to make a taxi request receiving a rapid response. The application will furthermore provide a programmatic interface for the integration with future systems and extensions.



2.1.1 System Interfaces

1. MyTaxiService mobile application will run on tablets and smartphones
2. MyTaxiService web application will run on every terminal which is provided an internet connection and a web browser

2.1.2 User Interfaces

Both the web and mobile applications will have a similar user interface, rearranged to fit the device screen.

1. **Application for passengers**

   - It has to provide a **login page** that allows users to be recognized by the system, along with a registration form that let the user to sign up to the service and a link for resetting the user password. In the home page after the login, it is possible to see the map of the zone in which the user is and the taxis that are nearby, along with a menu for accessing the other functionalities. If a taxi is available, the user can request a ride by selecting the button "Call Now". If a taxi is not available, a text message will display the estimated waiting time and the button will let the user to queue. When the user select the request button, a **confirmation dialog** will ask the user to insert the precise meeting location and, in the case the user is making a reservation, the meeting time. After the user has made a taxi request, the system will show directly in the home page the estimated waiting time for that request. In the case he has made a reservation, the home page is still usable for making a request, while clicking on **"your reservation"** a page will display the current reservation status.
   - **Passenger menu**

     Home Shows the home page from which is possible to make a new request or check the arrival time and the taxi code of the pending request.

     Your Reservation Shows the list of pending reservations along with the status or estimated arrival time/taxi code.

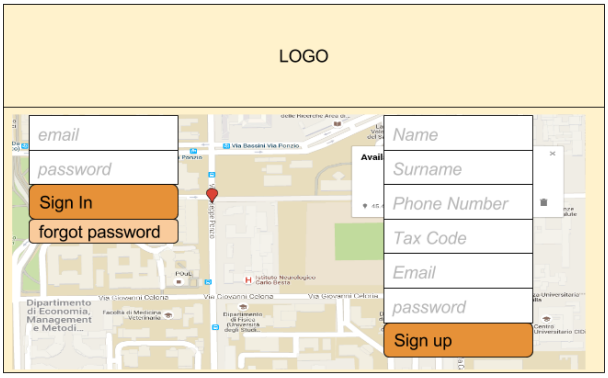
     Reserve a Taxi Display the **confirmation dialog** in which the passenger can set the meeting time and location

     Your Account Lead to the account administration page in which the user can show and edit his account

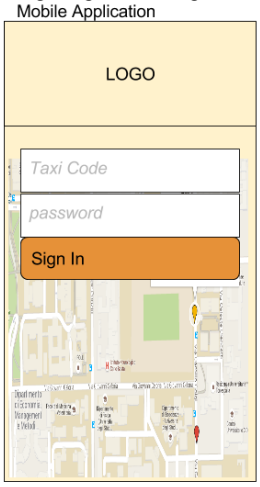
     Logout Ends the session and leads to the login page

2. **Example of the mobile application for passengers:**

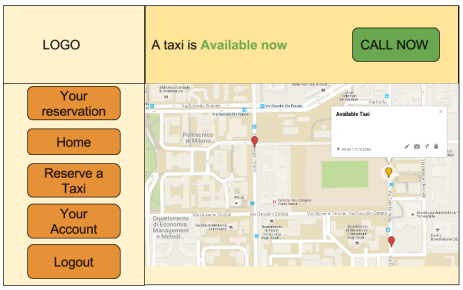
Login Page for passengers - Web Application



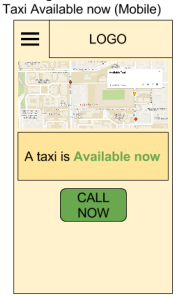
Login Page for Passenger Mobile Application



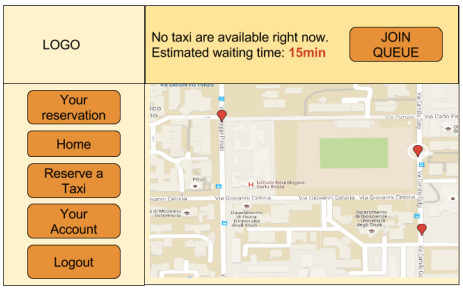
Passenger Home - Taxi Available now (Web Application)



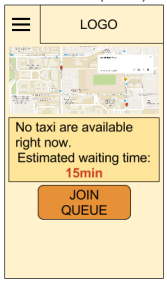
Passenger Home
Taxi Available now (Mobile)



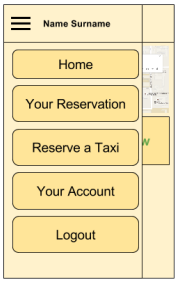
Passenger Home - Taxi not Available (Web Application)

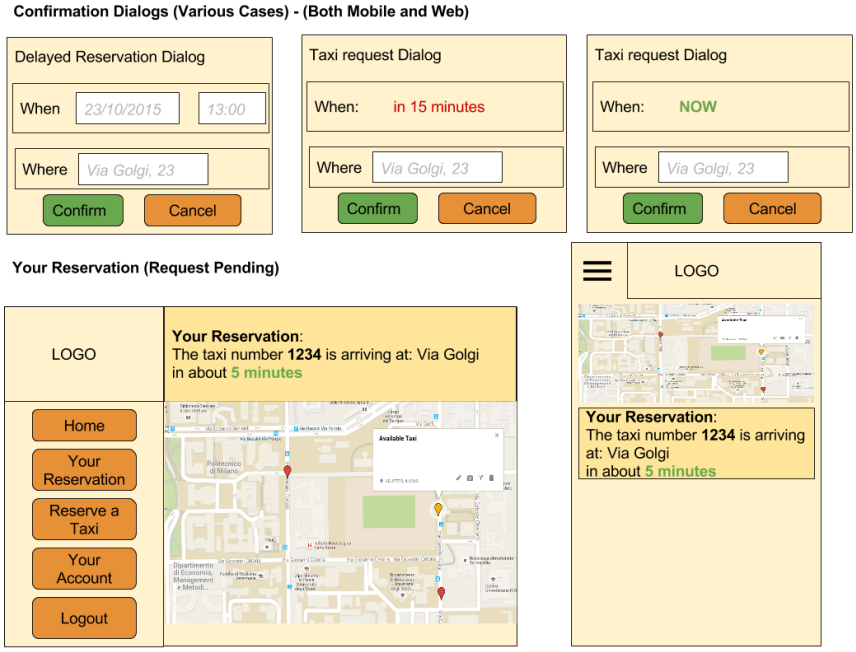


Passenger Home
Taxi not Available (Mobile)



Sliding Menu (Mobile)



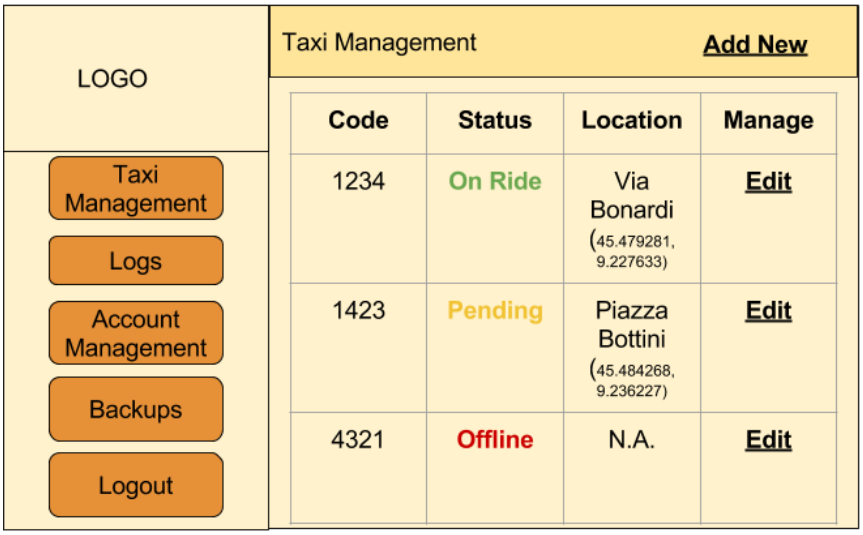


3. **Application for System Administrators (Web)**

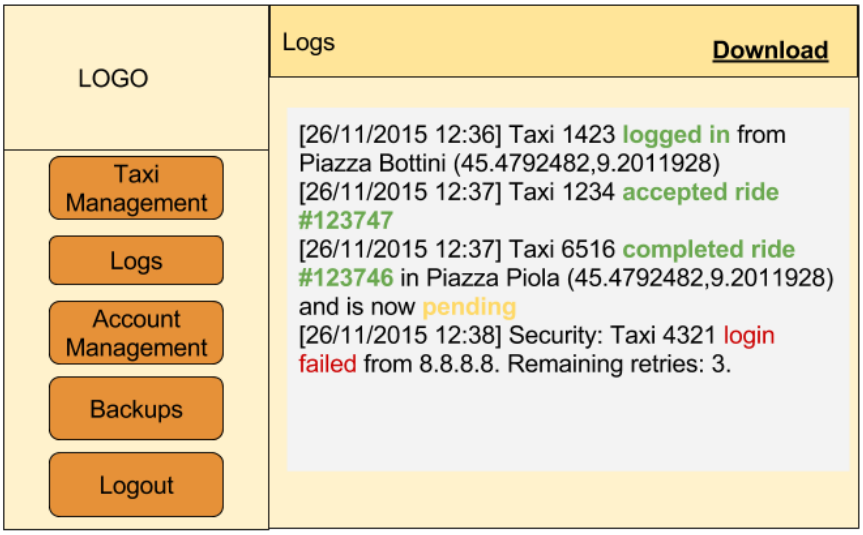
   - From the same **login page** for the users, system administrators can access the administration page by inserting their login data. From the administration panel the user can see the list of all the taxis and their current status. From the menu they can access the log page from which it is possible to read all the system history, the account administrator page or the backup/restore facility.

4. **Example of the web application for administrators:**

Administration - Taxi Management (Web Application)



Administration - Logs

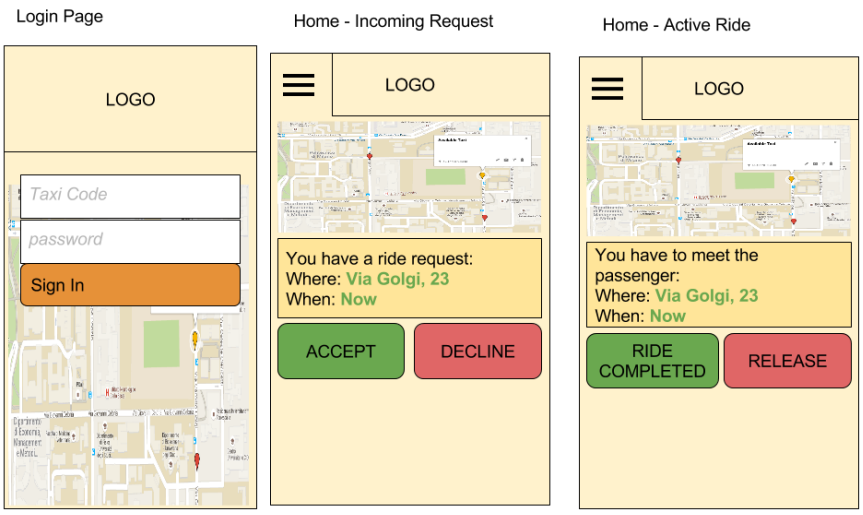


5. **Application for taxi drivers (Mobile)**

   - The taxi drivers have a particular login page that allows them to access by taxi code/email and password.
     The taxi application interface is as simple as can be, for avoiding loss of time and incomprehensions. In the homepage the user

can see a map of the nearby area. When a request arrives, the user is notified about the time and location of the request. Two buttons allow to accept/decline the request, and when a request is accepted the taxi driver can state he completed the ride or release the ride for another drive.

6. **Example of the mobile application for taxi drivers:**



2.1.3 Hardware Interfaces

The MyTaxiService mobile application will require a smartphone or tablet and will support the most common screen resolutions.
For the correct functionality of MyTaxiService mobile and web application (both for passenger and taxi drivers) the device will be required to support the GPS localization and a WiFi or data internet connection.

2.1.4 Software Interfaces

- MyTaxiService Application Server
  - Database Management System (DBMS):

  Name : MySQL.
  Version : 5.6.19
  Source : http://www.mysql.it/

- Application server:

  Name : Jboss application server J2EE open source.

  Source : http://www.jboss.org/

- Operative Systems:
  - : Ubuntu/Debian Linux
  - : Windows Server, Windows 7, Windows 8, Windows 10

- MyTaxiService Application Server
  - Mobile Application

    OS : Android 4.4.2 or higher, iOS, Windows Mobile.

  - Web Application

Web Browsers : Chrome, Firefox, Safari, Opera, Internet Explorer

Other : JRE 1.7 or higher

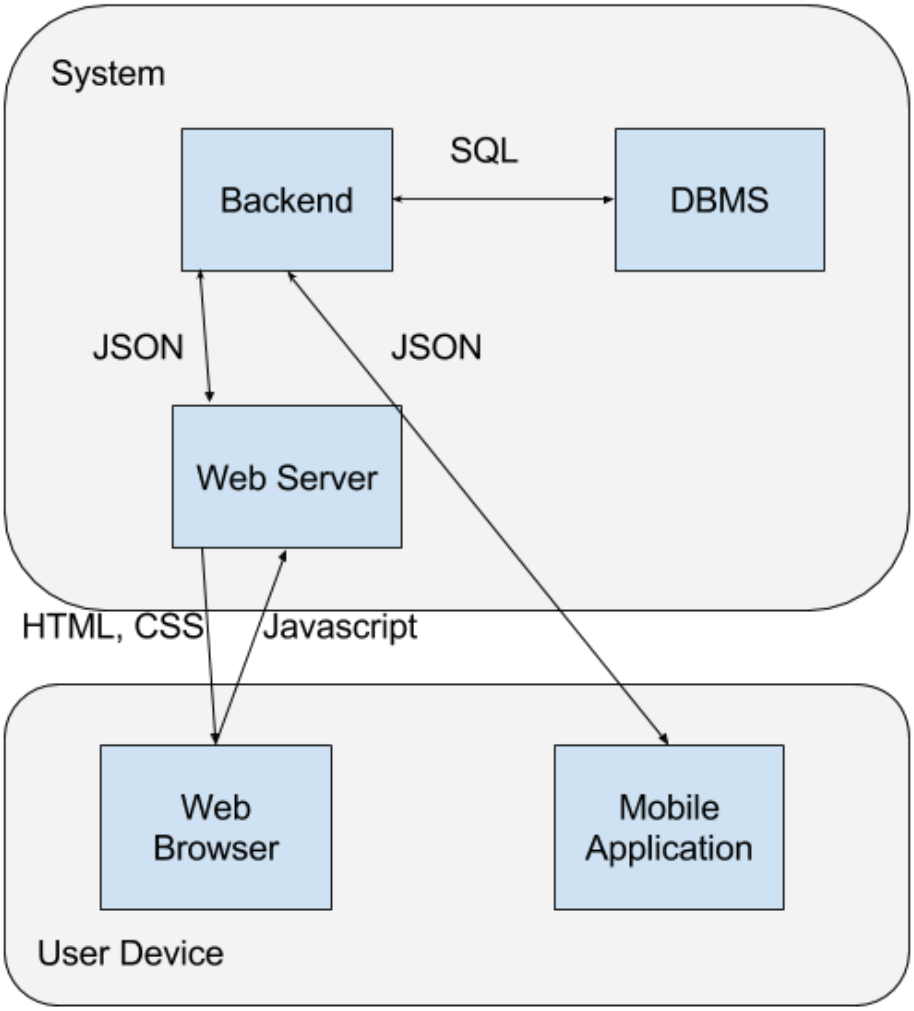
The internal system must make use of these tecnologies for the internal component interfacing

2.1.5 Communications Interfaces

The communication between the server and mobile application will make use of the following ports:

Protocol	**Application**	**Port**
TCP	HTTP	80
TCP	HTTPS	443
TCP	DBMS	3306

The default communication protocol will be HTTPS.
The data exchange between the server and the mobile applications will occur by the JSON data format, while the web interface will make use of both JSON and AJAX for the communication between the Web Browser and the server.



