Politecnico di Milano

A.Y. 2017-2018

TRAVLENDAR+

RASD

Requirements Analysis and Specification Document

Version 1.0

Greco Sonia *matricola* *893706*

Guzzo Francesco *matricola* *898035*

Release Date: 29.10.2017

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PURPOSE

Travlendar+ is a project which aims to create a calendar interface that automatically computes and handles the appointments of a person to make sure that the user is never late for his/her meetings and everyday activities. Travlendar+ also ensures that the user utilizes the best mobility option available at current time to reach the locations of his/her meetings, including walking, taking into account possible disabilities of the user, strikes and weather conditions. The user will be able to give personal constraints (e.g. It won't be suggested to use the car if the user doesn't have a driving license; walking distances should be less than a given threshold...) by activating or deactivating each travel means. The application will compute times between meetings and if the user wants to create a meeting which overlaps with events already in the calendar, the event won't be added. The application will also send a Warning message to the user if he wants to add an event that won't be reached in the due time because it is too far away, or there's a strike of the transport means, etc. The project also aims to keep a spot during the day if the user wants to have lunch in a given slot of time. A Warning message is also created when the user requests to add events during the slot of time that must be kept free for lunch. A user can specify if he/she wants to use the most ecological mobility mean, or the cheapest one, and insert his/her own mobility means.

1.2 SCOPE

Travlendar+ gives a useful support to users' life. Achieving the features explained above, the application helps the user making the best decision every day. It doesn't just keep track of all the appointments and meetings of a person, but it also makes sure that she/he can actually reach the different locations in time, it suggests the optimal way to reach them according to various aspects (weather, personal disabilities, owned mobility means, time of the day...).

1.3 REVISION HISTORY

First release of the document: 29th October 2017.

1.4 DOCUMENT STRUCTURE

The project will be formally analyzed in this document. Section 2 gives an overall description of the project. It includes the most important requirements, domain and text assumptions, class diagram and state chart of the main features of the application.

Section 3 analyzes in deeper details the goals of the projects, and the respective domain assumptions. Use cases are also included in this area and carefully explained.

Section 4 includes some possible scenarios of users who need this project in their everyday lives.

In section 5 we include our formal analysis of the project made using Alloy. We also show some *worlds* obtained by running our model.

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

Figure 2.1.1 and Figure 2.1.2 show, respectively, the Class Diagram and the State Chart of the overall project.

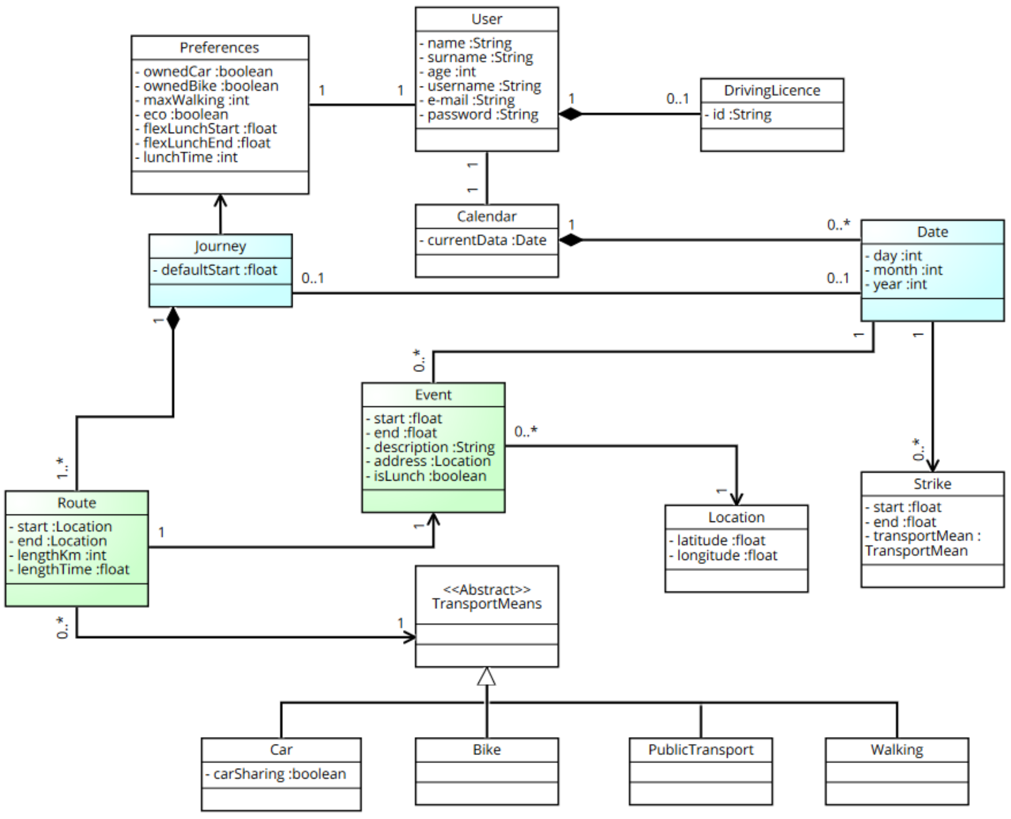


Figure 2.1.1. Class Diagram

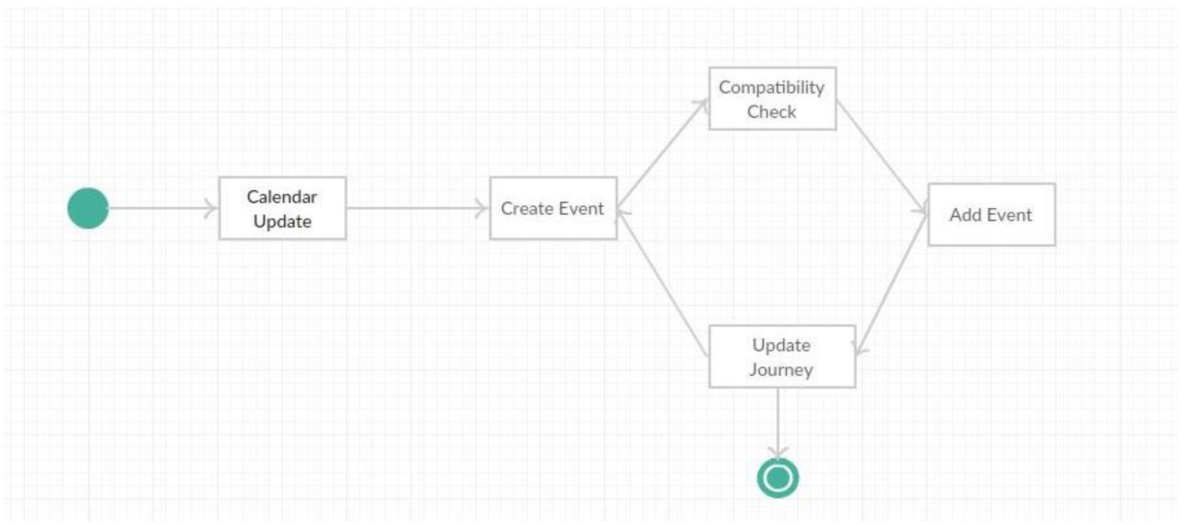


Figure 2.1.2. State Chart

2.2 PRODUCT FUNCTIONS

The most important requirements of the project are:

- the computed way from the starting point to the goal must be optimal:

  - it should be the shortest
  - it should choose the best mobility mean, taking into account the weather forecast
  - it should consider also other World related phenomena.

- the optimal way suggested, should also consider user's personal preferences, such as minimizing the carbon footprint, etc.

2.3 USER CHARACTERISTICS

The user of Travlendar+ is a person who needs help to schedule his/her daily appointments and to find the best way to reach them, according to a series of variables and personal constraints.

2.4 ASSUMPTIONS, DEPENDENCIES AND CONSTRAINTS

2.4.1. Domains

We suppose that these *domain properties* hold in the considered World:

• [D1] The Username chosen during the registration is unique.

• [D2] The login process needs User's Username or E-mail, and the corresponding password.

• [D3] The GPS of user's device is always on and always gives the correct position.

• [D4] Time, Date and Address are necessary to define uniquely a meeting.

• [D5] A public transport strike implies the impossibility to use the related public transport means.

• [D7] If there's a meeting in a given day, there must be at least an optimal way to reach the goal.

• [D8] The application refers to Google API in order to faithfully represent delays, arrivals and departures of every public transport mean, possible works on the roads or traffic jams at current time.

• [D9] The application should take into account the weather forecast when suggest the optimal way to reach the goal with a given mobility mean.

• [D10] During the working journey, there is always an amount of time kept for lunch break.

• [D11] Google's API considers only car, bike, walk and public transport means as transport means that can be chosen to travel through an itinerary.

• [D12] A journey can be created if and only if there is at least one appointment.

• [D13] Meetings and appointments must be already defined by the user before the computation of the best path.

• [D14] For the computation of the best ecological path, available travel means are considered in this order, from the minimum to the maximum carbon footprint: walking/bike, public transports, car/taxi.

• [D15] The system could also suggest the user to take a bike or a car from a mobility sharing system. Thus, we assume that in the interested area there are bike and car sharing systems and that the respective apps exist.

2.4.2. Assumptions

We assume that these *domain assumptions* hold in the considered World:

- There is one and only one calendar corresponding to one user.

- The public transport means are related to Milan's ATM.

- Initially the app will set available for the user all the transport means considered by google's API to calculate the best path.

- The system is able to get information about the current date and time from the used device.

- Maximum time walking for a person is 30 minutes.

- Maximum time biking for a person is 45 minutes.

- The application will calculate the best path for the user at the begin of the day considering information gathered from Google's API, weather news and user's set preferences.

- If the user wants to make ecological choices, the path will be also calculated considering the carbon footprint level of the different transport means.

- If the day will be rainy, even if the user wants to minimize carbon footprint, he/she will travel by car or public transports.

- The application will recalculate the path of the day only if the route is manually refreshed by the user.

- The application will recalculate during the day the optimal solution only to travel between the appointments already defined.

- The application doesn't keep track of the position of the user.

- The time for lunch considered by the app is fixed and chosen by the user.

- The user has to be able to fix a range of time during the day in which he/she could have lunch. It is 3 hours, by default. He/she can specify the length of lunch time, which is by default 30 minutes.

- If the user chooses to exploit a bike/car sharing mobility system, he/she already has the app for the interested service and handles the request of a transport mean oh his/her own.

- The application is also able to compute the best path in order to reach the least expensive journey, if the user chooses to activate that feature.

We suppose that these *constraints* hold in the considered world:

- Regulatory policies: the system must require to user the permission to get his position and he has to manage sensible data (position, credentials, e-mail) respecting the privacy law. Furthermore, the system must not use notifications to send SPAM respecting the privacy law.

- It is not possible to create meetings covering the all slot of time reserved for lunch break.

- Hardware limitations:

    - USER:

        - 3G connection
        - GPS
        - space for app package

- Interfaces to other applications

    - interface with Google API to get information about transport means, traffic jams, mobility schedules, etc.
    - interface with Weather Forecast

- the user shouldn't create overlapping meetings or unreachable ones. A meeting can be unreachable if it not possible to go there in a given slot of time (e.g. it is too far or there are no available transport means).

