Software Engineering 2 Project
Requirements Analysis and Specification Document

PowerEnJoy

Team: Nico Montali, Enrico Fini

Contents

Introduction

1.1 Description of the problem

We are to project and develop a car-sharing system, PowerEnJoy. This service only relies on electric cars, so more attention is paid to the driver behaviour to reduce infrastructure cost, minimizing the number of necessary charging stations. The system has to interact primarily with the users that want to rent the car, but also with "**workers**", employees of the company PowerEni, that are responsible for the management of cars themselves. All the cars in the system are connected to the internet through a mobile data receiver and are equipped with several sensor (GPS, speedometer). The system must also provide an **administration console**, accessible by few users inside PowerEni company. This console provides administration functionalities (e.g. registering new workers, log the state of all the vehicles) and business statistics (e.g. users per day, total driven km). The main users of the system are

User Users interact with the system primarily through a mobile application, given that users usually rent cars while they are on the move. The application allows users, or "**drivers**", to reserve a car by presenting a list or map of all the near-by available cars (using the user GPS position). The application allows the user to effectively use the car (find where it is parked, open the doors, find useful informations, e.g. near-by charging stations). The system also provides a web portal, where the user can manage all his informations (payment settings, license, birthdate, view payment history etcetera). Users can access the system only through a registration, that can be done both on the web portal and on the mobile application. During the registration, the user need to provide payment informations and a valid driving license.

Workers Workers interact with the system through a specific mobile

application, released only to them. Workers are paid by the company to execute "tasks", that include:

- Recharging on-site a car that was not left in a charging station
- Move cars to more popular places to maximize usage
- Pick up a car in the case of an accident or normal maintenance

Administrator Interact with the system through a web application. Given that this is a main asset for the company, because it can effectively block the system, the security of this web application should be very high: possibly a web application accessible only inside PowerEni network with a 2 factor authentication. Once logged in, the administrator can perform these operations

- Register a new worker newly employed in the system, generating a username and a password (and unregister them)
- View the status of a single car (e.g. position, battery charge) and request exceptional tasks on it (not the ones generated automatically by the system)
- View statistics of the system to make better business decisions

1.2 Goals

Drivers goals

- [**GOAL1**] Log in the system.
- [**GOAL2**] Allow drivers to reserve a car up to one hour before they pick it up
- [**GOAL3**] Allow drivers to open the reserved car
- [**GOAL4**] Allow drivers to pay correctly for the service
- [**GOAL5**] Allow drivers to cancel a reservation

Workers goals

- [**GOAL50**] Log in the system
- [**GOAL51**] Dispatch tasks to workers
- [**GOAL52**] Allow workers to get informations about an assigned task

- [**GOAL53**] Allow workers to open and control the assigned car
- [**GOAL54**] Allow workers to keep track of the state of the task (assigned, in progress, done)

Administrator goals

- [**GOAL100**] Log in the administration console
- [**GOAL101**] Allow to register new workers
- [**GOAL102**] Allow to view the status of a specific vehicle
- [**GOAL103**] Allow to request an exceptional task on a specific vehicle

1.3 Domain assumptions

- Every car is equipped with some wireless communication system to connect it the the internet.
- Every car is equipped with GPS, speedometer, passenger presence sensors, engine state sensors, data is sent remotely to the system through the internet.
- Every car is equipped with a remotely controllable locking system.
- Every car is legally usable on public street (insurance, taxes, maintenance)
- Every time the user reserves a car, the payment is pre-authorized by an amount equal to the rent of a day [MAYBE SHOULD NOT BE WRITTEN HERE]
- Workers are registered in the system by the company itself. The worker application is only distributed to them not on public stores.
- Every user can reserve a car at a time.
- Every worker must have only one task assigned at a time.
- Drivers always respect the driving rules and are able to drive.
- Driver-provided license informations are assumed to be truthful
- Safe areas are considered already defined by the management system.

1.4 Glossary

Driver Client of the system, i.e. the one that uses the service, reserves and drives. Every Driver must provide personal informations (name, surname, email, birthdate, a valid license and payment information.

Ride A single usage of the service, starts when the user turn on the engine, stops when the car is locked inside a Safe Area;

Blocked driver A driver that has an invalid license or invalid payment coordinates

Worker Employee of the company, perform physical actions (moving, charging etc) on cars

Report Car issue reported by the Driver during a Drive. Will later be assigned to a Worker.

Task Piece of work assigned to a worker. Different type of tasks are described later. Every task assigns a single car to a single worker.

Administrator Employee of PowerEni that is in charge of the administration of the system through the administration console.

Drop off The act of leaving the car inside a safe area. This ends the service provided to the driver and effectively make the driver pay

Safe Area An area in which the Driver can park the car and stop the Service.

1.5 Stakeholders

The stakeholder is the PowerEni company, who needs the system to provide the service itself.

1.6 References

Actors

The principal actors involved in the system are

Driver Drivers are the main users, external to the PowerEni company, that want to use the PowerEnJoy service. They need to register to the service providing all the needed informations (payment settings and licenses to drive) through the application or the website.

Worker Employee of PowerEni, registered by the company itself through the administration console

Administrator Person inside PowerEni that has access to the administrator console.