2.1.6 Memory Constraints

The suggested memory requirements are:

- Application Server

Main Memory : 4GB
Secondary Memory : 2GB

- Mobile Application

Main Memory : 500Mb
Secondary Memory : 100Mb

2.1.7 Operations

In this section will be described what operations each kind of user can do by interacting with the system interface:

1. Passengers (Web and Mobile Application)

   Registration The user can create a new account filling in his personal informations

   Login The user can access the main page by inserting his email and password.

   Map The user can visualize the map of the zone he is in.

   Request The user can request a taxi immediately, or adding to the queue according to the taxi availability

   Reservation The user can make a reservation of a taxi ride for a later moment

   Pending Reservation The user can visualize data such the estimated taxi arrival time about the reservation he made and cancel a reservation.

   Information The user can access some information about the system usage and rules

   Profile The user can access its profile personal data

   Logout The user can logout from the system

2. Taxi Drivers (Mobile Application Only)

   Login The user can access the main page by inserting his taxi code and password. This will state the availability of accepting requests.

   Map The user can see a map of the nearby area. When a request arrives the map will show the meeting point

   Request Incoming The user can accept or refuse an incoming request

   Active Ride The user can complete a ride when the passenger is at his desired location, or release the ride to another taxi driver if some unexpected event occurs.

   Logout The user can logout from the system and state his unavailability of taking requests.

3. System Administrators (Web Interface Only)

   Login The user can login and access the administration page

Taxi Management The user can acess the list of all active and unactive taxi accounts. He can Add and Remove taxi driver accounts or change their passwords.

Account Management The user can access user profiles to provide support to end users and reset their passwords.

Logs The user can view and download the logs associated to each accepted/refused request, all ride data such passenger account, departure/arrival date and location, and the taxi driver who made the ride.

Backup/Restore The user can make security backups of account data and restore them.

4. System Operations

Store user information Creates or updates a passenger profile

Store taxi information Creates or updates a taxi profile

Update Map Update the map every 30 seconds with each taxi location

Registration Confirmation The system sends a confirmation email to the user upon registration

Request Confirmation The system sends a confirmation email about the acceptance of a taxi request

Reservation Confirmation The system sends a confirmation email about the acceptance of a taxi reservation

Reservation Reminder The system sends a remainder email to the user who reserved a taxi 2 hour before the meeting time

Time estimation The system estimate the time needed to move by car from a location A to a location B. It can make use of external map services for this purpose.

5. System Support Operation

Backup Database The system make a backup of the database and logs

Restore Database The system make a restore of a previous backup of the database

2.2 Product Functions

This section will list the general functions of the system, with an eventual reference to the goal they contribute to satisfy.

1. *Relative to goal **1***: The system shall provide the passengers both a web and a mobile application with equivalent functionalities for requesting a ride

2. *Relative to goal **5***: The system shall provide a mobile application for taxi drivers, in which they can inform the system about their availability and confirm or refuse the proposed request.

3. *Relative to goal **6***: The system shall associate a queue of taxis to each zone according to the GPS location of the taxis

4. *Relative to goals **3** and **4***: The system shall provide a response to the passenger who requested a ride, the notification will contain the code of the taxi that accepted the request and the estimated waiting time.

5. *Relative to goal **7***: The system shall comprehend a programmatic interface that offer the access for the main functionality of the system itself.

6. *Relative to goal **2***: The system interface for passengers shall provide the functionality of taxi reservation for rides that will take place after 2 hours. The information required by the system include the origin and destination and the departure date.

7. Relative to goal **6** and security: The system shall keep track of any active ride, along with the current taxi position and history, for the system administrators.

8. *Relative to goal **5** and Administration*: The system should provide a web interface for system administrators, in which they can manage the taxi drivers and the passenger profiles

9. *Relative to goal **3*** The system sends a remainder email or push notification to the user who reserved a taxi 2 hour before the meeting time, just after that it has assigned a taxi to that request.

10. *Relative to goal **6***: The system should manage exceptional situations such the availability of taxis or accidents/vehicle failures.

11. ***Not provided functionality***: The system will not manage the payment for the taxi, which will occour directly to the taxi driver via cash or credit card and it will be managed separately by the transportation company

2.3 User charateristics

There are 4 types of system users:

passenger Who access the system for requesting a transportation. For this purpose, the user must have an internet connection on his smartphone or computer. In the case he's using a smartphone, the user can have the MyTaxiService application installed or using the web interface. For using the web interface the user must have a compatible browser.

Taxi Driver Who provide the transportation to the passengers. The user interacts with the system for receiving transportation tasks. The user will be provided a compatible device in advance, and must be in possess of his login credentials, which are generated by the system and managed by system administrators.

System Administrator Who manages the transportation service. The system administrator is in charge to give support to taxi drivers and passengers, in particular to create and supervise taxi drivers' accounts. It's the only figure that have access to the system logs and taxi managements.

System Developers Who is in charge to further extends the system. System developers will make use of this document and the Design document for planning the addition of new functionalities.

2.4 Constraints

2.4.1 Regulatory Policies

The system shall comply with the following regulatory policies:

- CODICE IN MATERIA DI PROTEZIONE DEI DATI PERSONALI "Decreto legislativo 30 giugno 2003, n. 196"

2.4.2 Reliability Requirements

- In case of system faliure, the system must be recoverable from a previous backup
- The system infrastructure (data centers, servers, network infrastructures, etc) must be sized accordingly to the expected user base size.

2.4.3 Criticality of the application

The main criticalities of this project are:

- The compliance of the interface according to the usability metrics explained in section 3.
- The fair management of taxis. In particular the system must not waste resources such as not reallocating unemployed taxi when necessary and must allocate requests according to well defined precedence rules.

A failure in one or more of this criticalities will lead to the loss of clients.

2.4.4 Safety and Security Consideration

MyTaxiService should comply to the serurity statements listed above:

- Only logged users can request a taxi ride. This is for preventing any malicious behaviour in which unknown users can make taxi request without making use of the transportation service.
- At the time of registration, a user must read and accept any privacy policy about the user data management
- The system must undertake to not divulge the user data to third parties, in addition the user data will be used only for the supply of the service
- For the reason stated before, the passenger must supply at least:
  - Name
  - Surname
  - e-mail address
  - Phone number or Tax Code
  - Password
- While the taxi drivers must supply at least:
  - Taxi Code
  - e-mail address
  - Phone number or Tax Code
  - Password
- The system should therefore warn the user about the regulation policy about false impersonation.
- The system will hash the user password before storing it into the database, preventing unauthorized access in the case of data leak.
- The system will ask the user the email address in order to reset his password. The new password will be sent on the user e-mail address.
- In the case of taxi drivers, the password is generated by the system and it's updated periodically. The taxi drivers will receive their new password by email.
- Every request must be associated to the user identity and kept in a log for security reasons and kept available for judicial authorities.

2.5 Assumptions and Dependancies

2.5.1 Assumptions

1. The city is already divided in Taxi Zones of approximately 2x2 km each.

2. Taxis or Taxi Drivers are assumed to be in possession of a mobile device that

   1) have GPS functionalities

   2) is capable of establishing an internet connection with the facility in which the system is hosted. In the case one of these assumption does not hold, the Taxi driver must contact his referent in order to solve the issue.

3. The number of taxi in service are at least equal to the number of zones. **Rationale**: In this way, in the case that a zone queue is empty, an incoming request is guaranteed to forwarded by at least one taxi in an estimable time by propagating the request to adjacent zone and forwarding the request to the nearest first taxi that is available.

4. The service is currently running a traditional system based on phone calls

5. The payment will occur directly to the taxi driver by cash or credit card and it will be managed by the transportation company

6. The taxi drivers don't need to know in advance (when they accept a request) what the ride destination will be, as usually the passenger informs directly the taxi driver about the desired arrival location.

7. The taxi Drivers are supposed to accept one request or reservation at a time.

8. The maximum time length of a ride is 1 hour and 45 minutes.

9. All zones have a square o rectangular shape

10. There are no area in which the service operates that are not covered by a zone.

11. Zones do not overlap.

3 Specific Requirements

3.1 External Interfaces

This section will list the main input/outputs of the system.

Name	Login
Context	Login
Description	A user requests to Login
Source	Passenger
Destination	System
Relations	None
Channel	Internal to application (via Web)
Data Format	• email: string • password: string

Name	TaxiProbe
Context	Taxi Request or Reservation
Description	The passenger application retrives informations on taxi availability
Source	Passenger Application
Destination	System
Relations	None
Channel	Internal to application (via Web)
Data Format	• User Location (GPS)

Name	TaxiProbeResponse
Context	Taxi Request or Reservation
Description	The system replies to a TaxiProbe with the current taxi availability informations
Source	System
Destination	Passenger Application
Relations	Response to TaxiProbe
Channel	Internal to application (via Web)
Data Format	• taxisAvailable: boolean • taxiLocations(taxiCode, taxiLocation, isAvailable): list of locations • waitingTime: integer, in seconds

Name	TaxiRequest
Context	Taxi Request
Description	The user makes a real-time taxi request
Source	Passenger
Destination	System
Relations	None
Channel	Internal to application (via Web)
Data Format	• User Email: string • currentTime: long • Meeting Location: location

Name	TaxiReservation
Context	Taxi Reservation
Description	The user makes a taxi reservation
Source	Passenger
Destination	System
Relations	None
Channel	Internal to application (via Web)
Data Format	• User Email • currentTime: Timestamp • meetingTime: Timestamp • meetingLocation: Location • arrivalLocation: Location

Name	Confirmation
Context	Taxi Request or Reservation
Description	The system confirms a Taxi request or reservation
Source	System
Destination	Passenger
Relations	Response to a TaxiRequest or TaxiReservation
Channel	Email or Internal to application (optional, for push notifications)
Data Format	• User Email • requestId • requestTime: Timestamp • meetingTime: Timestamp • meetingLocation: Location • arrivalLocation: Location • accepted: boolean

Name	Notification
Context	Taxi Request or Reservation
Description	The system updates the information about a request or a reservation
Source	System
Destination	Passenger
Relations	Triggered by a pending reservation or request
Channel	Email or Internal to application (optional, for push notifications)
Data Format	• User Email • requestId • requestTime: Timestamp • meetingTime: Timestamp • meetingLocation: Location • arrivalLocation: Location • accepted: boolean • taxiArriving: boolean • taxiCode • waitingTime: integer

Name	DriverRequest
Context	Taxi Request or Reservation
Description	The system inform the taxi driver of an incoming ride request
Source	System
Destination	Taxi Driver
Relations	Triggered by a pending reservation or request
Channel	Internal to application
Data Format	• requestId • requestTime: Timestamp • isReservation: boolean • meetingLocation: Location • arrivalLocation: Location (optional) • meetingTime: Timestamp (only if isReservation)

Name	DriverResponse
Context	Taxi Request or Reservation
Description	The taxi driver accepts or refuses a DriverRequest
Source	Taxi Driver
Destination	System
Relations	Response to a DriverRequest
Channel	Internal to application
Data Format	• requestId • email • requestTime: Timestamp • isReservation: boolean • meetingLocation: Location • arrivalLocation: Location (optional) • meetingTime: Timestamp (only if isReservation)

Name	DriverNotification
Context	Taxi Request or Reservation
Description	The taxi driver inform the system he has completed a ride or he is releasing it.
Source	Taxi Driver
Destination	System
Relations	Only accepted if an affermative DriverResponse arrived to the system
Channel	Internal to application
Data Format	• requestId • email • requestTime: Timestamp • currentLocation: Location • isCompleted: boolean • isReleased: boolean

Name	LocationUpdate
Context	Taxi KeepAlive
Description	The taxi application inform the system about its location.
Source	Taxi Driver Application
Destination	System
Relations	Only possible if the driver is Logged
Channel	Internal to application
Data Format	• email • currentLocation

Name	NewPassword
Context	Security
Description	The system provides the user his new password
Source	System
Destination	User
Relations	Only if the password has been Resetted
Channel	Email
Data Format	• email • newPassword

3.2 Functions

This section will list specific functional requirements needed in order to accepting and processing inputs and processing and generating the outputs.

1. *Relative to goal **6***: The mobile application for Taxi Drivers shall transmit the taxi GPS location to the system every 30 seconds, in order to keep the taxi queues updated.

2. *Relative to goal **6***: When a taxi becomes available, the system shall store the taxi identifier in the queue of the corresponding zone

3. *Relative to goal **6***: When a request arrives from a certain zone and there is at least one available taxi in that zone, the system shall forward it to the first taxi queuing in that zone.

4. *Relative to goal **6***: Upon an incoming request, If the Taxi Drivers confirms the system shall inform the passenger, if he doesnt confirm then the system will forward the request to the next driver in queue

and move the taxi driver which refused in the last position in the queue.

5. *Relative to goal **3***: The passenger will be informed of any unexpected event by email if he made the request via the web interface and via email and push notification if hes using the mobile application.

6. *Relative to goal **3***: In any case, the passenger can check the request details and information by accessing the web or mobile interface

7. *Relative to goal **6***: If the taxi driver who accepted a request is unable to reach the passenger in a reasonable amount of time (i.e. due to an accident or a vehicle failure) then the taxi driver should be able to release its request. The request is forwarded to the first available taxi and the passenger will receive an update about the new taxi code and the new estimated waiting time.

8. *Relative to goal **2***: The reservation request is accepted only if it is made at least two hours before the ride. An earlier reservation should be made impossible to select from the GUI and double checked in the backend.

9. *Relative to goals **6** and **2***:For guaranteeing the precedence of reservation over real-time requests, a reservation is forwarded to a taxi 2 hours before the meeting time: in this way the taxi will result not available for eventual real-time requests (which has a maximum time length, as stated in the assumption **??**) that occur in the meanwhile and the request will be forwarded to the next taxi. The taxi is supposed to be at the meeting point at least 10 minutes before the meeting time.

10. *Relative to goal **6***: Along with the taxi queue, there will be a Request Queue, with a FIFO policy. In this way, the first passenger who makes a real-time request is the first that is served by the first available taxi.