3. SPECIFIC REQUIREMENTS

3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 User interfaces

The following graphic interfaces represent a basic idea of what the mobile app will look like.

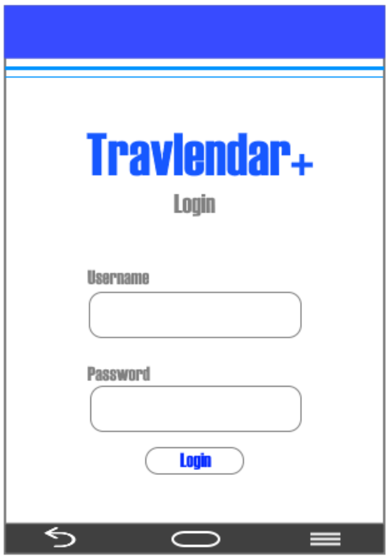


Figure 3.1.1.1. Login

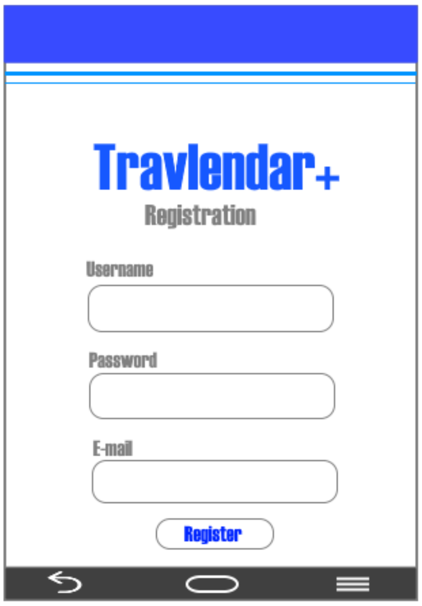


Figure 3.1.1.2. Registration

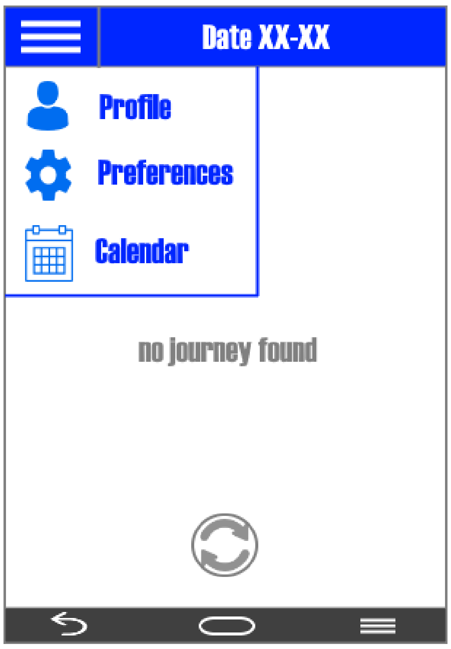


Figure 3.1.1.3. Menu

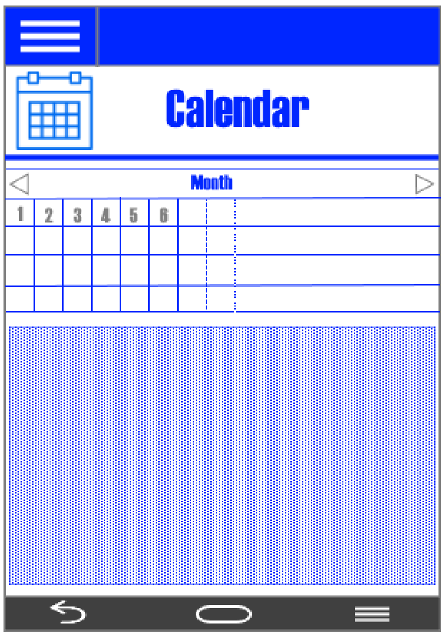


Figure 3.1.1.4. Calendar

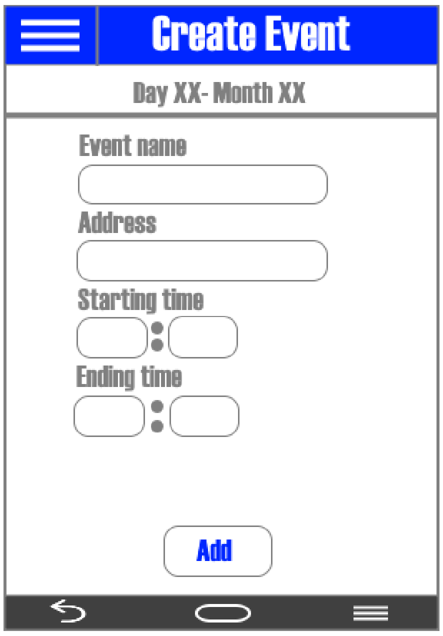


Figure 3.1.1.5. Event creation

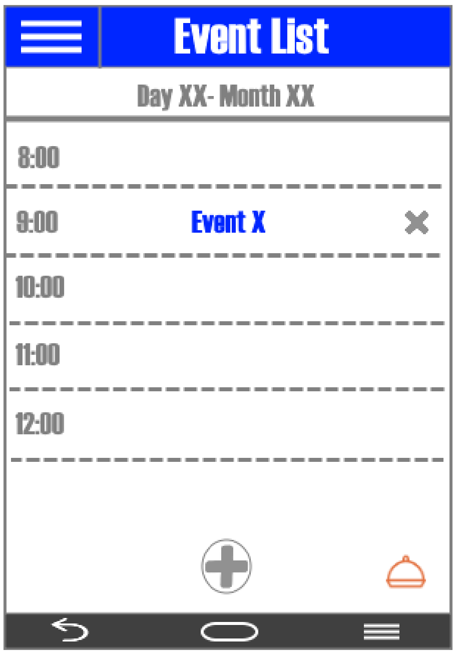


Figure 3.1.1.6. Event List (with lunch time option activated)

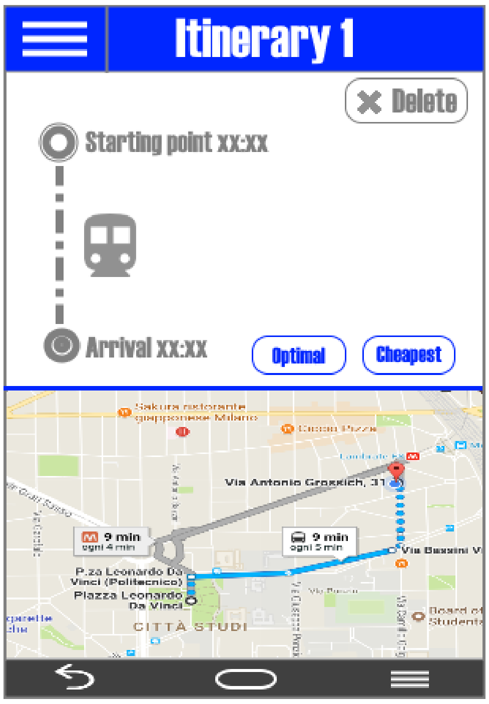
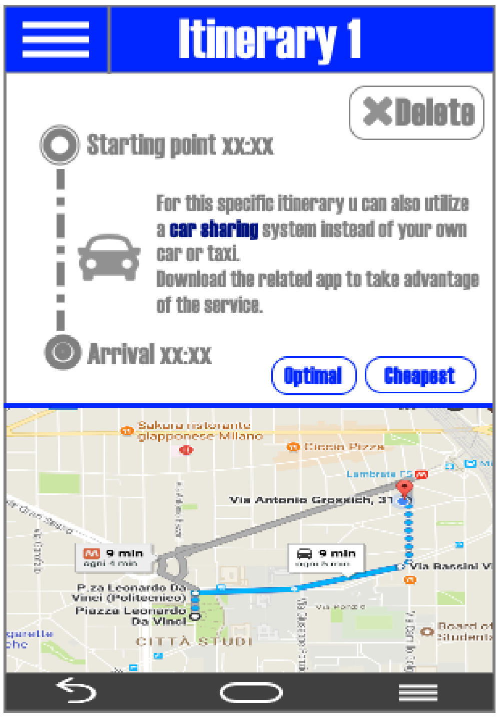
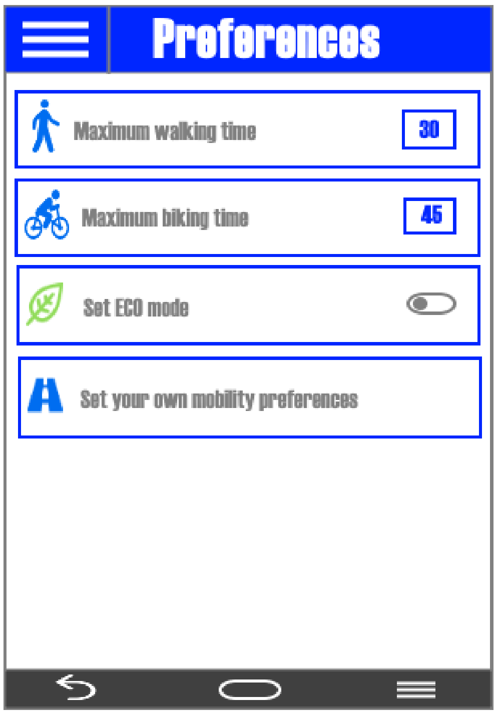


Figure 3.1.1.7. Itinerary (Public Transport)



For this specific itinerary u can also utilize a **car sharing** system instead of your own car or taxi.
Download the related app to take advantage of the service.

Figure 3.1.1.8. Itinerary (Car)



Preferences

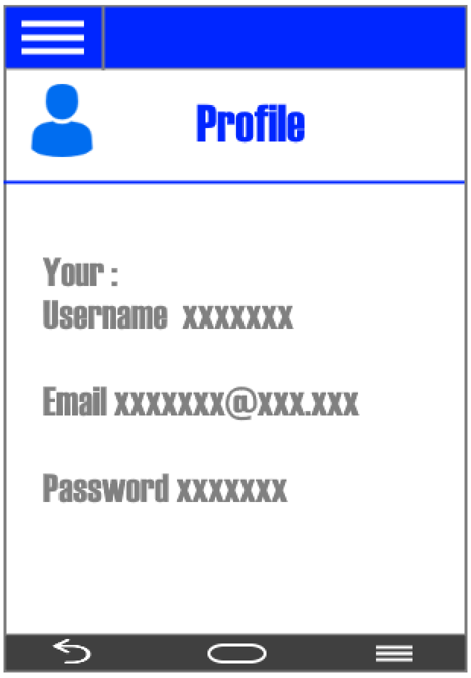
Maximum walking time 30

Maximum biking time 45

Set ECO mode

Set your own mobility preferences

Figure 3.1.1.9. Preferences



Profile

Your :
Username xxxxxxx

Email xxxxxxx@xxx.xxx

Password xxxxxxx

Figure 3.1.1.10. Personal profile

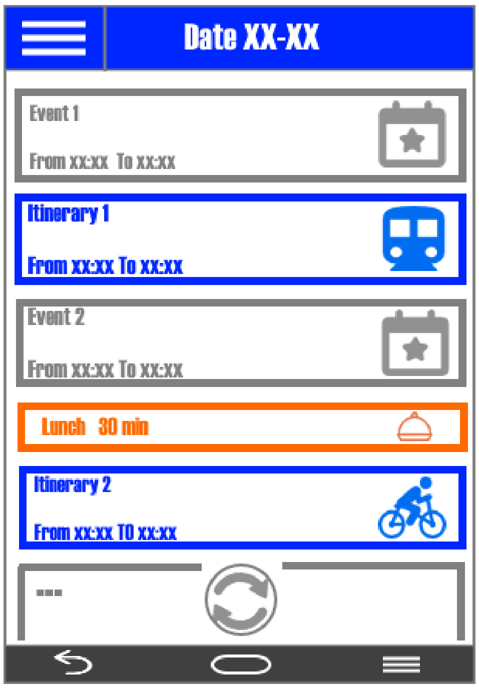


Figure 3.1.1.11. Journey schedule suggested

3.2 FUNCTIONAL REQUIREMENTS

3.2.1 [G1] Registration.

