POLITECNICO
MILANO 1863

Software Engineering 2
Politecnico di Milano, A.A. 2017/2018

TravIendar+

Requirement Analysis and Specifications Document

Version: 2.0
Release date: 7/1/2018

Riccardo Novic (mat. 829184)
Lorenzo Pratissoli (mat. 893980)

1. **Introduction**
   - 1.1. About this document
   - 1.2. Purpose
   - 1.3. Scope
     - 1.3.1. World and shared phenomena
     - 1.3.2. Actors
     - 1.3.3. Goals
   - 1.4. Definitions, Acronyms, Abbreviations
     - 1.4.1. Definitions
     - 1.4.2. Acronyms
     - 1.4.3. Abbreviations
   - 1.5. Revision history
   - 1.6. Reference Documents
   - 1.7. Document Structure
2. **Overall Description**
   - 2.1. Product perspective
   - 2.2. Product functions
   - 2.3. User characteristics
   - 2.4. Assumptions, dependencies and constraints
     - 2.4.1. Assumptions
     - 2.4.2. Dependencies
     - 2.4.3. Constraints
3. **Specific Requirements**
   - 3.1. External Interface Requirements
     - 3.1.1. User Interfaces
     - 3.1.2. Hardware Interfaces
     - 3.1.3. Software Interfaces
       - 3.1.3.1. API
     - 3.1.4. Communication Interfaces
   - 3.2. Functional Requirements
   - 3.3. Performance Requirements
   - 3.4. Design Constraints
     - 3.4.1. Hardware limitations
     - 3.4.2. Any other constraint
   - 3.5. Software System Attributes
     - 3.5.1. Reliability
     - 3.5.2. Availability
     - 3.5.3. Security
     - 3.5.4. Mantainability
     - 3.5.5. Portability
     - 3.5.6. Scalability
4. **Scenarios**
5. **Uml Modeling**
   - 5.1. Use case diagram
   - 5.2. Use case descriptions
   - 5.3. Uml class diagram
   - 5.4. Sequence Diagrams
6. **Formal analysis using Alloy**
   - 6.1. Purpose of the model
   - 6.2. Alloy code
   - 6.3. World generated
7. **Effort Spent**

1 Introduction

1.1 About this document

This document is the Requirement Analysis and Specification Document. Its aim is to completely describe the Travlendar+ system, its components, functional and non-functional requirements, constraints, and relationships with the external world, and to provide typical use cases and scenarios for the users involved. Aim of this document is also to provide a visualization of the structure of the system using UML and a formal specification of application features, using the Alloy language.
This document is written for developers, testers and Quality Assurance. It may be used in a contractual requirement.

1.2 Purpose

The aim of the project is to develop a new mobile calendar-based application that automatically computes and accounts for travel time between appointments to make sure that the user is not late for them.
Furthermore it should support the user in his/her travels by suggesting the best mobility option to get to the location of the appointment.
The user should be able to create different types of calendar on which he/she can insert appointments.
On each calendar he/she can define various kinds of preferences. The user can select from a multitude of travel means he/she wants to use for the calendar and can provide reasonable constraints on them.
He/she can set constraint on distance(e.g., walking distances should be less than 3 km) and on time(e.g., public trains should not be used after 20.00pm or for more than two hours totally). User can also select combinations of transportation means that minimize carbon footprint.
Besides calendar specific preferences there are also global preferences related to the user. For example, a user without a car sharing subscription would not have this travel option activated.
Following the calendar division and the preferences set on them by the user the system will suggest the best combination of travel means to take. For example, imagine that the user would like to bike to the office in the morning (event added in the “Work” calendar with bike as an activated travel option) but the car is better to take the children to soccer practice (event belonging to the “Family” calendar where the car is preferred due to fact that you have to move more people).
In giving suggestions the system should also take into account weather forecast(avoid biking during rainy days). Another feature of the application is that when a meeting is created at location that is unreachable in the allotted time, a warning is shown to the user.
Finally the application also offers the possibility for a user to define a flexible appointment. For instance, a user could be able to specify that lunch must be possible every day between 11:30- 2:30, and it must be at least half an hour long, but the specific timing is flexible.

1.3 Scope

1.3.1 World and shared phenomena

Many endeavors require scheduling meetings at various locations all across a city or a region, whether in support of a mobile job or a busy parent. Choosing the best combination of travel means for reaching different appointments during the days could be complex.
There are a lot of different travel means in the world from public to personal and even shared ones. Every travel option has its own pros and cons and while a public transportation system could be cheaper, a private one could be preferred for route not covered publicly and a shared one could be

preferred in a crowded city, to avoid parking costs. Travlendar+ can help the user overcome this complexity.
However, even if the user organizes its events in different calendars with different travel means, it might happen that the system is not able to satisfy all user's preferences, due to the fact that some types of transportation are just unavailable in some cities. Travlendar+ has to take this into account following user's calendar event locations.
There are other complexities in the world that can constrain Travlendar+ effectiveness and need to be taken into account in order to give proper and reliable information to the user.
An important one is locating a personal vehicle. It's not easy to trace the position of a private vehicle without using a GPS installed on it. This problem is addressed by Travlendar+ by giving the user an automatic tracing system that predicts owned vehicles position based on the user's travel history. Besides the user can manually set the location of his vehicle if he/she does not follow for an event the suggested travel instructions.
Another one, for instance, is giving suggestions for future events on travel sharing system(suppose a car sharing system). A shared car could be the best travel option to move to the next event's location, but when this occurs there could be no sharing car available in the surroundings. That's why Travlendar+ will assume that a shared car will be available for reaching a future event but will provide anyway another alternative to the shared option in case it won't. The system will always let the user find nearby shared mean so when the user has to reach a meeting can choose the alternative according to the position of shared means.
Finally it is worth mentioning that giving the user the possibility to change with an high granularity and at any time the calendar preferences can bring to some problems. Suppose a user has a work calendar with the car and the subway as activated travel means and changes the preferences to have activated only the bike. It would be possible that different events in the calendar become unreachable in time. Travlendar+ solves this problem by letting the user see a warning on all of the events that are unreachable so that the user can fix his/her schedule.
Travlendar+ only supports ground transportation and can be used worldwide.
It supports: public transports(bus, subway, train, tram and light raill), taxis, Enjoy(car sharing system) and Mobike(bike sharing system) if the user travel routes for moving from one meeting to another include the city of Milan, a private car and/or a private bike eventually inserted by the user. Finally it also considers walking in travel calculations.

1.3.2 Actors

- Guest User: guest users, also known as anonymous users, can only see the login and the registration page. They have to register in order to become registered users. From now on a guest user will be referred in the document with the word Guest.
- Registered User: after logging in, users can access all the functionality of the application. From now on a a registered user will also be referred in the document as User.

1.3.3 Goals

The goals of the Travlendar+ system are:

- [G1] Allow a guest to become a registered user
- [G2] Allow a user to log in to the application
- [G3] Allow a logged user to logout of the application
- [G4] Allow a logged user to remove his/her account
- [G5] Allow a logged user to modify his/her account personal data
- [G6] Allow a logged user to create a calendar
- [G7] Allow a logged user to change preferences of a calendar.
- [G8] Allow a logged user to change global preferences.
- [G9] Allow a logged user to specify time and mileage preferences for each specific travel option in calendar preferences.
- [G10] Allow a logged user to select the option to minimize carbon footprint in calendar preferences.

- [G11] Allow a logged user to insert information about his/her personal vehicles in global preferences.
- [G12] Allow a logged user to delete a calendar.
- [G13] Allow a logged user to add a new event in a calendar.
- [G14] Allow a logged user to modify an event in a calendar.
- [G15] Allow a logged user to remove an event from a calendar.
- [G16] Allow a logged user to see the events set on a calendar.
- [G17] Allow a logged user to see all the events set on all the active calendars.
- [G18] Allow a logged user to see, upon event insertion, modification, delete or active calendar/global preferences modification, a warning if there's no feasible travel solution to reach an event in the current schedule.
- [G19] Allow a logged user to see, after event insertion, modification, delete, or active calendar/global preferences modification, the updated travel instructions to reach the location of the events in the active calendars.
- [G20] Allow a logged user to see updated location positions of his/her personal vehicles in global preferences.
- [G21] Allow a logged user to have Flex events on his / her calendar.
- [G22] Allow a logged user to see a travel solution on the Travlendar+ map.
- [G23] Allow a logged user to see the travel instructions to follow at the current time for the very next event.

1.4 Definitions, Acronyms, Abbreviations

1.4.1 Definitions

- Login credentials: credentials used by the user for logging in. They are composed by an email and a password.
- Event: a place the user has to reach at a given time. It is characterized by a start time and an end time. The difference between the start time and the end time defines the event duration. An event is also referred with the word appointment and meeting. The present event is the one the user is attending right now. All the events chronologically before the present event are past while all the ones chronologically after are future.
- Flex event: an event that is characterized by a fixed duration that can be inserted anywhere inside a bigger time window of the event duration.. This window is referred in the document as the flexible sliding window. For example, a flex event that has a duration property of 1 hour and a flexible sliding window that goes from 1pm to 3pm could be allocated, for instance, in the 1:30pm-2:30pm time frame.
- Calendar's base location: the default user location in a calendar. For example in a 'Job' calendar it can be the location of the office. The user can set a next event location to be the calendar's base without creating every time an event to get back to the office.
- Active/inactive calendar: the status of a calendar. The system computes and accounts for travel time only between appointments on active calendar. As an example, a user can make a 'School' calendar inactive while on holidays.
- Previous event: given an event its previous event is the closest past event to it.
- Calendar preferences: set of rules set by the user on a specific calendar that the application will follow.
- Travel Option: a ground transport mean. It is also referred as travel mean. The possible travel options in this document are: public transportation (bus, subway, train, tram, light rail, taxi), shared transports (Mobike and Enjoy), personal vehicles (bike, car, motorbike) and walking.
- Global preferences: set of rules set by the user for all calendars the application will follow.
- Travel Solution: a combination of travel trips suggested to the user for reaching an event in time. Each travel trip is composed by a travel mean the user has to take at a pickup time/location combination has to leave at a drop off time/location. Following all the travel

directions in the solution brings the User to an event. The pickup location of the first trip coincides with the departure location and the drop off location of the last trip corresponds to the event position. The sum of the duration of all the trips defines the total travel duration of the solution. A travel solution can be feasible or unfeasible. A solution is feasible only if the user, by following the combination of travel options suggested, arrives at the event in time. Between all the possible travel solutions found by the system the best solution is the one that let the User reach the event location in the shortest possible time.

- Schedule: all the events on the active calendars. It can also be defined as all the events that have a travel solution, feasible or unfeasible.
- Travel calculation: calculation done by the system on a set of events to find the travel solutions for moving between them.
- Very next event: it is the first event in the chronologically order of the future events. It is the next event the user has to reach.
- User estimated location: it is the location of the user estimated by the application. It corresponds to the present event location or the calendar's base location.
- Recurrence Rule: A combination of:
  - A recurrence policy that can be set on an event that allows the user to repeat with a given frequency the same event. The allowed values are: normal (one-timer), daily, weekly, monthly, yearly.
  - An ending date that must be set to avoid events that repeat forever .The replications of an event are referred as recurrent events.
- User Session: The session of activity that an authenticated user spends communicating with the Travlendar+ during a specified period of time.

1.4.2 Acronyms

- API: Application Program Interface
- MTTF: Mean Time To Failure
- MTTR: Mean Time To Repair
- OS: Operative System

1.4.3 Abbreviations

- [Gn]: goal number n.
- [Dn]: domain assumption number n.
- [Rn]: functional requirement number n.
- [NFn]: non-functional requirement number n.

1.5 Revision history

- **Version 2.0**
  - Fixed an error flow in login use case
  - Fixed calendar and user classes in the domain model
  - Fixed sequence diagrams and related use cases according to what has been done in the design document

1.6 Reference Documents

- AA 2017-2018 Software Engineering 2—Mandatory Project goal, schedule, and rules
- IEEE std 29148 Systems and software engineering - Life cycle processes - Requirements engineering

1.7 Document Structure

This document is structured in the following way:

- Section 2: an overall description of the software product is given. Besides domain assumptions here are listed in an informal way the major functional requirements.
- Section 3: functional and non-functional requirements are listed in a formal way.
- Section 4: here are listed typical scenarios.
- Section 5: the functionality of the software are described with uml modeling. In the use case descriptions mapping on requirements is provided.
- Section 6: presents a formal description of the system's structure using alloy.

2. Overall Description

2.1 Product perspective

There is no legacy system to integrate with and the application will be created from scratch.
In order to retrieve information regarding the travel means accessible in a given location and calculating travel time the system will use Google Maps API.
Thanks to this API it is possible for the system to support accurate travel calculations for all the travel options.
In order to gather information about weather forecast, the system will use OpenWeatherMap API.
The system will also interface with Enjoy API(Milan car sharing system) and Mobike API (Milan bike sharing system).

2.2 Product functions

The system shall:

- Provide a sign up functionality
- Provide a sign in(and sign out) functionality
- Let the logged user modify his/her account
- Let the logged user manage calendars
- Let the logged user manage events on calendars
- Let the logged user manage flex events
- Let the logged user set global preferences
- Let the logged user set calendar preferences
- Warn the logged user if a created event is unreachable in time in the schedule
- Let the user see updated travel instructions for the events in his/her schedule
- Let the user see updates positions of his/her personal vehicle

2.3 User characteristics

Travlendar+ targeted user is a person that has different appointments during the day and needs a easy-to-use tool that automatically computes travel time and suggests the best combination of travel means to get to destination.