11. *Relative to goal **1***: The system should provide passengers a "password reset" service.

12. *Relative to goal **1***: The system should let the user cancel a previous reservation he made until 2 hours before the meeting time.

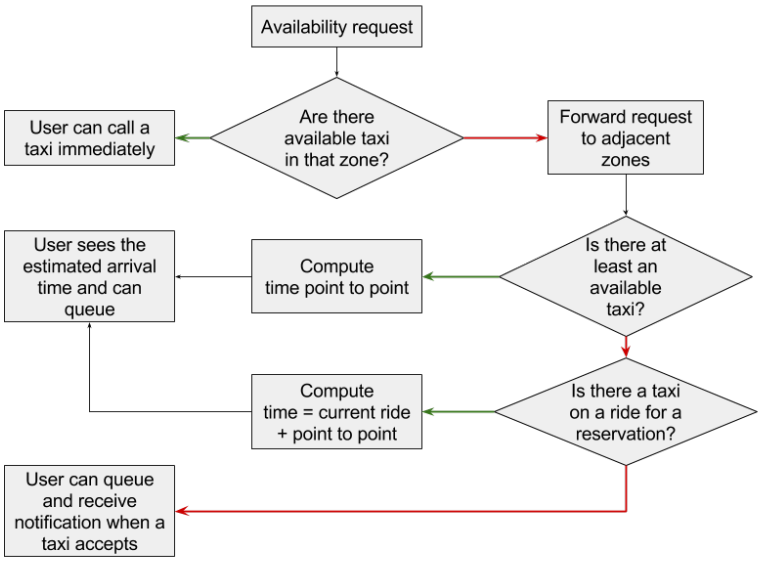
13. *Relative to goal **3***: The system will send notification to a user as soon as the request has been accepted by a taxi driver

14. The GPS localization of the passenger is used merely for providing a better GUI experience (i.e. showing the right portion of the map) and it is an optional requirement for the passenger. The actual localization occurs by the information that the user provides by inserting the meeting location while making a request.

15. If the Taxi Driver doesn't reply to a request within a minute, the request is considered refused.

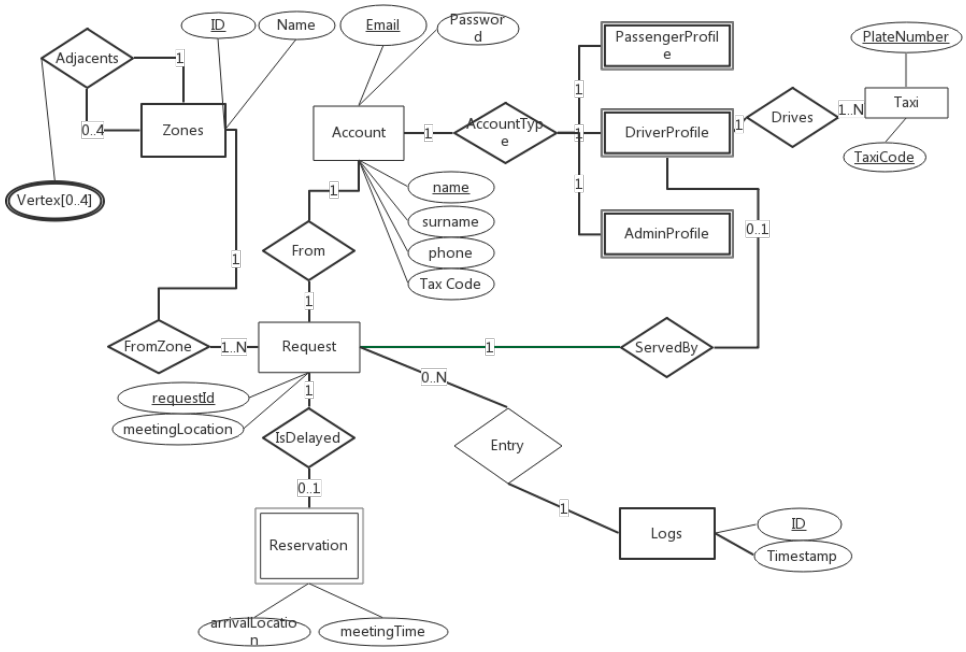
3.2.1 Exceptions Management

In the following diagram will be described the logical flow of steps to follow in some particular cases:



3.3 Logical Database Requirements

This section contains a ER Schema draft for the database, meant for giving the designers a better understanding of the crucial application data.



3.4 Design Constraints

The client application will be a **thin client**.

3.4.1 Standard Compliance

The mobile application for passenger must comply with the application store regulation policy, depending on the mobile OS for which will be deployed.

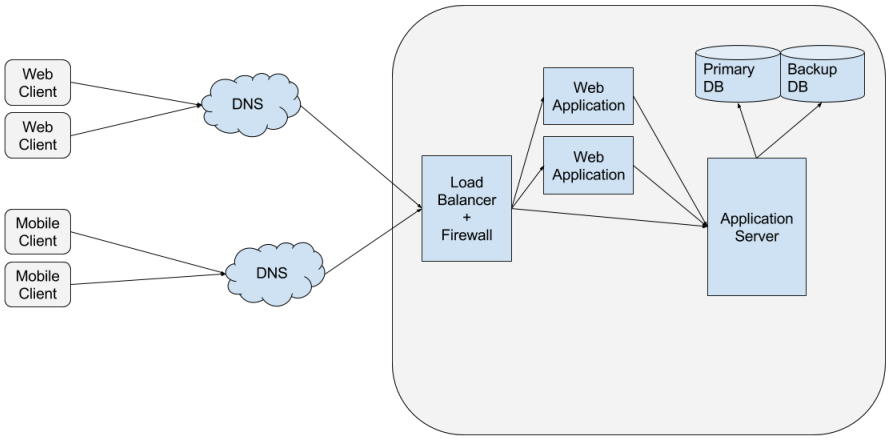
3.5 Software System Attributes

3.5.1 Performance Requirements

- The server shall process the 95% of operations in less than 1 second.
- In optimal network conditions the user shall get a response for a confirmation in less than 5 seconds. This not includes email confirmations.

3.5.2 Availability

Here follows a draft of the interal system phisical architecture, which aids to bettere comprehend the availability analysis.



- Each database and web application server shall have at least a 3-nine availability.
- The load balancer, firewall and application server shall have at least a 5-nine availability since they are crucial for the business process and they are more cost-effective.

3.5.3 Security

1. The login information and the user data must be kept safe by the system. In particular, the user data can be accessed only by the user itself or a system administrator, which can see and edit the entire user profile except the password.
2. A taxi request or reservation can be made only by registered and logged users, in order to avoid unintentional or malicious requests.
3. The transport layer will make use of SSL and HTTP (HTTPS)
4. The passenger password can be reset by inserting the user email. The server will then provide a new temporary password and send it by email to the user.
5. The taxi driver password is managed directly by the system and can be resetted only by contacting a system administrator. This is for avoiding disturbance in the case a malevolent user learns about the taxi driver email and tries to reset the taxi driver password. The taxi driver password is reset every six month and sent automatically to the taxi driver email recipient

6. All user password must have a costraint in the number of characters, which will be between 8 and 20.

3.5.4 Portability

The MyTaxiService service will be written in Java. All system modules will have to be developed accordingly to the inherent java portability, in particular the testing phase will comprehend the following systems:

1. Android 4.4 and later
2. iOS
3. Internet Explorer, Chrome, Safari, Opera, Firefox

3.5.5 Usability

The system must comply with the following usability metrics (both for web and mobile application):

- At least a score of 95 percent in Completion Rates
- The mean Task Time for taxi request must be at most 1 minute
- The mean Task Time for taxi driver application (excluded the ride time) must be at most 10 seconds (From Logging to Ride Complete/Release).

Other usability metrics can be added during the design of the interface

3.6 Scenarios

1. Alice is at the Garibaldi railway station, as he has just arrived from Rome. She needs a lift to his office as soon as possible, so he decides to look for a taxi. On her smartphone she installed the MyTaxiService app. As soon as she get off of the train, Alice checks the application and control the situation of taxis in the zone of the station, then she requests a taxi service by inserting her position. After a few seconds the application replies with the code of a taxi which is near the station. When Alice will get out of the station, she will find the taxi with the corresponding code that will take her to his office.

2. Tomorrow Bob will have to go to the airport, as he has a flight at 23:00. He must be at the airport at 21:00 and he need to take a taxi. Since hes at home, Bob decides to register to MyTaxiService website from his computer in order to reserve a taxi for the airport. After inserting a few informations he is able to reserve a taxi for the 20:00 of the day after. The day after, a taxi driver receives a request at 18:00, the request indicates Bobs home address and the arrival time which

is at 19:50. When the taxi drivers accepts the request, Bob receives an email asserting that a taxi is on the way for his home and will be waiting there when he will get out at 20:00.

3. Carl works for an important society of import-export; today afternoon he will take the plane for going to Usa to close an important deal. His friend Daniel will bring him to the airport. Unfortunately his friend had an accident so he cant bring Giovanni to the airport. The inconvenient occurs at 14:30 and Giovanni must be at the airport at 15:30. Carl uses the web application to request a taxi at 14:45; but he sees on the web page that no taxis are available in that zone right now, but the nearest available will arrive in 25 minutes. He still decides to queue for a taxi anyway and the system accepts his request.

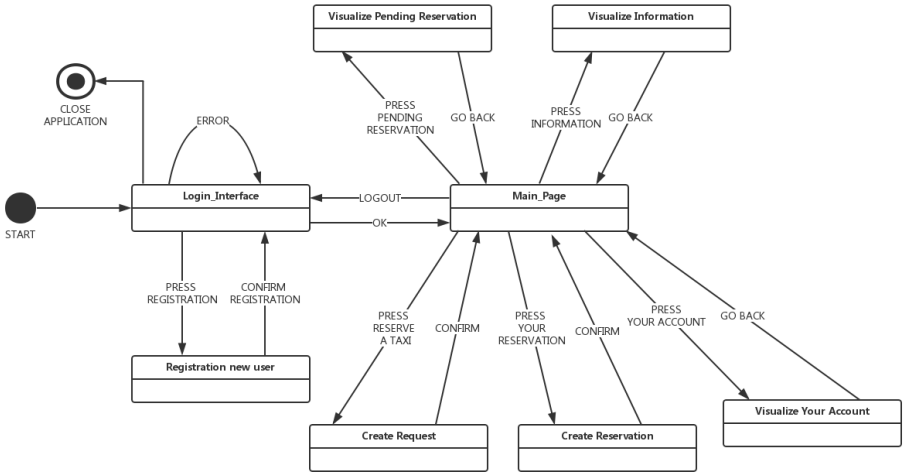
4. Kevin is a taxi driver and on his smartphone has installed MyTaxiService taxi application. He has just done a ride for bring Mrs Thompson from the station to her house. As soon as he leave Mrs Thompson and receives the payment, he select the Ride Completed button. After a few seconds, a new request arrives. He accepts the request and he leaves for the new meeting location.

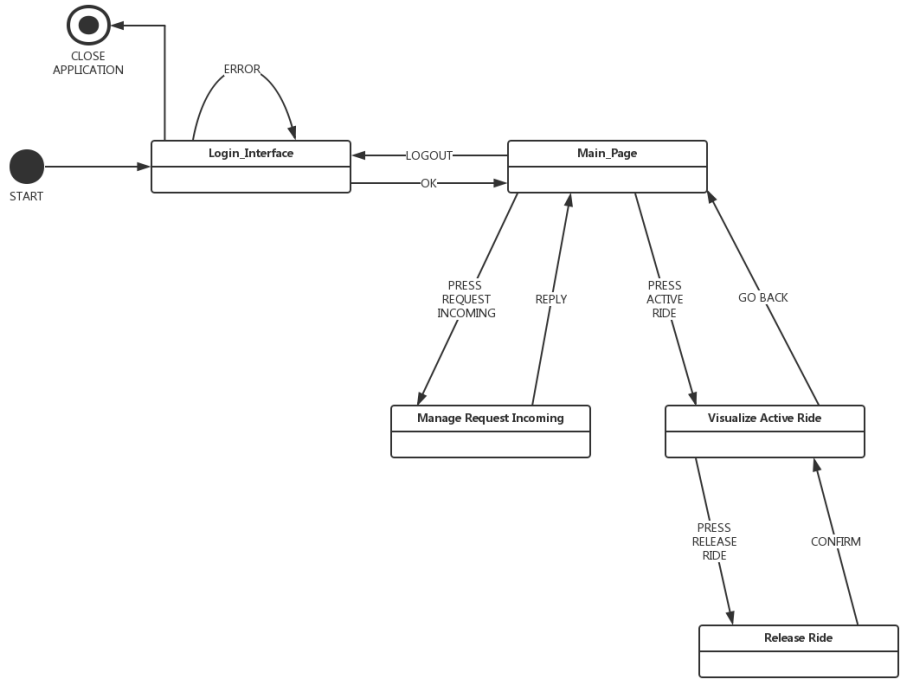
5. Samuel is a taxi driver and on his smartphone has installed MyTaxiService taxi application. He has just done a ride for bring Ms Rossi from the station to her house in the other side of the city. When he left her, on his smartphone, a request for a ride arrives. As his shift is about to end, he refuses the request and logs out, so the system contacts another taxi driver that accepts the request.

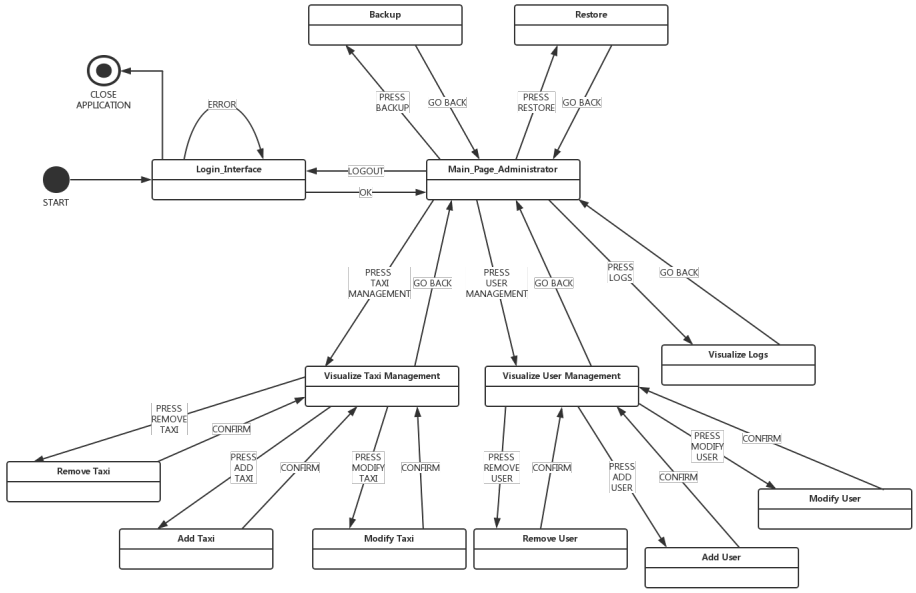
6. Giovanna is a passenger. Since she hasnt used the service for a long time, she cannot remember her password, so she press the forgot password button in the login page and inserts her email address. After a few seconds the system sends her a new password, that she uses to log-in and access the profile page where she can input a new password inserting the generated password and the new one.