• [R1] A visitor must be able to begin the registration process.

• [R2] The password chosen for the registration must be at least of 7 characters and it must include at least one number and one capital letter.

• [D1] The Username chosen during the registration must be unique.

3.2.2 [G2] Login.

• [R3] The visitor must be already registered to go through the login.

• [R4] A registered User must be able to login to the system using his/her credentials.

• [D2] The logic process needs User's Username or E-mail, and the corresponding password.

3.2.3 [G3] Allow an User to create/delete meetings and send warning messages if needed.

• [R4] A registered User must be able to login to the system using his/her credentials.

• [R5] The system must ask the User to insert Date, Time and Address of the meeting he/she wants to create.

• [R6] An User must be able to delete a meeting if and only if it is already in the calendar.

• [R7] The system must notify the User with a *warning message* if the event he/she wants to create is not reachable (e.g. it is too far away or there is not enough time). If the event overlaps with events already in the calendar, it won't be created.

• [D3] The GPS of user's device is always on and always gives the correct position.

• [D4] Time, Date and Address are necessary to define uniquely a meeting.

• [D10] During the day, there is a fixed slot of time kept for breakfast/lunch time, chosen by the user.

3.2.4 [G4] Allow an User to notify a public transport strike.

• [R4] A registered User must be able to login to the system using his/her credentials.

• [R8] The User must be able to insert a public transport strike in the calendar to influence the calculation of the best path for those days.

• [D5] A public transport strike implies the impossibility to use public transport means.

3.2.5 [G5] Allow an User to insert personal preferences to modify the calculation of the best path.

• [R4] A registered User must be able to login to the system using his/her credentials.

• [R9] The User must be able to insert personal preferences and constraints:

- A maximum amount of time for walking
- A maximum amount of time for biking
- Choose not to use public transport means after a certain hour of the day

• [D6] The application should allow the user to register with his/her personal data, preferences and owned mobility means.

3.2.6 [G6] Allow an User to choose to minimize carbon footprint of the journey.

• [R4] A registered User must be able to login to the system using his/her credentials.

• [R10] A user has to have activated the Ecological feature in the preferences.

• [D6] The application should allow the user to register with his/her personal data, preferences and owned mobility means.

• [D14] For the computation of the best ecological path, available travel means are considered in this order, from the minimum to the maximum carbon footprint: walking/bike, public transports, car/taxi.

3.2.7 [G7] Allow an User to insert in the Preferences his/her owned mobility means.

• [R4] A registered User must be able to login to the system using his/her credentials.

• [D6] The application should allow the user to register with his/her personal data, preferences and owned mobility means.

3.2.8 [G8] Allow an User to choose the least expensive path in the computation of the best path.

• [R4] A registered User must be able to login to the system using his/her credentials.

• [R11] A user has to have activated the "LeastExpensive" feature in the preferences.

• [D6] The application should allow the user to register with his/her personal data, preferences and owned mobility means.

3.2.9 [G9] Allow an User to acknowledge the best path to follow to reach the daily meetings.

• [R4] A registered User must be able to login to the system using his/her credentials.

• [R12] There must be at least a meeting during the day.

• [D3] The GPS of user's device is always on and always gives the correct position.

• [D7] If there's a meeting in a given day, there must be at least an optimal way to reach the goal.

• [D8] The application refers to Google API in order to faithfully represent delays, arrivals and departures of every public transport mean, possible works on the roads or traffic jams at current time.

• [D9] The application should take into account the weather forecast when suggest the optimal way to reach the goal with a given mobility mean.

• [D15] The system could also suggest the user to take a bike or a car from a mobility sharing system. Thus, we assume that in the interested area there are bike and car sharing systems and that the respective apps exist.

3.2.10 [G10] Allow the application to notify a User if there are changes on the pre-defined route.

• [R4] A registered User must be able to login to the system using his/her credentials.

• [R12] There must be at least a meeting during the day.

• [R13] The system must have already computed the best route at least once.

• [D3] The GPS of user's device is always on and always gives the correct position.

• [D7] If there's a meeting in a given day, there must be at least an optimal way to reach the goal.

• [D8] The application refers to Google API in order to faithfully represent delays, arrivals and departures of every public transport mean, possible works on the roads or traffic jams at current time.

• [D9] The application should take into account the weather forecast when suggest the optimal way to reach the goal with a given mobility mean.

• [D15] The system could also suggest the user to take a bike or a car from a mobility sharing system. Thus, we assume that in the interested area there are bike and car sharing systems and that the respective apps exist.

3.2.11 [G11] Allow the application to keep a slot of time for lunch break, if requested.

• [R4] A registered User must be able to login to the system using his/her credentials.

• [R14] The user has to have chosen to activate the “LunchBreak” in the preferences, and a certain amount of time to reserve for it. It is by default 30 minutes.

• [D6] The application should allow the user to register with his/her personal data, preferences and owned mobility means.

3.2.12 Sequence diagrams

Figure 3.2.12.1, Figure 3.2.12.2, Figure 3.2.12.3 and Figure 3.2.12.4 show sequence diagrams of the four main actions a user can do with this application: register, login, create an event and receive the comptation of the best path to reach all the event of the day.