Requirements

3.1 Functional Requirements

Given all the assumptions, domain properties and constraints from chapter 1, these functional requirements can be stated, listed by the goal they fulfill

- [**GOAL1**] Driver must log in the system
  - The system must check if the provided user exists and if the provided password matches
  - The system must provide further access only if the user is logged
- [**GOAL2**] Allow drivers to reserve a car up to one hour before they pick it up
  - The system must access driver GPS position or use a user-provided position to recommend the best vehicle to choose
  - The system must check if the user is logged in and if it is not a blocked driver
  - The system must check that the user has no other active reservation
  - The system must check payment service by pre-authorizing a payment of a fixed amount (to be defined)
  - The system must reserve the chosen vehicle, i.e. no other user is allowed to reserve the car until it is available again, once the payment is done. The vehicle is considered available again in case of: drop off, cancel registration, one hour passed from the reservation without any driver interaction
- [**GOAL3**] Allow drivers to open the reserved car

  - The system must check if the user is nearby the reserved car to open it
  - The system must be able to open the car remotely, only if the user effectively reserved it and is nearby

- [**GOAL4**] Allow drivers to pay correctly for the service
  - The system must log the whole drive by the driver (elapsed time, driven km)
  - The system must compute the amount the driver must pay for the service
  - The system must request a payment of the correct amount

- [**GOAL5**] Allow drivers to cancel a reservation
  - The system must be able to cancel a reservation if the driver asks it, given that the reservation has not exceed the 1 hour limit, without making him pay

- [**GOAL50**] Worker must log in the system
  - The system must check if the provided worker id exists and if the password matches
  - The system must provide further access only to logged workers

- [**GOAL51**] Dispatch tasks to workers
  - The system must be able to create a new task. The new task can be created from the administration console or generated automatically by a set of pre-defined rules (e.g. maintenance, usage balance)
  - The system must select the nearest available worker and assign him the task
  - The system must notify to the selected user that a new task has been assigned to him

- [**GOAL52**] Allow workers to get informations about an assigned task
  - The system must provide through the PowerEniWorker App all the needed information for the Worker. These include the GPS position of the car, the reported issue and additional information the user could have provided

- [**GOAL53**] Allow workers to open and control the assigned car
  - The System, during the assignment of a Task, gives to the chosen Worker the complete control of the car, i.e. normal actions (turn on/off engine, lock/unlock) but also priviliged actions (open hood, access maintenance interfaces, e.g. OBD)
- [**GOAL54**] Allow workers to update the state of the task (assigned, in progress, done)
  - The System must provide, through the PowerEniWorker App, buttons to change the state of the assigned Task, also with an optional description. Task can be completed or unfixable.
- [**GOAL100**] Log in the administration console
  - The system must check if the provided Admin ID exists and the password matches
  - The system must provide further access only to logged Admins
- [**GOAL101**] Allow to register new workers
  - The System must provide a form to fill in the Admin Portal to insert all the information of the new Worker
  - The System must create a new Worker account once the previous information are provided.
  - The System must print the badge used for Worker authentication with the unique Worker signature on it.
- [**GOAL102**] Allow to view the status of a specific vehicle
  - The System must provide an interface to select a car among all.
  - The System must provide analytics and current status of the chosen car.
- [**GOAL103**] Allow to request an exceptional task on a specific vehicle
  - The System must provide an interface to select a car among all.
  - The System must provide a button to create a new Task, along with specification of the Task itself, inserted manually by the Admin.
  - The System assigns the newly generated Task to the most suitable available Worker as in a normal Task

3.2 Non-functional requirements

3.2.1 Usability

The target user for the service is heterogeneous. It ranges from students, who have a deep understanding of technology, to senior citizens, who find it more difficult. Hence the user interface should be simple and understandable for everyone.

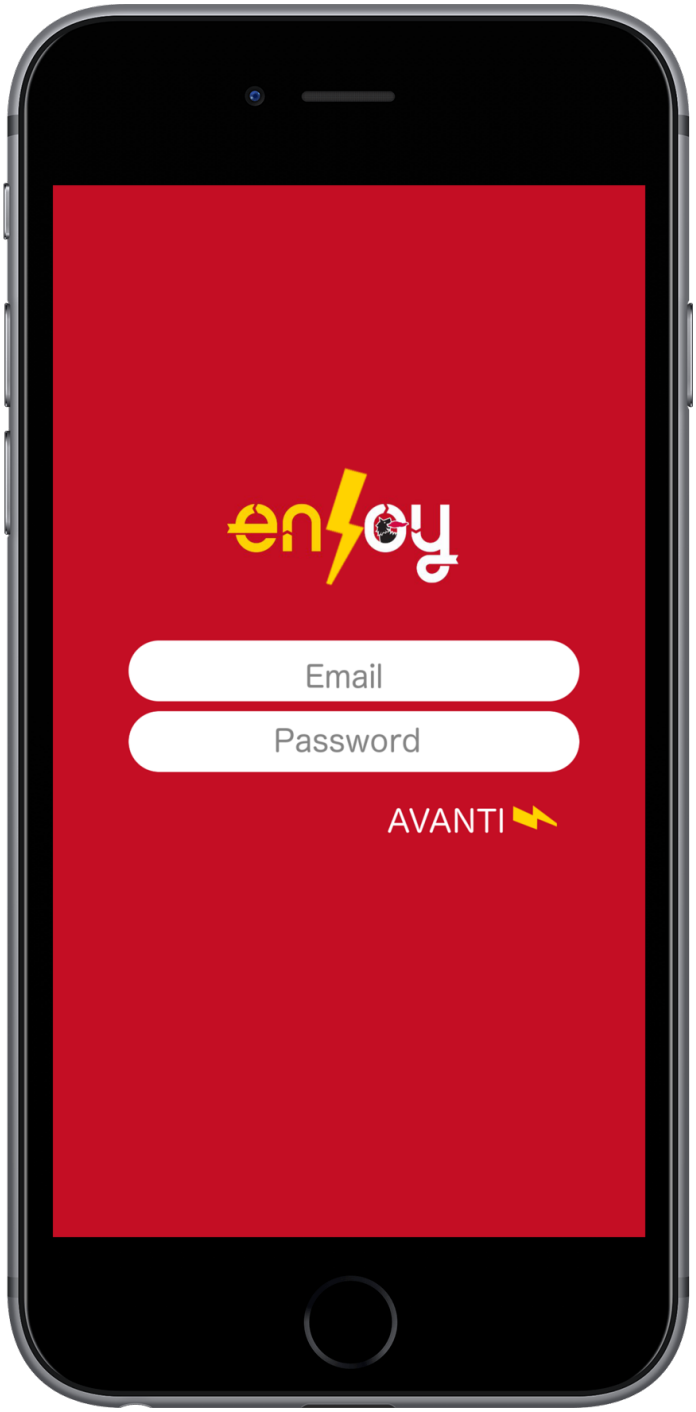
3.2.2 Privacy and Security

The system should be able to protect users' personal data, in particular login credentials, payment information and location of the Drivers, which are sensible information nowadays. At the same time we need to find a trade-off between security of our assets (cars and vans) and accessibility for the final user. Therefore our goal is to develop a vulnerability free System using good programming practices.

