



Politecnico di Milano  
AY 2017/2018

# Travlendar<sup>+</sup>

Requirement Analysis and Specification Document

Mirko Salaris, Piervincenzo Ventrella, Pietro Cassarino

October 29, 2017

---

**Deliverable:** RASD  
**Title:** Requirement Analysis and Verification Document  
**Authors:** Mirko Salaris, Piervincenzo Ventrella, Pietro Cassarino  
**Version:** 1.0  
**Date:** 27-October-2017  
**Download page:** <https://github.com/mirkosalaris/CassarinoSalarisVentrella/>  
**Copyright:** Copyright 2017, M. Salaris, P. Ventrella, P. Cassarino  
All rights reserved

---

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Scope . . . . .	4
1.2.1	Goals . . . . .	4
1.3	Definitions, Acronyms, Abbreviations . . . . .	5
1.3.1	Definitions . . . . .	5
1.3.2	Acronyms . . . . .	5
1.3.3	Abbreviations . . . . .	5
1.4	Revision history . . . . .	6
1.5	Document Structure . . . . .	6
<b>2</b>	<b>Overall Description</b>	<b>7</b>
2.1	Product perspective . . . . .	7
2.2	Product functions . . . . .	7
2.2.1	Scenario 1 . . . . .	7
2.2.2	Scenario 2 . . . . .	7
2.2.3	Scenario 3 . . . . .	7
2.2.4	Scenario 4 . . . . .	8
2.2.5	Scenario 5 . . . . .	8
2.2.6	Scenario 6 . . . . .	8
2.3	User characteristics . . . . .	8
2.3.1	Actors . . . . .	8
2.4	Assumptions, dependencies and constraints . . . . .	9
2.4.1	Text Assumptions . . . . .	9
2.4.2	Domain Assumptions . . . . .	9
<b>3</b>	<b>Specific Requirements</b>	<b>10</b>
3.1	External Interface Requirements . . . . .	10
3.1.1	User Interfaces . . . . .	10
3.1.2	Hardware Interfaces . . . . .	11
3.1.3	Software Interfaces . . . . .	11
3.1.4	Communication Interfaces . . . . .	11
3.2	UML modeling . . . . .	11
3.2.1	Use Case diagram . . . . .	11
3.2.2	Class diagram . . . . .	17
3.2.3	Activity diagrams . . . . .	18
3.3	Functional Requirements . . . . .	21
3.4	Performance Requirements . . . . .	23
3.5	Design Constraints . . . . .	23
3.5.1	Standard compliance . . . . .	23
3.5.2	Hardware limitations . . . . .	23
3.5.3	Any other constraint . . . . .	23
3.6	Software System Attributes . . . . .	23
3.6.1	Reliability . . . . .	23
3.6.2	Availability . . . . .	23
3.6.3	Security . . . . .	23
3.6.4	Maintainability . . . . .	24
3.6.5	Portability . . . . .	24
<b>4</b>	<b>Formal Analysis Using Alloy</b>	<b>25</b>
<b>5</b>	<b>Effort Spent and Team Work</b>	<b>35</b>
<b>6</b>	<b>References</b>	<b>36</b>
6.1	Software and Tools . . . . .	36
6.2	Reference Documents . . . . .	36

## List of Figures

1	Login . . . . .	10
2	Select solution . . . . .	10
3	Visualize schedule . . . . .	11
4	Person Use Case Diagram . . . . .	11
5	User Use Case Diagram . . . . .	12
6	"Specify Details" Use Case . . . . .	13
7	Manager Use Case Diagram . . . . .	13
8	General Class Diagram . . . . .	17
9	Registration Activity Diagram . . . . .	18
10	Creation appointment Activity Diagram . . . . .	19
11	Solution selection Sub-Activity Diagram . . . . .	20
12	Ticket handling Sub-Activity Diagram . . . . .	20

# 1 Introduction

## 1.1 Purpose

This document is the Requirement Analysis and Specification Document for the Travlendar+ application. Its aim is to inform about what the application offers, about requirements and goals that the system must present. This document offers also an analysis of the world and of the shared phenomena regarding Travlendar+. RASD contains class diagram to show domain model and other diagrams which illustrate, with more details, transactions of the functionalities of the application.

## 1.2 Scope

Travlendar+ is an application that allows people to organize and to track their appointments and meeting by registering them into the application. A person becomes a User of Travlendar+ by registering himself to the app. After this phase, users can start to use basic functionalities of the app (e.g. register appointments and organize meetings).

The app allows users to create appointments, eventually inviting other users of the app, facilitating communication issues. The goal of Travlendar+ is to organize in the best way all daily commitments of its users considering all the possible problems that can influence travels and trips (e.g. weather conditions, strikes, etc.).

When a User creates an event, he can add, for the travel, eventual passengers or baggage, so that the app can suggest better trip choices. Travlendar+ allows users to visualize their planned schedule too.

The dynamicity of the software allows users to set some personal preferences, for example, to set a flexible break window time for having free time or lunch, to choose eco-friendly solutions for his trips, to deactivate some transportation means.

The system interfaces with other firms (e.g. transportation companies, territory maps companies, sharing transportation companies) to offer a more comprehensive user experience. In this way, a User has the possibility to buy tickets and passes for transportation means. Moreover, the system offers a location service for vehicles of affiliated shared transport companies but to proceed with the renting the User is redirected to the company's app.

### 1.2.1 Goals

[G1] a Person should be able to have his/her own Travlendar+ agenda

[G2] a User should be able to customize the offered service

[G2]#1 specify his/her preference for eco-friendly solution

[G2]#2 define break time windows, either flexible or fixed

[G2]#3 define time slot in which the use of specific transportation means should be avoided

[G2]#4 define a minimum distance below which a specific transportation mean should be avoided

[G2]#5 define a maximum distance beyond which a specific transportation mean should be avoided

[G2]#6 disable permanently specific transportation means

[G3] a User should be able to take note of all his/her appointments and their details

[G4] a User should be able to manage his/her appointments

[G5] for each appointment, the User should be assisted in the choice of the travel solution

[G5]#1 travel solution suggestions must take into account traffic, weather conditions/forecast, strikes, type of appointment, baggage, passengers

[G6] a User should be able to invite other persons to his/her appointment

[G7] a User is assisted in the purchase of a ticket when it is required

[G8] a User should be able to locate nearest vehicle of a vehicle sharing system, if that is the transportation mean of choice of an incoming appointment

[G9] a User should be able to rent a shared vehicle, if that is the transportation mean of choice of an incoming appointment

[G10] a User should always be aware of the incoming appointments and how to reach them

[G10]#1 the User should be aware of eventual complications (bad weather, traffic, strikes)

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

Here we provide a list of definitions of words and expression used in the documents. Every time such words or expressions will be used they will be preceded by the symbol "(↑)" that will be a link to this section.

- **welcome page:** the page where the user is redirected after completing the sign up process and logging in for the first time. In this page the app sequentially asks to the user to insert credit-card data and set his preferences. The user can skip any of this phases and complete them in a second time. After this process, the initial settings configuration is completed.

- **personal preferences:** with this term we mean that here the user can:

- specify break time windows;
- specify the interest or not for eco-friendly solutions;
- specify constraints, such as avoiding bike, on the travel means solutions.

- **supported languages:** the set of languages that the S2B will be able to use to communicate with its users. English and Italian are included.

- **valid credentials:** Name, surname, personal email address and password.

Name: it should be non empty and it should contains only alphabetical characters

Surname: it should be non empty and it should contains only alphabetical characters

Email: It should be a valid email address, with an alphanumerical string followed by a '@', followed by an alphanumerical string, a dot, and a domain name

Password: It should be a string with at least 8 characters

- **appointment details:** time, date, type of appointment, location, number of passengers and precence of baggage.

- **incoming appointment:** the next scheduled appointment for which the S2B send a remind to the user. The remind is sent a certain amount of time before the appointment starts, to allow the user to get on time in the location.

- **affiliated company:** a company (transport company or vehicle sharing company) that has deals with the S2B.

- **break window:** it is a time slot in which the User specify he/she want to be absolutely free. In this time slot he does not want to have any appointment nor to be travelling. This is a fixed break window. A flexible break window is a time slot in which he/she specify he want to have some free time, but not the full window free.

### 1.3.2 Acronyms

- **S2B:** System to Be;

- **API:** Application Programming Interface.

### 1.3.3 Abbreviations

- **[Gn]:** n-th goal. Apart when it is actually defined, it is always a reference to the definition of the goal

- **[Dn]:** n-th domain assumption. Apart when it is actually defined, it is always a reference to the definition of the domain assumption

- **[Rn]:** n-th requirement

## **1.4 Revision history**

## **1.5 Document Structure**

After purpose and scope , used to briefly introduce the topic, we decided to state the goals that the StB should achieve coupled with a list of usefull definitions and acronyms. subsequently the text proceeds with an analysy of the functions that the app should provide. The analysy starts with a general exposition of the scenarios and become gradually more detailed passing trough the analysy of the actors that will interact with the S2B and the statement of domain assumptions. After that, the specific requirements are exposed focusing firstly on the external interfaces and then providing the models used to highlight the relations between actors and StB and describe the internal structure of the latter. After that, Functional and non Functional requirements are sequentially discussed. Before ending with the effort spent and the references we provide a formal analysy performed with alloy.