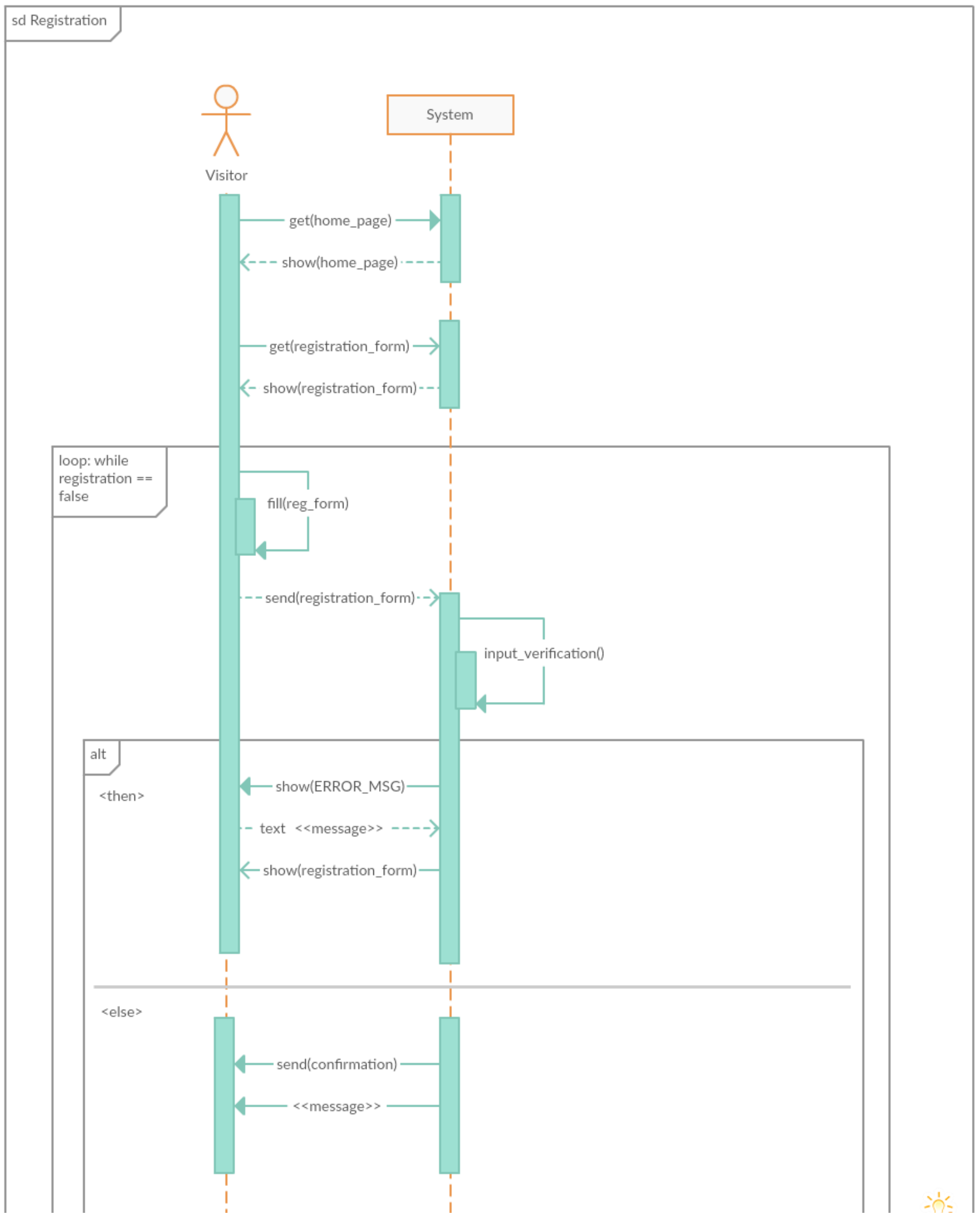


Figure 3.2.12.1. Registration – sequence diagram

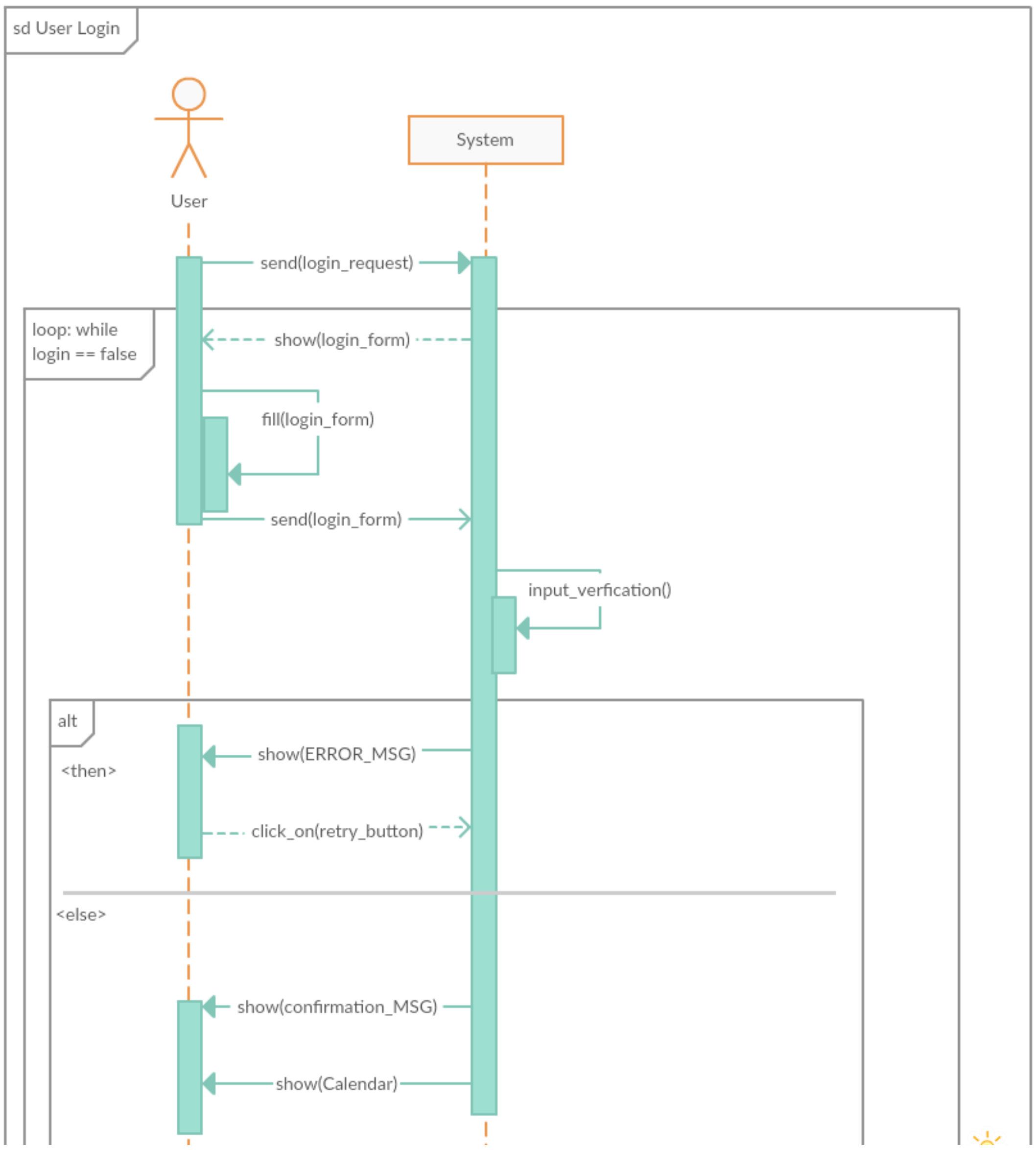


Figure 3.2.12.2. Login – sequence diagram

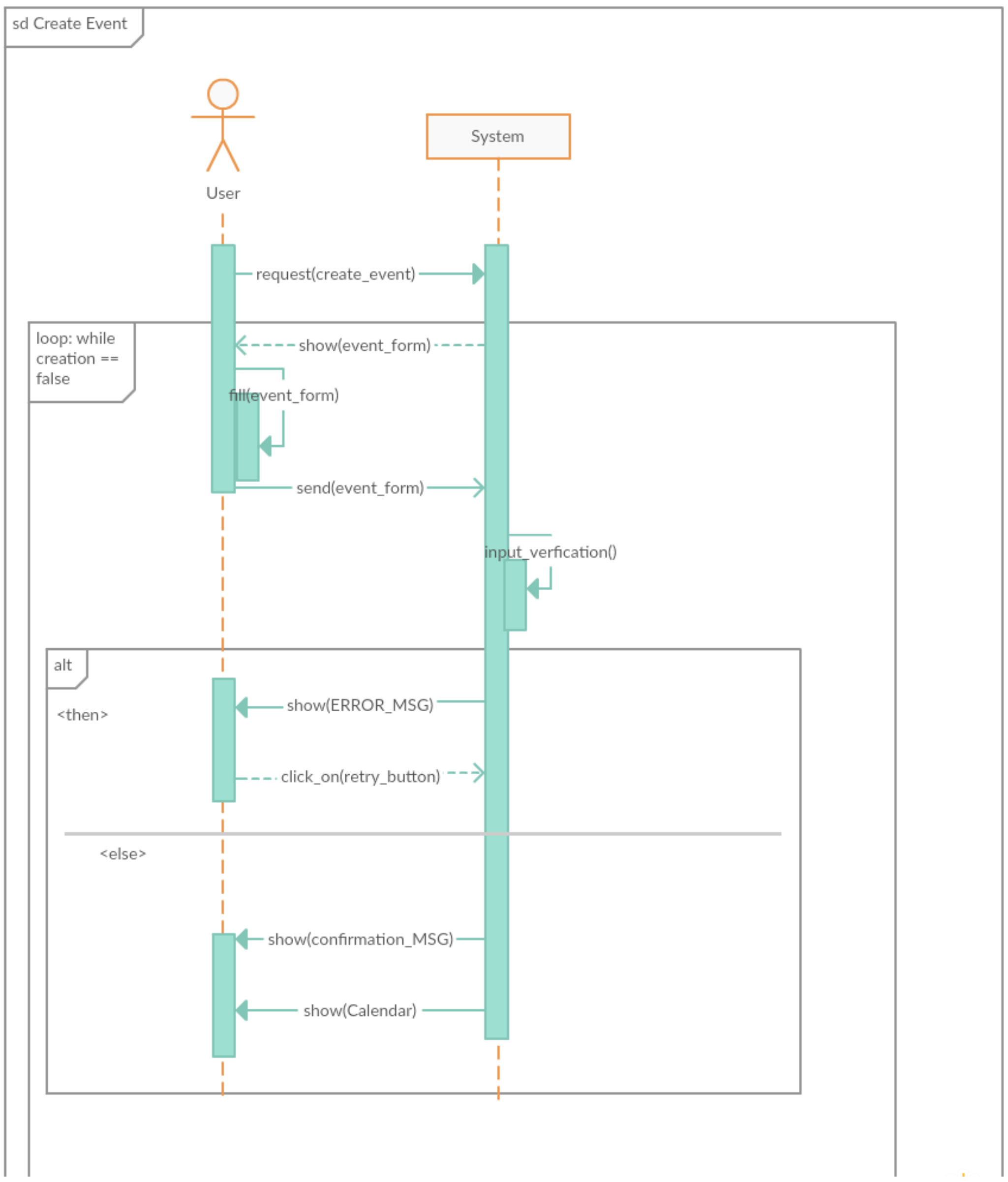


Figure 3.2.12.3. Event creation – sequence diagram

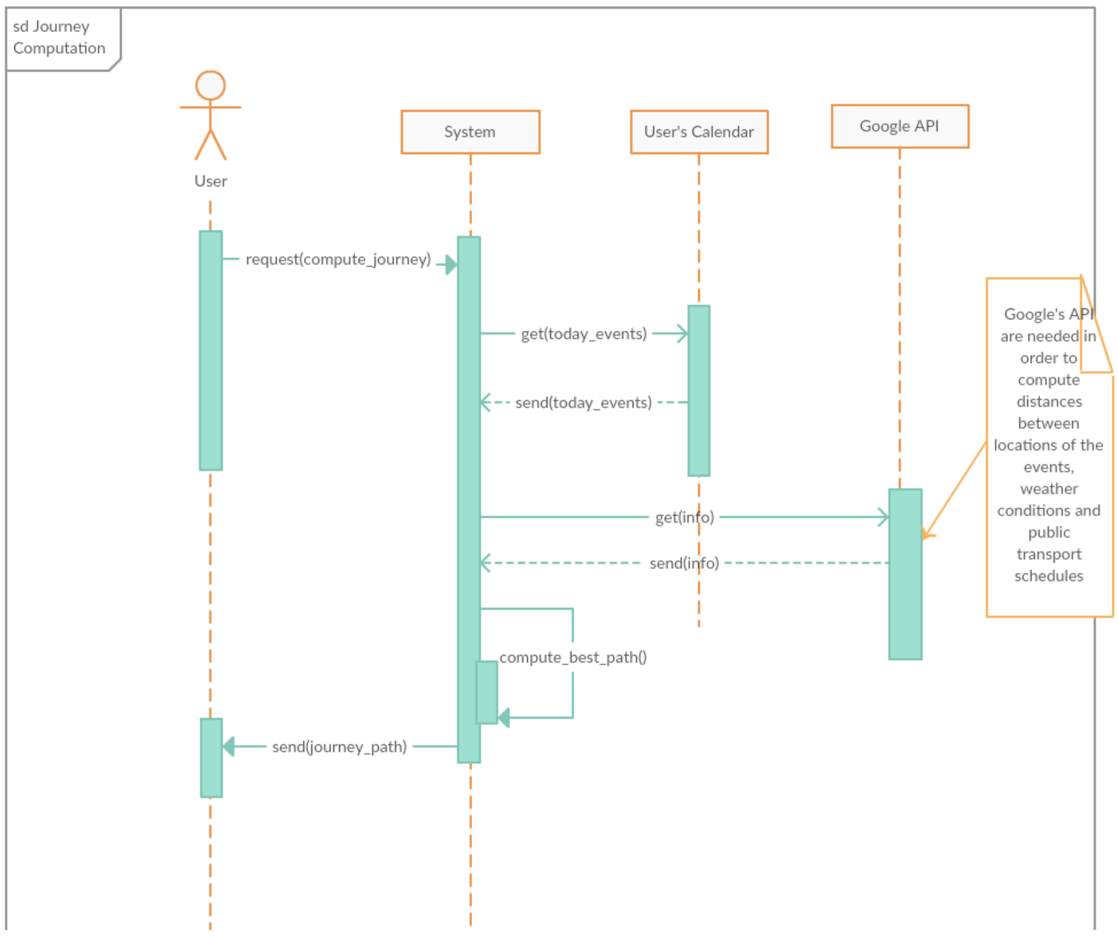


Figure 3.2.12.4. Computation of the best route – sequence diagram

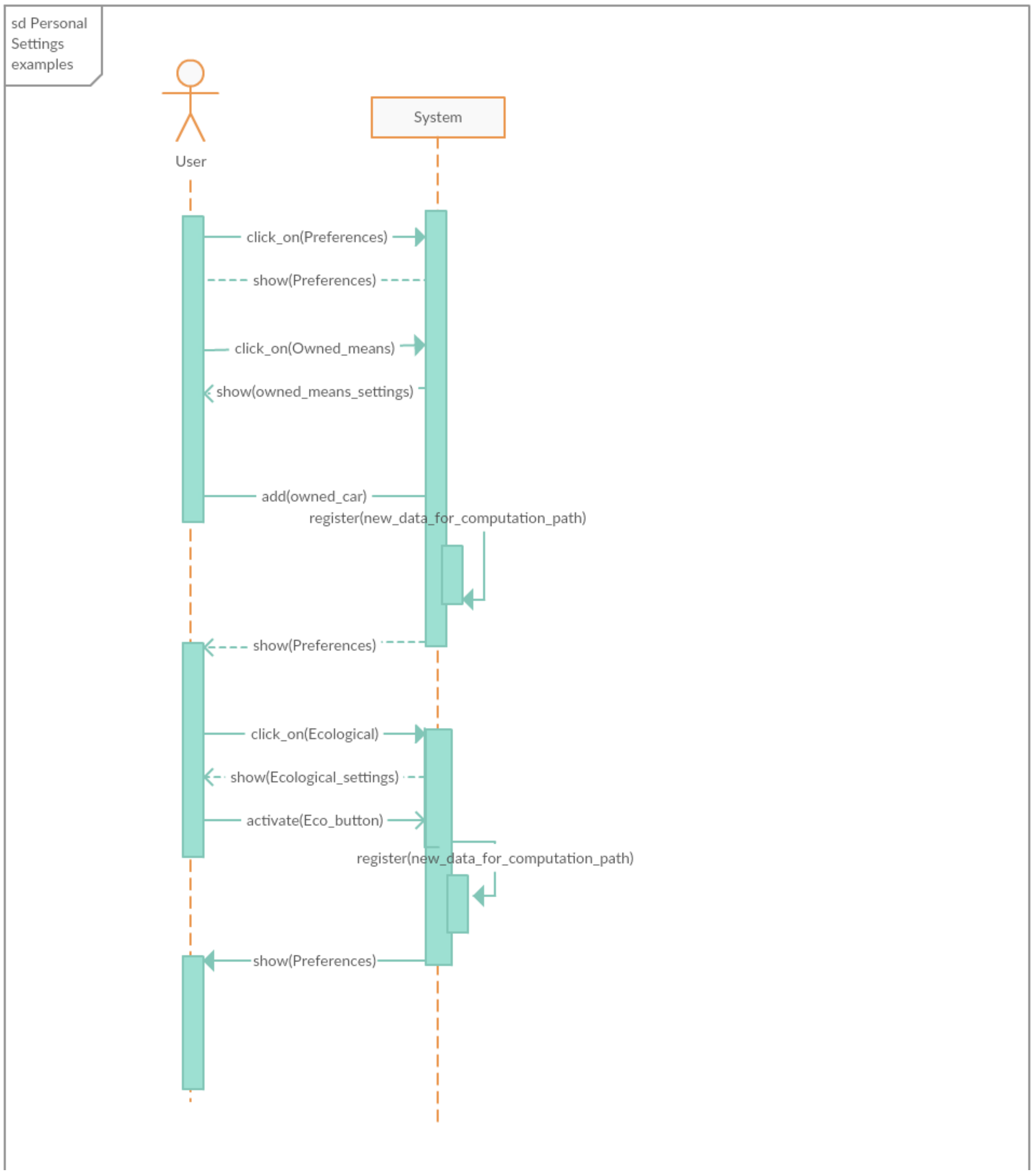

sd Personal Settings examples
User
System
click_on(Preferences)
show(Preferences)
click_on(Owned_means)
show(owned_means_settings)
add(owned_car)
register(new_data_for_computation_path)
show(Preferences)
click_on(Ecological)
show(Ecological_settings)
activate(Eco_button)
register(new_data_for_computation_path)
show(Preferences)

Figure 3.2.12.5. Setting personal preferences – sequence diagram

3.2.13 Use Cases

Figure 3.2.13.1 shows the complete use case diagram.

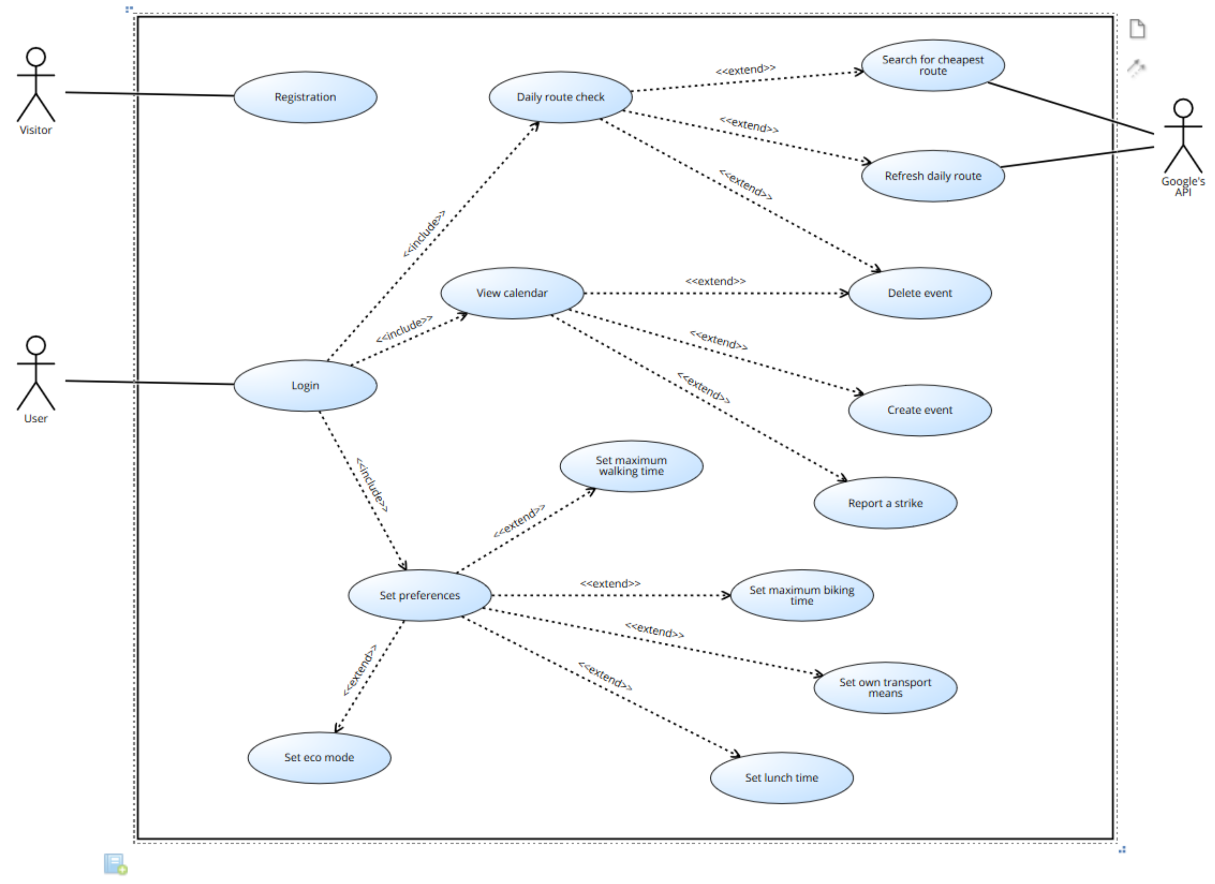


Figure 3.2.13.1. Use case diagram.

• Visitor's registration

ACTORS	Visitor
INPUT CONDITIONS	The Visitor has to have downloaded *Travlendar+*
EVENTS FLOW	1. The Visitor enters the home page ad clicks on the sign in button. 2. The Visitor starts the registration process filling all the mandatory fields with the requested information (user ID, password, email etc.). 3. The Visitor confirm the registration clicking on the Confirm button. 4. The System saves the data and sends an email to the Visitor to verify the previous registration.
OUTPUT CONDITIONS	The Visitor successfully ended the registration process and can now login in the application and start exploiting all its features.

EXCEPTIONS	1. The Visitor is already a User. 2. The Visitor did not insert correctly the information in the mandatory fields. 3. The username chosen by the Visitor already exists.

• User's login

ACTORS	User
INPUT CONDITIONS	The User must have successfully completed the registration process.
EVENTS FLOW	1. The User inserts his/her credentials: username and password. 2. The User clicks on the Login button. 3. The System redirects the User on the home page of the application.
OUTPUT CONDITIONS	The User is successfully redirected to the home page.
EXCEPTIONS	1.The User did not insert correctly the information in the mandatory fields (username or password).

• User creates an event

ACTORS	User
INPUT CONDITIONS	The User must have successfully completed the Login process.
EVENTS FLOW	1. The User enters the calendar section of the application from the main menu. 2. The User selects a specific day on the calendar in which he/she has to insert the appointment. 3. The User inserts all the information needed in the mandatory fields to complete the creation of an event: address, starting and ending time. 4. The User clicks on the button create event to successfully finish the event creation process. 5. The System gets the information and adds the event in the calendar. If the event is added in the current date, the system also recalculate the scheduled journey for that specific day. If the event is unreachable because of different reasons, a *warning message* is sent to the user.
OUTPUT CONDITIONS	The User has now successfully created an event and the relative schedule for that day.