2.4 Assumptions, dependencies and constraints

2.4.1 Assumptions

[D1] The user's always follows the suggested feasible travel solutions and if not he/she will update the personal vehicles' location positions.
[D2] Google Maps API is always available and completely trustable.
[D3] OpenWeatherMap API is always available and completely trustable.
[D4] Mobike API is always available and completely trustable.
[D5] Enjoy API is always available and completely trustable.

2.4.2 Dependencies

The app depends on Google Maps API and cannot provide updated travel instructions without it.
The app cannot provide travel means suggestions in relation to weather forecast without OpenWeather API.
The app cannot locate Enjoy cars without Enjoy API.
The app cannot locate Mobike bikes without Mobike API.

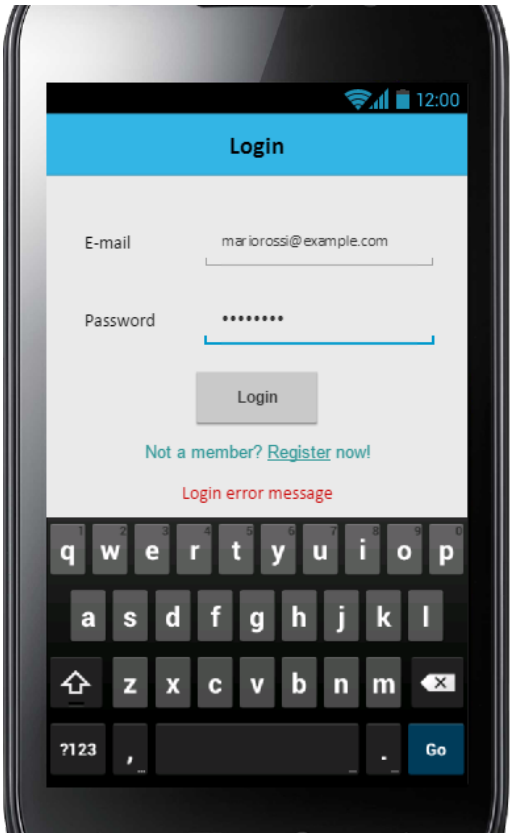
2.4.3 Constraints

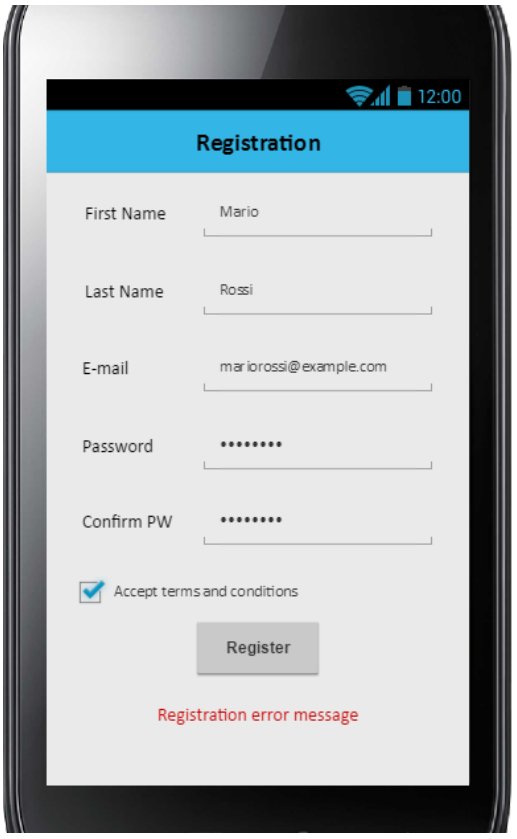
Travlendar+ travel calculations' map coverage is constrained by Google Maps API capability.

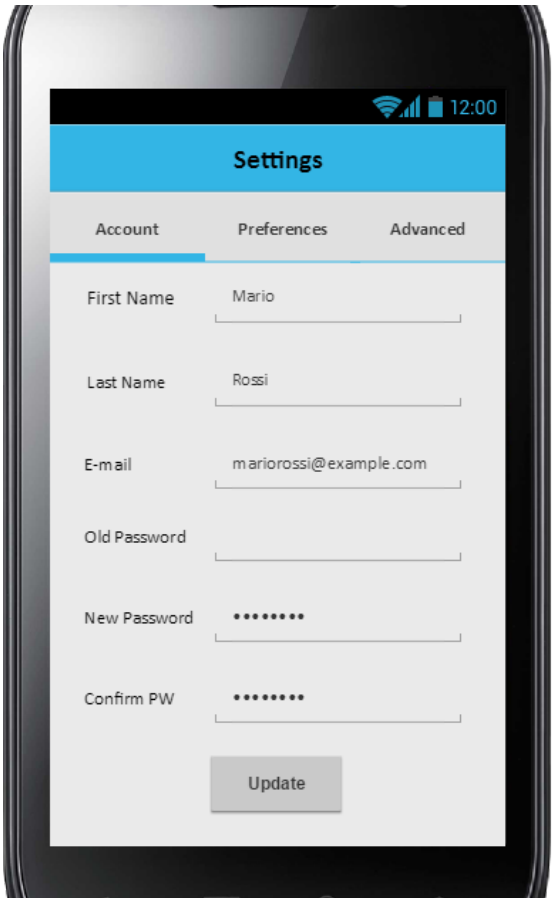
3. Specific Requirements

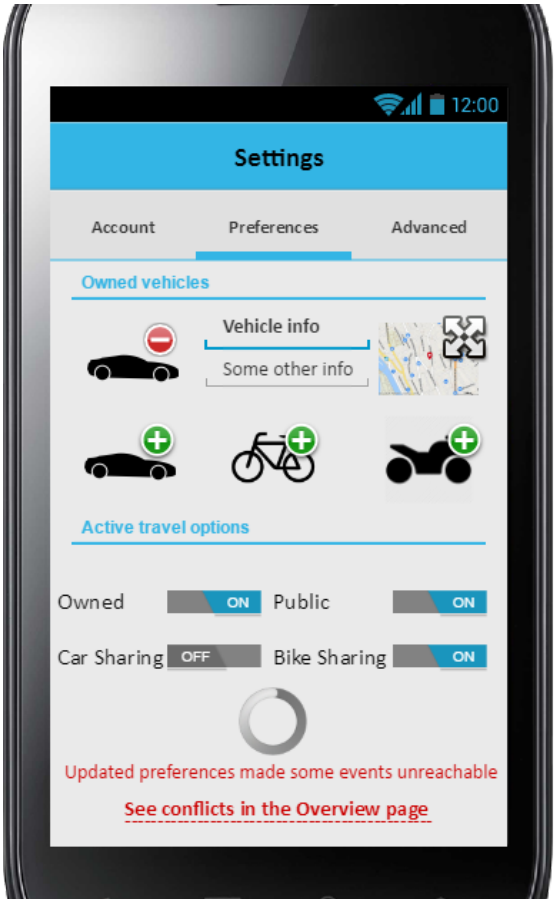
3.1 External Interface Requirements

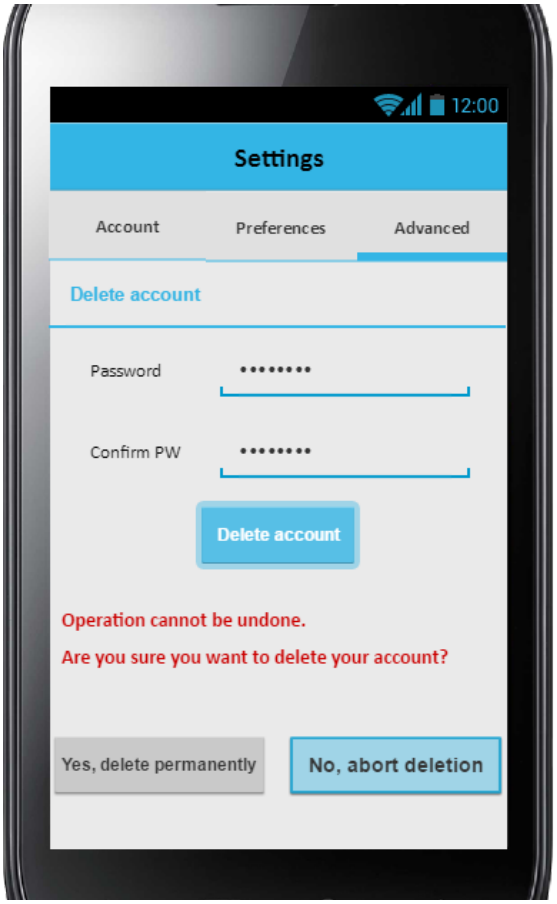
3.1.1 User Interfaces

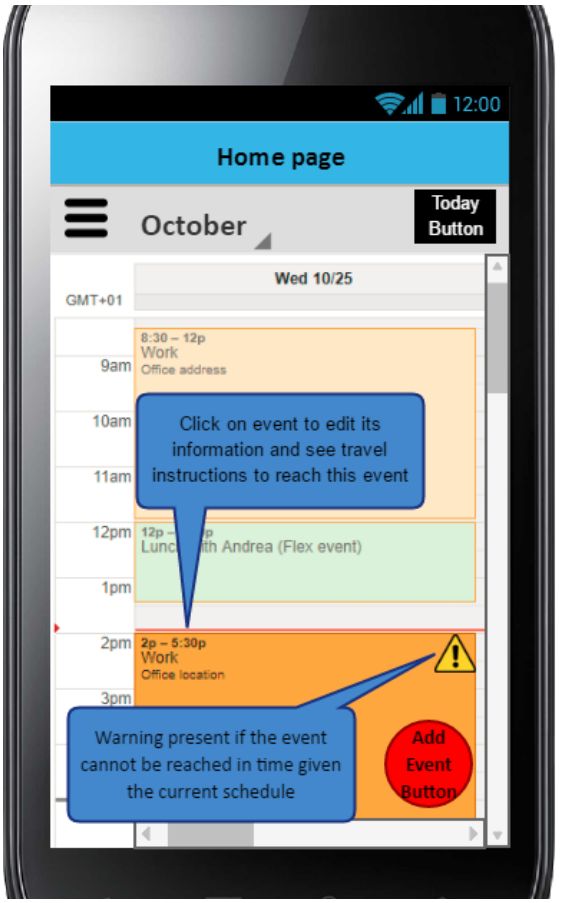


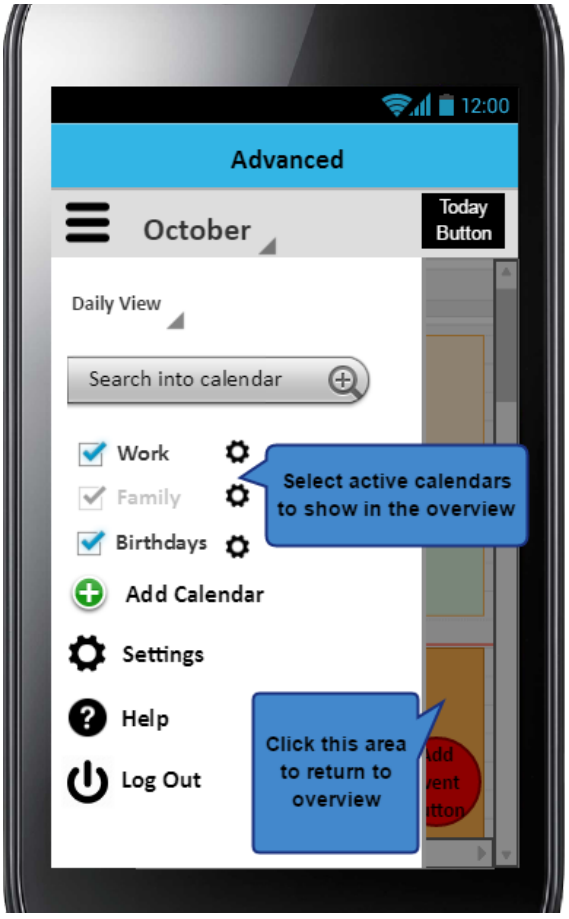



12:00
Settings
Account
Preferences
Advanced
First Name
Mario
Last Name
Rossi
E-mail
mariorossi@example.com
Old Password
New Password
••••••••
Confirm PW
••••••••
Update


12:00
Settings
Account
Preferences
Advanced
Owned vehicles
Vehicle info
Some other info
Active travel options
Owned
ON
Public
ON
Car Sharing
OFF
Bike Sharing
ON
Updated preferences made some events unreachable
See conflicts in the Overview page

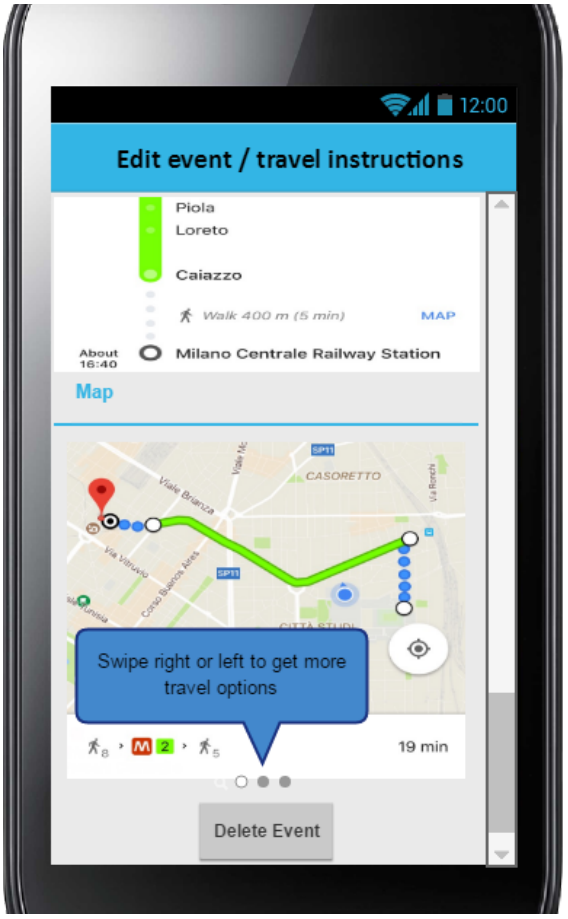

12:00
Settings
Account
Preferences
Advanced
Delete account
Password
••••••••
Confirm PW
••••••••
Delete account
Operation cannot be undone.
Are you sure you want to delete your account?
Yes, delete permanently
No, abort deletion