## 2 Overall Description

### 2.1 Product perspective

The system will be developed from scratch and it will use a lot of external services including Google services and the services provided by the (↑) *affiliated companies*. This is because the services provided by Google are of high quality and there is no point in trying to redevelop them. The services of affiliated companies are needed to interface with them and have information on tickets and vehicles (*for Sharing Transportation Companies*).

### 2.2 Product functions

Here we provide several scenarios to better delineate the purposes for which the app should be designed, the situations the S2B will deal with and more generally to have a better comprehension of the associated environment.

#### 2.2.1 Scenario 1

Mario is the director of his company and he has seen an ad of Travlendar+, so he wants to try it to organize the weekly meetings with his employees. After the installation, he has to register to the app. The first thing he creates is his weekly program. He works from Monday to Friday, from 8 am to 16 pm. There is an actual meeting coming up on Friday at 8 pm, so he creates a new meeting in the app. After setting the time and the day, he invites his employees to join the meeting. He will remain in his office, so he will not need to use any travel means.

Giovanni is one of Mario's employee and he is registered to Travlendar+. He receives a notification and accepts the invitation to the meeting. He chooses to reach the location by walk because he will already be nearby.

Alex is another employee and does not have an account on Travlendar. He receives an email with an invitation link to register to the app. After the registration, the app redirects him to the meeting's invitation and he will proceed by accepting it and choosing to go by car. Then he explores the app and decides to add his weekly program. The app finds out that he will be in the cafe near the metro at 6:40 pm, so it suggests Alex to take the metro instead of the car. He accepts the suggestion.

#### 2.2.2 Scenario 2

Paolo, a resident of Bergamo, has recently registered to Travlendar+ and during the initial setup has specified a flexible launch break from 11:00 am to 13:00 am, with at least 40 minutes of break.

Tomorrow he is going to have an audition at 12:30 pm, in Monza. He inserts the event in the app and after having specified that the audition will end at about 13:30pm, he looks at the suggestions of the app on the travel means to take: the app suggests him some travel solutions, but he does not specify which he's going to take because he wants to think about it overnight.

The next morning, the app sends Paolo a reminder with two travel solutions:

- go by car, leaving at about 11.45am, arriving at 12.27 pm;
- take the bus, passing at 10:49 am and arriving at 11:38 am.

He chooses the second option to avoid being late at the audition.

#### 2.2.3 Scenario 3

Alex is a professor of Bologna University, he has a short memory and is very badly organized, so he decides to rely on Travlendar+. Alex downloads it on occasion of a work trip. He signs up and decides to insert his credit card data for an eventual purchase from the app.

He needs to reach the University of Parma to hold a conference. Alex sets up the app to arrive in Parma by train. Travlendar+ asks Alex the kind of event and he specifies it is a formal work meeting. The app asks him which transport means he wants to take in order to get to the university from the railway station. Alex opts to go by bicycle, although the app suggest not to, because of the formal type of meeting.

The departure day Alex is in a shopping center with his family, he has completely forgotten that he has a train to take, but Travlendar+ solves the problem by notifying him of the appointment. At that point, Alex has no more time and chooses to buy the train tickets using the app. While he is on the



train, Travlendar+ suggests him to choose another transport means (instead of a bike) because of the bad weather conditions. Alex accepts the advice and decides to take a bus.

#### **2.2.4 Scenario 4**

Luca is a meteorologist who works for a laboratory in Venice. He knows very well all the climatic problem that humans are creating in their Country. Luca finds Travlendar+ very appropriate to help in solving this problematic. He likes to opt for an Eco-friendly solution by setting up this preference in the app settings. In this way he can avoid, at least in this aspect, further damaging the environment. His favorite functionality is bike sharing because of its innovative localization system and its low environmental impact.

#### **2.2.5 Scenario 5**

Mark, son of Lucas, asks to his father to bring him to the basketball tournament of Sunday morning. Lucas checks the daily schedule for Sunday and he notices he already has an appointment with the hairdresser but, of course, spending time with his son is more important, so he decides to delete the previous event on the agenda and set a new one.

Mark asks to the father if also his team mate Mike can come. Of course Mark and Mike have to bring with them the bags with the jersey and the basketball shoes, so Lucas, creating the event on Travlendar+, after specifying the location of the basketball court, specifies also that he will bring with him baggage and passengers.

Unfortunately his car is broken, so Lucas use the app to look for alternative solutions.

Travlendar+, taking into account the constraints previously settled by the man, suggests to him to use Enjoy or SmartToGo, two well known car sharing companies that will solve his problem.

Lucas accepts the suggested solution and proceeds with the creation of the event.

#### **2.2.6 Scenario 6**

Mary, John's wife, one week ago, asked to her husband to pick the children up to school on Monday at 13.00 and, because she knows John, forced him to take note of that with Travlendar+.

So John planned this event on the app specifying that he will use the car to do it. He specified also the location of the school.

On Monday morning, as usual, Travlendar+ shows to John his daily program reminding him about his children and showing the previous travel mean planned.

John, still intentioned to pick the children up with the car, does not modify the plan, closes the app and goes to work in the other side of Milan.

At 12.00 Travlendar+, according to the GPS position of the man, suggests him to leave in 15 minutes. Travlendar+ also suggests to avoid to go through Viale Gioia because of the traffic and take the SS1.

Thanks to Travlendar+ John manages to be on time, collect the children and make his wife proud of him.

### **2.3 User characteristics**

#### **2.3.1 Actors**

- Person: a person that doesn't have a registered account. The only thing that he/she can do is to proceed with the Sign Up process;
- User: a person passed through a successful registration process and now able to use all the Travlendar+ services. He/she can login to the system and, after that, use all the platform's functionalities.
- Credit Institution: the institution that checks the credit-card validity and reports it to the S2B;
- Google: the system with whom the S2B retrieves the maps and related information about routes, real-time traffic situations, estimated travel time and weather conditions;
- Transport Company System: the system of the affiliated companies with whom the S2B interacts to allow the user to buy the tickets for the associated travel mean;

- Shared Vehicle Company System: the system with whom the S2B interacts to allow the user to visualize the map of the available vehicles to rent and locate the nearest one. Vehicle Company System provides the GPS locations of the vehicles. To proceed with the renting the user is redirected to the vehicle Company System/App.
- Manager: the person able to manage the system. His activities consists in adding or deleting affiliated companies (both transport companies or shared vehicle companies) and adding new type of tickets according to the services provided by affiliated transport companies.

## **2.4 Assumptions, dependencies and constraints**

### **2.4.1 Text Assumptions**

- credit cards do not have an expiration date

### **2.4.2 Domain Assumptions**

- [D1] the user's device should allow the app to retrieve the language settings
- [D2] when the registration process begin, the Person always inserts his/her credentials
- [D3] when the S2B sends an email, it is always received by the receiver
- [D4] every Person has an email address
- [D5] the User shall remember his password
- [D6] the User knows only his password
- [D7] the User's device has a working GPS installed, to which the app has access
- [D8] the affiliated shared vehicle companies provide a localization service APIs
- [D9] Google Maps services take traffic into consideration

## 3 Specific Requirements

### 3.1 External Interface Requirements

#### 3.1.1 User Interfaces

Here we provide some basic mockups to show how the interface should appear to the user:

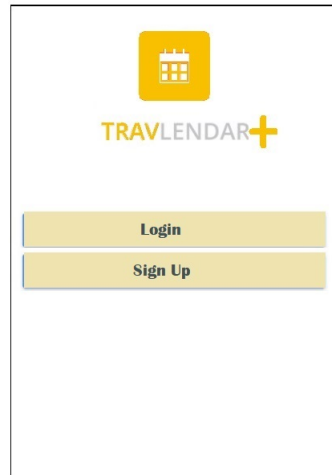


Figure 1: Login



Figure 2: Select solution

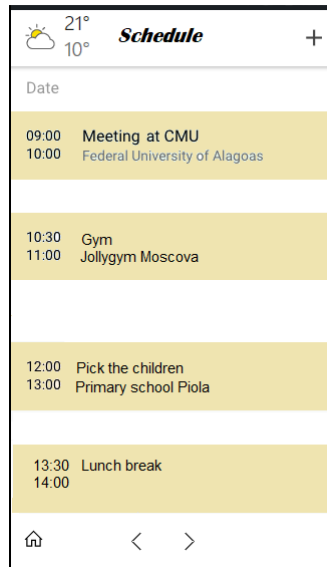


Figure 3: Visualize schedule

### 3.1.2 Hardware Interfaces

The main hardware interface of the system consists in the access to the GPS data in the mobile application. The application also requires Internet connectivity and internal storage access.

### 3.1.3 Software Interfaces

The mobile application must support Android, iOS and the remaining main OSs (further details are discussed in paragraph 3.6.5 Portability). The web application works on any web server that supports Java. The back-end stores its data in a DBMS and can run on every platform that supports the JVM.