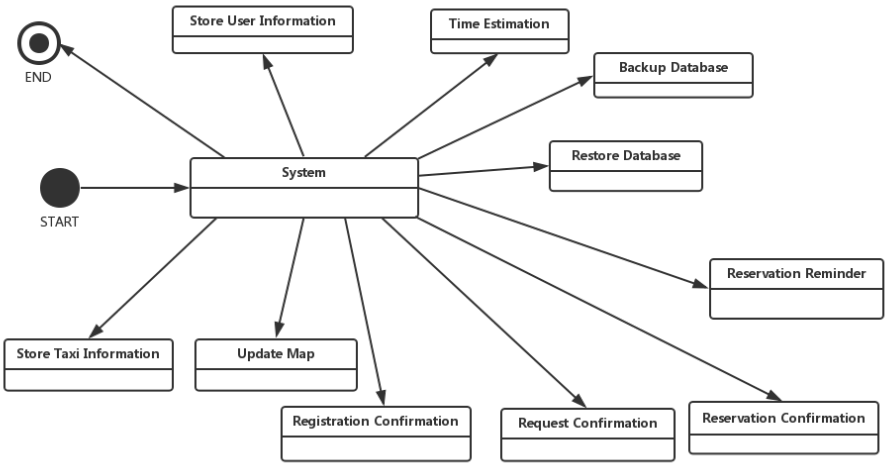
7. As the trains are on strike, the taxi service is literally overwhelmed by requests. Luca wants to request a ride from the station in the zone 1 of the city so he opens the web application home page. In his zone there are no taxi available, so the system checks if theres an available taxi in the other zones but no available taxis are found.
   Luca sees a notification about that and he decides that he can wait a bit, so he commit request and the system puts that in a queue.
   When a taxi will be available, it will receive the request, and if the taxi driver accepts, Luca will receive an email informing him that a taxi is on the way, along with the estimated waiting time.

3.7 State Machines

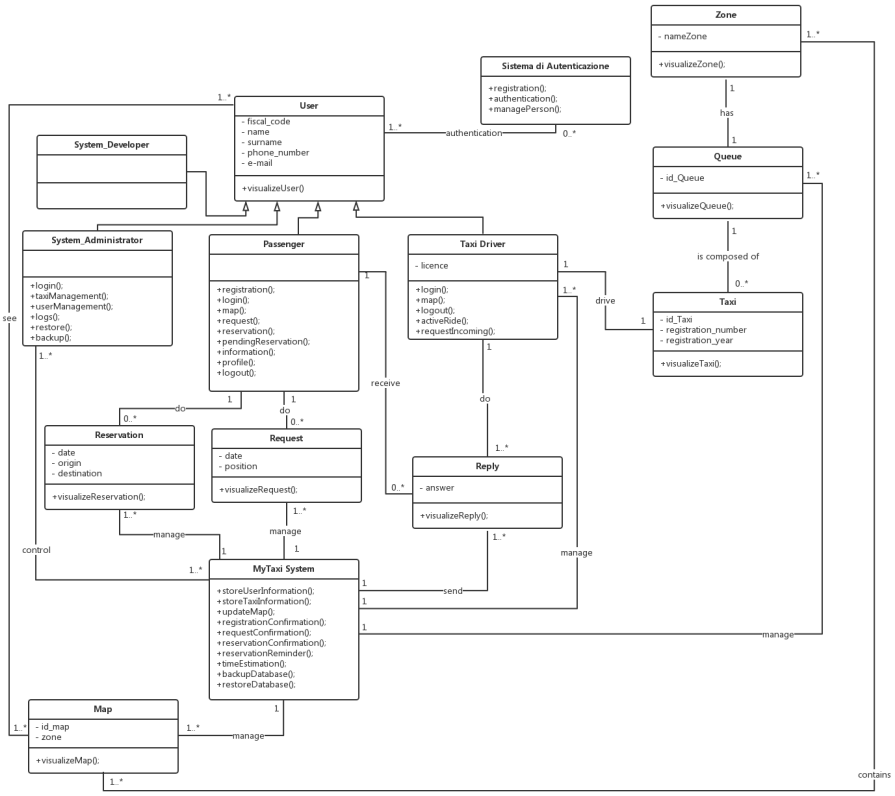





Backup
Restore
CLOSE APPLICATION
ERROR
PRESS BACKUP
GO BACK
PRESS RESTORE
GO BACK
Login_Interface
LOGOUT
OK
Main_Page_Administrator
START
PRESS TAXI MANAGEMENT
GO BACK
PRESS USER MANAGEMENT
GO BACK
PRESS LOGS
GO BACK
Visualize Logs
Visualize Taxi Management
Visualize User Management
PRESS REMOVE TAXI
CONFIRM
Remove Taxi
PRESS ADD TAXI
CONFIRM
PRESS MODIFY TAXI
CONFIRM
Add Taxi
Modify Taxi
PRESS REMOVE USER
CONFIRM
Remove User
PRESS ADD USER
CONFIRM
Add User
PRESS MODIFY USER
CONFIRM
Modify User

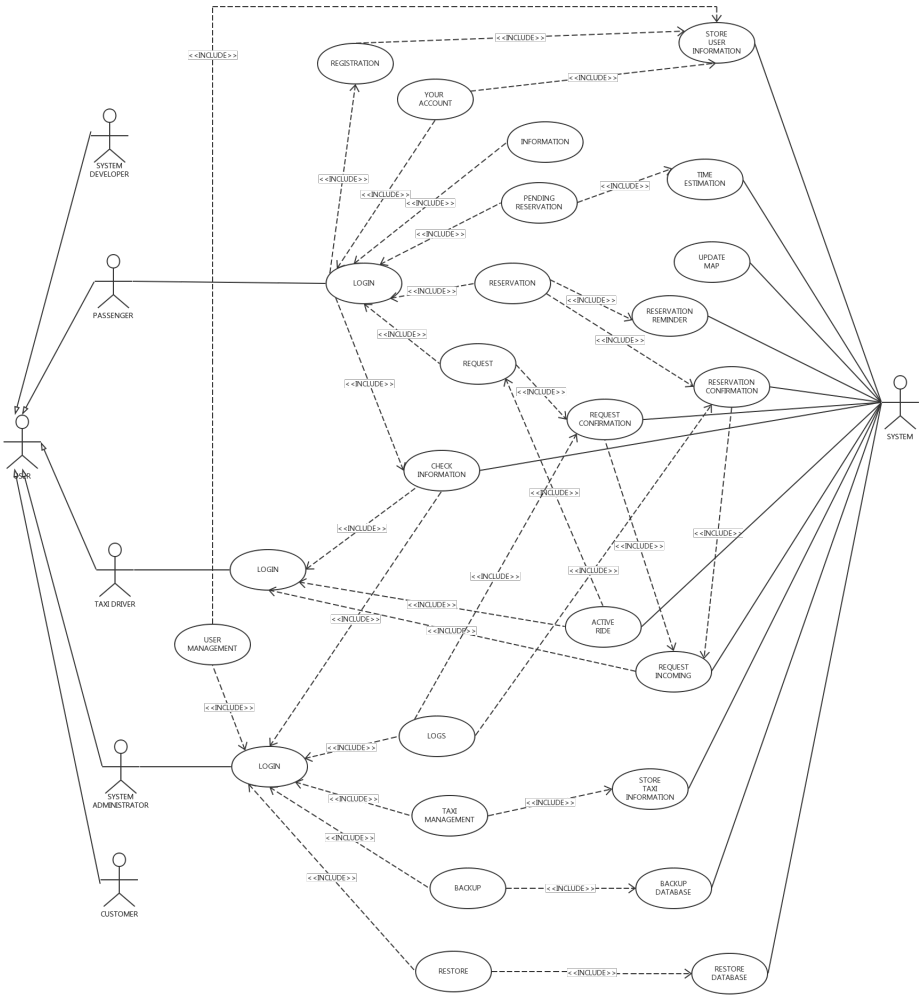

Store User Information
Time Estimation
Backup Database
END
System
Restore Database
START
Reservation Reminder
Store Taxi Information
Update Map
Registration Confirmation
Request Confirmation
Reservation Confirmation

3.8 Class Diagram



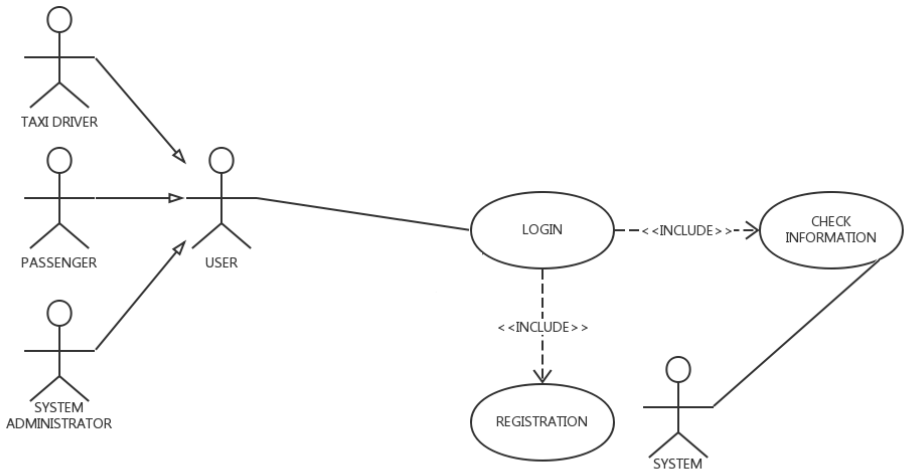
3.9 Use Cases

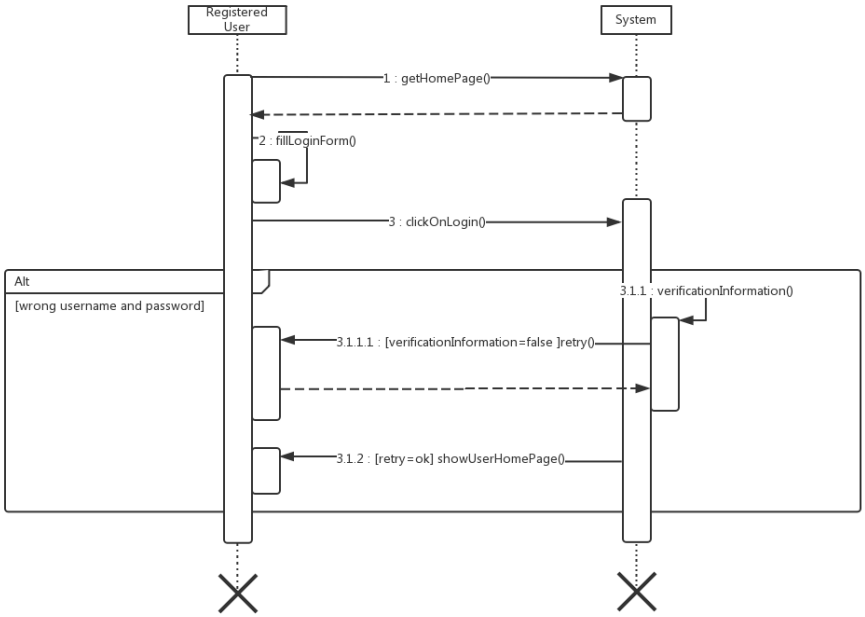
3.9.1 Use Case: General



3.9.2 Use Case: Login

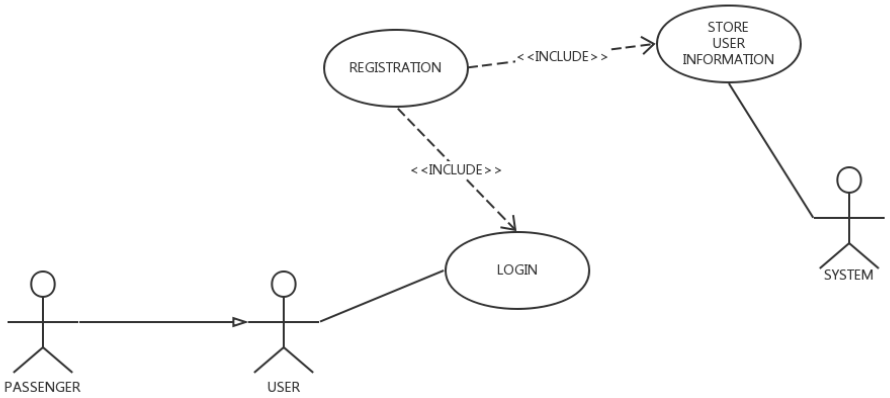
Name	Login
Actors	User (Passenger, Taxi Driver, System Administrator), System
Entry Conditions	The user has successfully inserted username and password in the login form
Flow of events	1. The user goes to the home page of the application. 2. The user clicks the button login. 3. The user enters his username and password in the form provided 4. The system shows the private page of the user 5. The system controls the informations that the user inserts.
Exit conditions	The user clicks on logout.
Exceptions	The information inserted in the form is wrong, an error message is shown and so the user re-inserts the information.