12:00
Home page
October
Today Button
GMT+01
Wed 10/25
9am
10am
11am
12pm
1pm
2pm
3pm
8:30 – 12p
Work
Office address
Click on event to edit its information and see travel instructions to reach this event
Andrea (Flex event)
2p – 5:30p
Work
Office location
Warning present if the event cannot be reached in time given the current schedule
Add Event Button


12:00
Advanced
October
Today Button
Daily View
Search into calendar
Work
Family
Birthdays
Add Calendar
Settings
Help
Log Out
Select active calendars to show in the overview
Click this area to return to overview


12:00
Edit event / travel instructions
Event info
Calendar associated with this event
Family
Milano Centrale Station
Pickup sister at station
Start 16:30
End 17:00
Arriving gate: 2
Set event location here (redirects to map)
Travel instructions
19 min
16:19
Politecnico di Milano
Walk 650 m (8 min)
MAP
Lambrate FS
at 16:27
at 16:31


12:00
Edit event / travel instructions
Booking PIN: 12345
Travel instructions
10 minutes late
19 min
16:19
Warning present if the best solution for this trip is unfeasible (i.e. cannot arrive in time)
Lambrate FS
at 16:27
at 16:31
Piola
Loreto
Caiazzo
Walk 400 m (5 min)
MAP
About 16:40
Milano Centrale Railway Station


12:00
Edit event / travel instructions
Piola
Loreto
Caiazzo
Walk 400 m (5 min)
MAP
About 16:40
Milano Centrale Railway Station
Map
CASORETTO
Swipe right or left to get more travel options
19 min
Delete Event

3.1.2 Hardware Interfaces

The system does not support any hardware interface.

3.1.3 Software Interfaces

3.1.3.1 API

- Google Maps API
    - Provides accurate real-time information for navigation and places on a global coverage
    - For messages content and format please refer to https://developers.google.com/maps/
- OpenWeatherMap API
    - Gives access to current weather data for any location including over 200,000 cities frequently updated based on global models and data from more than 40,000 weather stations.
    - For messages content and format please refer to https://openweathermap.org/api
- Mobike API
    - Provides accurate information on Mobike bikes' positions in Milan.
- Enjoy API
    - Provides accurate information on Enjoy cars' positions in Milan.

3.1.4 Communication Interfaces

- Http
- Https
- TCP

3.2 Functional Requirements

3.2.1 [G1] Allow a guest to become a registered user

[R1] The system shall provide a sign up functionality.
[R2] The system shall provide fields for the first name and the last name, the email, the password and the password confirmation the user can fill to register.
[R3] The system shall check the email is unique.
[R4] The system shall check the password confirm field matches the password one.
[R5] The system shall store the fields filled by the user in the registration.

3.2.2 [G2] Allow a user to log in to the application

[R6] The system shall provide a sign in functionality.
[R7] The system shall provide fields for the email and the password the user can fill for logging in.
[R8] The system shall check the correctness of the login credentials by checking that the email is a registered one and that the password matches that email.

3.2.3 [G3] Allow a logged user to logout of the application

[R9] The system shall provide a sign out functionality to let the user logout from the application.

3.2.4 [G4] Allow a logged user to remove his/her account

[R10] The system shall provide the user a function to remove the account from the system.
[R11] The system shall be sure that the user really wants to delete the account.

3.2.5 [G5] Allow a logged user to modify his/her account personal data

[R12] The system shall provide a functionality to let the user modify all the data he/she inserted in the registration process.
[R3] The system shall check the email is unique.

[R13] To change the password, the system shall double check the password inserted by the user using a password confirm field and asks the user for the old password.

3.2.5 [G6] Allow a logged user to create a calendar

[R14] The system shall provide a functionality to let the user create a calendar.
[R15] The system shall let the user initialize calendar preferences.
[R16] The system shall provide text fields the user can fill for setting the calendar's name and description.
[R17] The system shall check that the calendar's name is unique.
[R18] The system shall provide a field the user can fill with the calendar's base location.
[R19] The system shall provide a set of colors to let the User choose his favorite one for the calendar.
[R20] The system shall provide a list of travel means the user can choose from to use on the calendar. The list generated is constrained by travel means activated in global preferences.
[R21] The system shall provide a toggle switch the User can toggle on/off to let the calendar become active/inactive.
[R22] The system shall check that the calendar's base location is always expressed in terms of coordinates, street and number or a landmark name, allowing the system to calculate its position. If the check correctness fails, the system shall ask the user to re-insert the location.

[G7] Allow a logged user to change preferences of a calendar.

[R23] The system shall provide a functionality to let the user change the calendar preferences.
[R24] The system shall provide a functionality to modify of all the parameters related to the calendar (name, description, color, base location, active/inactive toggle switch).
[R25] The system shall let the user change the travel means he/she wants to use on the calendar. The list of travel means is constrained by the one in global preferences.
[R17] The system shall check that the calendar's name is unique.
[R22] The system shall check that the calendar's base location is always expressed in terms of coordinates, street and number or a landmark name, allowing the system to calculate its position. If the check correctness fails, the system shall ask the user to re-insert the location.

[G8] Allow a logged user to change global preferences.

[R26] The system shall provide a functionality to let the user change the global preferences.
[R27] The system shall let the user activate or deactivate a travel mean between all the possible travel means supported.
[R28] The system shall provide default global preferences (all possible travel means activated) if no changes are done.

[G9] Allow a logged user to specify time and mileage preferences for each specific travel option in calendar preferences.

[R29] The system shall provide a functionality to let the User add/remove a time frame where the selected travel option must not be allowed. Each selected time frame shall start/end either at xx:00 or xx:30.
[R30] The system shall provide a number field to set a maximum distance limit allowed for each specific option.

[G10] Allow a logged user to select the option to minimize carbon footprint in calendar preferences.

[R31] The system shall provide a way to activate and deactivate this option.
[R32] When this option is active, the system shall adjust the feasible travel solution in travel calculations to display the travel options in ascending value of carbon footprint.

[G11] Allow a logged user to insert information about his/her personal vehicles in global preferences.

[R33] The system shall provide a functionality to let the user insert cars, bikes and/or motorbikes.
[R34] The system shall provide a field the user can fill to insert a name to identify the car/bike/motorbike.
[R35] The system shall provide a field the user can fill to insert his/her car/bike/motorbike location.
[R36] The system shall check that the vehicle's name is unique.
[R37] The system shall check that the vehicle location is always expressed in terms of coordinates, street and number or a landmark name, allowing the system to calculate its position. If the check correctness fails, the system shall ask the user to re-insert the location.

[G12] Allow a logged user to delete a calendar.

[R38] The system shall provide a functionality to let the user delete a calendar.
[R39] The system shall be sure that the user really wants to delete the calendar.
[R40] The system shall remove all of the events belonging to the calendar.

[G13] Allow a logged user to add a new event in a calendar.

[R41] The system shall provide a functionality to let the user add a new event on a calendar.
[R42] The system shall provide a list of fields the user can fill with the events details.
[R43] The system shall provide fields for the name, the location, the start time, the end time and the recurrence rule of the event.
[R44] The system shall check that the event created start time is in the future and the end time is greater than the start time.
[R45] The system shall only provide feasible recurrence options, in such a way that all the recurrence rules that would make the event overlap with itself if applied are disabled.
[R46] The system shall provide a toggle switch the user can toggle on to make the next event location to be the calendar's base location.
[R47] The system shall replicate the event in the calendar following its recurrence rule.
[R48] The system shall check that the event location is always expressed in terms of coordinates, street and number or a landmark name, allowing the system to calculate its position. If the check correctness fails, the system shall ask the user to re-insert the location.
[R49] The system shall prevent the user to insert an event that overlaps (even partially) with the current schedule.

[G14] Allow a logged user to modify an event in a calendar.

[R50] The system shall provide a functionality to modify of all the parameters related to the event (calendar, start time, end time, location, recurrence rule, toggle switch for the next event location).
[R44] The system shall check that the event created start time is in the future and the end time is greater than the start time.
[R45] The system shall only provide feasible recurrence options, in such a way that all the recurrence rules that would make the event overlap with itself if applied are disabled.
[R49] The system shall prevent the user to insert an event that overlaps (even partially) with the current schedule.
[R47] The system shall replicate the event in the calendar following its recurrence rule.
[R51] The system shall modify all the recurrent events.
[R48] The system shall check that the event location is always expressed in terms of coordinates, street and number or a landmark name, allowing the system to calculate its position. If the check correctness fails, the system shall ask the user to re-insert the location.

[G15] Allow a logged user to remove an event from a calendar.

[R52] The system shall provide a functionality to let the user remove a specific event.
[R53] The system shall delete the event in the calendar following its recurrence rule.
[R54] The system shall be sure that the user really wants to delete the calendar.
[R55] If the deleted events has a recurrence rule set, the system shall ask the user if he/she wants to eliminate all the recurrent events.

[G16] Allow a logged user to see the events set on a calendar.

[R56] The system shall provide a functionality to let the user see all the events set on a calendar in chronologically order.
[R57] The system shall let the user, viewing events on a calendar, see all the details of the event.

[G17] Allow a logged user to see all the events set on all the active calendars.

[R58] The system shall provide a functionality to let the user have an overview of all the events in chronologically order he/she inserted in the application.

[G18] Allow a logged user to see, upon event insertion, modification, delete or active calendar/global preferences modification, a warning if there's no feasible travel solution to reach an event in the current schedule.

[R59] The system shall not consider events on inactive calendar for travel calculations.
[R60] The system shall not consider past events for travel calculations.
[R61] At event creation/modification, the system shall fire a warning if any travel solution became unfeasible.
[R62] At active calendar/global preferences modification the system shall warn the user if any travel solution became unfeasible.
[R63] The system shall consider as location from which to reach an event the next location of the previous event in an active calendar, if such an event exists, otherwise the calendar base location, even if the considered event presents an unfeasible solution.
[R64] The system shall perform travel calculations considering all events across all active calendars simultaneously.
[R65] While doing travel calculations, the system shall move a flex event start and end time inside its flexible sliding window for finding a feasible travel solution.
[R66] The system shall consider Mobike and Enjoy service only if the travel route includes the city of Milan.
[R67] The system shall suppose, at travel calculation, that a Mobike bike and an Enjoy car will be available from the location from which to reach an event.
[R68] The system shall highlight all the events with an unfeasible travel solution.
[D2] Google Maps API is always available and completely trustable.

[G19] Allow a logged user to see, after event insertion, modification, delete, or active calendar/global preferences modification, the updated travel instructions to reach the location of the events in the active calendars.

[R69] The system shall provide for every event the best travel solution, following the User preferences, to reach it in time. The system shall consider the calendar preferences of that event in giving the solution.
[R70] The system shall prefer the travel solution with the first travel option that departs as late as possible while still giving the user the possibility to reach the location of the event before its start time.

[R63] The system shall consider as location from which to reach an event the next location of the previous event in an active calendar, if such an event exists, otherwise the calendar base location, even if the considered event presents an unfeasible solution
[R71] If the event next location is set to be the calendar's base one, the system shall also provide the best travel solution to leave the event and reach the calendar's base location.
[R64] The system shall perform travel calculations considering all events across all active calendars simultaneously.
[R59] The system shall not consider events on inactive calendar for travel calculations.
[R60] The system shall not consider past events for travel calculations.
[R72] The system shall not show any travel instruction for events on inactive calendars.
[R66] The system shall consider Mobike and Enjoy service only if the travel route includes the city of Milan.
[R67] The system shall suppose, at travel calculation, that a Mobike bike and an Enjoy car will be available from the location from which to reach an event.
[R68] The system shall highlight all the events with an unfeasible travel solution.
[R65] While doing travel calculations, the system shall move a flex event start and end time inside its flexible sliding window for finding a feasible travel solution.
[R73] If the best travel solution includes bike, Mobike or walking, and rain is forecast the system shall also show, if it exists, the next best solution that does not include bike, Mobike, and walking.
[R74] If the best travel solution includes a sharing service the system shall also show, if it exists, the next best solution without a sharing travel mean as an alternative.
[R75] If the best travel solution includes both bike, Mobike or walking with forecast rain and a shared option the system shall also show, if it exist, the next best solution without a shared mean and without bike, Mobike or walking.
[D3] OpenWeatherMap API is always available and completely trustable.
[D2] Google Maps API is always available and completely trustable.

[G20] Allow a logged user to see updated location positions of his/her personal vehicles in global preferences.

[R76] The system shall update the user's personal vehicles positions in global preferences according to the travel solutions suggested to the user across all calendars.
[R77] If more feasible travel solutions are shown to the user(for example because of [R73] and [R74] and [R75]) and a personal vehicle is included in them the system shall ask the user, after an event start time, which option he/she did choose. If the event next location is the calendar's base one the system shall also ask the user which option he/she is going to take for getting to the base location. The system shall update the location of the vehicle in global preferences accordingly.
[R37] The system shall check that the vehicle location is always expressed in terms of coordinates, street and number or a landmark name, allowing the system to calculate its position. If the check correctness fails, the system shall ask the user to re-insert the location.
[D1] The user's always follows the suggested feasible travel solutions and if not he/she will update the personal vehicles' location positions.

[G21] Allow a logged user to have Flex events on his / her calendar.

[R78] The system shall provide a toggle switch to set a regular event to become a Flex one.
[R79] When the Flex option is toggled on, the system shall provide a field to insert the flex event duration, which has to be less than the difference between the start and end times.