### 3.1.4 Communication Interfaces

The communication between clients and server should be HTTP requests/responses based.

## 3.2 UML modeling

In this section we formalize the S2B in terms of UML models.

We divided the Use Case Diagrams into parts to slightly improve the readability of the Diagrams. After the Diagrams, descriptions of the main Use Cases are provided.

After that we provide a Class Diagram of the whole system and then some Activity Diagrams, to better explain the structure of the S2B and its behavior.

Other types of diagrams have been considered, but we eventually realized them to not provide any other notable information about the system at this stage.

### 3.2.1 Use Case diagram

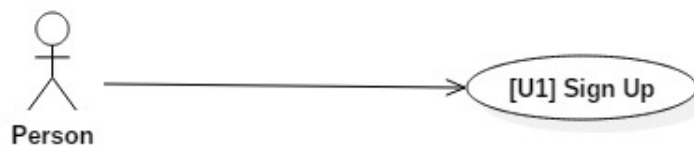


Figure 4: Person Use Case Diagram

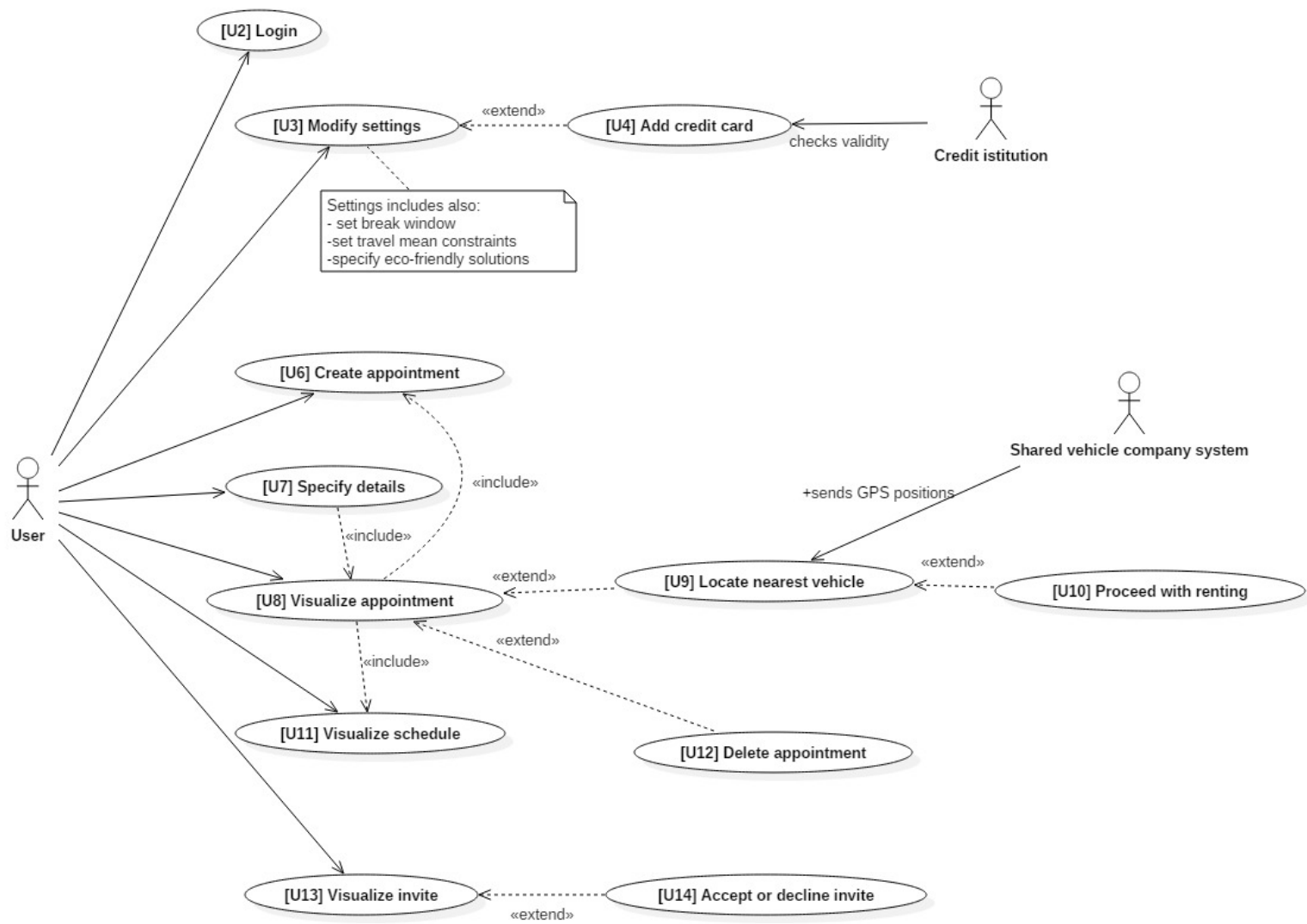


Figure 5: User Use Case Diagram

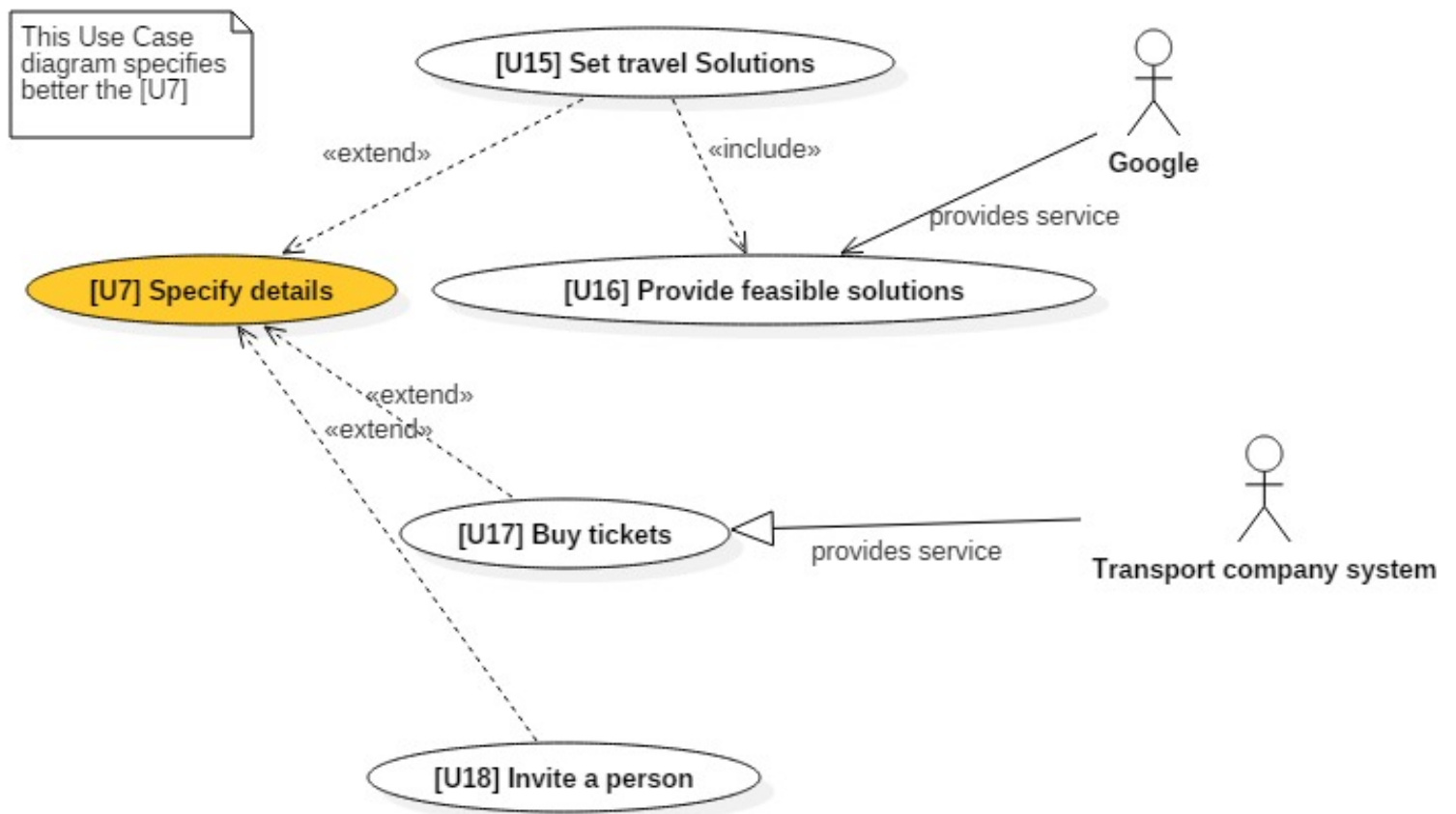


Figure 6: "Specify Details" Use Case

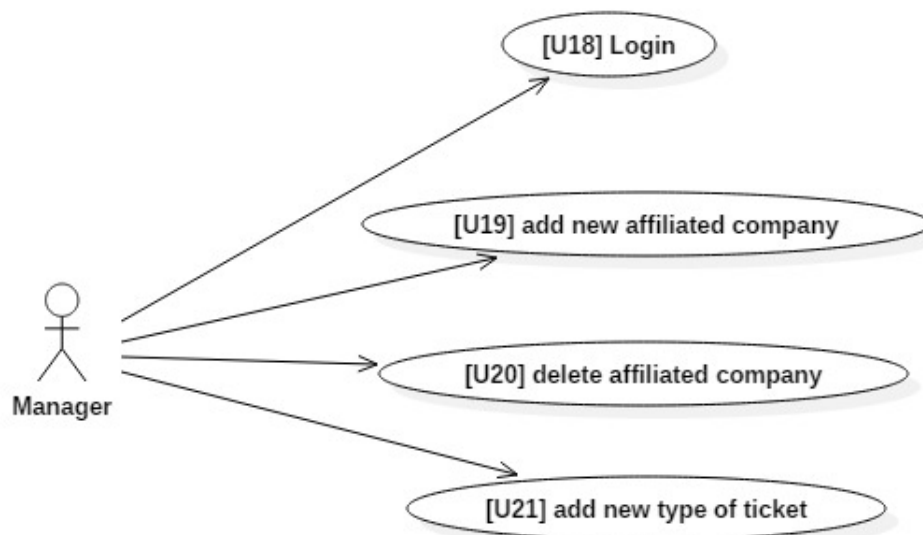


Figure 7: Manager Use Case Diagram

### User creates appointment

<b>Use case:</b>	User creates an appointment
<b>Actors:</b>	User
<b>Entry condition:</b>	The user must be logged
<b>Flow of events:</b>	<p>The user creates an appointment giving it a name;  User specifies the time and date of the appointment;  User specifies the location of the appointment;  User specifies the type of the appointment;  User specifies details such as passengers or baggage;  User selects a travel mean taking to account apps suggestion;  The app takes note of the settings and send a confirmation;  The app redirects the user to the main page.</p>
<b>Secondary flows:</b>	<p>User does not specify a travel mean and let it blank;  The app takes anyway note of the setting and alert the user of the missing information;  The app redirects the user to the main page.</p>
<b>Exceptions:</b>	<p>Warnings messages are created in the following cases:</p> <ul style="list-style-type: none"> <li>• User creates an appointment that overlaps with another appointment;</li> <li>• User creates an appointment with a location that is unreachable in the allocated time;</li> <li>• User creates an appointment that violates the set constraints about the break windows.</li> </ul>
<b>Post conditions:</b>	The user is successfully redirected to the main page.

### Sign Up

<b>Use case:</b>	Sign Up
<b>Actors:</b>	Person
<b>Entry condition:</b>	none;
<b>Flow of events:</b>	<p>the person inserts (↑) <i>valid credentials</i>;  the app sends an email with the confirmation link;  the person gives the confirmation through the link on the mail;  the app shows the (↑) <i>welcome page</i> to the new user.</p>