3.2.3 Architectural consideration

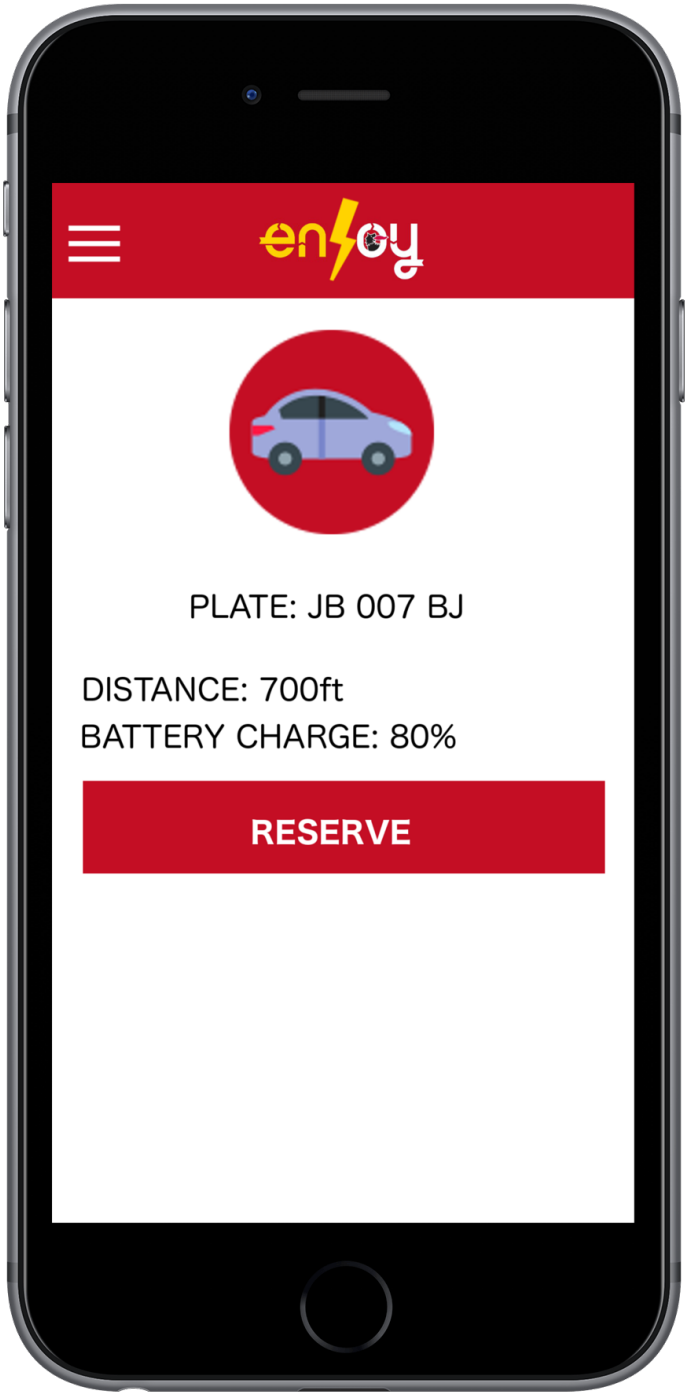
The fundamental part of the system is obviously the server-side application, that provides API to interact with the System. This server-side application will be written in NodeJS and will use a PostgreSQL DB to store and retrieve data. This application will be deployed on Amazon AWS, in a replicated, distributed and load-balanced way (to grant availability, fault tolerance, scalability), with the DB on a dedicated cluster of servers. Three endpoints will be provided, for the 3 categories of user that the System has: an endpoint for the Drivers, one for the Workers and one for the Admins. The Admin endpoint, for security reasons, will only be accessible from the inside PowerEni network (setting AWS firewall rules). The web-portal (both for the Driver and for the Admin), will be provided by the same server-side application: it will be built using EJS templates inside NodeJS and JS for client-side scripting. The mobile applications, both for Drivers and Workers, will be built natively for iOS, Android and Windows Phone. This applications will communicate with the backend using JSON format.