[G22] Allow a logged user to see a travel solution on the Travlendar+ map.

[R80] The system shall provide a functionality to let the user select a travel solution suggested on an event and view it on a map so that the user can visually seen how he/she should move and which travel means he/she has to take.
[R81] If the travel solution includes Mobike and/or Enjoy the system shall show in the Map the positions of the shared means.
[D4] Mobike API is always available and completely trustable.
[D5] Enjoy API is always available and completely trustable.

[G23] Allow a logged user to see the travel instructions to follow at the current time for the very next event.

[R82] The system shall let the user see the travel solutions to reach the very next event according to the current time and the user's estimated location.
[R83] The system shall let the user see the solutions till the event becomes past.
[R84] The system shall pick the travel solutions from the ones already calculated in the calendar.
[D4] Mobike API is always available and completely trustable.
[D5] Enjoy API is always available and completely trustable.

3.3 Performance Requirements

[NF1] The system shall support up to 10000 connected users.
[NF2] The system shall support up to 1 terminal for each user.
[NF3] 99% of the requests shall be processed in less than 5s under normal peak workload.
[NF4] 99% of the requests shall be processed in less than 10s under high peak workload.

3.4 Design Constraints

3.4.1 Hardware limitations

The device running the app must support Android OS and must have a data connection.

3.4.2 Any other constraint

The app must follow privacy's law and must not share users sensible data.

3.5 Software System Attributes

3.5.1 Reliability

[NF5] The system shall present a MTTF >> MTTR on long-term analysis.

3.5.2 Availability

[NF6] The system shall be five nines up and running.

3.5.3 Security

[NF7] The system shall not store user plain password but rather store a hash of them. The hashing function to be used shall be able to protect against rainbow table attacks and brute force attacks.
[NF8] The system shall communicate with the user using a secure communication protocol.
[NF9] The system shall do proper checking on user input in order to prevent XSS, CRSF and SQL Injection.
[NF10] The system shall sign cryptographically user sessions in order to prevent session hijacking.

3.5.4 Maintainability

[NF11] The system shall be modular and easily extensible.
[NF12] The system shall be fully tested and should not manifest unexpected behaviours.

3.5.5 Portability

[NF13] The system application logic software shall run on any Linux OS.

[NF14] The system application logic shall not be in the client side so that client's code hitting Travlendar+ endpoint is fully portable.

3.5.6 Scalability

[NF15] The system shall be able to scale horizontally in response to increasing workloads.

4 Scenarios

Scenario A

Mr. Black is a business manager with a very tight schedule, thus he needs to optimize his movements from one location to another to avoid being late to his events, which are carefully organized in his Travlendar+ app. He's currently attending a conference on digital currencies in the lecture hall, but sadly Dr. Brown (who is the speaker in said meeting) has just announced that the conference will end one hour earlier, due to personal issues. Thanks to Travlendar+, Mr. Black just needs to open the app, which will display today's schedule, and tap on the next event (*Afternoon work session, location: office*) to get the travel instructions and plan ahead of time his movement options.

Scenario B

Mrs. Pink is Mr. Black's wife, currently pregnant and thus she's not working at the moment. Her husband dropped her downtown at 9am before going to work because she needed to make some shopping for the baby to come, therefore she's alone by foot in the city center. She has reserved an appointment with the gynecologist after the shopping session, so she opens the Travlendar+ app to get the needed travel information in order to reach the doctor. In her situation, the preferred system would be booking a shared car, but the closest one is 2.5 kilometers away and she doesn't feel like walking much. Luckily for her, Travlendar+ is suited to provide multiple travel options that satisfy all the user demands, so she just needs to swipe her finger on the screen to see another suitable choice to reach the appointment. A bus, departing right from Shopping St., is going to bring her directly to the doctor (and she will also have right to a preferred seat on the bus, neat!).

Scenario C

Ms. White is Mr. Black's secretary, it is her duty to allocate his superior appointments properly, in order to avoid him to have a tight schedule that adds more difficulties into moving from one location to another. Mr. Black also likes to have lunch at the Spaghetteria Italiana, a Michelin 2-star restaurant where he his an usual customer. Ms. White knows that lunch is a very important time for a business manager, so at least 1 hour should be allocated for this event, that can happen anytime between 12:30pm to 2:30pm (restaurant opening hours at lunch). Today her director has a meeting until 12:15pm, and Mr. Green (the firm's lawyer) has just called to reserve an urgent appointment for today to discuss an ongoing lawsuit where Mr. Black is involved, so this is going to take at least 2 hours in order to prepare for the incoming trial. This would usually take Ms. White to perform a manual calculation in order to allocate all the appointments, but thanks to the Flex feature of Travlendar+ the system automatically acknowledges that the 2 hour appointment should be postponed after lunch, otherwise the Spaghetteria will close and Mr. Black will not be able to eat those delicious spaghetti. Sorry Mr. Green, but law must wait until lunch is over!

Scenario D

Mr. Blue is Mr.Black's brother, an environmentalist who is all about saving the world and being eco-friendly. Thanks to Travlendar+, he can organize his events in different calendars and specify the preferences according to the calendar type: he likes to always bike to the office (even when it's

raining, he just needs his raincoat!), so he disables all the other travel means for the *Work* calendar. But today is a special day, because he promised his kids that he would've picked them up to bring them to a birthday party. By placing this event in the *Family* calendar where the only allowed travel option is the car sharing, the system will automatically provide a solution that involves a car, without having Mr. Blue to worry any further, he just needs to toggle the preferences and that's it. In addition, the system will save the position of his bike that is left at the parking lot near his office, so the travel information needed to go back home after work is over will still be updated!

Scenario E

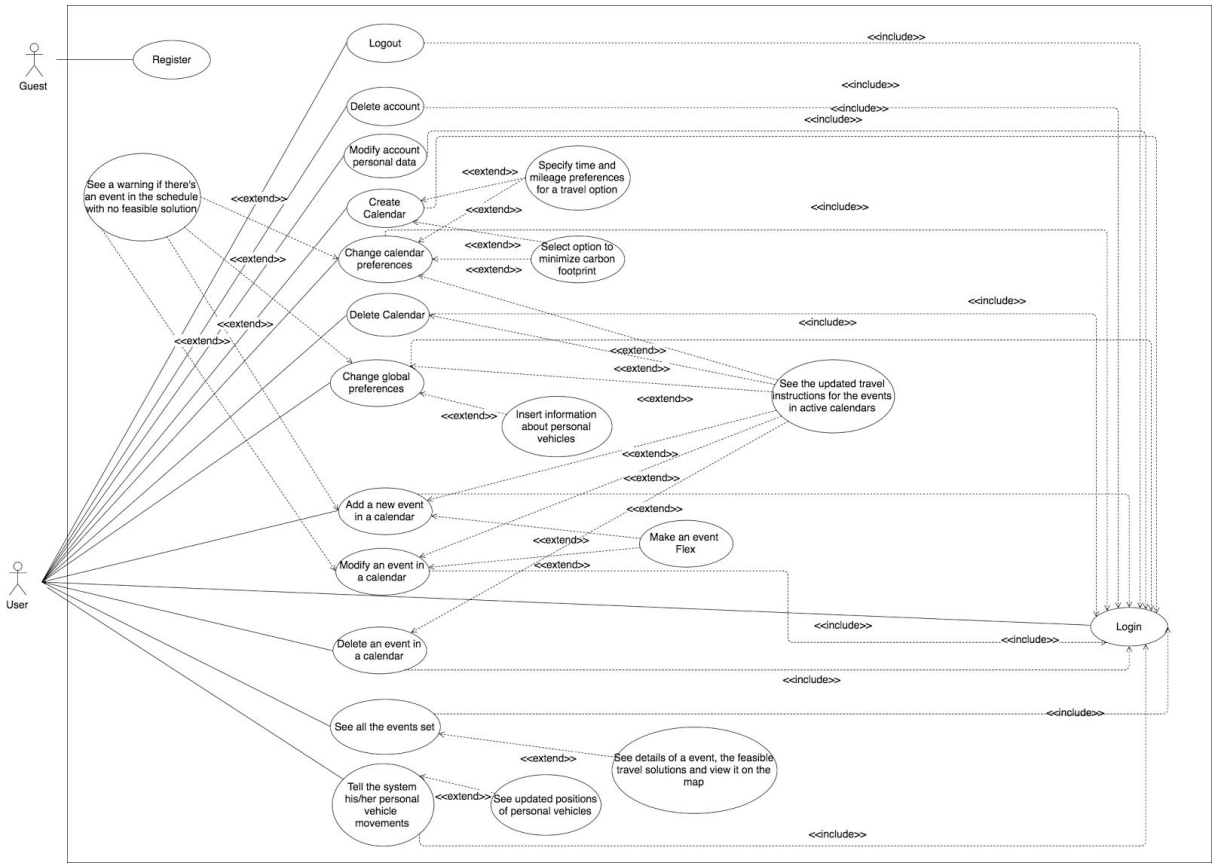
Mrs. Orange is Mr. Blue's wife and she's being influenced by her environmental-friendly husband about all the pollution and emissions problems. She's really satisfied with the Travlendar+ feature that has a sorting option for travel methods that is called *"Carbon footprint minimization".* She proceeded to insert all the vehicle information about her cars in the *Preferences* page of the app: a *Slimboy* (motorbike produced by the famous brand *Harly Dravidson*) and the brand new *Thunderbolt by Besla* (a really famous company that produces electric cars). As fun as the bike might be, the app will suggest Mrs. Orange to use the electric vehicle: some sacrifices need to be made in order to save the planet!

Scenario F

Mr. Gray is a professor at Haart University and needs to organize his schedule with an efficient app, that must also be flexible enough to adapt to sudden timetable changes in a fast and intuitive way. Travlendar+ comes to support once again, with the option to apply a recurrence rule to any event created: all he has to do is toggle the option in the Event Page and set the "*Repeat Weekly*" option, the system will then proceed to automatically create future events that match this rule. The *Software Engineering 2* lesson on Thursday has been move permanently to another classroom, or maybe the lecture hours have been changed? Just tap on a single event, edit the necessary info and click *Yes* when the option to change all future recurrences is prompted. On the other side, deleting a single event that belongs to a recurrent context (for example, a lecture cancelled because it happens to be under graduation day) is as simple as answering *No* when the same prompt as before is asked. This way Mr. Gray only needs to focus on developing the young minds he's teaching to!

5 Uml Modeling

5.1 Use case diagram



5.2 Use case descriptions

Guest Registers

Name: Guest registers
Related to requirements of goals: [G1]
Actors: Guest
Entry Conditions: the guest views the Login Page
Flow of events:

- The guest clicks on the Register button.
- The system redirects the User to the Registration Page.
- The guest inputs first name and last name, email, password and confirmation password.
- The guest clicks on the Submit button.
- The system creates the account and redirects the guest to the Login Page.

Exit Conditions: The guest is registered.
Exceptions: If the email is already present in the system and/or the password is invalid or does not match the confirmation password, the system displays an error to the guest and let him insert again the registration data.

User logs in

Name: User logs in
Related to requirements of goals: [G2]
Actors: User
Entry Conditions: the User views the Login Page
Flow of events:

- The User inputs the email and the password.
- The User clicks on the Submit button.
- The system logs in the User and redirects the User to the Home Page.

Exit Conditions: the User is logged in.
Exceptions: If the email and the password inserted do not correspond to a registered account, the system displays an error and let the User reinsert the login data.

User logs out

Name: User logs out
Related to requirements of goals: [G3]
Actors: User
Entry Conditions: the User is logged in and views the sidebar.
Flow of events:

- The user clicks on the Logout button.
- The system logs out the User and redirects the User to the Login Page.

Exit Conditions: the User is logged out.
Exceptions: there are no exceptions.

User deletes the account

Name: User deletes the account
Related to requirements of goals: [G4]
Actors: User
Entry Conditions: the User is logged in and views the Sidebar
Flow of events:

- The User clicks the Account Settings button
- The system redirects the User to the Account Settings Page.
- The User clicks on the delete account button.
- The system asks the User if he/she really wants to delete the account.
- The User clicks the Confirm button.
- The system deletes the User's account and redirects him/her to the Login Page.

Exit Conditions: the User is not registered anymore on the system.
Exceptions: if the user does not confirm the system let the user try to delete the account again.

User modifies account personal data

Name: User modifies account personal data
Related to requirements of goals: [G5]
Actors: User
Entry Conditions: the User is logged in and views the Sidebar
Flow of events:

- The User clicks the Account Settings button.
- The system redirects the User to the Account Settings Page.

- The User can modify the first name, last name, email and password by inserting new values.
- To change the old password the User also have to insert the old password and fill a confirmation password field.
- The User clicks the Update button.
- The system changes the user account data.
- The system redirects the User to the Home Page.

Exit Conditions: The User's account data is modified.
Exceptions: if the email is already registered or the password inserted does not match the confirmation password field or the old password is wrong, the system displays an error message and let the user reinsert the data.

User creates a calendar

Name: User creates a calendar
Related to requirements of goals: [G6][G9][G10]
Actors: User
Entry Conditions: the User is logged in and views the Sidebar.
Flow of events:

- The User clicks the Create Calendar button.
- The system redirects the User to the Calendar Preferences Page.
- The User inputs the calendar's name, description and base location.
- The User chooses the color of the calendar.
- The User chooses the travel options he/she wants to use on the calendar.
- For each travel option chose the User can specify time and mileage preferences.
- The User can select the option to minimize carbon footprint.
- The User choose if he/she wants the calendar to be active/inactive.
- The User clicks the Submit button.
- The systems creates the calendar and redirects the User to the Home Page.