<b>Secondary flows:</b>	none.
<b>Exceptions:</b>	<p>the person inserts non (↑) <i>valid credentials</i>;  the sign up cannot proceed.</p>
<b>Post conditions:</b>	the person is successfully signed up and become an actual logged user.

### Initial settings configuration

<b>Use case:</b>	Initial settings configuration
<b>Actors:</b>	User
<b>Entry condition:</b>	the User just has just completed the sign up process; user must be logged.
<b>Flow of events:</b>	User insert sequentially the following information: <ul style="list-style-type: none"><li>• Credit card;</li><li>• Break time windows;</li><li>• Interest for Eco-friendly solutions on the travel means.</li><li>• Constraints on travel means.</li></ul> The app, for each step, check the info and send a confirmation; The app redirects the user to the main page.
<b>Secondary flows:</b>	User skips to specify one or more information that could be specified later in the settings. The app notifies the user about the missing information and redirects anyway the user to the main page.
<b>Exceptions:</b>	user inserts inconsistent information (incorrect credid-card information, break time shorter than 30 minutes); The app allerts the user and asks him to insert again the info.
<b>Post conditions:</b>	the set configurations are successfully saved and the user is redirected to the main page.

### User specifies the appointment

<b>Use case:</b>	User specifies appointment's details.
<b>Actors:</b>	User
<b>Entry condition:</b>	user must be logged; the appointment must exist;
<b>Description:</b>	user specifies or modifies basic details (time, date, location, type of appointment, number of passengers and the precence of baggage) and eventually: <ul style="list-style-type: none"><li>• If not specified yet, sets a travel mean;</li><li>• Modifies the previous travel mean;</li><li>• Buys a ticket for the travel mean;</li><li>• Invites a person.</li></ul>



#### User buy travel ticket

<b>Use case:</b>	User buy travel ticket
<b>Actors:</b>	User
<b>Entry condition:</b>	user must be logged; user must have added a payment card .
<b>Flow of events:</b>	User selects an appointment; User, through the app, searches for tickets for the specified travel mean; User selects the tickets option and picks one; User proceeds with the payment; The payment operation ends successfully; The app sends a confirmation and redirects the user to the main page;
<b>Secondary flows:</b>	none
<b>Exceptions:</b>	The payment is rejected (not enough credit, expired card, ..); The app notifies the user; The app redirects the user to the home page;
<b>Post conditions:</b>	User successfully books the tickets and is redirected to the main page.