3.2.4 Mobile application



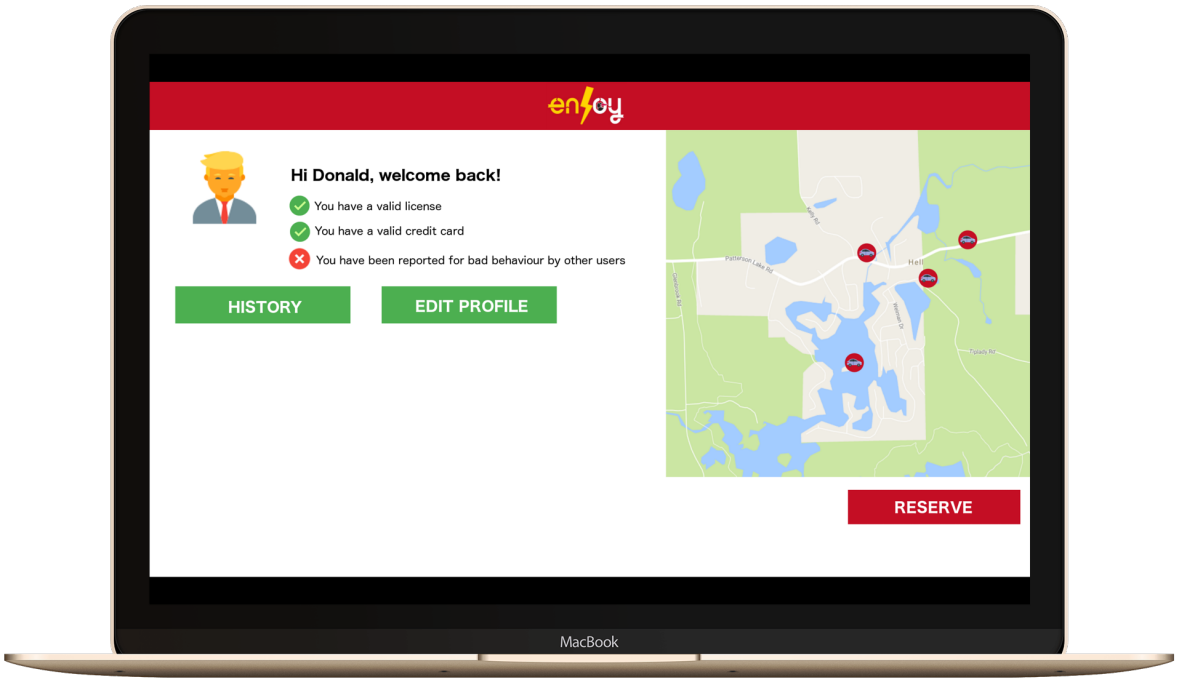
View 1: login

View 2: map



View 3: reservation

3.2.5 Web portal



Main view

Scenario Identification

Here we list some possible scenarios of usage of this service:

Scenario 1

It's about 9 in the morning and Ned, a well-known consultant, is leaving home to go to an important meeting. When he gets in his car and tries to start the engine the car refuses to turn on. Ned is very concerned because the meeting starts at 10 o'clock. Then he remembers of an advertising seen the day before, so he takes out his phone and looks for PowerEnJoy, a new, reliable car sharing service, on the App Store. He is in a hurry but he discovers that the registration mechanism is fast and intuitive. Since the system can recognise and extract information from pictures, he only needs to take a photo of his credit card and driving licence and that's it, few seconds later he is ready to make a reservation using the map of near-by available cars on the application. He is really lucky, the nearest car is only 100m away from his position. When he reaches it he can unlock it directly with his phone. Thanks to this new service Ned managed to arrive on time to the meeting.

Scenario 2

Anna is a lawyer and every week she must attend some court hearings at the central court of her city. Unfortunately, the central zone of the town has been declared off-limits to traffic to reduce air pollution. Since she is always carrying with her some important documents she cannot use public transport services. Recently she discovered that PowerEnJoy cars, being electric, can enter the city center. Now she uses PowerEnJoy portal every week to find and reserve cars near her house whenever she needs to go to the court for a hearing. Moreover, when she needs to go back home she uses PowerEnjoy application on her Android smartphone.

Scenario 3

Riccardo is an exchange student who wants to go and see the basketball match at the Arena tonight. Since he has no car he wants to use PowerEnJoy car sharing service to get there. In order to reduce the cost of the ride he decides to pick up some friends on his way. When the new passengers get into the car the seats automatically detect them and the system applies a 10% discount. When he arrives near the Arena the screen on the car notifies Riccardo the presence of a recharging station close to his position. Hence he decides to park at the station and plug the car to the power station. Then, with his phone, he terminates the rental and the application informs him that he obtained a 40% total discount. He is really happy with the service and decides to use it also to go back home.

Scenario 4

Roman is the system administrator of PowerEni. Today, as usual, he is working at PowerEni Headquarter. During his workday he receives a request from the human resources department saying that a new worker, Karim, has been hired. Roman is required to register the new employee in the system. To do that he opens an ad hoc tool on his computer. The tool provides an intuitive user interface for the management of workers' accounts. After having processed the information of the new worker the system prints his new badge and writes a unique NFC signature on it. Few minutes later Karim arrives at the Headquarter and picks up his badge. Then he downloads PowerEniWorker application, creates an account, activates it using the NFC tag and finally registers his fingerprint for future usage. Karim is ready to start working for PowerEni