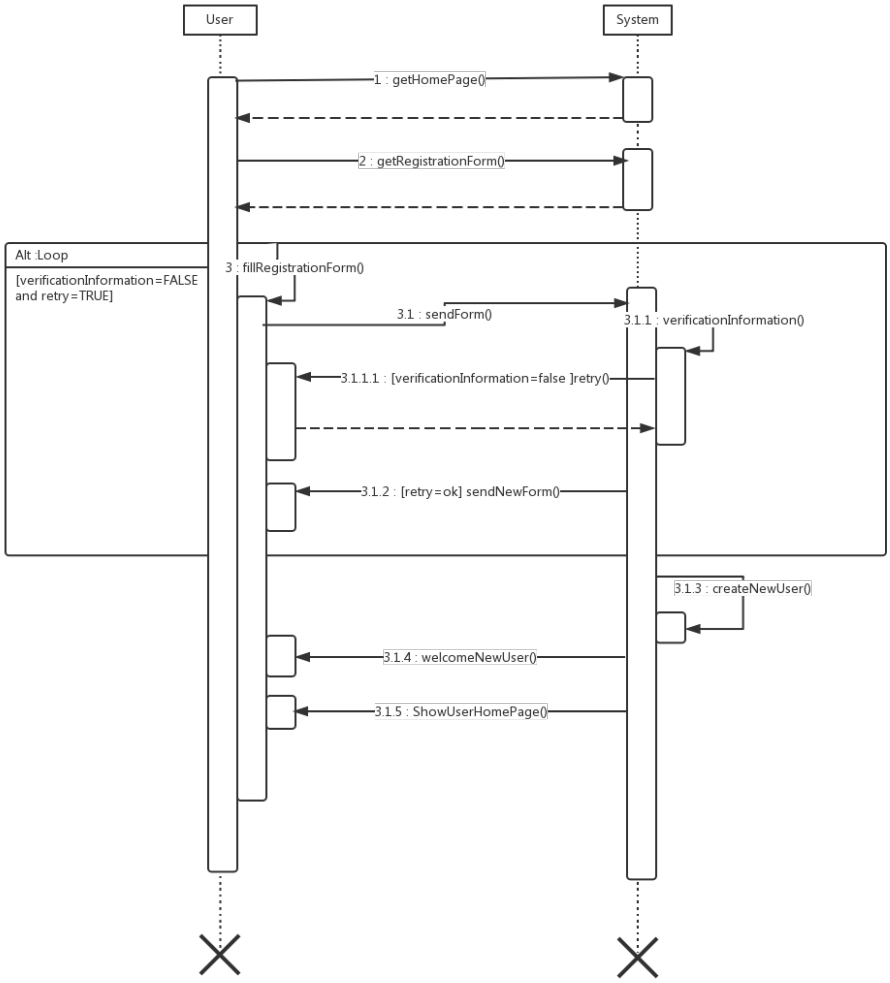



Registered User
System
1 : getHomePage()
2 : fillLoginForm()
3 : clickOnLogin()
Alt
[wrong username and password]
3.1.1 : verificationInformation()
3.1.1.1 : [verificationInformation=false]retry()
3.1.2 : [retry=ok] showUserHomePage()

3.9.3 Use Case: Registration

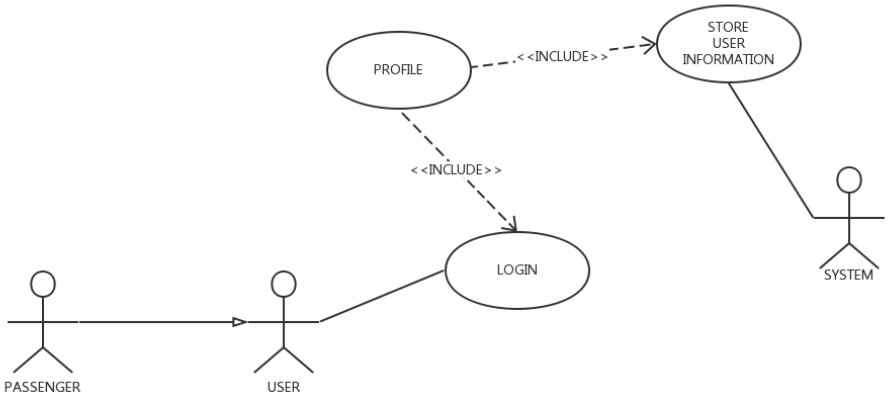
Name	Registration
Actors	User(Passenger), System
Entry Conditions	The user isnt registered.
Flow of events	1. The user clicks on Registration button to start the registration process. 2. The user fills in the form with : • e-mail • name • surname • cell phone number • tax code • password • verification password 3. User clicks on submit button. 4. The application will save the data in the DataBase. 5. User can now enter in the home page and he can make a reservation for a taxi service.
Exit conditions	The user clicks on cancel or ”submit” button.
Exceptions	• One or more mandatory fields are not valid. • Password not match • Password don’t meet the requirements • Email address is already associated to another user.

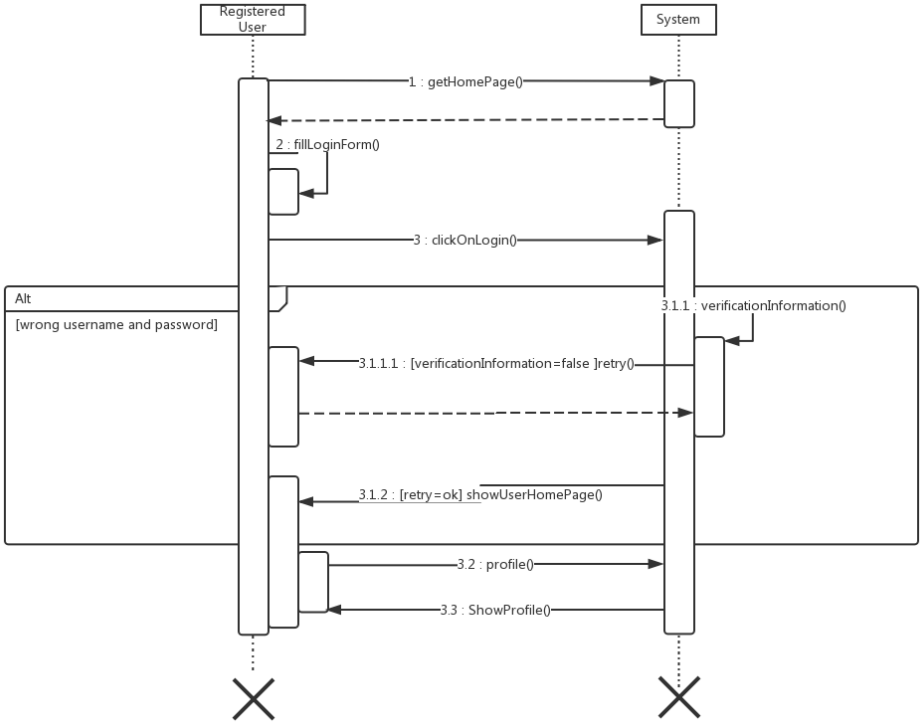

REGISTRATION
<<INCLUDE>>
STORE
USER
INFORMATION
<<INCLUDE>>
LOGIN
SYSTEM
PASSENGER
USER


User
System
1 : getHomePage()
2 : getRegistrationForm()
Alt :Loop
[verificationInformation=FALSE
and retry=TRUE]
3 : fillRegistrationForm()
3.1 : sendForm()
3.1.1 : verificationInformation()
3.1.1.1 : [verificationInformation=false]retry()
3.1.2 : [retry=ok] sendNewForm()
3.1.3 : createNewUser()
3.1.4 : welcomeNewUser()
3.1.5 : ShowUserHomePage()

3.9.4 Use Case: Profile

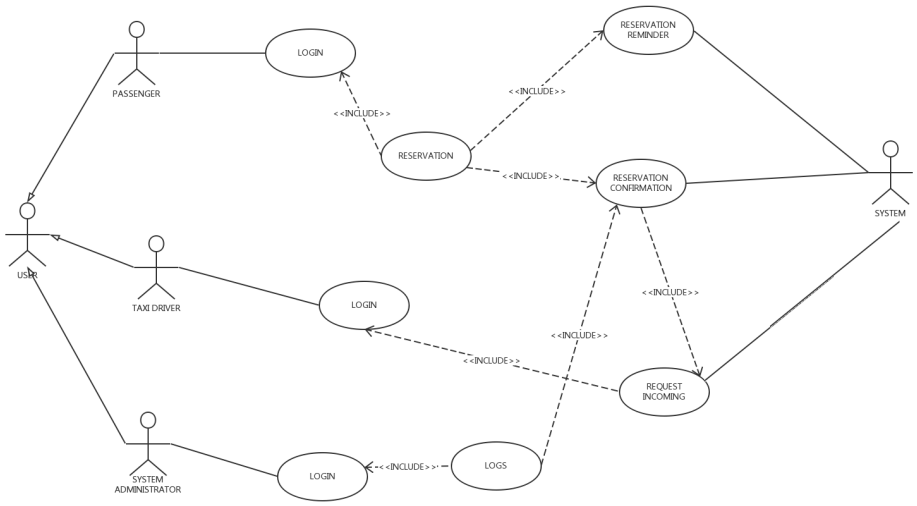
Name	Profile
Actors	User(Passenger), System
Entry Conditions	Registered user must be already logged in.
Flow of events	1. The user clicks on Profile button to see the private information that he inserts when he registered to the application. 2. If user wants to see the information, the system had to store them in the database.
Exit conditions	The user clicks on logout or another menu item
Exceptions	• The system can not show the user information.

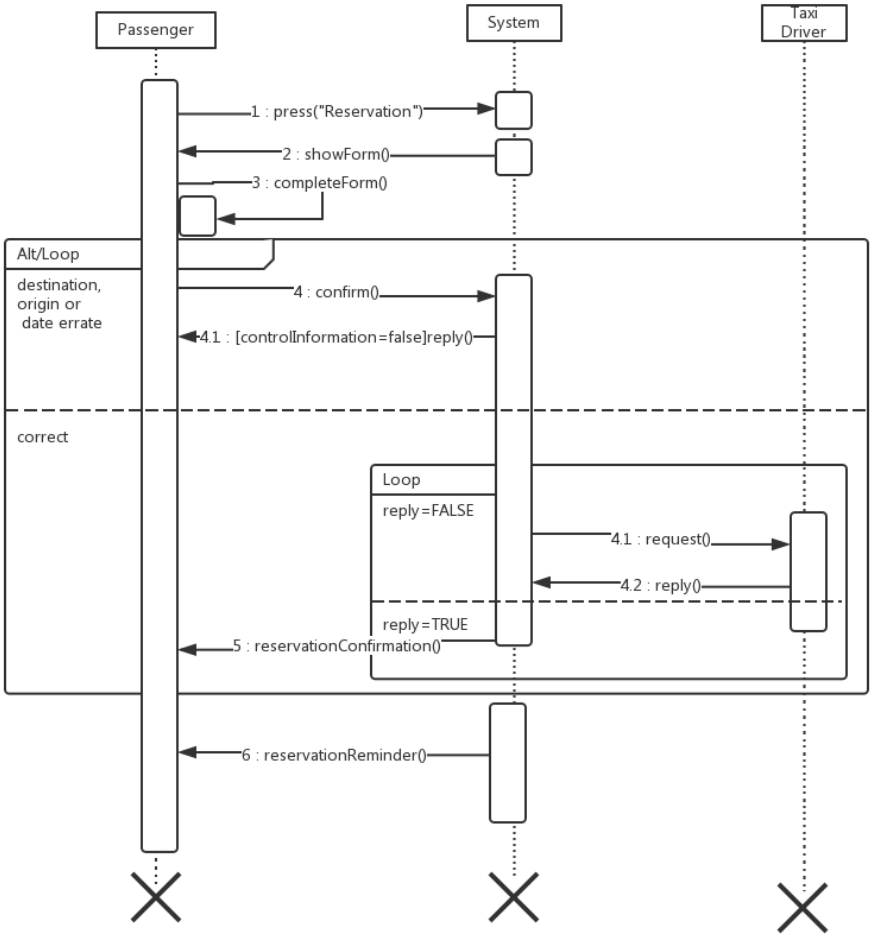



Registered User
System
1 : getHomePage()
2 : fillLoginForm()
3 : clickOnLogin()
Alt
[wrong username and password]
3.1.1 : verificationInformation()
3.1.1.1 : [verificationInformation=false]retry()
3.1.2 : [retry=ok] showUserHomePage()
3.2 : profile()
3.3 : ShowProfile()

3.9.5 Use Case: Reservation

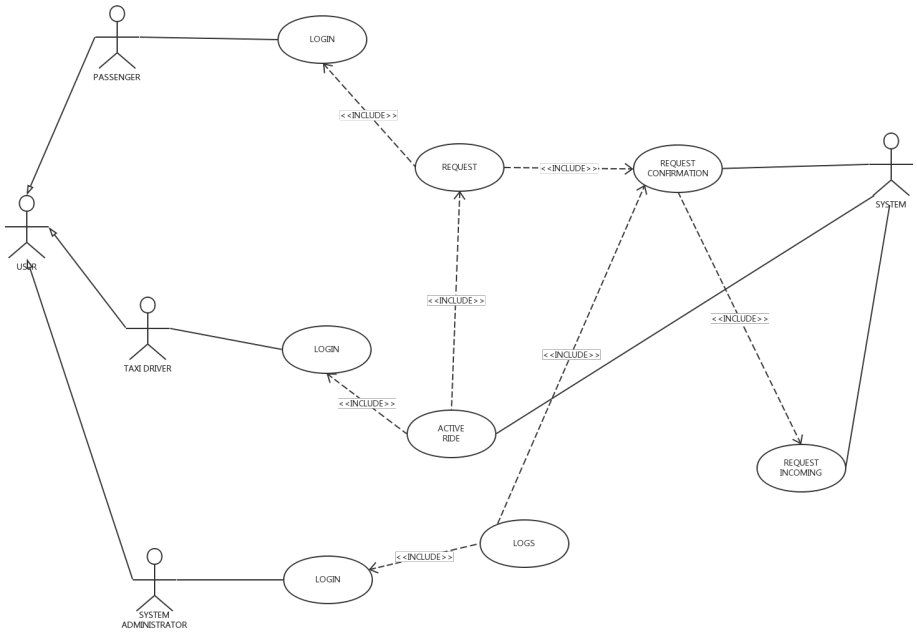
Name	Reservation
Actors	User(Passenger), System
Entry Conditions	Registered user must be already logged to the application.
Flow of events	1. Passenger clicks on Reservation button to request a taxi. 2. For requesting a taxi the user must insert position and destination of his ride. 3. The system sends an email to confirm a reservation. 4. The system sends a reminder e-mail to the user to remind him the taxi reservation two hours before the meeting time.
Exit conditions	The user clicks on logout or another menu item
Exceptions	• The system can not do a reservation because the request of reservation arrive after the limit time to do a reservation..

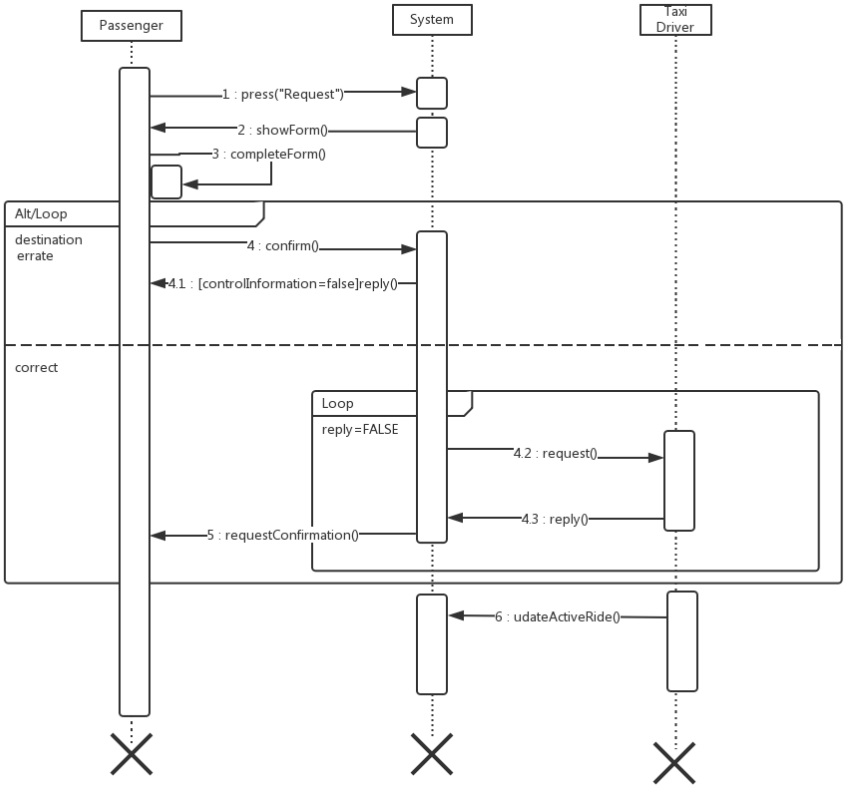



Passenger
System
Taxi Driver
1 : press("Reservation")
2 : showForm()
3 : completeForm()
Alt/Loop
destination, origin or date errate
4 : confirm()
4.1 : [controlInformation=false]reply()
correct
Loop
reply=FALSE
4.1 : request()
4.2 : reply()
reply=TRUE
5 : reservationConfirmation()
6 : reservationReminder()

3.9.6 Use Case: Request

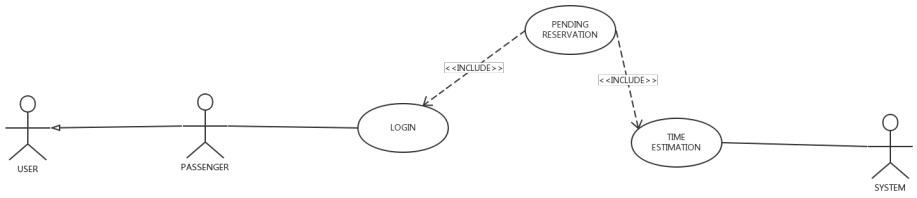
Name	Request
Actors	User(Passenger), System
Entry Conditions	Registered user must be already logged to the application.
Flow of events	1. Passenger clicks on Request button to request a taxi. 2. For requesting a taxi the user must insert his position. 3. The system sends an email to confirm a request when it has a reply of a one taxi driver available for a ride
Exit conditions	The user clicks on logout or another menu item
Exceptions	• The system can not answer to the request because no taxi driver are available to reply.