### 3.2.2 Class diagram

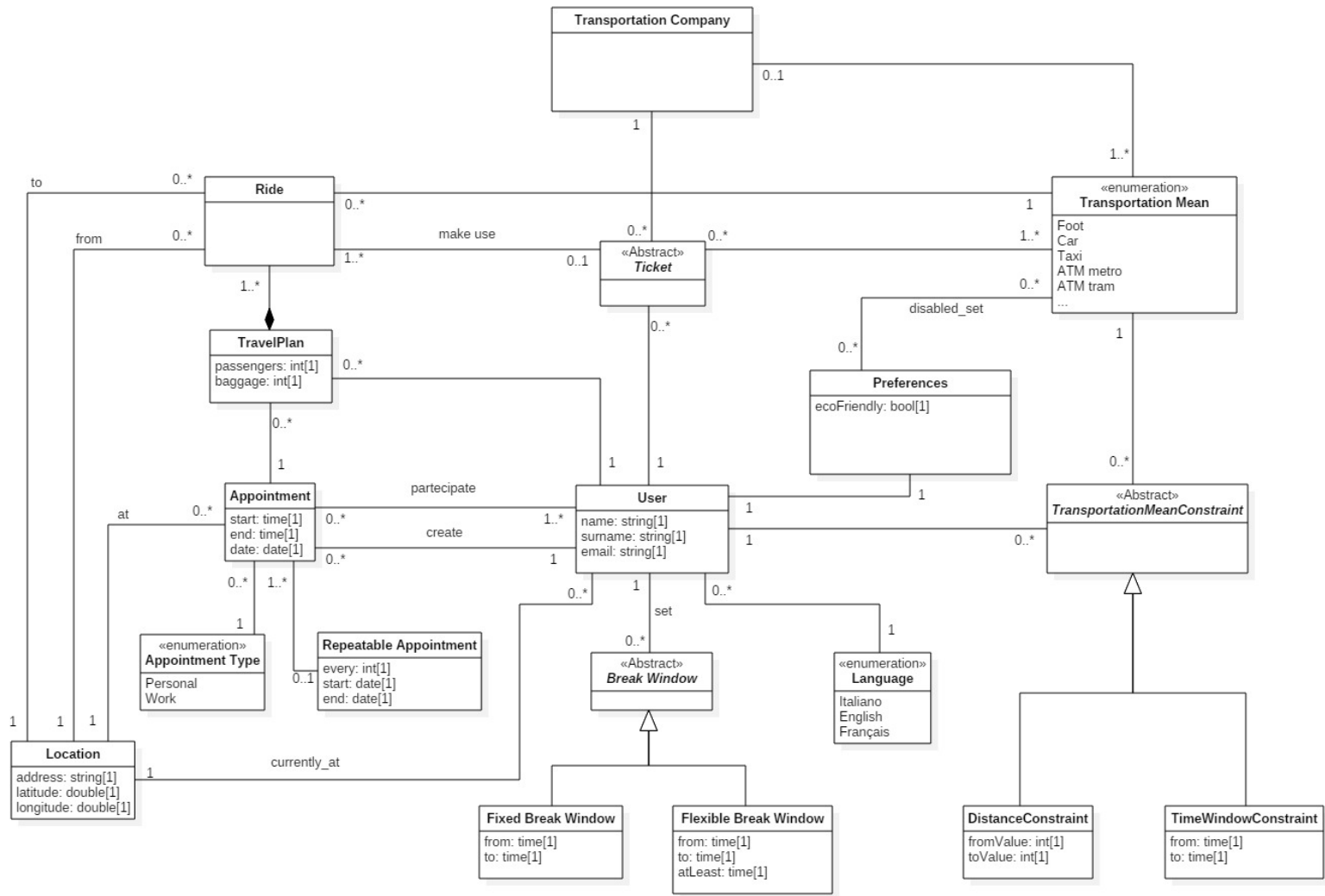


Figure 8: General Class Diagram

### 3.2.3 Activity diagrams

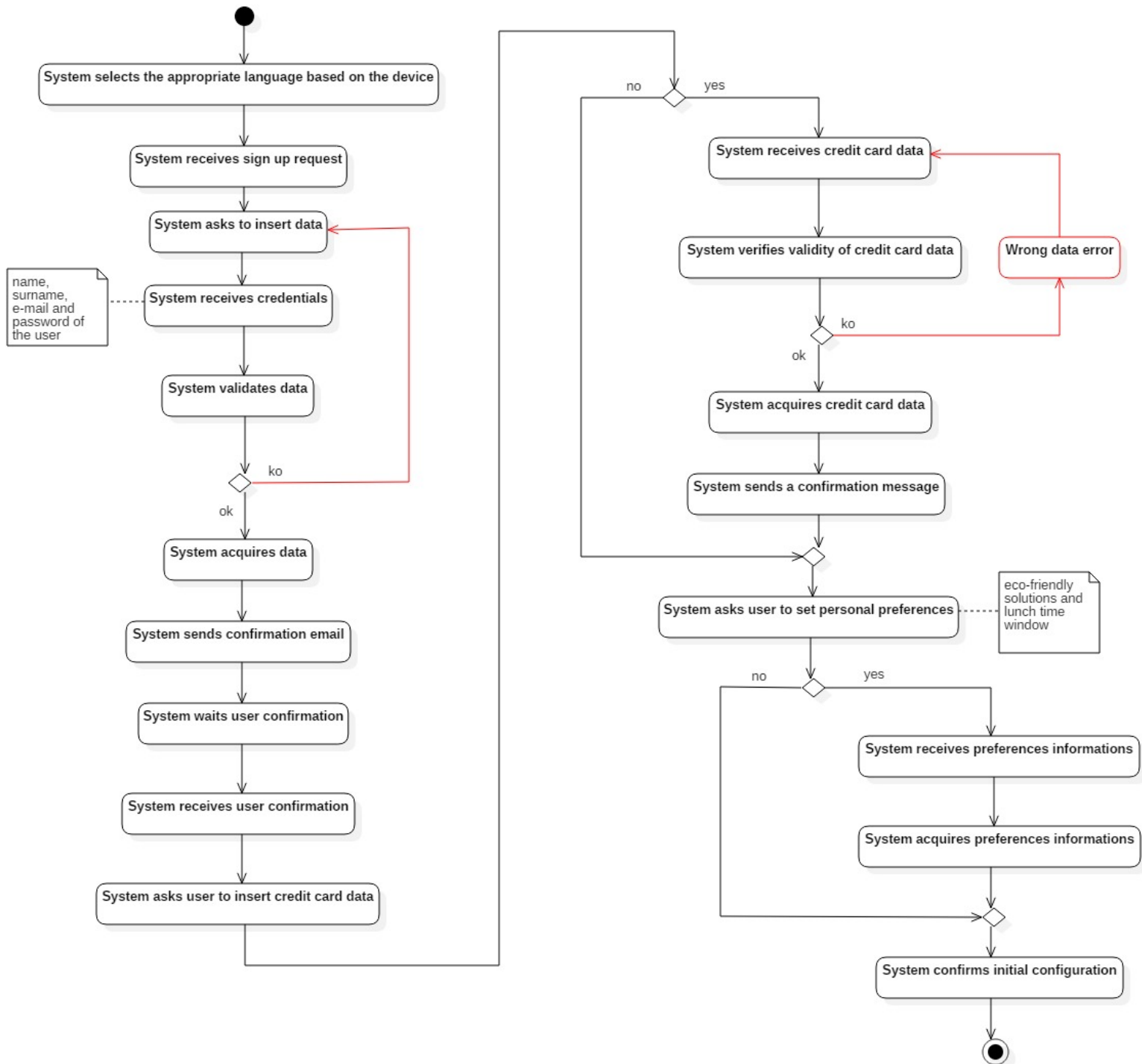


Figure 9: Registration Activity Diagram

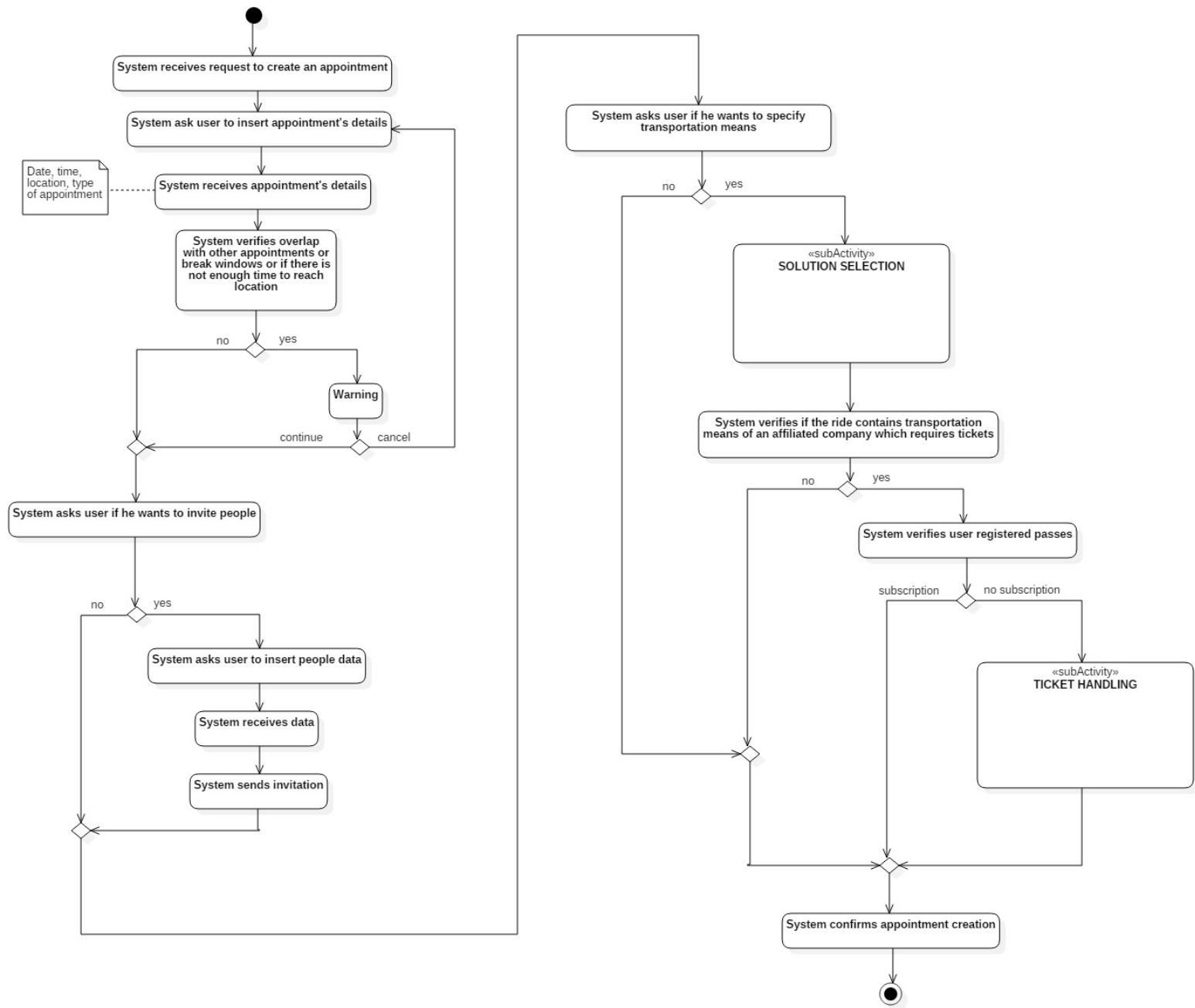


Figure 10: Creation appointment Activity Diagram

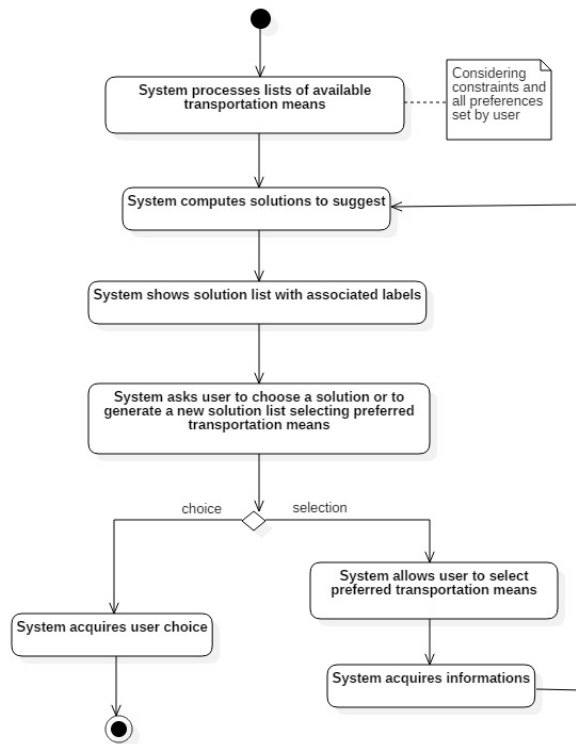


Figure 11: Solution selection Sub-Activity Diagram

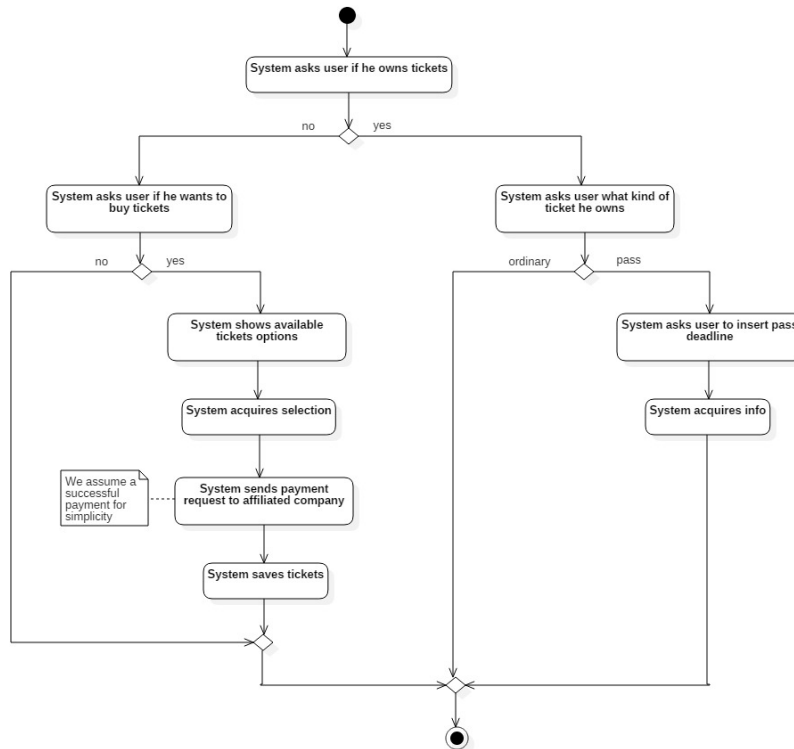


Figure 12: Ticket handling Sub-Activity Diagram

### 3.3 Functional Requirements

#### [G1] a Person should be able to have his/her own Travlendar+ agenda

- [R1] the S2B must provide to every Person a way to begin the registration process
- [R2] after the insertion of the credentials and their (↑)*validation*, the S2B has to send to the provided address an email with an activation link
- [R3] the registration fails if the inserted email is already associated to an account
- [R4] when the Person confirms through the activation link, he/she becomes a User
- [R5] in the case of non (↑)*valid credentials*, the system must reject them and restart the registration process
- [R6] the S2B must grant access to the User if and only if the User inserts an existing email and the associated password
- [D2] when the registration process begin, the Person always inserts his/her credentials
- [D3] when the S2B sends an email, it is always received by the receiver
- [D4] every Person has an email address
- [D5] the User shall remember his password
- [D6] the User knows only his password

#### [G2] a User should be able to customize the offered service

- [R7] for each type of (↑)*preference*, the S2B must provide the User the possibility to set or change the value(s)
- [R8] for each type of (↑)*preference*, the S2B must store the preference value(s)

#### [G3] a User should be able to take note of all his/her appointments and their details

- [R9] the system must provide a way to start the creation of a new appointment
- [R10] during the process the user shall insert the (↑)*appointment details*

#### [G4] a User should be able to manage his/her appointments

- [R11] existing appointments can be viewed together as a schedule view
- [R12] the schedule can be daily or weekly
- [R13] from the schedule view, the system provides a way to visualize a single appointment and its details
- [R14] after visualizing an appointment, the User who created it, can choose to edit its details
- [R15] in the schedule view the User can select one or more appointments
- [R16] in the schedule view, selected appointments can be deleted

#### [G5] for each appointment, the User should be assisted in the choice of the travel solution

- [R17] when time and location of the current appointment are set, the S2B produces a list of travel solutions with associated suggestions

- [R18] the S2B provide the user the possibility to choose one of the suggested travel solutions or leave the travel plan unspecified
- [R19] the S2B also provides the possibility to choose a preferred transportation mean
- [R20] when a new preferred transportation mean is selected the S2B has to recompute the list of solutions according to the new preference
- [R21] if weather forecast are bad: foot, bicycle motorbike are discouraged
- [R22] if strikes have been announced, public transport is discouraged
- [R23] in case of baggage or passengers a car is recommended
- [R24] in case of a work appointment, bicycle is discouraged

[D9] Google Maps services take traffic into consideration

**[G6] a User should be able to invite other persons to his/her appointment**

- [R25] when time and location of the current appointment are set, the S2B offers the possibility to invite other Users or Persons, through their emails
- [R26] when a User or a Person is invited, the S2B will inform him/her sending an email

[D3] when the S2B sends an email, it is always received by the receiver

**[G7] a User is assisted in the purchase of a ticket when it is required**

- [R27] the S2B must accept credit card data from the User
- [R28] the S2B must forward the credit card data to a credit institution to validate them
- [R29] the S2B must let the User use the credit card if and only if the inserted credit card data are valid
- [R30] when the user selects a travel solution for which a ticket is expected, the S2B asks the User to specify if he/she owns a ticket (either ordinary or a pass, in which case the deadline has to be inserted)