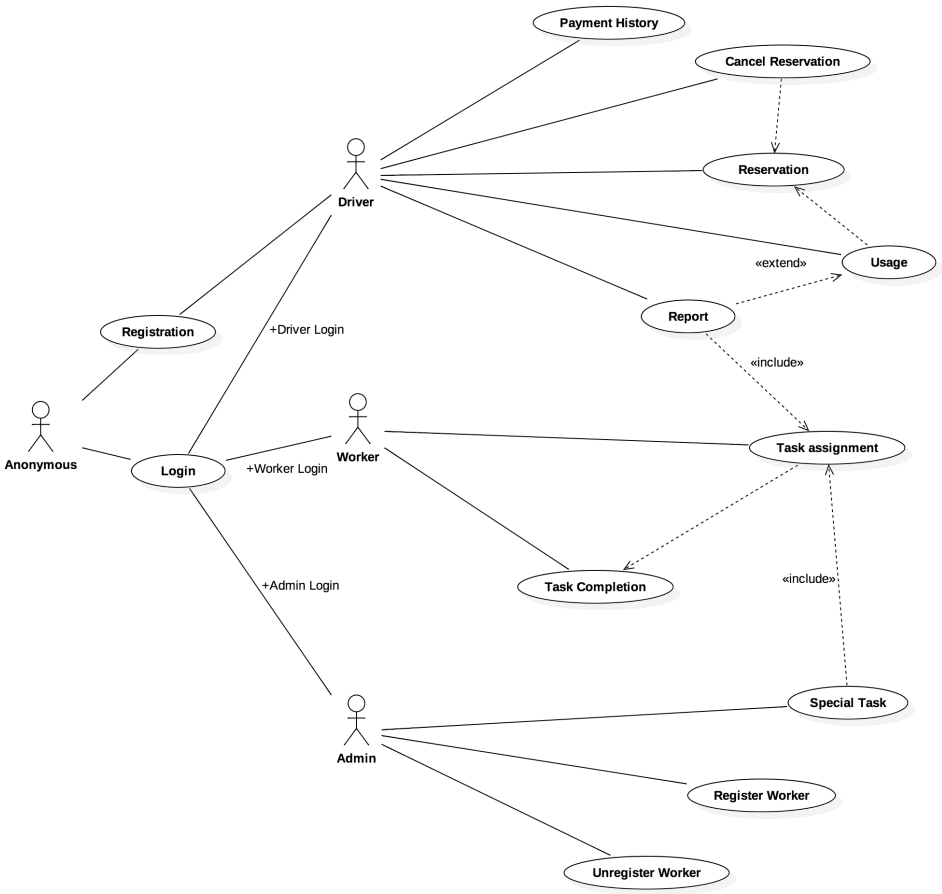
Scenario 5

Karim is a worker at PowerEni. At the beginning of his workday he opens his PowerEniWorker application and logs in using either his fingerprint or the NFC signature of his identification badge. Automatically the system marks his status as "available" and starts computing a suitable personalised task for the worker. Meanwhile, not far away from Karim's position, Alan, a user, is reporting a flat tire on his car using the dedicated button on PowerEnJoy application. The system terminates Alan's rental and marks the car as "to be repaired". Since Karim is the nearest available worker, the server sends him a push notification with all the information

needed to reach and repair Alan's car. Therefore Karim accepts the task, changing his status to "busy", and gets in his electric van (provided by PowerEni). Once he reaches the faulty car he fixes the flat tire. Finally, always using PowerEniWorker application, he marks the task as "done". The system reacts assigning a new task to Karim and changing the status of the car to "available".

Use Cases

5.1 UML use case diagram



5.2 Driver use cases

Here we list relevant use cases for the Driver:

Driver registration

Name: Driver registration
Actors: Driver
Entry conditions:

- The Driver has already downloaded the application or he/she is on the web portal.

Flow of events:

- The Driver provides an email and a password;
- The Driver provides information about himself/herself (full name, date of birth, address, sex);
- The Driver clicks the sign up button;
- The System verifies if the email is valid and not already in use;
- The System verifies if the password matches the rules (lenght, special characters);
- The System asks for driving license and payment information;
- The Driver uploads pictures of his license and credit card;
- The System shows a "success" message and sends a confirmation email.

Exit conditions: the Driver is registered and logged into the system.
Exceptions:

- If submitted data is not valid the System asks for them again.

Driver login

Name: Driver login
Actors: Driver
Entry conditions:

- The Driver has already downloaded the application or he/she is on the web portal.

Flow of events:

- The Driver provides username and password;
- The Driver clicks the login button;
- The System verifies the credentials.

Exit conditions: the Driver is logged into the system.
Exceptions:

- If credentials are wrong the system asks for them again.

Car reservation

Name: Car reservation
Actors: Driver
Entry conditions:

- The Driver is logged in;
- The Driver must not be a Blocked Driver.

Flow of events:

- The System automatically fetches the Driver's position and presents a list of available nearby cars;
- The Driver selects one of the available cars;
- The System pre-authorizes the fixed amount from the credit card and marks the car as reserved;
- The System shows a "success" message and sends an email with reservation information (car location, car plate, cancel reservation link)

Exit conditions:

- The Driver has a reservation and cannot reserve other cars
- The car is not available for other Drivers.

Exceptions:

- If the Driver does not have enough money the reservation in not successful.

Driver uses the service

Name: Driver uses the service
Actors: Driver
Entry conditions:

- The Driver has already reserved a car.

Flow of events:

- The System waits till the user is nearby the car then enables the open button on the application;
- The Driver opens the car by pressing the button;
- The Driver gets inside the car, turns it on and begins to drive;
- The System logs the route and the duration of the drive;
- The System presents to the Driver his current due amount and all the nearby recharging stations;
- When the Driver stops the car in a Safe Area and he gets outside, the car waits 30 seconds and then locks itself automatically. The Ride is terminated.
- The System calculates the correct amount to be paid (depending on discount policies) and charges the Driver's credit card accordingly.

Exit conditions: the Driver got to the final destination and the car is marked again as "available".
Exceptions:

- If the Driver does not start his ride within one hour, the reservation expires and his credit card is charged 1 Euro and the car status is reset to "available".

Cancel reservation

Name: Cancel reservation
Actors: Driver
Entry conditions:

- The Driver has already reserved a car;
- The reservation happened within one hour.

Flow of events:

- The user clicks either on the "cancel reservation" button on the application or on the "cancel reservation" link provided in the reservation email;
- The System cancels the reservation without charging the user.

Exit conditions:

- The car is marked as "available" again;
- The Driver can reserve cars again.

Exceptions: there are no exceptions for this use case.

Report a problem

Name: Report a problem
Actors: Driver
Entry conditions:

- The Driver started his ride;
- The car has a problem.

Flow of events:

- The Driver clicks on the "report" button and specifies the type of problem;
- The ride terminates and the Driver pays accordingly;
- The Driver gets outside the car.

Exit conditions:

- A Task is generated and is assigned to the most suitable available Worker;
- The Driver can reserve cars again.

Exceptions: there are no exceptions for this use case.

View payment history

Name: View payment history
Actors: Driver
Entry conditions:

- The Driver is logged into the System.