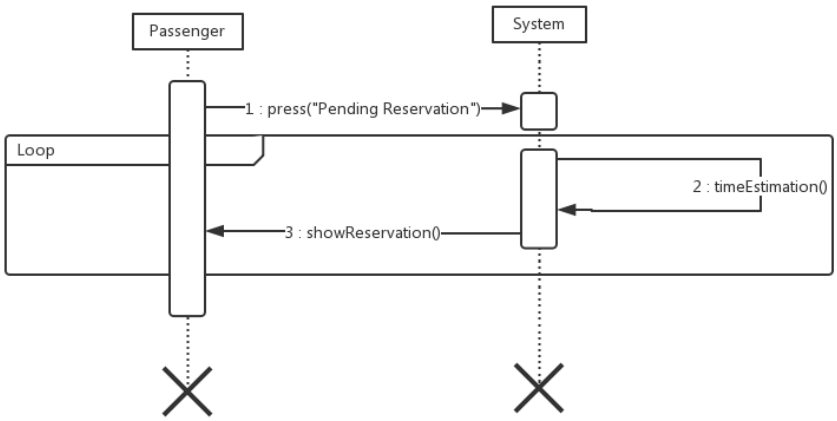



Passenger
System
Taxi Driver
1 : press("Request")
2 : showForm()
3 : completeForm()
Alt/Loop
destination errate
4 : confirm()
4.1 : [controlInformation=false]reply()
correct
Loop
reply=FALSE
4.2 : request()
4.3 : reply()
5 : requestConfirmation()
6 : udateActiveRide()

3.9.7 Use Case: Pending Reservation

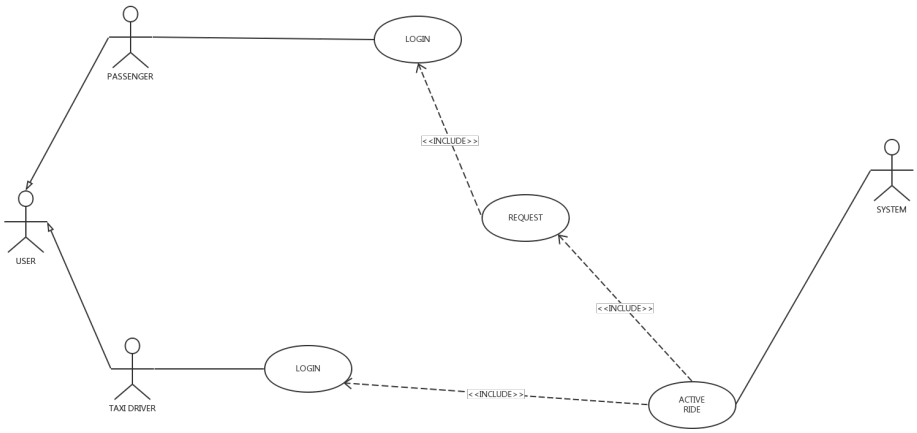
Name	Pending Reservation
Actors	User(Passenger), System
Entry Conditions	Registered user must be already logged to the application.
Flow of events	1. Passenger clicks on Pending Reservation button to see the estimated taxi arrival time of the reservation he made. 2. from here, the passenger can also cancel a reservation.
Exit conditions	The user clicks on logout or another menu item
Exceptions	• The user has no pending reservation

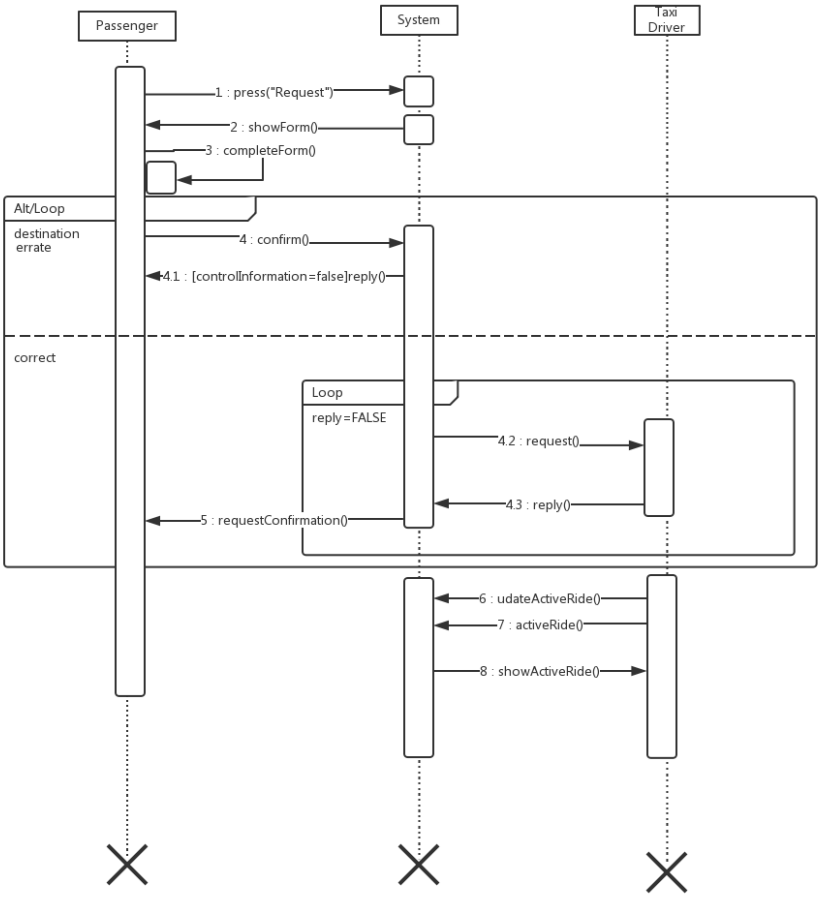




3.9.8 Use Case: Active Ride

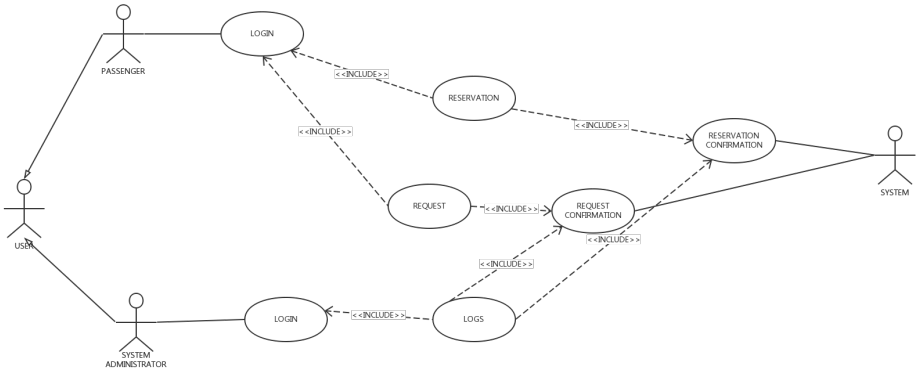
Name	Active Ride
Actors	User(Taxi Driver), System
Entry Conditions	Registered user must be already logged to the application and he accepted a request
Flow of events	1. Taxi driver clicks on Accept Request button to see the information of the ride.. 2. Taxi driver can release the ride to another taxi for problems that prevents him from reaching the passenger.
Exit conditions	The user clicks on logout or release the request successfully
Exceptions	• The taxi driver can not release his request, because there isn't any other taxi available in this moment.

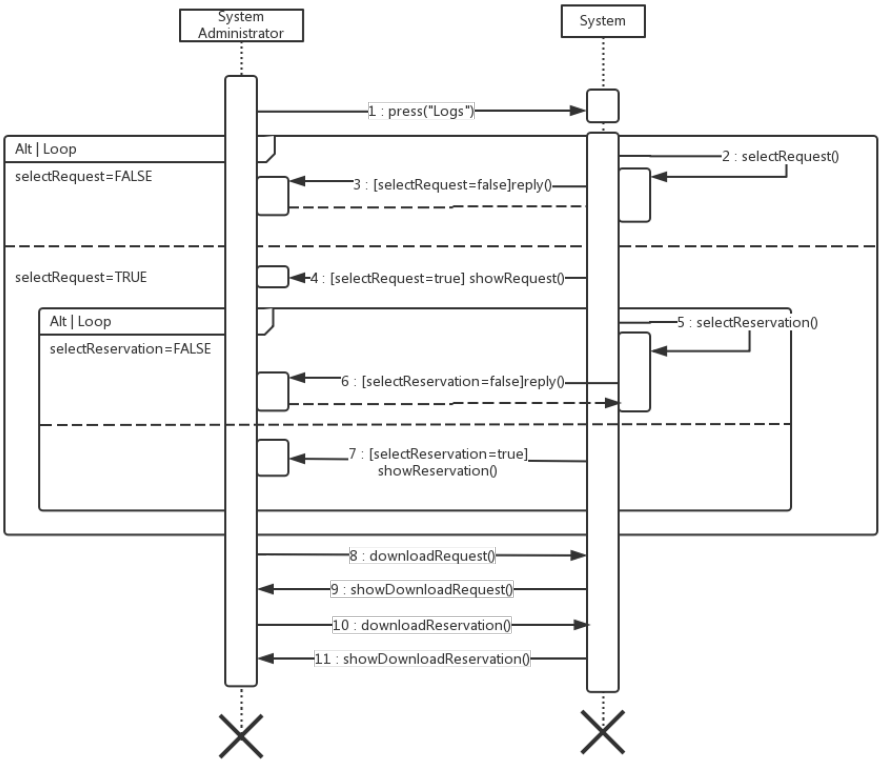



Passenger
System
Taxi Driver
1 : press("Request")
2 : showForm()
3 : completeForm()
Alt/Loop
destination errate
4 : confirm()
4.1 : [controlInformation=false]reply()
correct
Loop
reply=FALSE
4.2 : request()
4.3 : reply()
5 : requestConfirmation()
6 : udateActiveRide()
7 : activeRide()
8 : showActiveRide()

3.9.9 Use Case: Logs

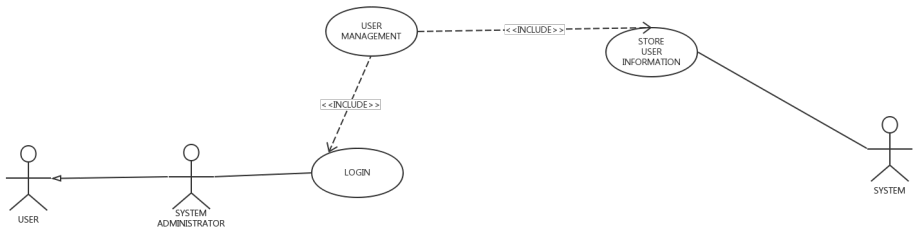
Name	Reservation
Actors	User(System Administrator), System
Entry Conditions	Registered user must be already logged to the application.
Flow of events	1. System administrator clicks on Logs button to view and download the logs associated to each accepted/refused request, all ride data such passenger account, departure/arrival date and location, and the taxi driver who made the ride.
Exit conditions	The user clicks on logout
Exceptions	• The system can not see all the information because not all of the information are available.

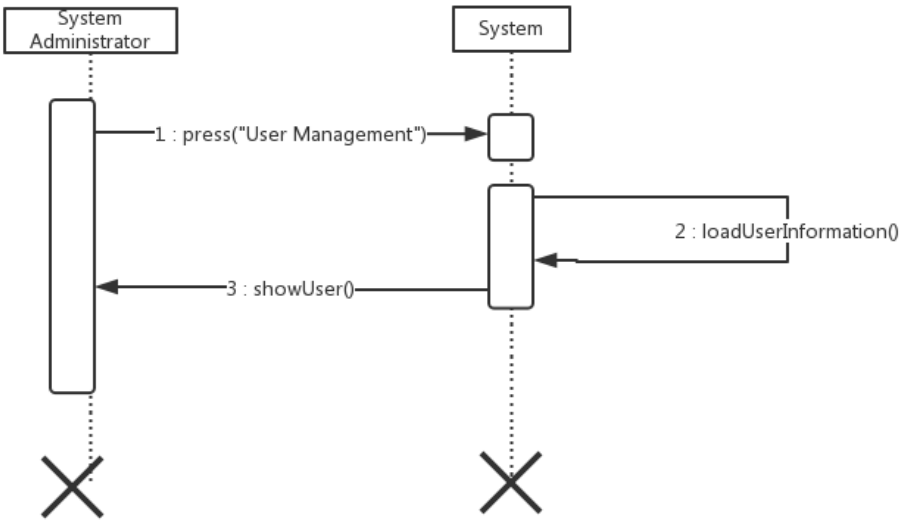



System Administrator
System
1 : press("Logs")
Alt | Loop
selectRequest=FALSE
2 : selectRequest()
3 : [selectRequest=false]reply()
selectRequest=TRUE
4 : [selectRequest=true] showRequest()
Alt | Loop
selectReservation=FALSE
5 : selectReservation()
6 : [selectReservation=false]reply()
7 : [selectReservation=true] showReservation()
8 : downloadRequest()
9 : showDownloadRequest()
10 : downloadReservation()
11 : showDownloadReservation()