- [R31] if the user has selected a travel solution for which a ticket is expected and the User said to not own a ticket, the S2B asks him/her to buy a ticket (options available only for transportation means of (↑) *affiliated companies*)

**[G8] a User should be able to locate nearest vehicle of a vehicle sharing system, if that is the transportation mean of choice of an incoming appointment**

- [R32] when a user visualizes an incoming appointment for which a shared vehicle of an affiliated company has to be used, the S2B provides a localization service

[D7] the User's device has a working GPS installed, to which the app has access

[D8] the affiliated shared vehicle companies provide a localization service APIs

**[G9] a User should be able to rent a shared vehicle, if that is the transportation mean of choice of an incoming appointment**

- [R33] after the User has localized a vehicle, the S2B offers the possibility to rent it
- [R34] when the User selects the vehicle to rent, the app redirects him/her to the right company's app or site

**[G10] a User should always be aware of the incoming appointments and how to reach them**

[R35] when an appointment becomes incoming, the S2B sends a notification to the User

[R36] if a travel plan has not already been set by the User, the notification suggests one

[R37] if a travel plan has already been chosen but some complications (bad weather, traffic, strikes) have arisen the User is informed and a new feasible solution is suggested

### **3.4 Performance Requirements**

The system has to be able to respond to a possibly great number of simultaneous requests and more generally to a great number of request throughout the day. The S2B, at least for the start, will only be available for the Lombardy region. Based on demographic analysis (number of inhabitants, number of people under the age of 60, number of smartphones sold over the past 2 years), it was decided to design the S2B to support 100,000 users simultaneously, but scalability needs to be guaranteed.

### **3.5 Design Constraints**

#### **3.5.1 Standard compliance**

To ensure interoperability the S2B will follow the W3C web standard and will be as adherent as possible to coding practices in relation to the use of HTML/XHTML, CSS and Java programming language. Moreover the use of non-opensource libraries will be avoided.

#### **3.5.2 Hardware limitations**

- Mobile App:
  - \* 3G connection
  - \* GPS
  - \* Space for app package
- Web App:
  - \* Modern browser able to retrieve user's location

#### **3.5.3 Any other constraint**

##### **Regulatory policies**

The system will ask for users' payment informations and obviously, in addition to store them safely, will use them only for fees and rides payments. Moreover, the system will have to ask for users' permission in order to retrieve and use their positions. Email addresses won't be used for commercial uses.

### **3.6 Software System Attributes**

#### **3.6.1 Reliability**

The system must guarantee a 24/7 service. Very small deviations from this requirement will be obviously acceptable.

#### **3.6.2 Availability**

The S2B must guarantees a 3-nines availability (99.9 percent) with a downtime not greater than 8 hours per year.

#### **3.6.3 Security**

User credentials and payment information will be stored. Data confidentiality is a primary concern. In addition, when the user wants to buy tickets or rent a shared vehicle, the stored information must be sent to affiliated transport Company systems or shared vehicle Company systems. To ensure the security and the confidentiality of this information, the S2B must be able to adopt access management protocols and communication protocols able to prevent not granted access and/or Sniffing/Spoofing activities performed by third.



#### **3.6.4 Maintainability**

The S2B must be designed in a way to easily correct defects or their cause, repair or replace faulty or worn-out components without having to replace still working parts, prevent unexpected working condition, maximize its useful life, maximize efficiency, reliability, and safety, meet new requirements, make future maintenance and cope with a changed environment.

#### **3.6.5 Portability**

The S2B must be able to run in all main mobile OS *Android, iOS, Windows-Phone OS*) and being supported by all the main Web Browser (*Google Chrome, Safari, Firefox, Microsoft Edge*).