Flow of events:

- The Driver clicks on the Personal History button inside the web portal
- The System presents a detailed log of all the previous reservation and payments.

Exit conditions:

- The Driver is informed on his/her history.

Exceptions: there are no exceptions for this use case.

5.3 Worker use cases

Here we list relevant use cases for the Worker:

Worker login

Name: Worker login
Actors: Worker
Entry conditions:

- The Worker has already downloaded the application;
- The Worker has been registered by an Admin.

Flow of events:

- The Worker places the NFC badge on his phone;

Exit conditions:

- The Worker is logged into the system and his workday begins.

Exceptions: there are no exceptions for this use case.

Task assignment

Name: Task assignment
Actors: Worker
Entry conditions:

- The Worker is logged into the System;
- The Worker is available;
- The System generated a Task which has not been assigned yet.

Flow of events:

- The System chooses the most suitable available Worker depending on its policies (distance, workload, van batteries);
- The System notifies with a push notification the selected Worker and sends him all the needed information (location, type of task, car info).

Exit conditions:

- The Task is assigned to the worker;
- The Worker is marked as busy.

Exceptions: there are no exceptions for this use case.

Task completion

Name: Task completion
Actors: Worker
Entry conditions:

- The Worker has been assigned a Task;
- The Worker completed the task;

Flow of events:

- The Worker informs the System that the work has been completed using the application;

Exit conditions:

- The Worker is marked as available again;

Exceptions:

- If the Worker did not manage to complete the Task he closes the Task providing a description. The System Admin will manage the situation manually (e.g. calling a tow truck).

5.4 Admin use cases

Here we list relevant use cases for the Admin:

Admin login

Name: Admin login
Actors: Admin
Entry conditions: there are no entry conditions for this use case.
Flow of events:

- The Admin provides his credentials to the System

Exit conditions:

- The Admin is logged into the System.

Exceptions: there are no exceptions for this use case.

Register employee

Name: Register employee
Actors: Admin
Entry conditions:

- The Admin is logged into the System.

Flow of events:

- The Admin provides the new employee's information to the System (full name, ID);
- The System prints the new Worker's badge and writes a unique NFC signature on it.

Exit conditions:

- The new Worker is registered.

Exceptions: there are no exceptions for this use case.

Unregister employee

Name: unregister employee
Actors: Admin
Entry conditions:

- The Worker has been fired or reassigned.

Flow of events:

- The Admin selects the employee to be unregistered.

Exit conditions:

- The Worker is no more registered and his credentials are invalidated.

Exceptions: there are no exceptions for this use case.

Get analytics

Name: get analytics
Actors: Admin
Entry conditions:

- The Admin is logged in.

Flow of events:

- The Admin selects the car or the worker to get analytics of;

Exit conditions:

- The System shows detailed analytics

Exceptions: there are no exceptions for this use case.

Request special Task

Name: request special Task
Actors: Admin
Entry conditions:

- The Admin is logged in.

Flow of events:

- The Admin writes a description of the special Task, including the car ID.

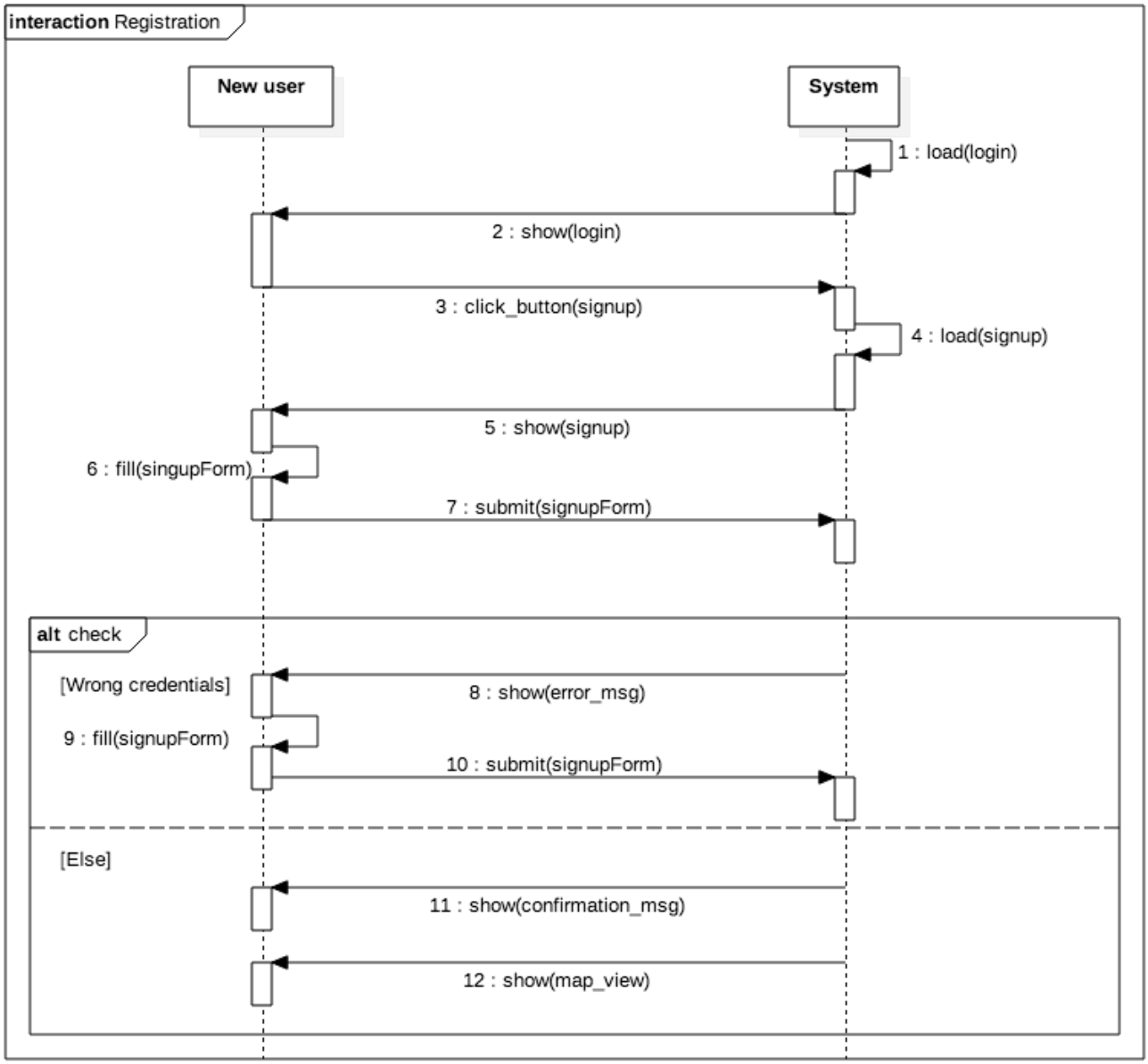
Exit conditions:

- A Worker is notified.

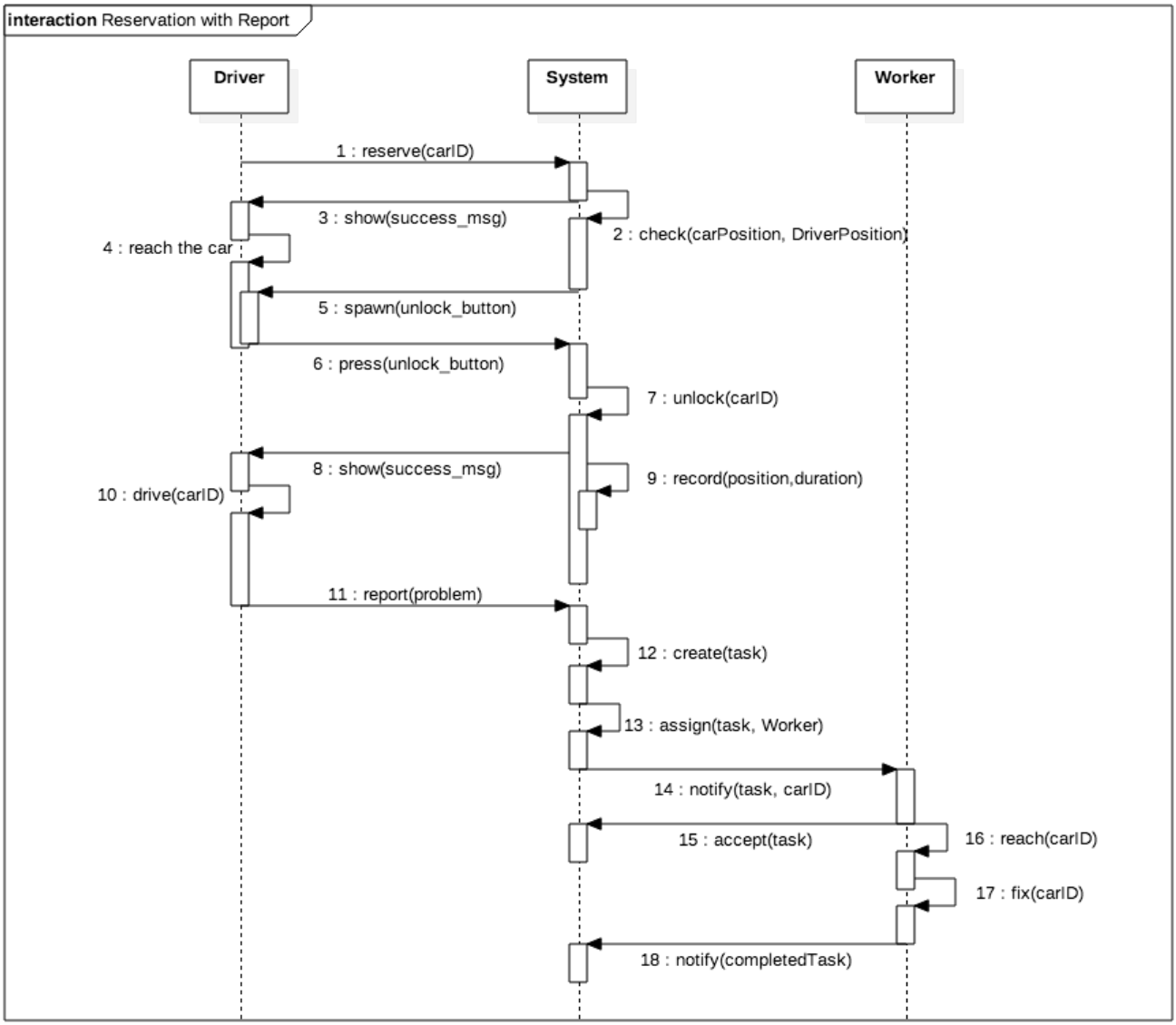
Exceptions: there are no exceptions for this use case.

5.5 Sequence diagrams

5.5.1 Registration



5.5.2 Reservation with Report



Model Coherence Analysis

In this chapter we will present the Alloy code and generated graphs used to check model consistency. The specification mostly relies on facts.