EXCEPTIONS	1.The User did not insert correctly the information in the mandatory fields. 2.The event can't fill the already existent schedule because overlapping previous appointments.

• User deletes an appointment

ACTORS	User
INPUT CONDITIONS	The User has to have successfully completed the login process.
EVENTS FLOW	1. The User enters the calendar section of the application from the main menu. 2. The User selects a specific day on the calendar. 3. The User selects an event/appointment existent in the chosen schedule that he wants to delete. 4. The User clicks on the button delete event to successfully finish the event deletion process. 5. The System gets the information and starts processing a new route for the remaining events in the schedule.
OUTPUT CONDITIONS	The User has now successfully deleted an event in the relative schedule for that day.
EXCEPTIONS	There are no possible exceptions.

• User reports a strike

ACTORS	User
INPUT CONDITIONS	The User has to have successfully completed the login process.
EVENTS FLOW	1. The User enters the calendar section of the application from the main menu. 2. The User selects a specific day on the calendar. 3. The User clicks on the button report strike. 4. The User fills all the mandatory fields reporting the type of the strike and its duration. 5. The System gets the information and starts processing a new route for the events in the schedule.
OUTPUT CONDITIONS	The User has now successfully reported a strike.
EXCEPTIONS	There are no possible exceptions.

• User checks the daily schedule

ACTORS	User
INPUT CONDITIONS	The User has to have successfully completed the login process and created a daily schedule.
EVENTS FLOW	1. The User enters the main page of the application. 2. The user can select the different sections of the programmed route to extinguish the daily journey.

	3. The user can gather information from the specific itinerary he selected and check on the map the route he will go through. 4. The User can check all the warnings related to previous recalculation of the daily route.
OUTPUT CONDITIONS	The User has acknowledged the daily path.
EXCEPTIONS	There are no possible exceptions.

- User refreshes the daily schedule

ACTORS	User
INPUT CONDITIONS	The User has to have successfully completed the login process and created a daily schedule.
EVENTS FLOW	1. The User enters the main page of the application. 2. The user can click on the button "Refresh" to recalculate the daily itineraries.
OUTPUT CONDITIONS	The User has refreshed the daily path.
EXCEPTIONS	There are no possible exceptions.

- User sets a maximum amount of walking time

ACTORS	User
INPUT CONDITIONS	The User has to have successfully completed the login process.
EVENTS FLOW	1. The User enters the main page of the application. 2. The User enters the main menu of the application. 3. The User go on his profile page e clicks on the button "Preferences". 4. The User enters the "Maximum walking time" section. 5. The User inserts the maximum time of walking he wants to effort.
OUTPUT CONDITIONS	The System has got the information that will influence next path calculations.
EXCEPTIONS	There are no possible exceptions.