## 4 Formal Analysis Using Alloy

---

```
1 // =====
2 // ===== PRIMITIVE SIGNATURES
3 sig Name {}
4
5 sig Surname {}
6
7 sig Email {}
8
9 sig Address {}
10
11 sig Double {}
12
13 enum Bool {
14     True,
15     False
16 }
17
18 // =====
19 // ===== SIGNATURES
20 sig Time {
21     value: Int
22     } { value ≥ 0 }
23
24 sig Date {
25     value: Int
26     } { value ≥ 0 }
27
28 sig Ride {
29     makeUseTicket: lone Ticket,
30     byTranMean: TransportationMean,
31     fromLocation: Location,
32     toLocation: Location
33     } { fromLocation ≠ toLocation }
34
35 sig TransportationCompany {}
36
37 abstract sig Ticket {
38     usedFor: some TransportationMean,
39     providedByCompany: TransportationCompany
40     }
41
42 sig User {
43     name: Name,
44     surname: Surname,
45     email: Email,
46     ownsTicket: set Ticket,
47     hasPreferences: Preferences,
48     hasConstraints: set TransportationMeanConstraint,
49     speaksLanguage: Language,
50     setBreakWindows: set BreakWindow,
51     createsAppointment: set Appointment,
52     participatesToAppointment: set Appointment,
53     hasTravelPlan: set TravelPlan,
54     currentlyAtLoc: Location
55 }
```

```

56
57 sig Appointment {
58     id: Int,
59     start: Time,
60     end: Time,
61     atLocation: Location,
62     hasType: AppointmentType,
63     isRepeatable: lone RepeatableAppointment,
64     isModified: Bool,
65     isIncoming: Bool
66 }
67
68 sig Location {
69     address: Address,
70     latitude: Double,
71     longitude: Double
72 }
73
74 enum AppointmentType {
75     Personal,
76     Work
77 }
78
79 sig RepeatableAppointment {
80     every: Int,
81     start: Date,
82     end: Date
83 } { every i 0
84     start.value i end.value }
85
86 sig TravelPlan {
87     passengers: Int,
88     baggage: Int,
89     startRide: Ride,
90     intermediateRides: set Ride,
91     endRide: Ride,
92     forAppointment: Appointment
93 } {
94     passengers ≥ 0
95     baggage ≥ 0
96
97     // structural constraints on start, intermediate and end
98     //   ↳ Rides
99     no ir: intermediateRides | startRide = ir or endRide = ir
100    lone ir: intermediateRides | startRide.toLocation = ir.
101        ↳ fromLocation
102    lone ir: intermediateRides | endRide.fromLocation = ir.
103        ↳ toLocation
104    no ir: intermediateRides | startRide.fromLocation = ir.
105        ↳ toLocation
106    no ir: intermediateRides | endRide.toLocation = ir.
107        ↳ fromLocation
108    all ir: intermediateRides | ir.toLocation = endRide.
109        ↳ fromLocation or
110        ↳ one ir1: intermediateRides | ir.toLocation = ir1.
111        ↳ fromLocation

```

```

105         all ir: intermediateRides | ir.fromLocation = startRide.
           ↪ toLocation or
106         one ir1: intermediateRides | ir.fromLocation =
           ↪ ir1.toLocation
107         #intermediateRides = 0 implies
108         (startRide = endRide or startRide.toLocation =
           ↪ endRide.fromLocation)
109     }
110
111     // retrieves the whole set of Rides of a travel plan
112     fun travelPlanRides[t: TravelPlan] : some Ride {
113         t.startRide + t.intermediateRides + t.endRide
114     }
115
116     abstract sig BreakWindow {}
117
118     sig FixedBreakWindow extends BreakWindow {
119         from: Time,
120         to: Time
121     } { from.value < to.value }
122
123     sig FlexibleBreakWindow extends BreakWindow {
124         from: Time,
125         to: Time,
126         atLeast: Time
127     } { from.value < to.value
128         (atLeast.value < 0 and atLeast.value < minus[to.@value,
           ↪ from.@value]) }
129
130     enum Language {
131         Italiano,
132         English,
133         Francais
134     }
135
136     abstract sig TransportationMean {
137         belongsToCompany: lone TransportationCompany
138     }
139
140     one sig Foot, MoBike, PersonalCar, EnjoyCar, Metro, Tram extends
           ↪ TransportationMean {}
141
142     sig Preferences {
143         ecoFriendly: Bool,
144         disabledTranMean: set TransportationMean
145     }
146
147     abstract sig TransportationMeanConstraint {
148         associatedToTranMean: TransportationMean
149     }
150
151     sig DistanceConstraint extends TransportationMeanConstraint {
152         fromValue: Int,
153         toValue: Int
154     } { fromValue ≥ 0
155         toValue ≥ 0
156         fromValue < toValue }

```

```

157
158 sig TimeWindowConstraint extends TransportationMeanConstraint{
159     from: Time,
160     to: Time
161     } { from.value ≤ to.value }
162
163 // =====
164 // ===== ADDITIONAL SIGNATURES
165 sig SuggestedSolutions {
166     suggestTo: User,
167     containsSolutions: some Solution
168 }
169
170 sig Solution {
171     suggestTranMean: some TransportationMean,
172     forAppointment: Appointment,
173 }
174
175 sig Device {
176     belongsTo: User,
177     language: Language
178 }
179
180 sig AppInstance{
181     installedOn: Device,
182     displayLanguage: Language
183 }{ let d = installedOn | (displayLanguage = d.language) or
184    (displayLanguage = English and d.language not in
185     ↪ SupportedLanguages.setOfLanguages) }
186
187 one sig SupportedLanguages {
188     setOfLanguages: set Language
189 }
190
191 sig Person {
192     name: Name,
193     surname: Surname,
194     email: Email,
195     isUser: lone User
196 }
197
198 sig Invitation {
199     fromUser: User,
200     toEmail: Email,
201     forAppointment: Appointment
202 }{
203     forAppointment in fromUser.createAppointment
204     fromUser.email ≠ toEmail
205 }
206
207 sig Notification {
208     toUser: User,
209     incomingAppointment: Appointment
210 }
211 // =====
212 // ===== FACTS

```

```

213 fact EmailsAreUnique {
214     all disjoint u1, u2: User | u1.email ≠ u2.email
215 }
216
217 fact NoOverlappingLocations {
218     all disjoint l1, l2: Location | (l1.latitude ≠ l2.latitude) ∨ (l1
        ↪ .longitude ≠ l2.longitude)
219 }
220
221 fact TimelsUnique {
222     all disjoint t1, t2: Time | t1.value ≠ t2.value
223 }
224
225 fact ATicketBelongsOnlyToOneUser {
226     all disjoint u1, u2: User | u1.ownsTicket & u2.ownsTicket = none
227 }
228
229 fact TicketMustBeAssociatedToRides {
230     all t: Ticket | some r: Ride | t in r.makeUseTicket
231 }
232
233 fact APreferenceBelongsOnlyToOneUser {
234     all disjoint u1, u2: User | u1.hasPreferences & u2.hasPreferences
        ↪ = none
235 }
236
237 fact TranMeanConstraintsRefersOnlyToOneUser {
238     all disjoint u1, u2: User | u1.hasConstraints & u2.hasConstraints
        ↪ = none
239 }
240
241 fact ABreakWindowIsSetOnlyByOneUser {
242     all disjoint u1, u2: User | u1.setBreakWindows & u2.
        ↪ setBreakWindows = none
243 }
244
245 fact AnAppointmentIsCreatedOnlyByOneUser {
246     all disjoint u1, u2: User | u1.createAppointment & u2.
        ↪ createAppointment = none
247 }
248
249 fact AppointmentsMustBeCreatedOnlyByUsers {
250     all a: Appointment | some u: User | a in u.createAppointment
251 }
252
253 fact ATravelPlanBelongsOnlyToOneUser {
254     all disjoint u1, u2: User | u1.hasTravelPlan & u2.hasTravelPlan =
        ↪ none
255 }
256
257 // if a User has disabled a transportation mean, it should never be
    ↪ suggested to him/her
258 fact DisabledTranMeansAreNotSuggested {
259     all p: Preferences, s: SuggestedSolutions, u: User |
260     p in u.hasPreferences and
261     s.suggestTo = u and