3.9.10 Use Case: User Management

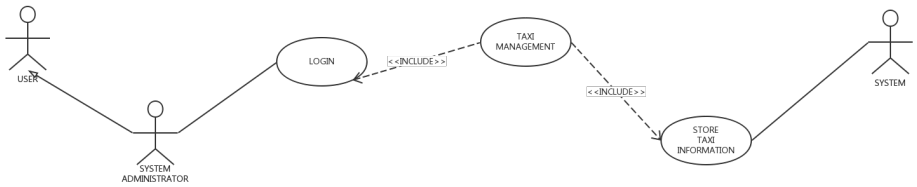
Name	Reservation
Actors	User(System Administrator), System
Entry Conditions	Registered user must be already logged to the application.
Flow of events	1. The system administrator can access user profiles to provide support to end users and reset their passwords by providing the user email. 2. The system had to update and store the passenger information.
Exit conditions	The user clicks on logout or another menu item.
Exceptions	• The system can not see all the information because not all of the information are available. • The user doesnt exists in the database

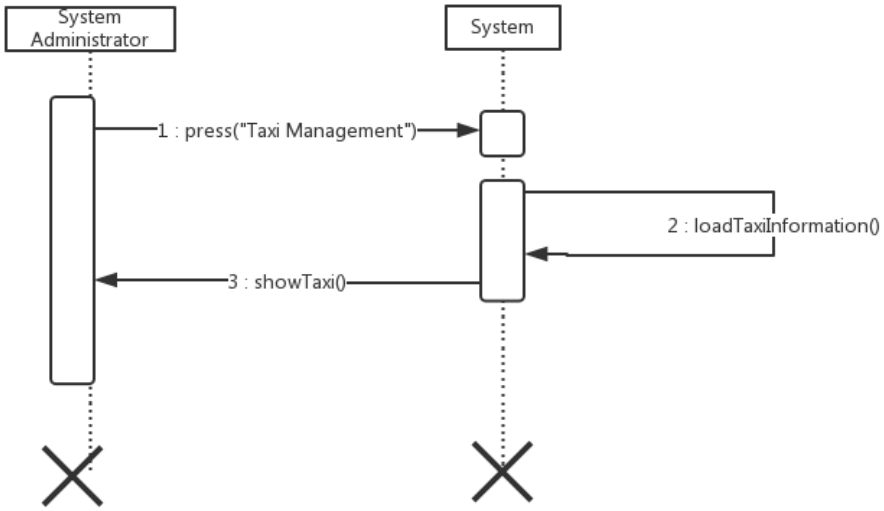



System Administrator
System
1 : press("User Management")
2 : loadUserInformation()
3 : showUser()

3.9.11 Use Case: Taxi Management

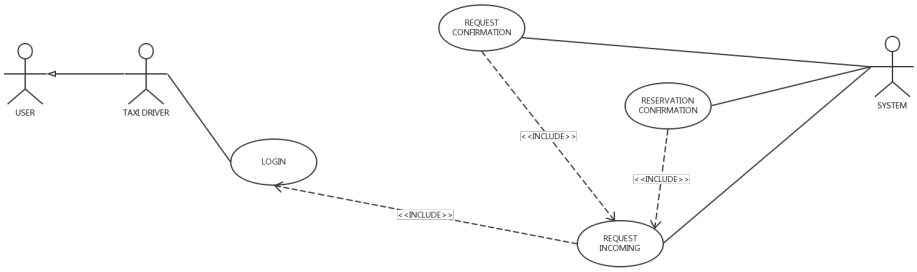
Name	Taxi Management
Actors	User(System Administrator), System
Entry Conditions	Registered user must be already logged to the application.
Flow of events	1. The system administrator can access the list of all active and inactive taxi accounts. 2. He can also add and remove taxi driver accounts or reset their passwords. 3. The system has to update and store the taxi driver information.
Exit conditions	The user clicks on logout or another menu item
Exceptions	• The system can not see all the information because not all of the information are available.

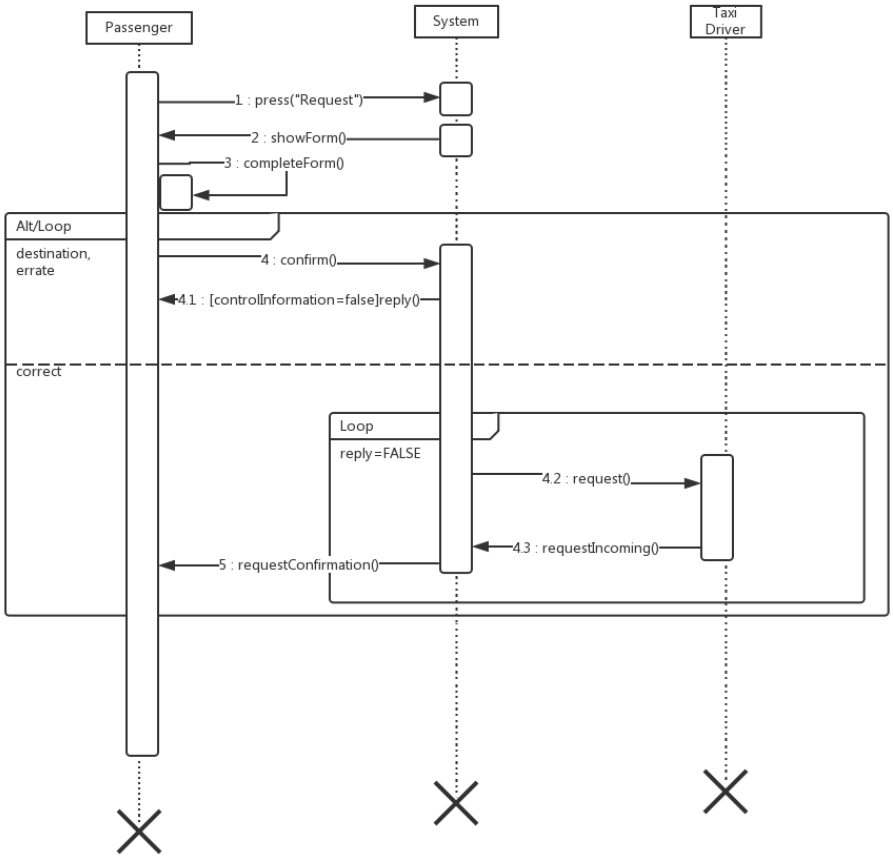



System Administrator
System
1 : press("Taxi Management")
2 : loadTaxiInformation()
3 : showTaxi()

3.9.12 Use Case: Request Incoming

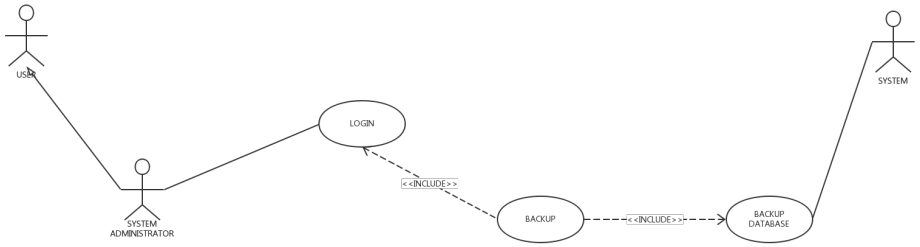
Name	Request Incoming
Actors	User(Taxi Driver), System
Entry Conditions	Registered user must be already logged to the application.
Flow of events	1. The taxi driver can accept or refuse a request that arrives from a passenger. 2. The system receives the reply of the taxi driver and inform the passenger if the taxi driver accepted
Exit conditions	The user clicks on logout or another menu item
Exceptions	• The system can not send a request to a taxi driver (Driver is offline or connection problem) • No response from the driver

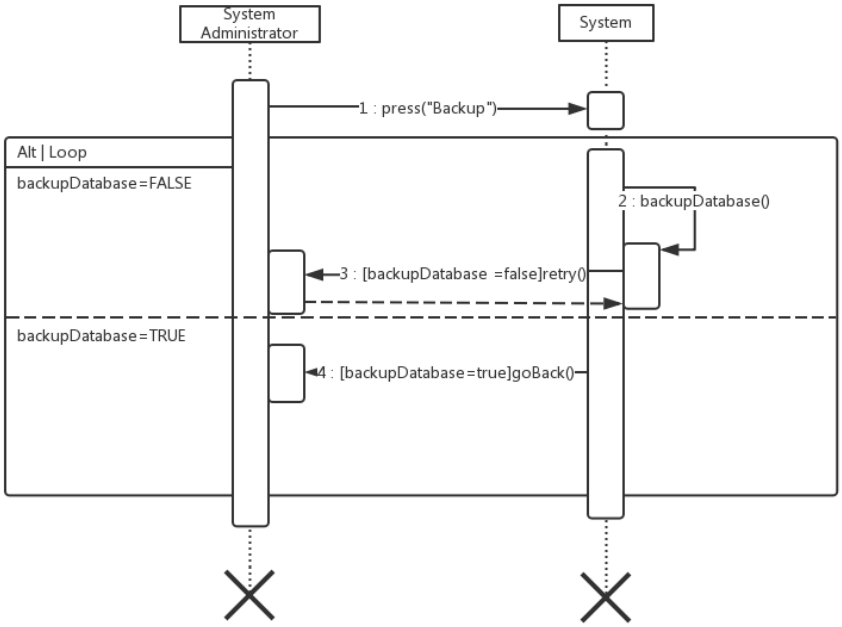



Passenger
System
Taxi Driver
1 : press("Request")
2 : showForm()
3 : completeForm()
Alt/Loop
destination, errate
4 : confirm()
4.1 : [controlInformation=false]reply()
correct
Loop
reply=FALSE
4.2 : request()
4.3 : requestIncoming()
5 : requestConfirmation()

3.9.13 Use Case: Backup

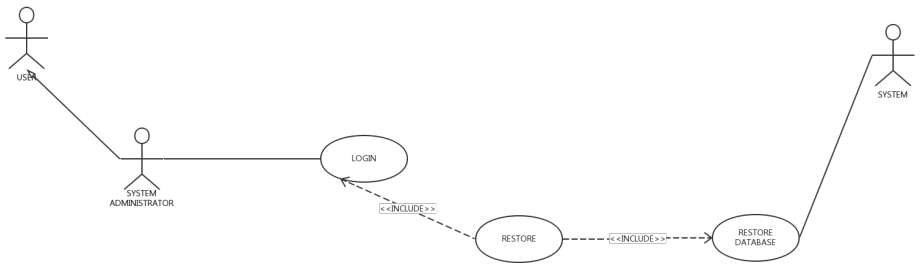
Name	Backup
Actors	User(System Administrator), System
Entry Conditions	Registered user must be already logged to the application.
Flow of events	1. The system administrator makes a security backup of the user information on the database. 2. The system stores the database in a backup device
Exit conditions	The user clicks on logout.
Exceptions	• No connection with the database • No enough space on device • DBMS failure

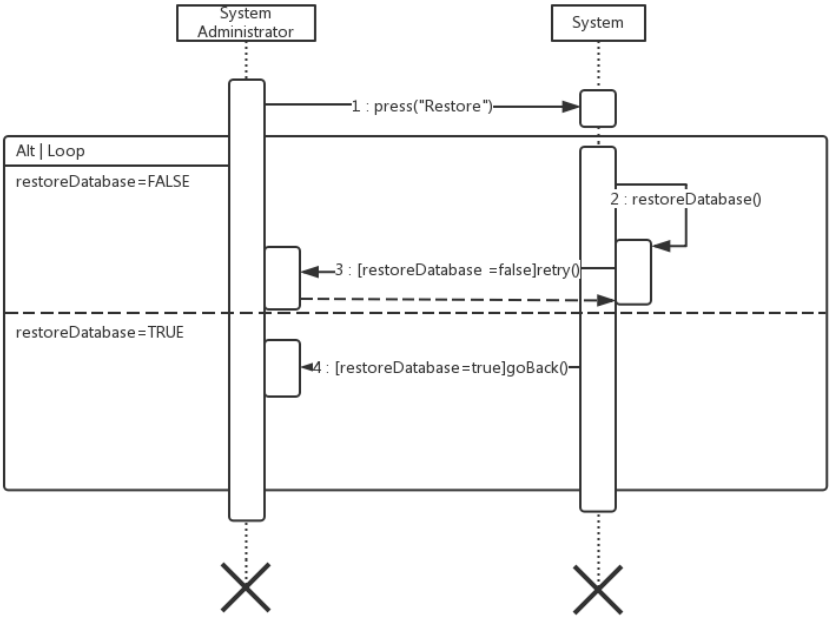



System Administrator
System
1 : press("Backup")
Alt | Loop
backupDatabase=FALSE
2 : backupDatabase()
3 : [backupDatabase =false]retry()
backupDatabase=TRUE
4 : [backupDatabase=true]goBack()

3.9.14 Use Case: Restore

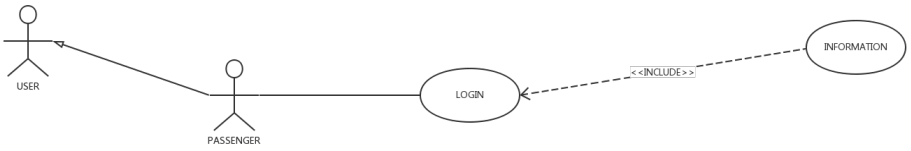
Name	Restore
Actors	User(System Administrator), System
Entry Conditions	Registered user must be already logged to the application.
Flow of events	1. The system administrator makes a restore of the user information on the database.
Exit conditions	The user clicks on logout or another menu item
Exceptions	• Corrupted Backup • No connection to database • Different data format

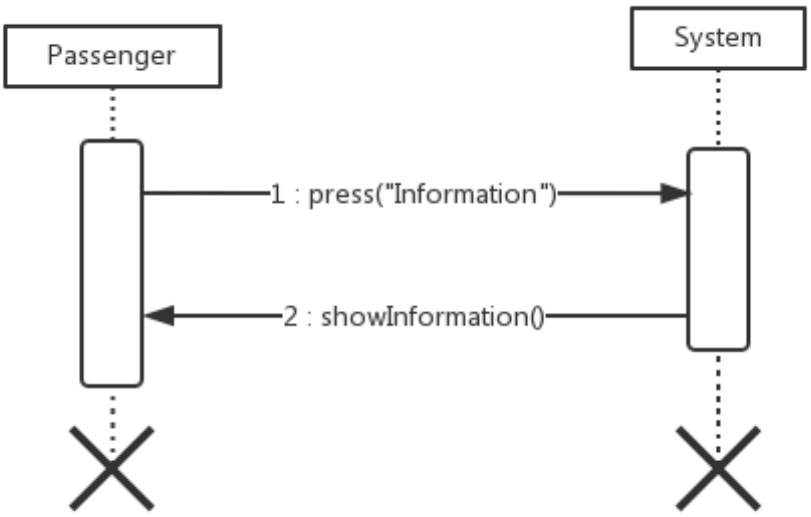



System Administrator
System
1 : press("Restore")
Alt | Loop
restoreDatabase=FALSE
2 : restoreDatabase()
3 : [restoreDatabase =false]retry()
restoreDatabase=TRUE
4 : [restoreDatabase=true]goBack()