- User sets a maximum amount of biking time

ACTORS	User
INPUT CONDITIONS	The User has to have successfully completed the login process.

EVENTS FLOW	1. The User enters the main page of the application. 2. The User enters the main menu of the application. 3. The User go on his profile page e clicks on the button “Preferences”. 4. The User enters the “Maximum biking time” section. 5. The User inserts the maximum time of biking he wants to effort.
OUTPUT CONDITIONS	The System has got the information that will influence next path calculations.
EXCEPTIONS	There are no possible exceptions.

• User sets the option to minimize the carbon footprint

ACTORS	User
INPUT CONDITIONS	The User has to have successfully completed the login process.
EVENTS FLOW	1. The User enters the main page of the application. 2. The User enters the main menu of the application. 3. The User go on his profile page e clicks on the button “Preferences”. 4. The User ticks the “Ecological choice” field. .
OUTPUT CONDITIONS	The System has got the information that will influence next path calculations.
EXCEPTIONS	There are no possible exceptions.

• User defines his/her own transport mean configuration

ACTORS	User
INPUT CONDITIONS	The User has to have successfully completed the login process.
EVENTS FLOW	1. The User enters the main page of the application. 2. The User enters the main menu of the application. 3. The User go on his profile page e clicks on the button “Preferences”. 4. The User goes in the “Available transport means” section. 5. The User defines his own transport means configuration.

OUTPUT CONDITIONS	The System has got the information that will influence next path calculations.
EXCEPTIONS	There are no possible exceptions.

- User sets a fixed lunch duration

ACTORS	User
INPUT CONDITIONS	The User has to have successfully completed the login process.
EVENTS FLOW	1. The User enters the main page of the application. 2. The User enters the main menu of the application. 3. The User go on his profile page e clicks on the button "Preferences". 4. The User can modify the fixed lunch duration and set up a time range during the day in which he prefers to have lunch.
OUTPUT CONDITIONS	The System has got the information that will influence next path calculations.
EXCEPTIONS	There are no possible exceptions.

- User searching for the cheapest route

ACTORS	User
INPUT CONDITIONS	The User has to have successfully completed the login process and created a daily schedule.
EVENTS FLOW	1. The User enters the main page of the application. 2. The user can click on the itinerary he is interested on. 3.He clicks on the button "Cheapest". 4.The map will show him an itinerary cheaper than the optimal one that was already found.
OUTPUT CONDITIONS	The User has refreshed the daily path.
EXCEPTIONS	There are no possible exceptions.

3.3 PERFORMANCE REQUIREMENTS

The system has to be able to correctly extract information from Google's API and compute the best route for the day exploiting them. The project needs to get information about weather, traffic jams conditions, schedules of the public transport means and sharing mobility systems in the considered area.

3.4 SOFTWARE SYSTEMS ATTRIBUTES

3.4.1 Reliability

The system must guarantee 24/7 service, especially during the part of the day included in the planned journey. Reliability of the application is also strictly related to the information got from Google's API about weather and traffic jams condition, schedules of public transport means and sharing systems mobility means.

3.4.2 Availability

The system must be available 24/7, allowing the user to create or delete, whenever he/she wants, meetings or appointments on the calendar. The application also needs to recalculate a journey if the user adds events in the current date, so this characteristic is highly required.

3.4.3 Security

User's credentials and passwords are stored in the system. Their security and user's privacy is highly taken into account among the primary concerns of this project's development.

3.4.4 Maintainability

The application will be periodically updated to fix possible bugs and improve user's experience.

3.4.5 Portability

The application runs in all the devices which have internet access, with at least 3G connection for smartphones (Android/iOS). The application could also be developed in order to become a web application.

4. SCENARIOS

4.1 SCENARIO 1 – Ecological feature

Walter is a chemistry professor. He has always gone to work using his car, but he now wants to use more environmental friendly mobility means. He doesn't have a bike, so he uses Travlendar+ daily to find out which is the available path with the less carbon footprint. Walter cannot arrive late at school and he doesn't know the schedule of the public transport means. He's happy to use the app, because it tells him when the chosen transport mean is coming and it gives him the possibility to buy the tickets for it.

4.2 SCENARIO 2 – Event scheduling

Laura is the director of a chain of clothing stores and has a very busy life. Besides her job in her office, she also has to schedule appointments with all the shop managers in the area of Milan. Every appointment must take place in a specific shop because Laura also has to check the conditions of the building and the quality of service of her employees. In order to schedule all her appointments during the day and plan her movements in the city, Laura relies on Travlendar+. She'll always be sure to have a slot of time for a lunch break.

4.3 SCENARIO 3 – Best path computation

Marco works in a bank from 8 AM until 4 PM every day. He goes to the gym on Mondays and Thurdays at 7 PM, every week. Lately, the director of the bank asked him to do extra hours at work. He needs assistence on planning the fastest way from work to the gym, because he doesn't want to give up on training, but he also has to be at home at latest at 9PM every day. He goes to work by car, so Travlendar+ is a useful mean for him, because it keeps him updated with the traffic situation and helps him finding the best way to reach the goal in the fastest time.

4.4 SCENARIO 4 – Warning message assistence

Chiara is a doctor. Her weekly duties are strictly dependent to her work schedule. She uses Travlendar+ to keep track of her daily shifts and appointments with private patients. Given the old age of most of her private patients, she would like to visit them in their houses, in the morning, between 8 and 11 AM, in order to let them rest in the afternoon. Thus, she needs help to plan the fastest route to reach all the location,in order to accomplish the goal of meeting all the patients and not make them wait for more than 10 minutes. As a matter of fact, the application will send her a warning message if she wants to organize an appointment in a given day and in a given location, but she won't be able to reach it in time.

4.5 SCENARIO 5 – Personal constraints feature

Filippo is a retired man of 72 years of age. He finds it difficult to walk for more than 10 minutes, without feeling tired and sore. Daniele, his nephew, helped him organizing his weekly schedule, adding all the meetings to which he has to participate, in the calendar of Travlendar+ and specifying all his personal needs. Wherever he has to go, the app will never tell him to make a journey which includes more than 10 minutes of walking, driving a bike or a car, because he specified so in the preferences section during the registration process. On the contrary, he will get suggestions about the fastest, or more suitable, public transport mean to take, to reach his goals.

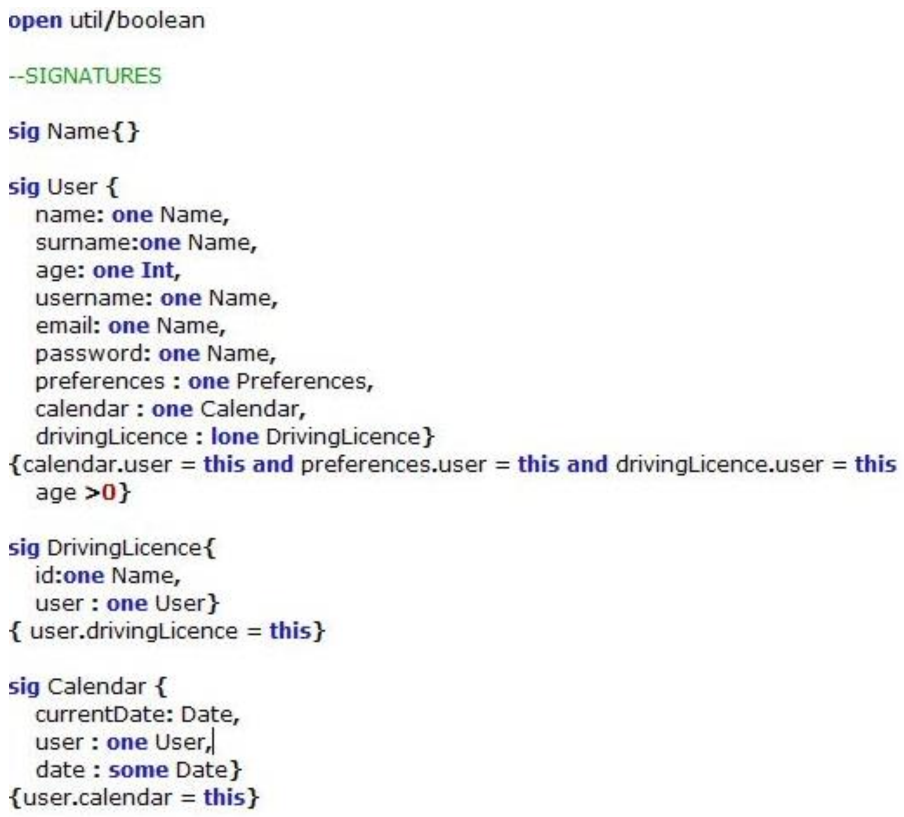
4.6 SCENARIO 6 – Fastest route computation

Elena is a convinced ecologist and she always uses her bike, wherever she has to move in the city of Milan. She works for a company which make cruelty-free and vegan cosmetics and her job is to go to customers' house, to give in what they ordered and bought in the online shop. Her priority is to deliver all the packages during her morning shift, between 9 and 12 AM, because she works part time and she won't get extra money for extra hours. Calculating the best route for a bike, for every situation would be hard without Travlendar+. Since she has it, she always manages to carry out all the deliveries in the given time.