```

```

262         u.hasPreferences.disabledTranMean & (s.containsSolutions).
           ↪ suggestTranMean = none
263     }
264
265     // if an appointment is associated to a travel plan of a User, the User
           ↪ must participate to the appointment
266     fact ConsistentUserTravelPlanAppointment {
267         all u: User, a: Appointment, tp: TravelPlan |
268         (tp.forAppointment = a and tp in u.hasTravelPlan) implies (a in u
           ↪ .participatesToAppointment)
269     }
270
271     fact AppointmentCreationImpliesParticipation {
272         all u: User, a: Appointment |
273         (a in u.createAppointment) implies (a in u.
           ↪ participatesToAppointment)
274     }
275
276     // there is not the possibility to have a name, surname, email, address,
           ↪ appointment
277     // type or transportation company without associations with something
278     fact AllNameMustBelongToUsers {
279         all n: Name | some u: User | u.name = n
280     }
281
282     fact AllSurnameMustBelongToUsers {
283         all s: Surname | some u: User | u.surname = s
284     }
285
286     fact AllEmailMustBelongToPersonos {
287         all e: Email | some p: Person | p.email = e
288     }
289
290     fact AllAddressesMustBelongToLocations {
291         all a: Address | some loc: Location | loc.address = a
292     }
293
294     fact TicketMustBelongToUsers {
295         all t: Ticket | some u: User | t in u.ownsTicket
296     }
297
298     fact AllTicketsMustBeProvidedByTranCompany {
299         all t: Ticket | some tc: TransportationCompany | t.
           ↪ providedByCompany = tc
300     }
301
302     fact TranCompanyMustBeAssociatedWithTranMean {
303         all tc: TransportationCompany | some tm: TransportationMean | tm.
           ↪ belongsToCompany = tc
304     }
305
306     // No tickets for personal and shared transportation means
307     fact TicketsUsedOnlyIfNecessary {
308         all t: Ticket | (Foot & t.usedFor = none) and
309         (MoBike & t.usedFor = none) and
310         (PersonalCar & t.usedFor = none) and
311         (EnjoyCar & t.usedFor = none)

```

```

312     }
313
314 fact NoTranCompanyForPersonalTranMeans {
315     (Foot.belongsToCompany = none) and
316     (PersonalCar.belongsToCompany = none)
317 }
318
319 fact TranMeanConstraintsMustBelongToUsers {
320     all tmc: TransportationMeanConstraint | some u: User | tmc in u.
321     ↪ hasConstraints
322 }
323
324 fact BreakWindowMustBeSetByUsers {
325     all bw: BreakWindow | some u: User | bw in u.setBreakWindows
326 }
327
328 fact RideMustBelongToTravelPlans {
329     all r: Ride | some tp: TravelPlan | r in travelPlanRides[tp]
330 }
331
332 fact RideBelongsToOnlyOneTravelPlan {
333     all disjoint tp1, tp2: TravelPlan | travelPlanRides[tp1] &
334     ↪ travelPlanRides[tp2] = none
335 }
336
337 fact TravelPlanMustBelongToUsers {
338     all tp: TravelPlan | some u: User | tp in u.hasTravelPlan
339 }
340
341 fact SolutionMustBeSuggested {
342     all s: Solution | some ss: SuggestedSolutions | s in ss.
343     ↪ containsSolutions
344 }
345
346 fact LocationAssociatedToRideAppointmentOrUser {
347     all l: Location | some r: Ride, a: Appointment, u: User |
348     l in (r.fromLocation + r.toLocation + a.atLocation + u.
349     ↪ currentlyAtLoc)
350 }
351
352 fact RepeatingAppointmentIsAnAppointment {
353     all ra: RepeatingAppointment | some a: Appointment | ra in a.
354     ↪ isRepeating
355 }
356
357 fact RepeatingAppointmentsAtTheSameTime {
358     all a1, a2: Appointment | (a1.isRepeating = a2.isRepeating)
359     ↪ implies
360     (a1.start = a2.start and a1.end = a2.end)
361 }
362
363 fact NoStartRideFromAppointmentLocation {
364     all tp: TravelPlan | tp.startRide.fromLocation ≠ tp.
365     ↪ forAppointment.atLocation
366 }
367
368 fact SameEmailImpliesSamePerson {

```



```

362         all p1, p2: Person | p1.email = p2.email implies (samePerson[p1,
363             ↪ p2])
364     }
365     fact SamePersonImpliesOldAndNew {
366         all disjoint p1, p2: Person | samePerson[p1, p2] implies
367             ((p1.isUser = none and p2.isUser ≠ none) or (p1.isUser ≠
368                 ↪ none and p2.isUser = none))
369     }
370     fact UserAndPersonSameData {
371         all p: Person | p.isUser ≠ none implies
372             let u = p.isUser |
373             p.email = u.email and
374             p.name = u.name and
375             p.surname = u.surname
376     }
377
378     fact EveryUserHasA2Person {
379         all u: User | some disjoint p1, p2: Person | p1.isUser = u and
380             ↪ samePerson[p1, p2]
381     }
382     fact IsModifiedImpliesAnotherAppointment {
383         all apOld: Appointment | some apNew: Appointment |
384             apOld.isModified = True implies
385             apOld.id = apNew.id and apOld ≠ apNew and apNew.
386                 ↪ isModified = False
387     }
388     fact SameAppointmentIdSameUser {
389         all disjoint ap1, ap2: Appointment | all u: User |
390             ap1.id = ap2.id and
391             (ap1 in u.participatesToAppointment implies (ap2 in u.
392                 ↪ participatesToAppointment))
393             and
394             (ap1 in u.createAppointment implies (ap2 in u.
395                 ↪ createAppointment))
396     }
397     fact AllUsersMustBeCreatorOrInvited {
398         all a: Appointment, u: User | a in u.participatesToAppointment
399             ↪ implies
400             (a in u.createAppointment or invitedToAppointment[u, a])
401     }
402     fact NotificationOnlyIfAppointmentIsIncoming {
403         all n: Notification | n.incomingAppointment.isIncoming = True
404     }
405     fact NotificationForAllIncomingAppointment {
406         all a: Appointment, u: User |
407             (a.isIncoming = True and a in u.participatesToAppointment)
408             implies
409             (one n1: Notification | n1.incomingAppointment = a and n1.toUser
410                 ↪ = u)
411     }

```

```

411
412 fact UserReceivesNotificationOfOwnedAppointments {
413     all n: Notification , u: User |
414         n.toUser = u implies n.incomingAppointment in u.
            ↪ participatesToAppointment
415 }
416
417 fact EachDeviceHasMaxOneAppInstance {
418     all d: Device | lone a: AppInstance | a.installedOn = d
419 }
420
421 fact EveryUserHasAtLeastOneAppInstance {
422     all u: User | some a: AppInstance | a.installedOn.belongsTo = u
423 }
424
425 // =====
426 // ===== UTILITY PREDICATES
427 pred samePerson[p1, p2: Person] {
428     p1.email = p2.email and p1.name=p2.name and p1.surname = p2.
            ↪ surname
429 }
430
431 pred invitedToAppointment[u: User , a: Appointment] {
432     some i: Invitation | i.forAppointment = a and i.toEmail = u.email
433 }
434
435 // =====
436 // ===== ASSERTIONS
437 assert CanDisplayInAllSupportedLanguages {
438     no l: AppInstance.installedOn.language |
439         l in SupportedLanguages.setOfLanguages and
440         no ap: AppInstance | ap.displayLanguage = l
441 }
442 check CanDisplayInAllSupportedLanguages
443
444 assert EveryPersonShouldBeAbleToHaveAnAccount {
445     #User < 0
446     implies
447     some p1, p2: Person , u: User |
448         p1.email = p2.email and p1.isUser = none and p2.isUser =
            ↪ u
449 }
450 check EveryPersonShouldBeAbleToHaveAnAccount
451
452 assert UserAlwaysNotified {
453     all a: Appointment , u: User |
454         (a.isIncoming = True and a in u.participatesToAppointment
            ↪ )
455         implies
456         (one n: Notification | n.toUser = u and n.
            ↪ incomingAppointment = a)
457 }
458 check UserAlwaysNotified for 2
459
460 // =====
461 // ===== RUNNABLE PREDICATES
462

```

```

463 pred showSomeAppointmentModified {
464     some ap: Appointment | ap.isModified = True
465 }
466 run showSomeAppointmentModified
467
468 pred showSomeMeeting {
469     some a: Appointment, disjoint u1, u2: User |
470         a in u1.participatesToAppointment and a in u2.
            ↪ participatesToAppointment
471 }
472 run showSomeMeeting for 4
473
474 pred show {
475     #Appointment = 2 and
476     #User = 1 and
477     #Notification  $\leq$  0 and
478     #TravelPlan  $\leq$  1 and
479     #Solution  $\leq$  1 and
480     #SupportedLanguages.setOfLanguages  $\leq$  1
481 }
482
483 run show for 5 but 4 Notification, 8 Int

```

---

## 5 Effort Spent and Team Work

We worked together most of the time, more than 30 hours. This is because in the beginning we struggled to find a way to divide responsibilities so we worked together and when we managed to work separately we still needed to discuss and to cross check our work.

Below, for each component of the team, we provide his list of main responsibilities, but it is important to say that we have absolutely collaborated on everything as said before.

**Cassarino Pietro**

**Salaris Mirko**

**Ventrella Piervincenzo**

## 6 References

### 6.1 Software and Tools

- L<sup>A</sup>T<sub>E</sub>X for typesetting document
- TeXstudio as L<sup>A</sup>T<sub>E</sub>X IDE
- GitHub for version control and team work
- Alloy latex highlighting package: <https://github.com/Angtrim/alloy-latex-highlighting>
- StarUML for uml models

### 6.2 Reference Documents

- Alloy reference: <http://alloy.mit.edu/alloy/documentation/book-chapters/alloy-language-reference.pdf>
- 29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering: <http://ieeexplore.ieee.org/document/6146379/>