Exit Conditions: the User has a new Calendar on which he/she can inserts events.
Exceptions: if the calendar's name is not unique and/or the base location is not valid, the system displays an error and let the User reinsert the data.

User changes calendar preferences

Name: User changes calendar preferences
Related to requirements of goals: [G7][G9][G10][G18][G19]
Actors: User
Entry Conditions: the User is logged, has at least one calendar and views the Sidebar
Flow of events:

- The User clicks on a Calendar Preferences button.
- The system redirects the User to the Calendar Preferences Page.
- The User can modify the calendar's name, description and base location.
- The User can modify the color of the calendar.
- The User can modify the the travel options he/she wants to use on the calendar.
- For each travel option chose the User can modify time and mileage preferences.
- The User can select/deselect the option to minimize carbon footprint.
- The User can make the calendar become active/inactive.
- The User click the Submit button.
- The system changes the calendar's preferences.
- If the calendar is active, the system updates the travel instructions for the user schedule and checks if a feasible travel solution exists for all the events in it.

- If the check fails the system shows a warning.
- The system redirects the User to the Home Page.

Exit Conditions: the User's calendar preferences are modified and the User can see the updated travel instructions according to the changes.
Exceptions: if the calendar's name is not unique and/or the base location is not valid, the system displays an error and let the User reinsert the data.

User deletes a calendar

Name: User deletes a calendar
Related to requirements of goals: [G12][G19]
Actors: User
Entry Conditions: the User is logged in, has at least one calendar and views the Sidebar
Flow of events:

- The User clicks on the Calendar Preferences button.
- The system redirects the User to the Calendar Preferences Page.
- The user clicks on the Delete Calendar button.
- The system asks the User if he/she really wants to delete the calendar.
- The User clicks on the Confirm button.
- The system deletes the calendar, updates the travel instructions for the user schedule(if the calendar is active) and redirects the user to the Home Page.

Exit Conditions: the User's calendar does not exist anymore and the User can see the updated travel instructions for the schedule.
Exceptions: if the User doesn't confirm the deletion, the system will not delete the calendar and let the User retry to delete the calendar.

User changes global preferences

Name: User changes global preferences
Related to requirements of goals: [G8][G11][G18][G19]
Actors: User
Entry Conditions: the User is logged in and views the Sidebar
Flow of events:

- The User clicks on the Global Preferences button.
- The system redirects the User to the Global Preferences Page.
- The User can change the position of an inserted personal vehicle.
- The User can click on the Add Personal Vehicle button to add a personal vehicle. If so the system let the User input the name, the type(car, bike or motorbike) and the location of the vehicle.
- The User can click on the Add Personal Vehicle button to add multiple vehicles.
- The User can activate or deactivate all the travel means listed.
- The systems changes the global preferences.
- The system recalculates travel options and displays a warning if the updated preferences bring to conflicts with events in the calendar due to unreachable events.
- The system redirects the User to the Home Page.

Exit Conditions: the User's global preferences are modified and the User can see the updated travel instructions for the schedule.
Exceptions: if a location inserted is not valid or the name inserted for a vehicle is not unique, the system displays an error and let the User reinsert the data.

User adds new event in a calendar

Name: User adds a new event in a calendar
Related to requirements of goals: [G13][G21][G18][G19]
Actors: User
Entry Conditions: the User is logged in, has at least one calendar and views the Home Page
Flow of events:

- The User clicks on the Add Event button.
- The system redirects the User to the Event Page.
- The User select the calendar on which he/she wants to add the event.
- The User can input the name, the location, the start time, the end time and the recurrence rule of the event.
- The User can toggle on a toggle switch to make the next event location to be the calendar's base location.
- The User can make the event of Flex type using a toggle switch. If so the system let the user inputs the flex event duration.
- The User clicks the Submit button.
- The system creates the event on the calendar.
- If the calendar is active, the system updates the travel instructions for the user schedule and checks if a feasible travel solution exists for all the events in it.
- If the check fails the system shows a warning.
- The system redirects the user to the Home Page.

Exit Conditions: the event is created in the calendar and replicated following the recurrence rule. The User can see the updated travel instructions for the schedule.
Exceptions:
If the event start time is not in the future and/or the end time is not greater than the start time or the recurrence rule can't be applied to the event or the location inserted is not valid or the flex event duration is not less than the difference between the start and end times, the system displays an error and let the User reinsert the data.

User modifies an event in a calendar

Name: User modifies an event in a calendar
Related to requirements of goals: [G14][G21][G18][G19]
Actors: User
Entry Conditions: the User is logged in, has at least one calendar with at least one event and views the Home Page
Flow of events: .

- The User clicks on the event in the calendar.
- The system redirects the User to the Event Page.
- The User can modify the calendar of the event.
- The User can modify the name, the location, the start time, the end time and the recurrence rule of the event.
- The User can toggle on/off the toggle switch to make the next event location to be the calendar's base location.
- The User can toggle on/off the toggle switch for making the event of Flex type. If the User modifies the event for making it a Flex one the system let the user inputs the flex event duration.
- The system modifies the event.
- If the calendar is active, the system updates the travel instructions for the user schedule and checks if a feasible travel solution exists for all the events in it.
- If the check fails the system shows a warning.

- The system redirects the user to the Home Page.

Exit Conditions: the event is modified. The User can see the updated travel instructions for the schedule.
Exceptions: If the event start time is not in the future and/or the end time is not greater than the start time or the recurrence rule can't be applied to the event or the location inserted is not valid or the flex event duration is not less than the difference between the start and end times, the system displays an error and let the User reinsert the data.

User deletes an event in a calendar

Name: User deletes an event in a calendar
Related to requirements of goals: [G15][G19]
Actors: User
Entry Conditions: the User is logged in, has at least one calendar with at least one event and views the Home Page.
Flow of events:

- The User clicks on the event in the calendar.
- The system redirects the User to the Event Page.
- The User clicks on the Remove Event button.
- The system asks the user if he/she really wants to delete the event.
- The User clicks on the Confirm button.
- The system deletes the event.
- The system then asks the user if he/she really wants to delete all the future events following the recurrence rule (if present).The user can click on the Confirm button or Cancel button.
- If the user confirms the system deletes also all the future events following the recurrence rule.
- The system updates the travel instructions for the user schedule(if the calendar is active).
- The system redirects the User to the Home Page.

Exit Conditions: the event is deleted in the calendar and the user sees the updated travel instructions for the schedule.
Exceptions: if the User doesn't confirm the deletion, the system will not delete the calendar and let the User retry to delete the calendar.

User sees all the events set

Name: User sees all the events set
Related to requirements of goals: [G16][G17][G22][G23]
Actors: User
Entry Conditions: the User is logged, has at least one calendar.
Flow of events:

- The user clicks the Home Page button.
- The system redirects the user to the Home Page.
- The User can see all the events across all the calendars.
- The User can see events of active and inactive calendars.
- The User can see the very next event in his/her schedule.
- The User can click on an Event, view its details and view the travel instructions on the Travlendar+ Map if it is of an active calendar.

Exit Conditions: the User sees the events set in the calendar.
Exceptions: there are no exceptions.

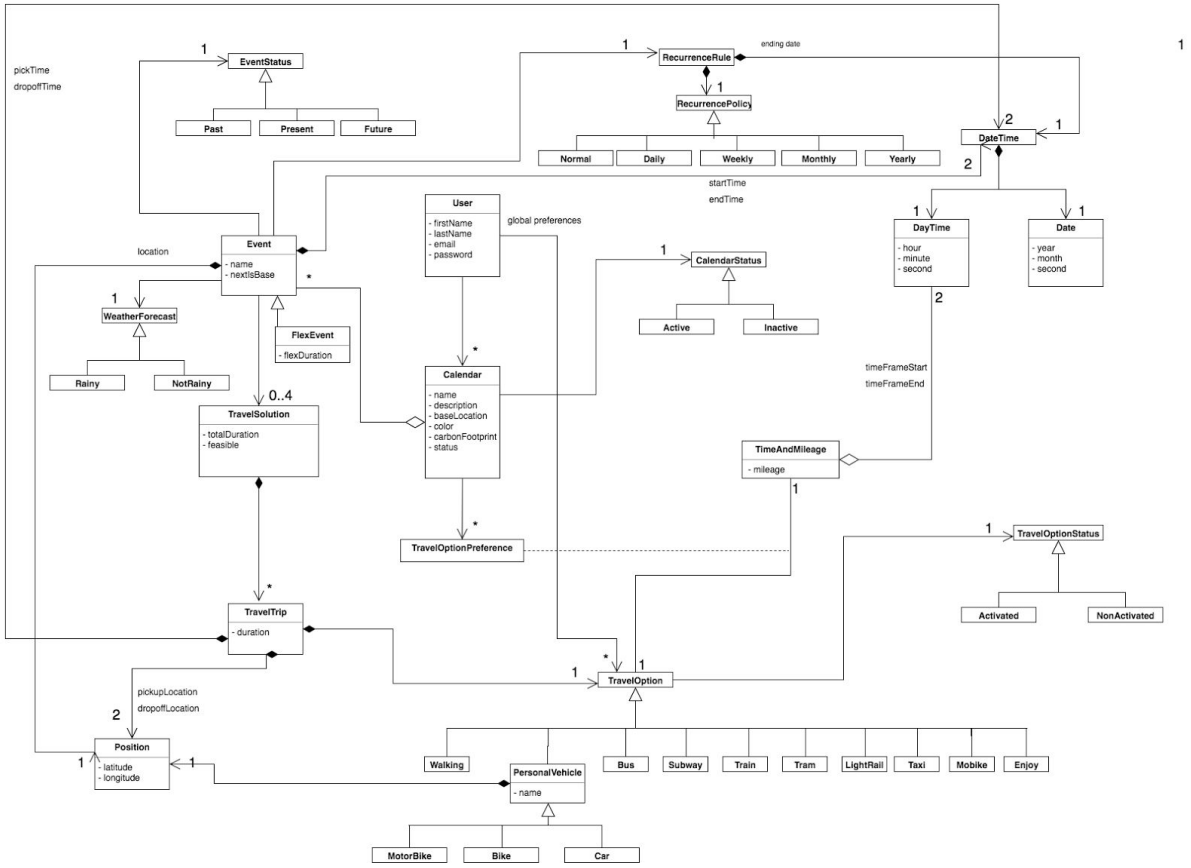
User tells the system his/her personal vehicles movements.

Name: User tells the system his/her personal vehicles movements.
Related to requirements of goals: [G20]
Actors: User
Entry Conditions: the User is logged and is viewing any Page.
Flow of events:

- An event, for which more feasible travel solutions are shown to the user and a personal vehicle is included in one of them, has started
- The system asks the User which solution he/she did choose.
- The User selects the solution.
- If the event next location is the calendar base one and a personal vehicle is included in the travel solution to get to the base location, the system asks the User which option he/she is going to take to get to the base location.
- The User selects the solution.
- The system updates the personal vehicles positions accordingly.
- The system redirects the User to the global preferences page.

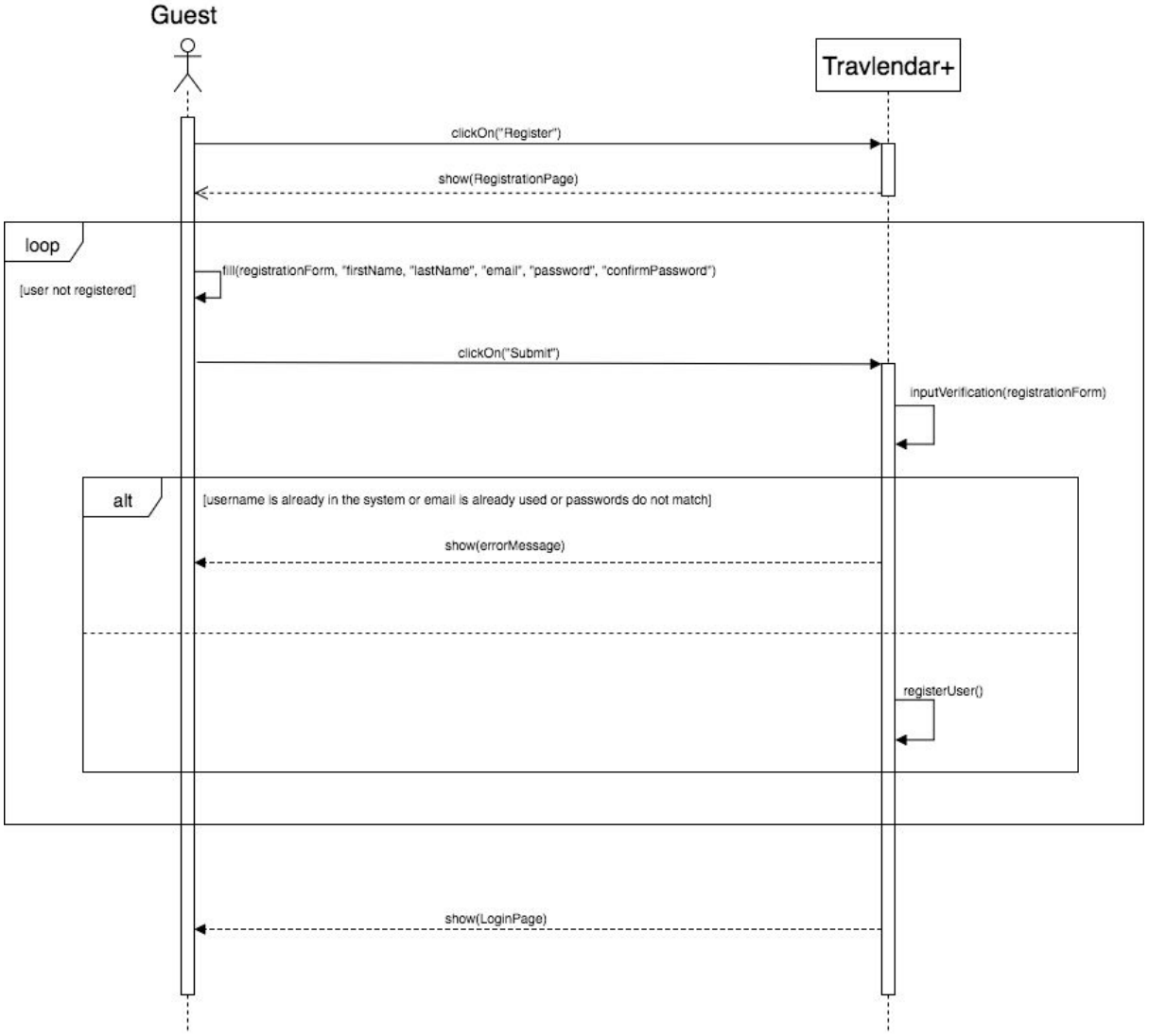
Exit Conditions: the User has told the system his/her personal vehicles movements and User sees the updated positions of personal vehicles
Exceptions: there are no exceptions.