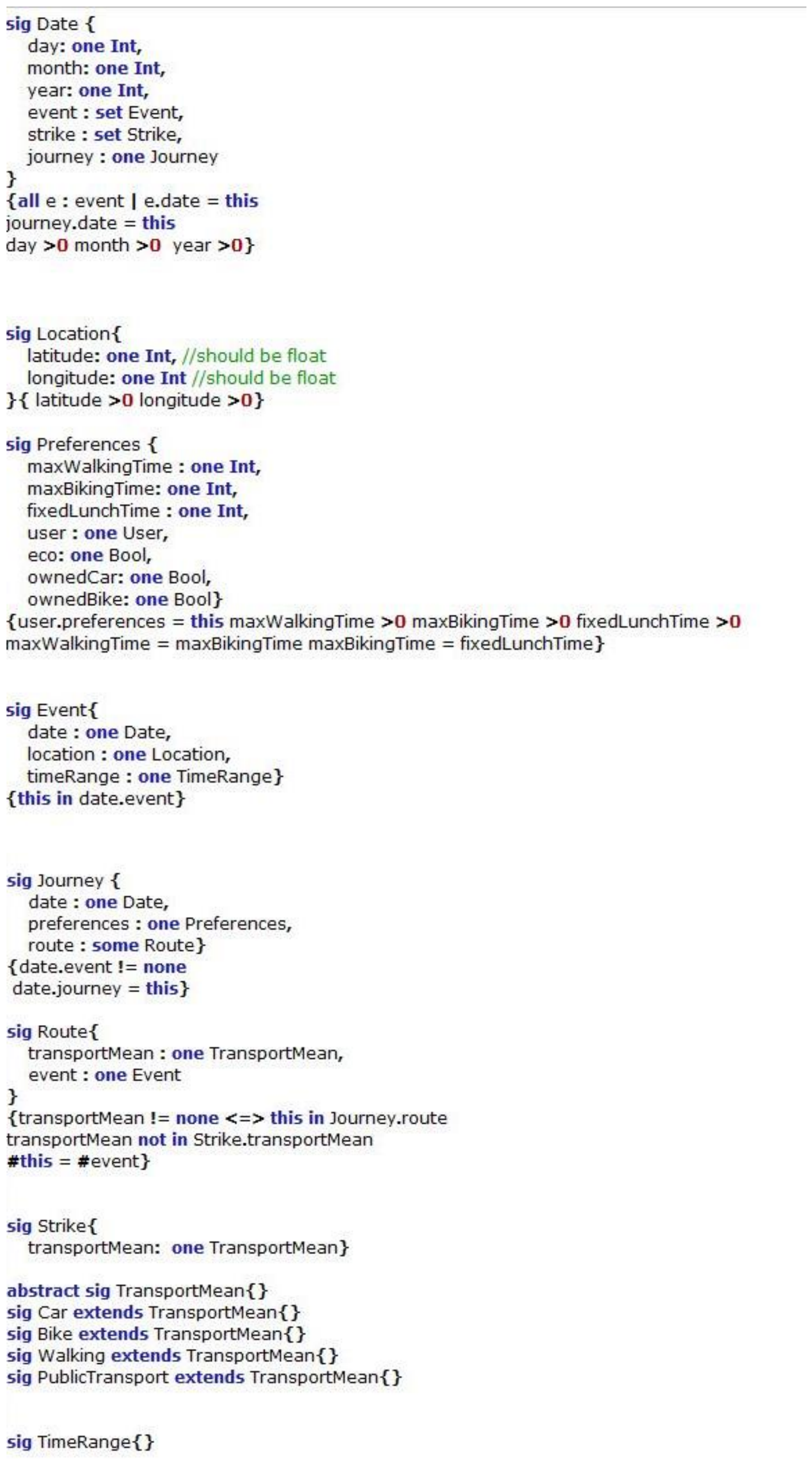
5. FORMAL ANALYSIS USING ALLOY

In this section we provide the formal analysis of the project made with Alloy. The original document is also included in the Delivery Folder.

We report the main Signatures included in the *world*.

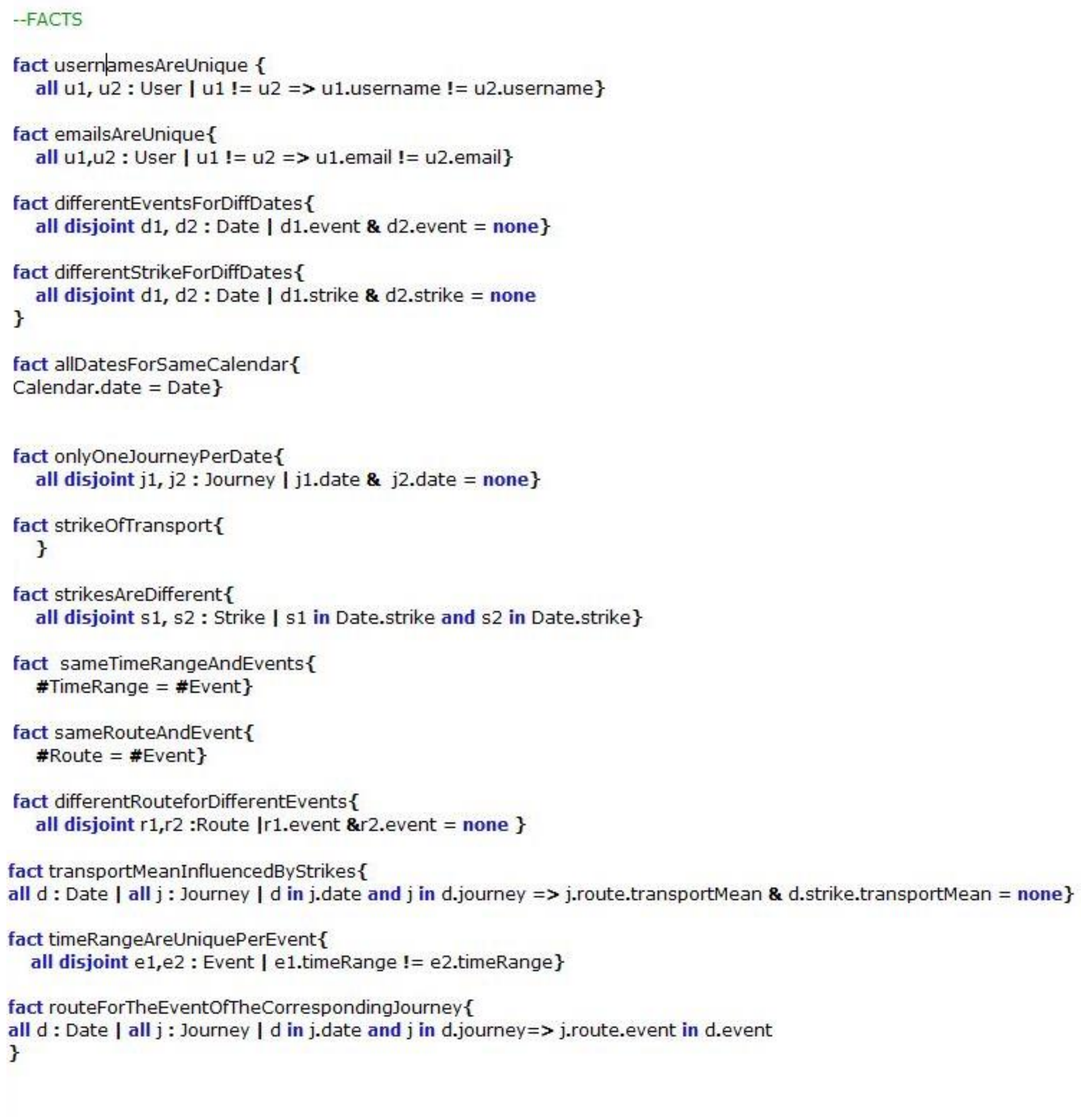


```
open util/boolean

--SIGNATURES

sig Name{}

sig User {
   name: one Name,
   surname:one Name,
   age: one Int,
   username: one Name,
   email: one Name,
   password: one Name,
   preferences : one Preferences,
   calendar : one Calendar,
   drivingLicence : lone DrivingLicence}
{calendar.user = this and preferences.user = this and drivingLicence.user = this
   age >0}

sig DrivingLicence{
   id:one Name,
   user : one User}
{ user.drivingLicence = this}

sig Calendar {
   currentDate: Date,
   user : one User,
   date : some Date}
{user.calendar = this}
```



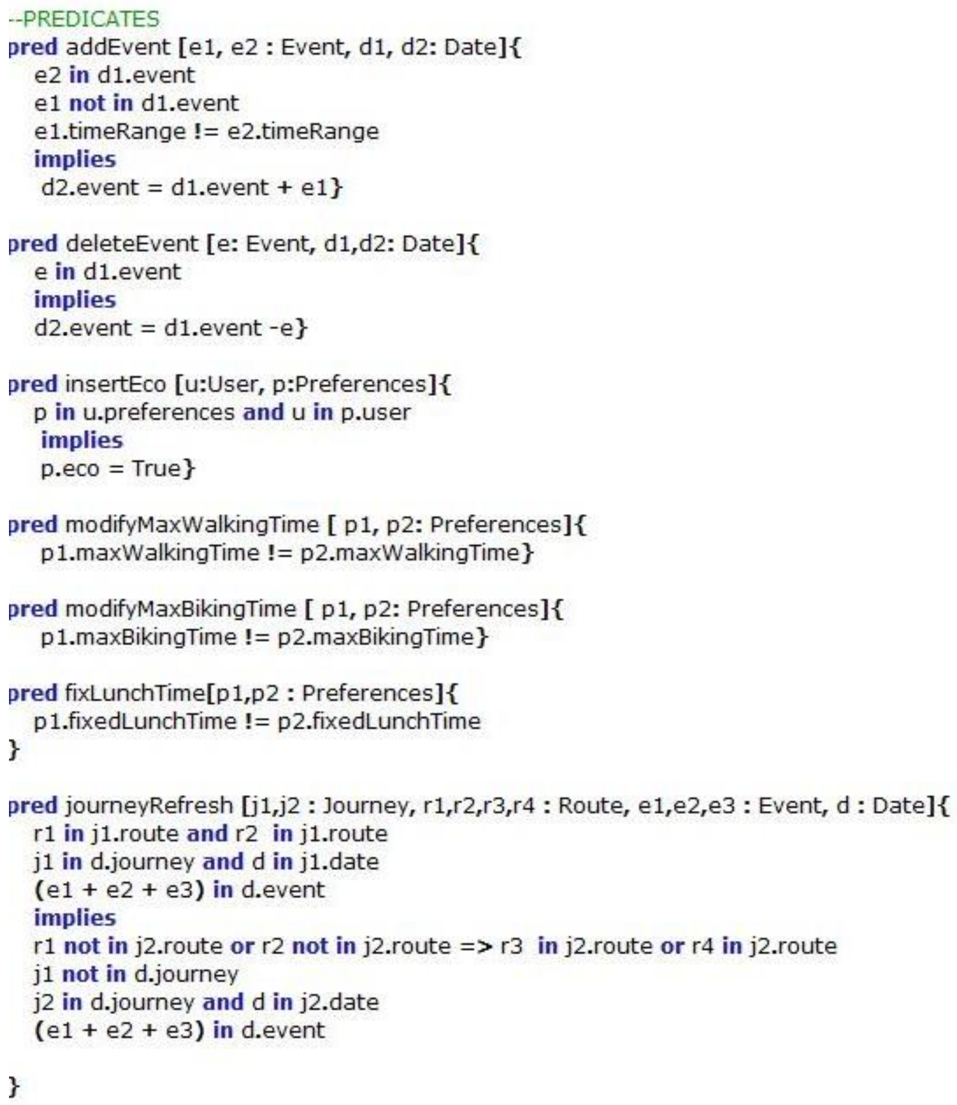
```
sig Date {
   day: one Int,
   month: one Int,
   year: one Int,
   event : set Event,
   strike : set Strike,
   journey : one Journey
}
{all e : event | e.date = this
journey.date = this
day >0 month >0  year >0}



sig Location{
   latitude: one Int, //should be float
   longitude: one Int //should be float
}{ latitude >0 longitude >0}

sig Preferences {
   maxWalkingTime : one Int,
   maxBikingTime: one Int,
   fixedLunchTime : one Int,
   user : one User,
   eco: one Bool,
   ownedCar: one Bool,
   ownedBike: one Bool}
{user.preferences = this maxWalkingTime >0 maxBikingTime >0 fixedLunchTime >0
maxWalkingTime = maxBikingTime maxBikingTime = fixedLunchTime}


sig Event{
   date : one Date,
   location : one Location,
   timeRange : one TimeRange}
{this in date.event}



sig Journey {
   date : one Date,
   preferences : one Preferences,
   route : some Route}
{date.event != none
 date.journey = this}

sig Route{
   transportMean : one TransportMean,
   event : one Event
}
{transportMean != none <=> this in Journey.route
transportMean not in Strike.transportMean
#this = #event}


sig Strike{
   transportMean:  one TransportMean}

abstract sig TransportMean{}
sig Car extends TransportMean{}
sig Bike extends TransportMean{}
sig Walking extends TransportMean{}
sig PublicTransport extends TransportMean{}


sig TimeRange{}
```