6.1 Alloy code

```
sig Vehicle {}

sig Email {}

sig License {}

sig PaymentInfo{}

sig Driver {
        email : Email
}

sig ValidDriver extends Driver {
        license : one License,
        paymentinfo : one PaymentInfo
}

sig Reservation {
        driver : one ValidDriver,
        vehicle : one Vehicle
}

fact uniqueDriverInfos {

        all d1,d2: Driver |
```

```
        d1 != d2 => d1.email != d2.email

        all d1,d2: ValidDriver |
        d1 != d2 => d1.license != d2.license

        all d1,d2: ValidDriver |
        d1 != d2 => d1.paymentinfo != d2.paymentinfo

        ValidDriver.license = License

        ValidDriver.paymentinfo = PaymentInfo

        ValidDriver.email = Email
}

fact singleReservation {

        all r1, r2: Reservation |
        r1 != r2 => r1.driver != r2.driver

        all r1, r2: Reservation |
        r1 != r2 => r1.vehicle != r2.vehicle
}

fact mainteinance {

        all r : Reservation |
        not (r.vehicle in Report.vehicle)

        all t1, t2 : Report |
        t1 != t2 => t1.vehicle != t2.vehicle
}

//Workers signals
sig Worker {
}

sig Report {
        vehicle : one Vehicle,
        driver : one ValidDriver
}
```

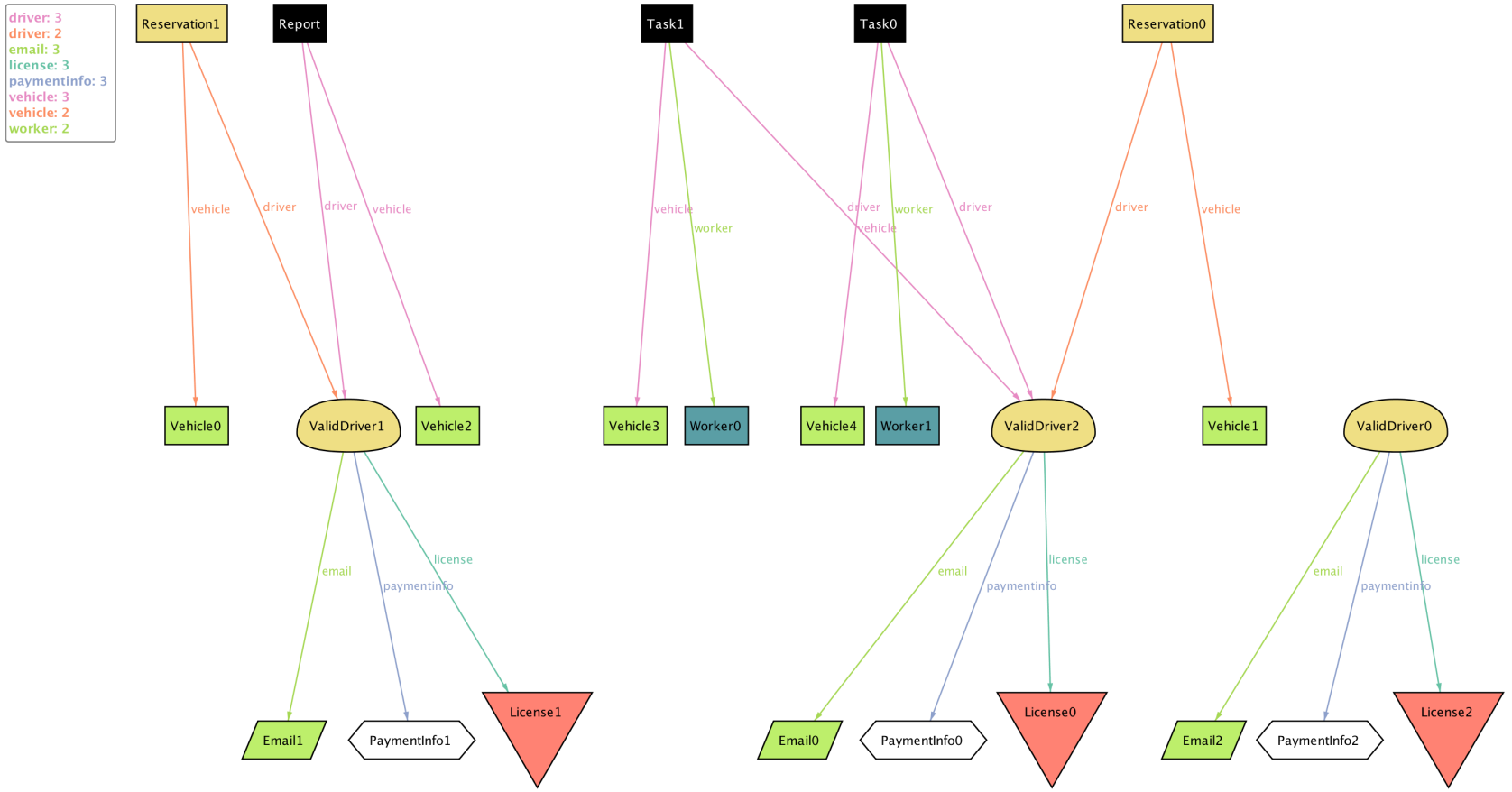
```
sig Task extends Report{
        worker : Worker
}


pred show() {
        #Vehicle = 5
        #ValidDriver = 3
        #Reservation = 2
        #Report = 3
        #Worker = 2
}

run show for 5
```

6.2 Generated world and results

The simulation returned a consistent system. The generated graph is presented below: in this world there are total of 5 vehicles, 3 Drivers (all valid), 2 Workers. At the current time of the system, there are 2 active reservations, 2 active tasks in progress, and 1 generated report that is waiting for an available Worker to be assigned to.


driver: 3
driver: 2
email: 3
license: 3
paymentinfo: 3
vehicle: 3
vehicle: 2
worker: 2
Reservation1
Report
Task1
Task0
Reservation0
vehicle
driver
driver
vehicle
vehicle
worker
driver
vehicle
worker
driver
driver
vehicle
Vehicle0
ValidDriver1
Vehicle2
Vehicle3
Worker0
Vehicle4
Worker1
ValidDriver2
Vehicle1
ValidDriver0
email
paymentinfo
license
email
paymentinfo
license
email
paymentinfo
license
Email1
PaymentInfo1
License1
Email0
PaymentInfo0
License0
Email2
PaymentInfo2
License2