5.3 Uml class diagram

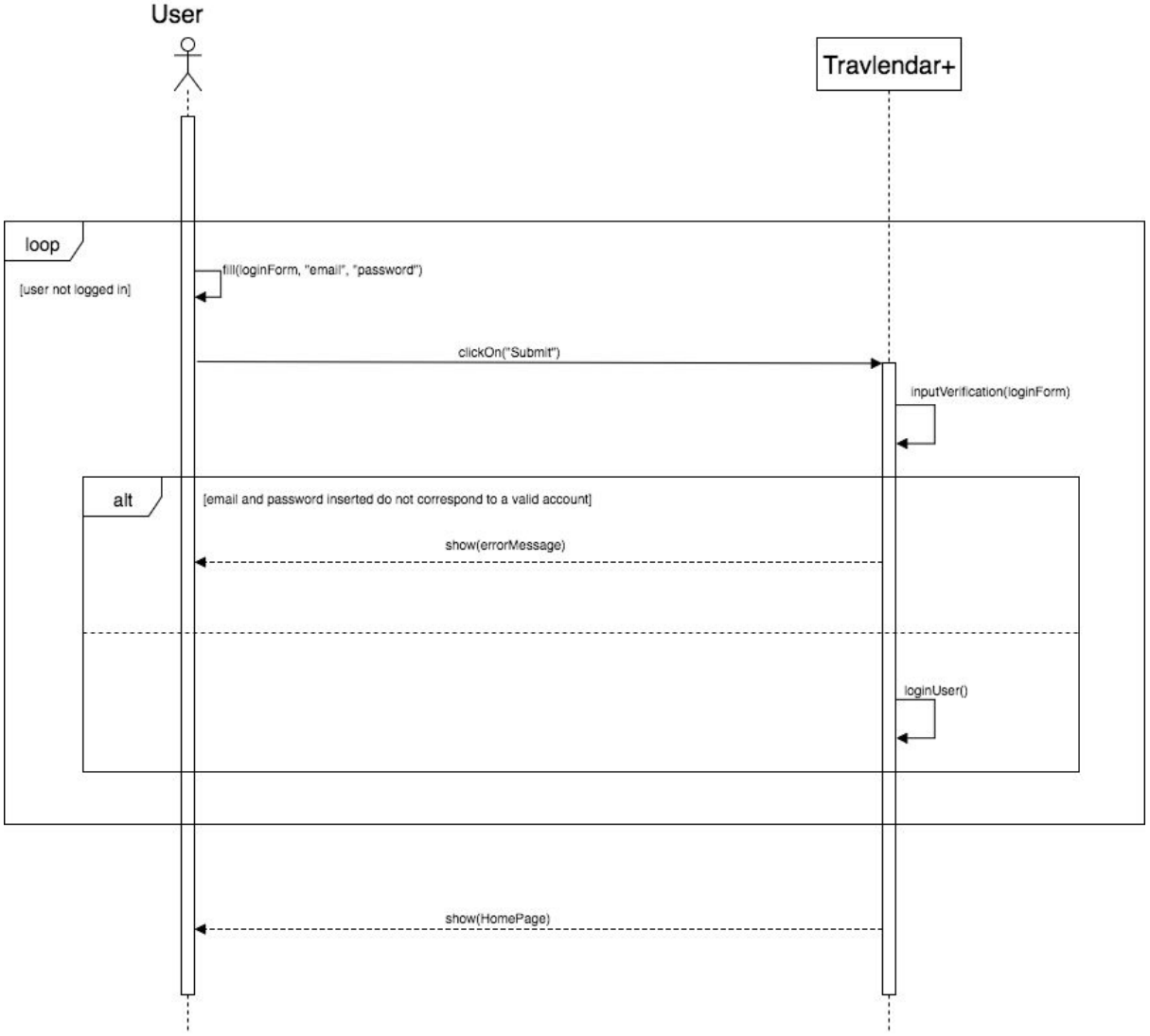


5.4 Sequence Diagrams

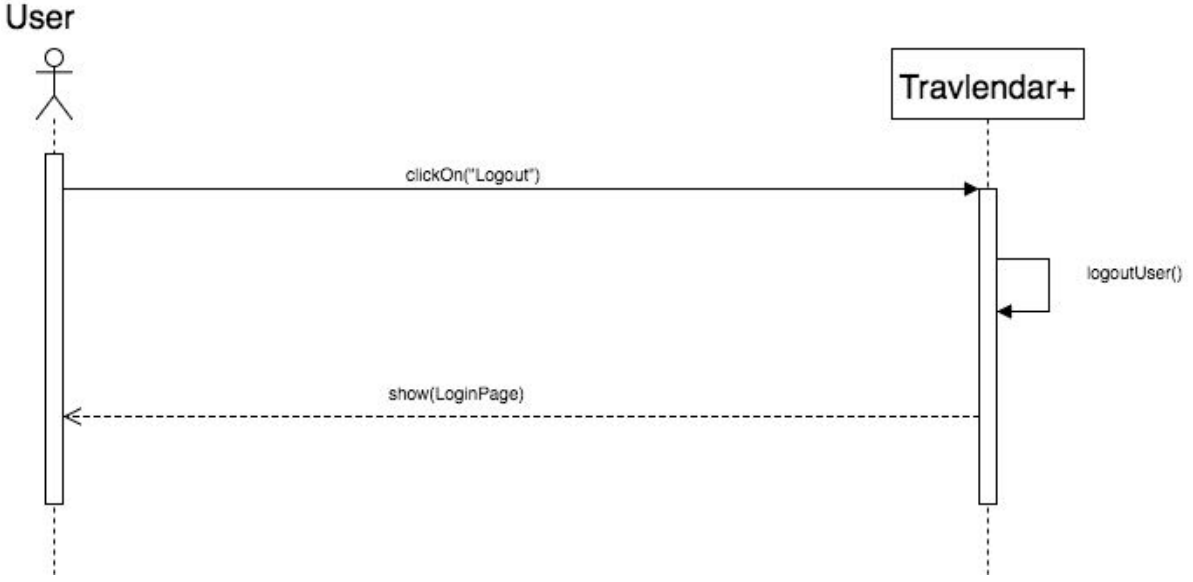
Guest Registers



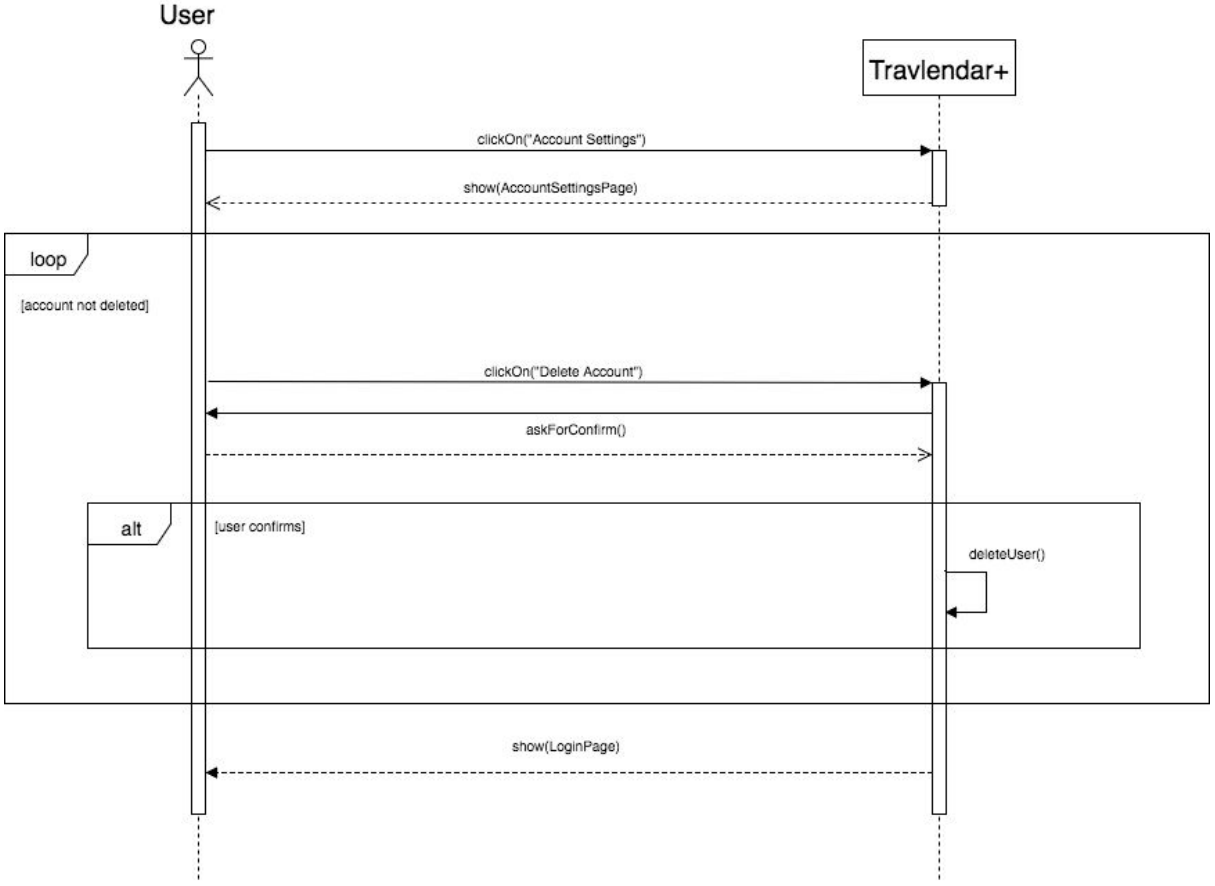
User logs in



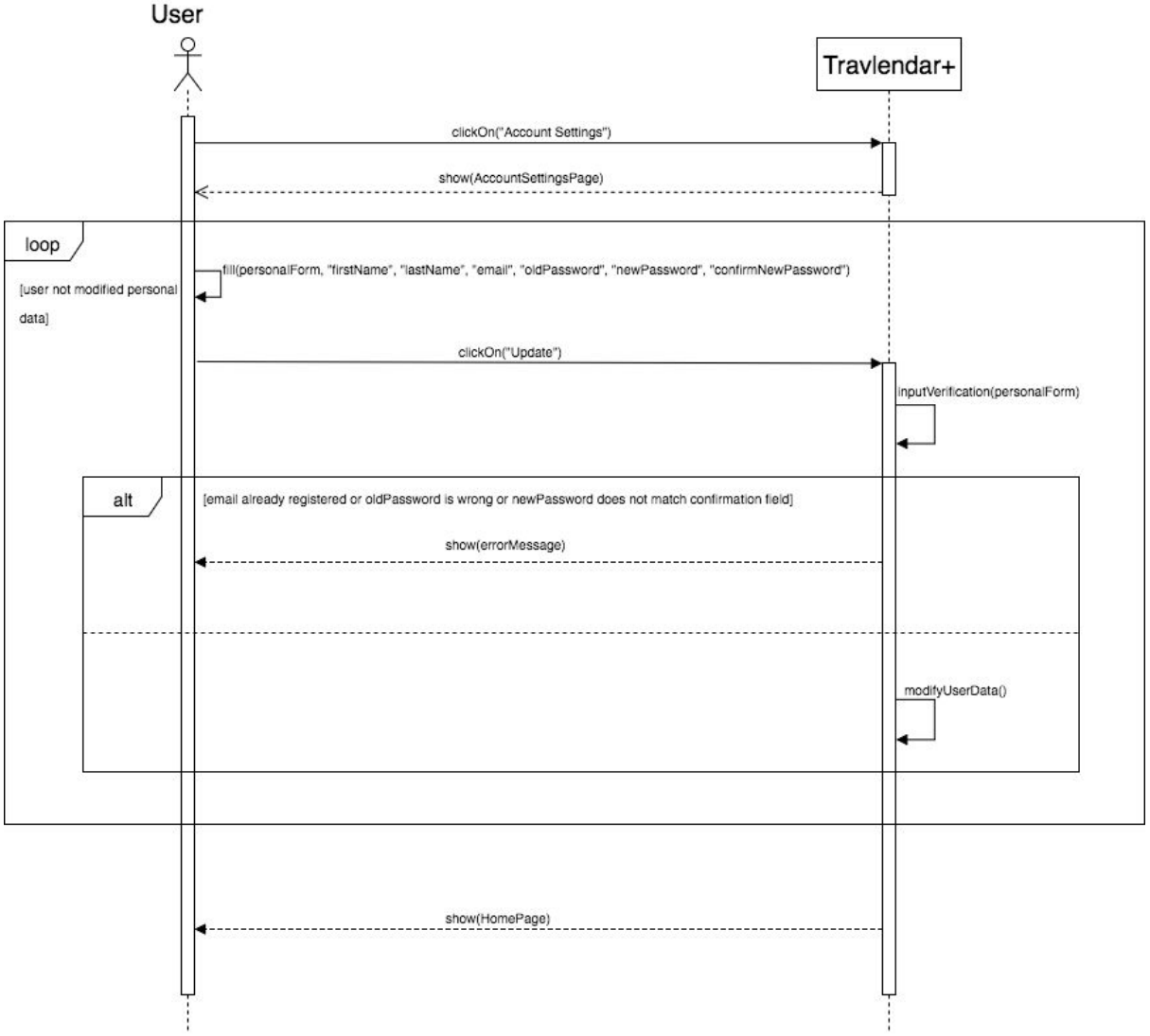
User logs out