3.9.15 Use Case: Information

Name	Information
Actors	User(Passenger), System
Entry Conditions	Registered user must be already logged to the application.
Flow of events	1. The passenger can see the information that can help him to request a taxi with the application.
Exit conditions	The user clicks on logout or another menu item
Exceptions	• Web server failure





3.10 Alloy

3.10.1 Definitions

```
//DEFINITION OF DATA TYPE
sig Strings{}
sig Integer{}
sig Zone{
nameZone : Strings,
map : one Map
}
sig City{
name : Strings,
popolation : Integer,
zone : some Zone
}

sig Map{
id_mappa : Strings,
}
sig Queue{
id_queue : Integer,
zone : one Zone,
taxi : some Taxi
}
sig Taxi{
id_Taxi : Integer,
registration_number : Strings,
registration_year : Integer,
}
sig Position{
street : one Strings,
number : Integer
}
sig Reservation{
date : Date,
origin : Position,
destination :Position,
user : one Passenger
}
sig Request{
date : Date,
position : Position,
user : one Passenger
```

```
}
sig Reply{
id_reply : Integer,
date : Date,
answer : Strings,
user : one Passenger,
taxi_driver : some Taxi_Driver
}
sig Date{
year : one Integer,
month : one Integer,
day : one Integer,
hour : one Integer,
minutes : one Integer
}
//ABSTRACT ENTITY
abstract sig User{
fiscal_code : Strings,
name : Strings,
surname : Strings,
phone_number : Integer,
email : Strings,
Password : Strings
}
//DEFINITION OF ABSTRACT ENTITY
sig Passenger extends User{}
sig Taxi_Driver extends User{
licence : Strings,
taxi : lone Taxi
}
sig System_Administrator extends User{}
sig System_Developer extends User{}
```

3.10.2 Facts

```
//FACTS
// no city with 2 zone equal
fact NoZone{
no c : City | some z1,z2 : Zone |z1!=z2 and c.zone=z1 and c.zone=z2
}
//no 2 taxi_driver with the same taxi
fact NoDoubleTaxi{
no t : Taxi| some t1,t2 : Taxi_Driver| t1!=t2 and t in t1.taxi and t in t2.taxi
}
```

```
//no 2 queue of the same zone
fact NoDoubleQueue{
no z : Zone| some q1,q2 : Queue|q1!=q2 and z in q1.zone and z in q2.zone
}
//no 2 queue with the same taxi
fact NoDoubleTaxiQueue{
no t : Taxi| some q1,q2 : Queue|q1!=q2 and t in q1.taxi and t in q2.taxi
}
// no 2 taxi with the same passenger
fact NoDoubleReservation{
no p : Passenger| some r1,r2 : Reservation|r1!=r2 and p in r1.user and p in r2.user
}
// no empty date
fact noEmptyDate{
all r : Reservation|(#r.date.hour=1)and(#r.date.minutes=1)
}
// no empty position
fact noEmptyPosition{
all r : Request|(#r.position.street=1)and(#r.position.number=1)
}
//no duplicate user
fact noDuplicateUser{
no disj u1,u2 : User | (u1.fiscal_code=u2.fiscal_code)
}
//no 2 different reply
fact noDoubleReply{
no p : Passenger|some r1,r2 : Reply|r1!=r2 and r1.user=p and r2.user=p and r1.date=r
}
// no duplicate password and email
fact noDuplicateEmail{
no disj u1,u2 : User | (u1.password=u2.password)and (u1.email=u2.email)
}
//no empty reply
fact noEmptyReply{
all r : Reply |(#r.answer=1)
}
//no empty reservation
fact noEmptyReservation{
all r : Reservation |(#r.date=1)and(#r.origin=1)and(#r.destination=1)
}
//no empty request
fact noEmptyReservation{
all r : Request |(#r.date=1)and(#r.position=1)
}
```

3.10.3 Assertions

```
//ASSERT
//check if the system don't do two reservation for the user
assert noDoubleReservation{
no disj r1,r2 : Reservation, p : Passenger | r1.user =p and r2.user=p and
r1.origin=r2.origin and r1.destination=r2.destination and r1.date=r2.date and r1=r2
}
check noDoubleReservation for 5
//check if the system don't do two request for the user
assert noDoubleRequest{
no disj r1,r2 : Request, p : Passenger | r1.user =p and r2.user=p and
r1.position=r2.position and r1.date=r2.date and r1=r2
}
check noDoubleRequest for 5
//check if the new user is insert
assert addUser{
all u,u1,u2 : User | (u not in u1)and addUser[u,u1,u2]implies (u in u2)
}
check addUser for 5
//check if city has two zone equal
assert NoZone{
no c : City | some z1,z2 : Zone |z1!=z2 and c.zone=z1 and c.zone=z2
}
check NoZone for 5
//check if the request is insert in the system
assert addRequest{
all r,r1,r2 : Request | (r not in r1)and addRequest[r,r1,r2]implies (r in r2)
}
check addRequest for 5
//check if the reservation is insert in the system
assert addReservation{
all r,r1,r2 : Reservation | (r not in r1)and addReservation[r,r1,r2]implies (r in r2
}
check addReservation for 5
//check to insert a taxi to a queue
assert addTaxiToQueue{
all t : Taxi, q,q1: Queue|(t not in q.taxi)and addTaxi[t,q,q1]implies(t in q1.taxi)
}
check addTaxiToQueue for 5
//check no double taxi for a taxi driver
assert NoDoubleTaxi{
no t : Taxi| some t1,t2 : Taxi_Driver| t1!=t2 and t in t1.taxi and t in t2.taxi
}
```

```
check NoDoubleTaxi for 5
```

3.10.4 Predicates

```
//PREDICATES
//pred for add user to the system
pred addUser(u,u1,u2 : Passenger){
#City=1
u not in u1 implies u2=u1+u
}
run addUser for 5
//pred for insert request of taxi
pred addRequest(r,r1,r2 : Request){
#City=1
r not in r1 implies r2=r1+r
}
run addRequest for 5
//pred for insert a reservation
pred addReservation(r,r1,r2 : Reservation){
#City=1
r not in r1 implies r2=r1+r
}
run addReservation for 5
//pred for insert a taxi in the queue
pred addTaxi(t : Taxi,q,q1 : Queue){
#City=1
t not in q.taxi implies q1.taxi=q.taxi+t
}
run addTaxi for 5
//pred show
pred show{
#City=1
#Passenger>1
#Taxi>1
#Taxi_Driver>1
}
run show for 10
```

3.10.5 Output

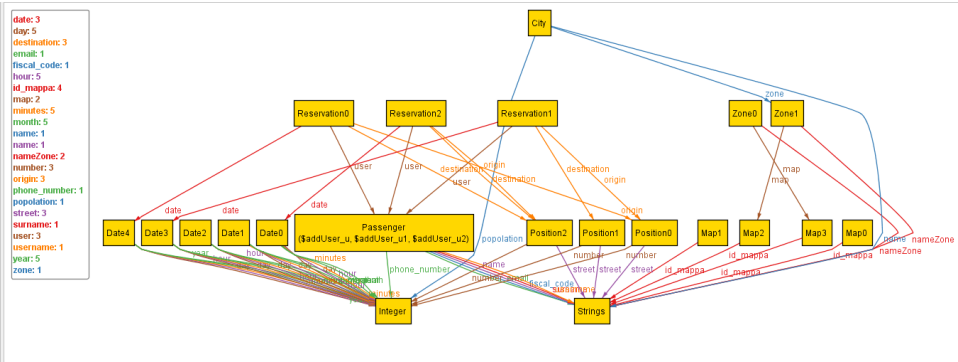
```
Executing "Check noDoubleReservation for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12567 vars. 1070 primary vars. 23687 clauses. 32ms.
No counterexample found. Assertion may be valid. 10ms.
```

```
Executing "Check noDoubleRequest for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12493 vars. 1070 primary vars. 23453 clauses. 40ms.
No counterexample found. Assertion may be valid. 0ms.
Executing "Check addUser for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12289 vars. 1070 primary vars. 22837 clauses. 43ms.
No counterexample found. Assertion may be valid. 20ms.
Executing "Check NoZone for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12390 vars. 1070 primary vars. 23011 clauses. 40ms.
No counterexample found. Assertion may be valid. 0ms.
Executing "Check addRequest for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12369 vars. 1070 primary vars. 22977 clauses. 40ms.
No counterexample found. Assertion may be valid. 0ms.
Executing "Check addReservation for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12369 vars. 1070 primary vars. 22977 clauses. 40ms.
No counterexample found. Assertion may be valid. 0ms.
Executing "Check addTaxiToQueue for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12429 vars. 1070 primary vars. 23187 clauses. 43ms.
No counterexample found. Assertion may be valid. 0ms.
Executing "Check NoDoubleTaxi for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12326 vars. 1070 primary vars. 22852 clauses. 40ms.
No counterexample found. Assertion may be valid. 20ms.
Executing "Run addUser for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12281 vars. 1070 primary vars. 22811 clauses. 42ms.
. found. Predicate is consistent. 73ms.
Executing "Run addRequest for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12361 vars. 1070 primary vars. 22951 clauses. 40ms.
. found. Predicate is consistent. 40ms.
Executing "Run addReservation for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12361 vars. 1070 primary vars. 22951 clauses. 52ms.
. found. Predicate is consistent. 40ms.
Executing "Run addTaxi for 5"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
12421 vars. 1070 primary vars. 23161 clauses. 73ms.
. found. Predicate is consistent. 42ms.
```

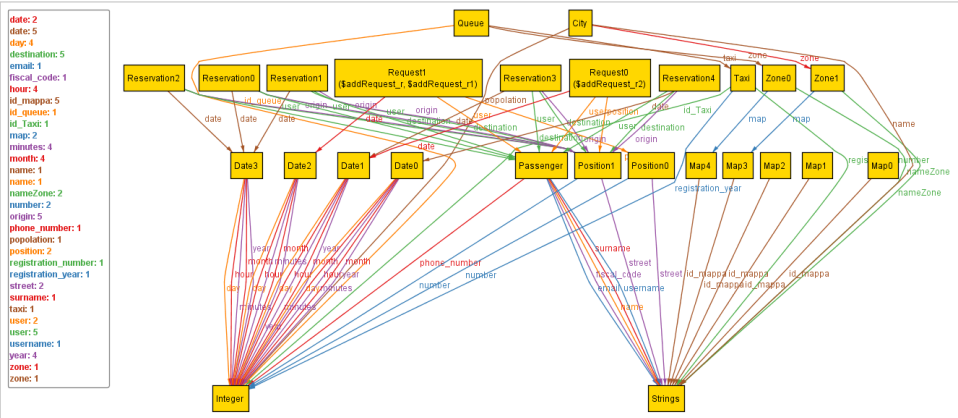
```
Executing "Run show for 10"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
53372 vars. 4060 primary vars. 109871 clauses. 338ms.
. found. Predicate is consistent. 517ms.
13 commands were executed.
```

3.10.6 Results

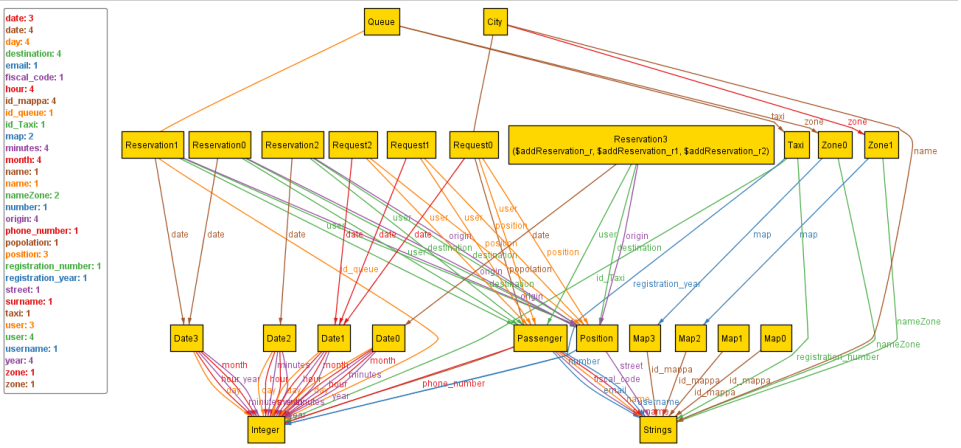
```
#1: No counterexample found. noDoubleReservation may be valid.
#2: No counterexample found. noDoubleRequest may be valid.
#3: No counterexample found. addUser may be valid.
#4: No counterexample found. NoZone may be valid.
#5: No counterexample found. addRequest may be valid.
#6: No counterexample found. addReservation may be valid.
#7: No counterexample found. addTaxiToQueue may be valid.
#8: No counterexample found. NoDoubleTaxi may be valid.

#9: Instance Found.addUser is consistent.
```



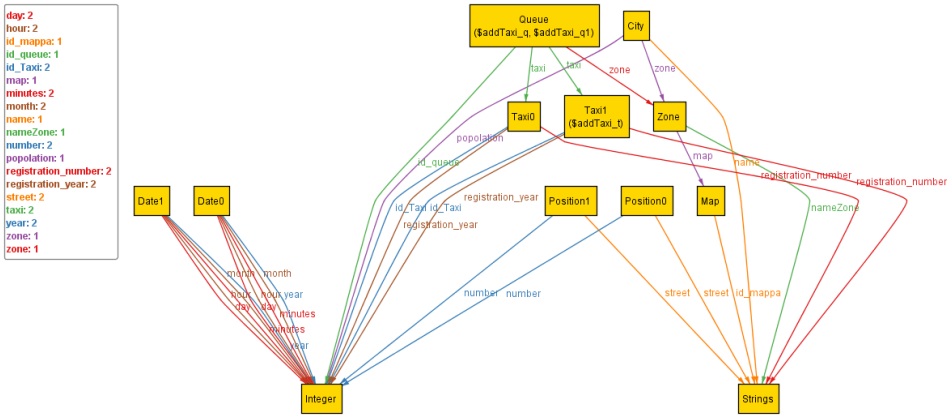

```
#10: Instance Found.addRequest is consistent.
```



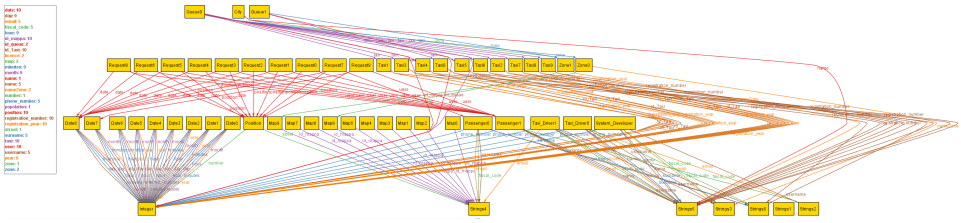
#11: Instance Found.addReservation is consistent.



#12: Instance Found .addTaxi is consistent.



#13: Instance Found .show is consistent.



4 Tools

For the redaction of this document, the following tools have been used:

- Alloy 4.2
- ProcessOn
- Google Drive
- TexStudio

5 Work Time

The work time for releasing the rev. 1.0 has been of

- Mattia Fontana: 42 Hours
- Edoardo Giacomello: 48 Hours