The facts included in the model are shown below.



```
--FACTS

fact usernamesAreUnique {
   all u1, u2 : User | u1 != u2 => u1.username != u2.username}

fact emailsAreUnique{
   all u1,u2 : User | u1 != u2 => u1.email != u2.email}

fact differentEventsForDiffDates{
   all disjoint d1, d2 : Date | d1.event & d2.event = none}

fact differentStrikeForDiffDates{
   all disjoint d1, d2 : Date | d1.strike & d2.strike = none
}

fact allDatesForSameCalendar{
Calendar.date = Date}


fact onlyOneJourneyPerDate{
   all disjoint j1, j2 : Journey | j1.date &  j2.date = none}

fact strikeOfTransport{
   }

fact strikesAreDifferent{
   all disjoint s1, s2 : Strike | s1 in Date.strike and s2 in Date.strike}

fact  sameTimeRangeAndEvents{
   #TimeRange = #Event}

fact sameRouteAndEvent{
   #Route = #Event}

fact differentRouteforDifferentEvents{
   all disjoint r1,r2 :Route |r1.event &r2.event = none }

fact transportMeanInfluencedByStrikes{
all d : Date | all j : Journey | d in j.date and j in d.journey => j.route.transportMean & d.strike.transportMean = none}

fact timeRangeAreUniquePerEvent{
  all disjoint e1,e2 : Event | e1.timeRange != e2.timeRange}

fact routeForTheEventOfTheCorrespondingJourney{
all d : Date | all j : Journey | d in j.date and j in d.journey=> j.route.event in d.event
}
```

Lastly, we include the predicates of the analysis.



```
--PREDICATES
pred addEvent [e1, e2 : Event, d1, d2: Date]{
   e2 in d1.event
   e1 not in d1.event
   e1.timeRange != e2.timeRange
   implies
    d2.event = d1.event + e1}

pred deleteEvent [e: Event, d1,d2: Date]{
   e in d1.event
   implies
   d2.event = d1.event -e}

pred insertEco [u:User, p:Preferences]{
   p in u.preferences and u in p.user
    implies
    p.eco = True}

pred modifyMaxWalkingTime [ p1, p2: Preferences]{
    p1.maxWalkingTime != p2.maxWalkingTime}

pred modifyMaxBikingTime [ p1, p2: Preferences]{
    p1.maxBikingTime != p2.maxBikingTime}

pred fixLunchTime[p1,p2 : Preferences]{
   p1.fixedLunchTime != p2.fixedLunchTime
}

pred journeyRefresh [j1,j2 : Journey, r1,r2,r3,r4 : Route, e1,e2,e3 : Event, d : Date]{
   r1 in j1.route and r2  in j1.route
   j1 in d.journey and d in j1.date
   (e1 + e2 + e3) in d.event
   implies
   r1 not in j2.route or r2 not in j2.route => r3  in j2.route or r4 in j2.route
   j1 not in d.journey
   j2 in d.journey and d in j2.date
   (e1 + e2 + e3) in d.event

}
```

With our analysis of the world, we wanted to prove that the logic behind our design held. Figure 5.1 shows the graph of the structure with the main attributes of the signatures.

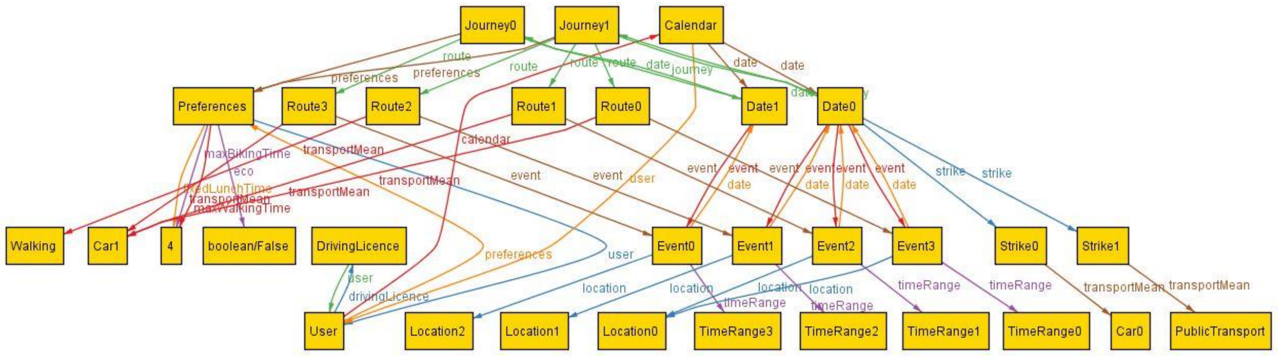
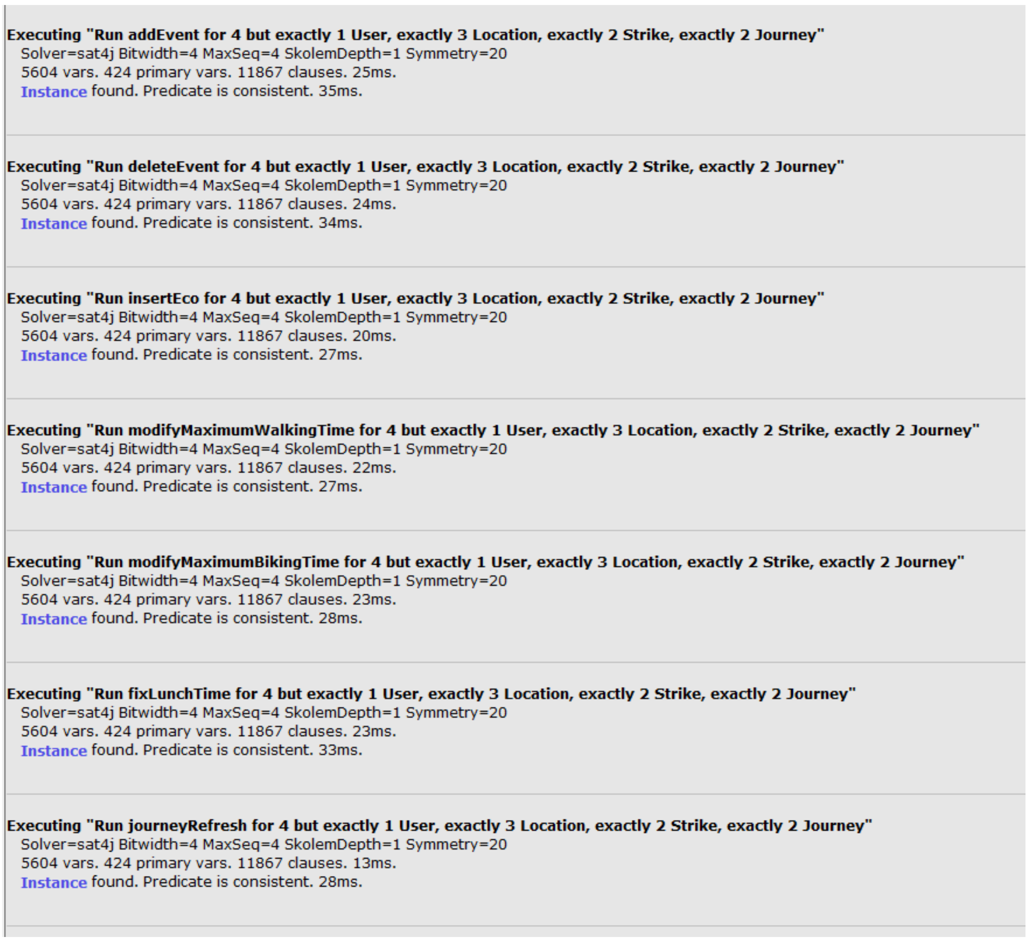


Figure 5.1. World graph from Alloy

We make the software analyze our model, as shown in Figure 5.2.



```
Executing "Run addEvent for 4 but exactly 1 User, exactly 3 Location, exactly 2 Strike, exactly 2 Journey"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   5604 vars. 424 primary vars. 11867 clauses. 25ms.
   Instance found. Predicate is consistent. 35ms.

Executing "Run deleteEvent for 4 but exactly 1 User, exactly 3 Location, exactly 2 Strike, exactly 2 Journey"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   5604 vars. 424 primary vars. 11867 clauses. 24ms.
   Instance found. Predicate is consistent. 34ms.

Executing "Run insertEco for 4 but exactly 1 User, exactly 3 Location, exactly 2 Strike, exactly 2 Journey"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   5604 vars. 424 primary vars. 11867 clauses. 20ms.
   Instance found. Predicate is consistent. 27ms.

Executing "Run modifyMaximumWalkingTime for 4 but exactly 1 User, exactly 3 Location, exactly 2 Strike, exactly 2 Journey"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   5604 vars. 424 primary vars. 11867 clauses. 22ms.
   Instance found. Predicate is consistent. 27ms.

Executing "Run modifyMaximumBikingTime for 4 but exactly 1 User, exactly 3 Location, exactly 2 Strike, exactly 2 Journey"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   5604 vars. 424 primary vars. 11867 clauses. 23ms.
   Instance found. Predicate is consistent. 28ms.

Executing "Run fixLunchTime for 4 but exactly 1 User, exactly 3 Location, exactly 2 Strike, exactly 2 Journey"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   5604 vars. 424 primary vars. 11867 clauses. 23ms.
   Instance found. Predicate is consistent. 33ms.

Executing "Run journeyRefresh for 4 but exactly 1 User, exactly 3 Location, exactly 2 Strike, exactly 2 Journey"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   5604 vars. 424 primary vars. 11867 clauses. 13ms.
   Instance found. Predicate is consistent. 28ms.
```

Figure 5.2. Predicates analys from Alloy

Our aim is to check if the actions of the User are correctly executed.

With the first predicate we check that, when the User adds an event, it is really added in the event set of the respective date and it doesn't overlap with any other event already included.

Then, we check if the deletion of an event is correctly executed, which means that after that point, the event won't be in the set of events of that day anymore.

Later on we control that the user can modify his/her personal preferences in the correct way and finally, that the refreshing of the computation of the best journey path is correct.

6. EFFORT SPENT

The total amount of hours spent working on the project:

Sonia Greco	Date	Time
	01.10.2017	3 h
	07.10.2017	1 h
	10.10.2017	5 h
	17.10.2017	5 h
	19.10.2017	4 h
	24.10.2017	6 h
	25.10.2017	5 h
	27.10.2017	6 h
	28.10.2017	4 h
Francesco Guzzo	Date	Time
	01.10.2017	3 h
	07.10.2017	1 h
	10.10.2017	5 h
	17.10.2017	5 h
	19.10.2017	4 h
	23.10.2017	7 h
	24.10.2017	6 h
	25.10.2017	3 h
	26.10.2017	4 h
	28.10.2017	3 h