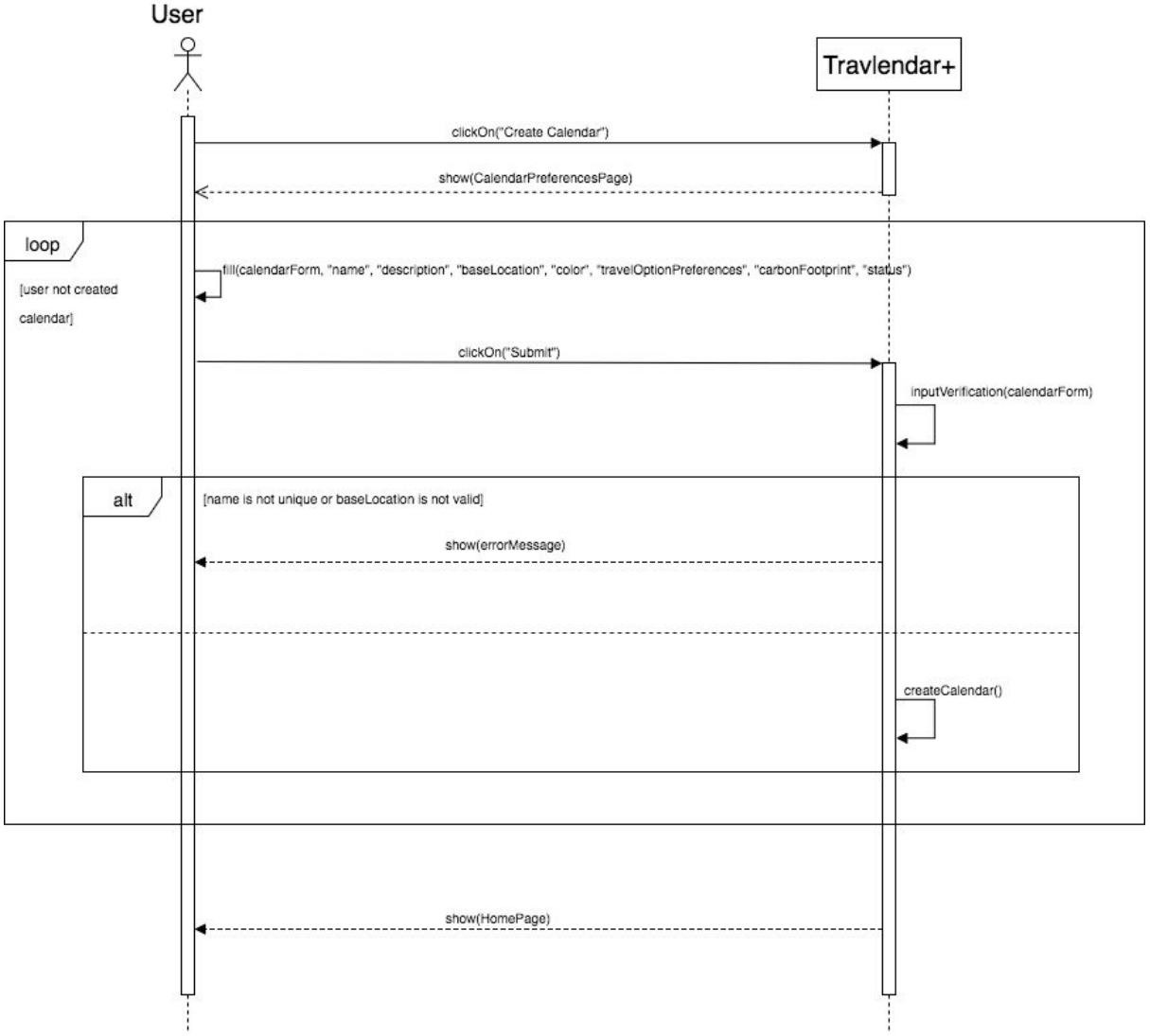
User deletes the account



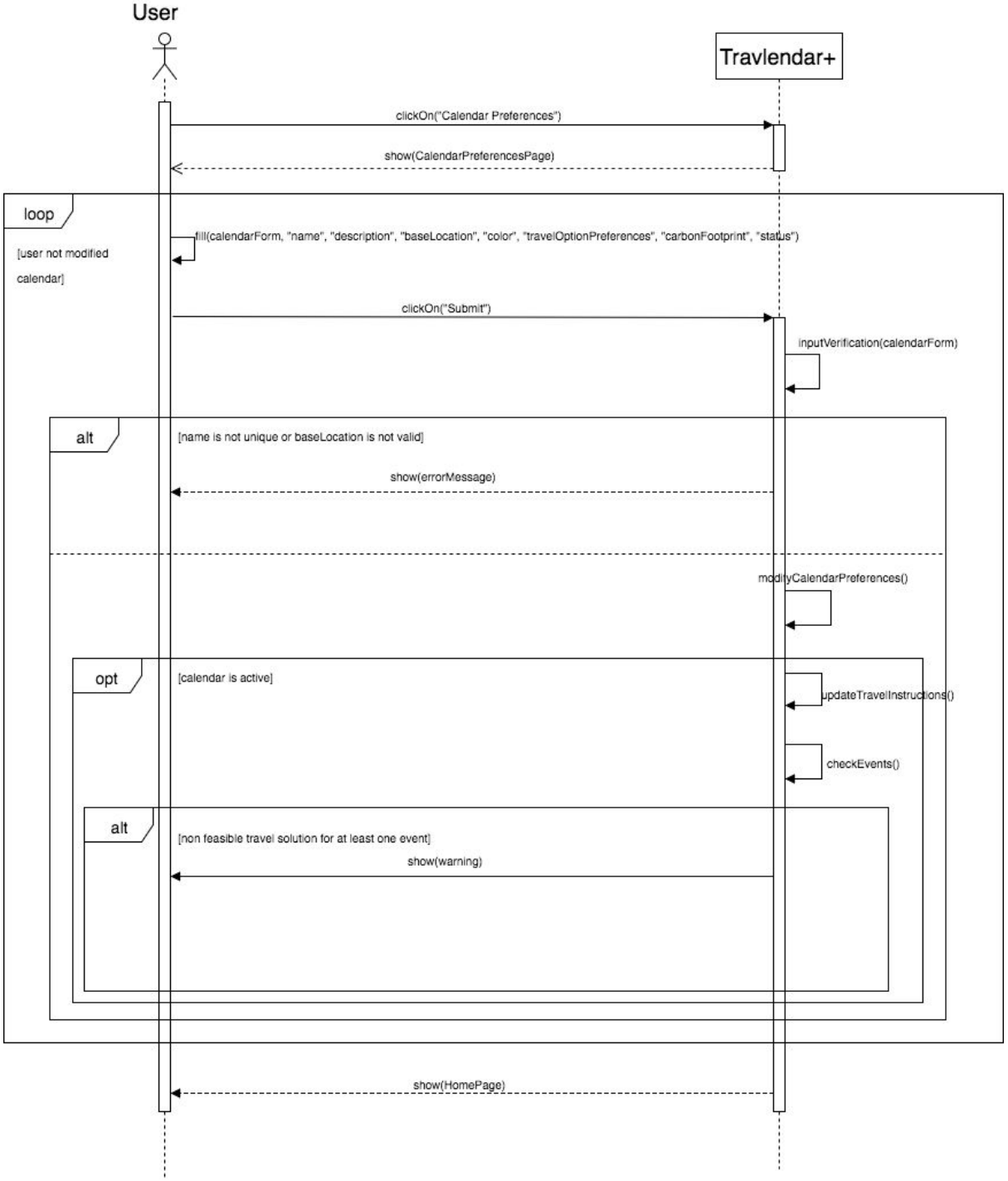
User modifies account personal data



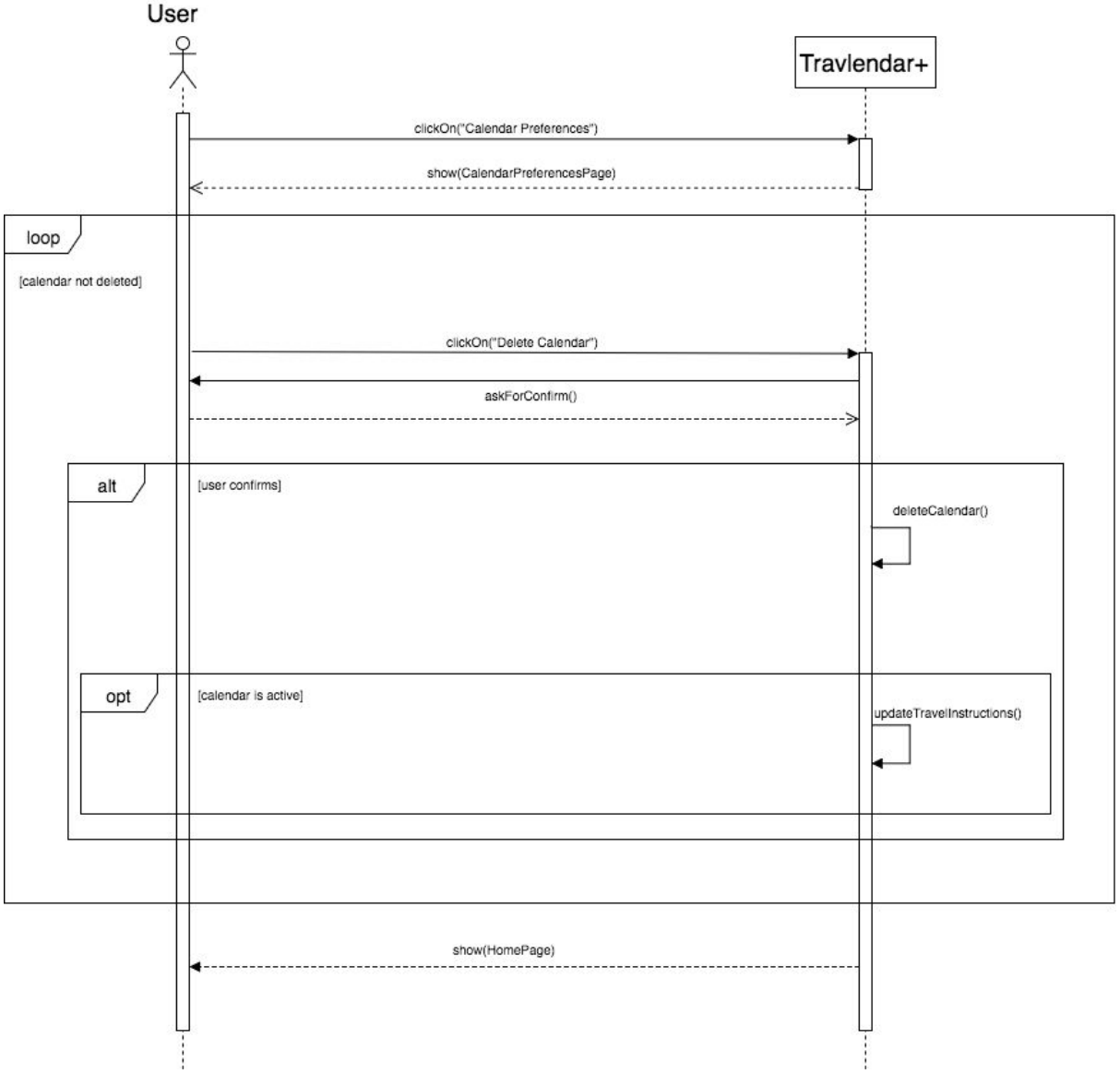
User creates a calendar



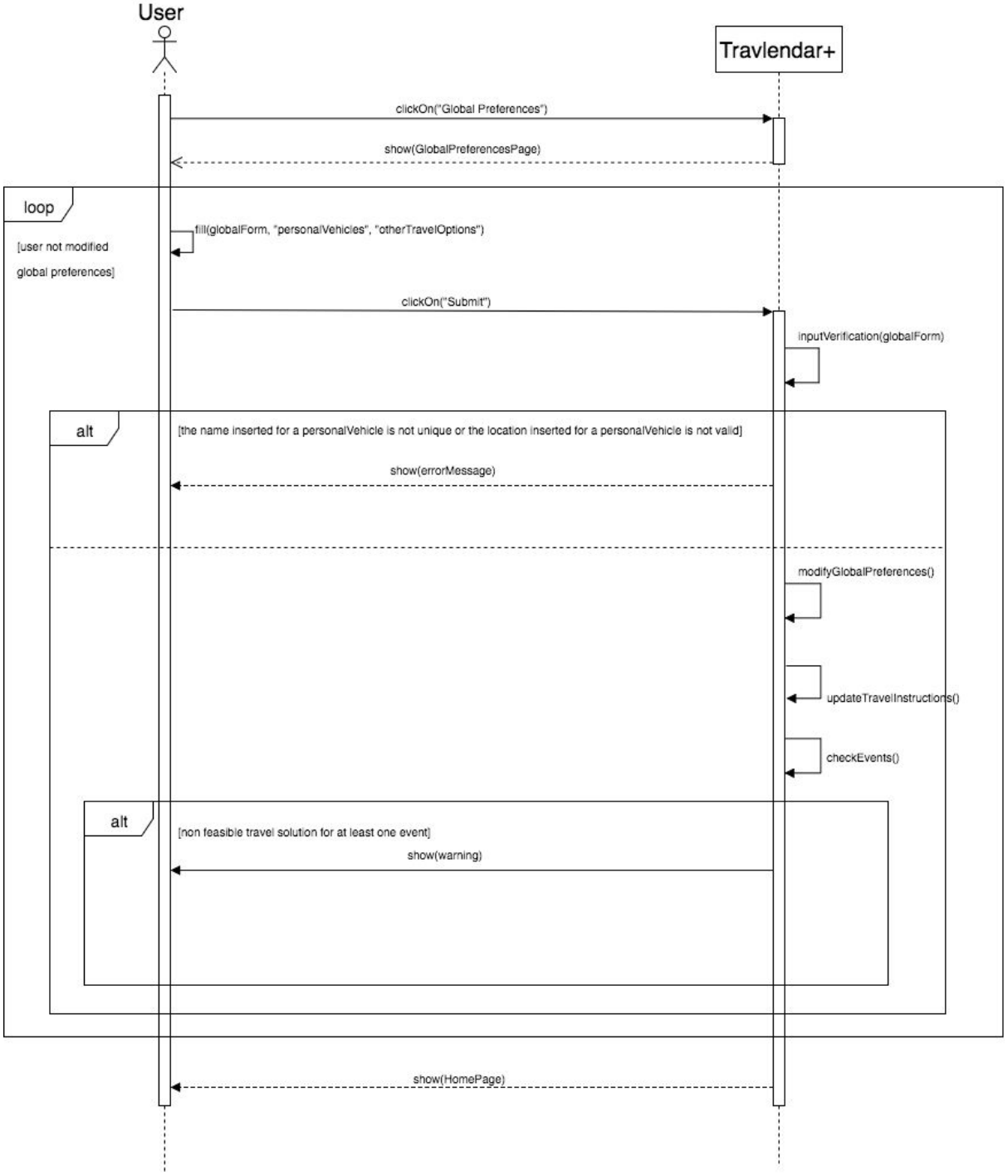
User changes calendar preferences



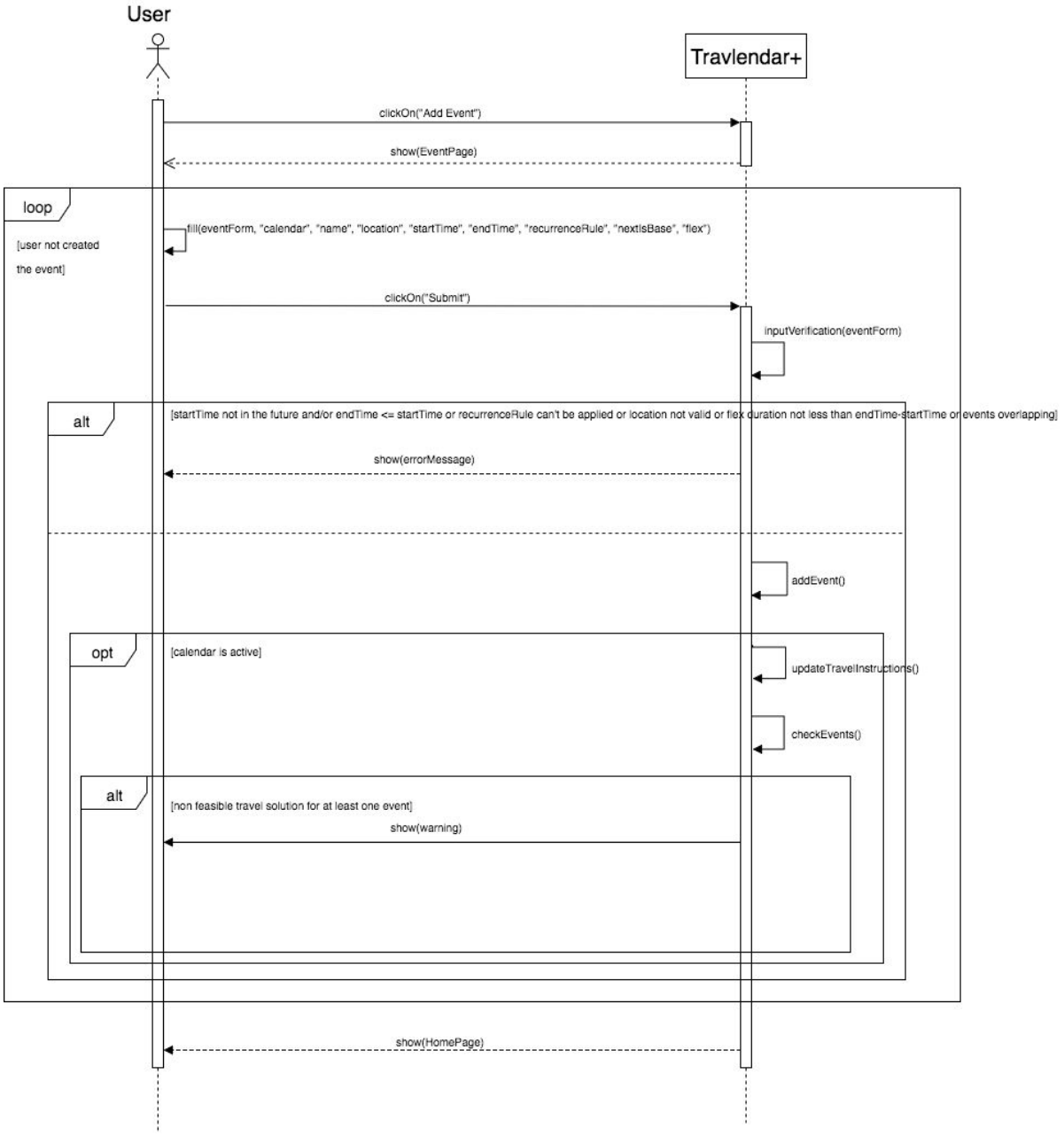
User deletes a calendar



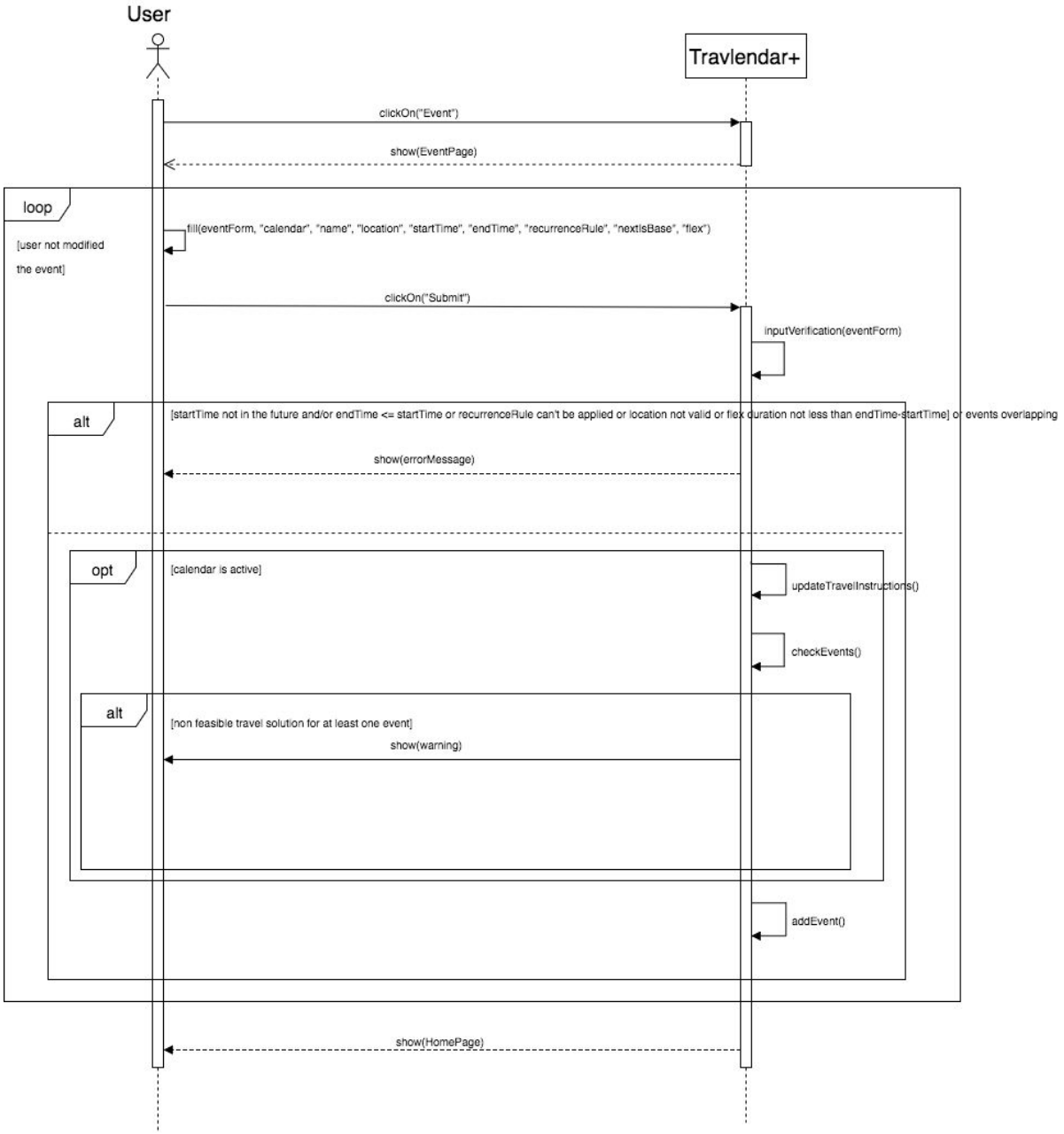
User changes global preferences



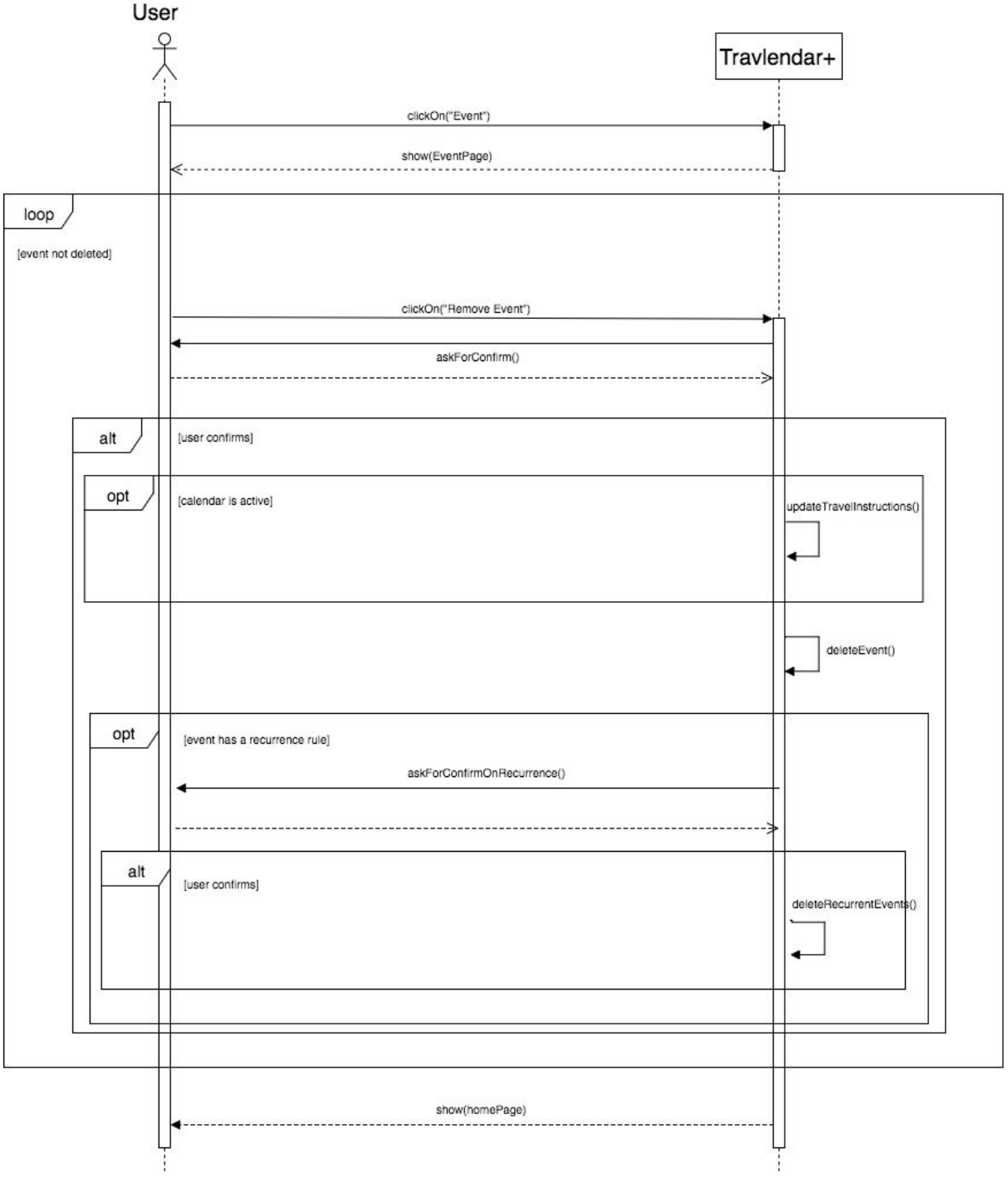
User adds new event in a calendar



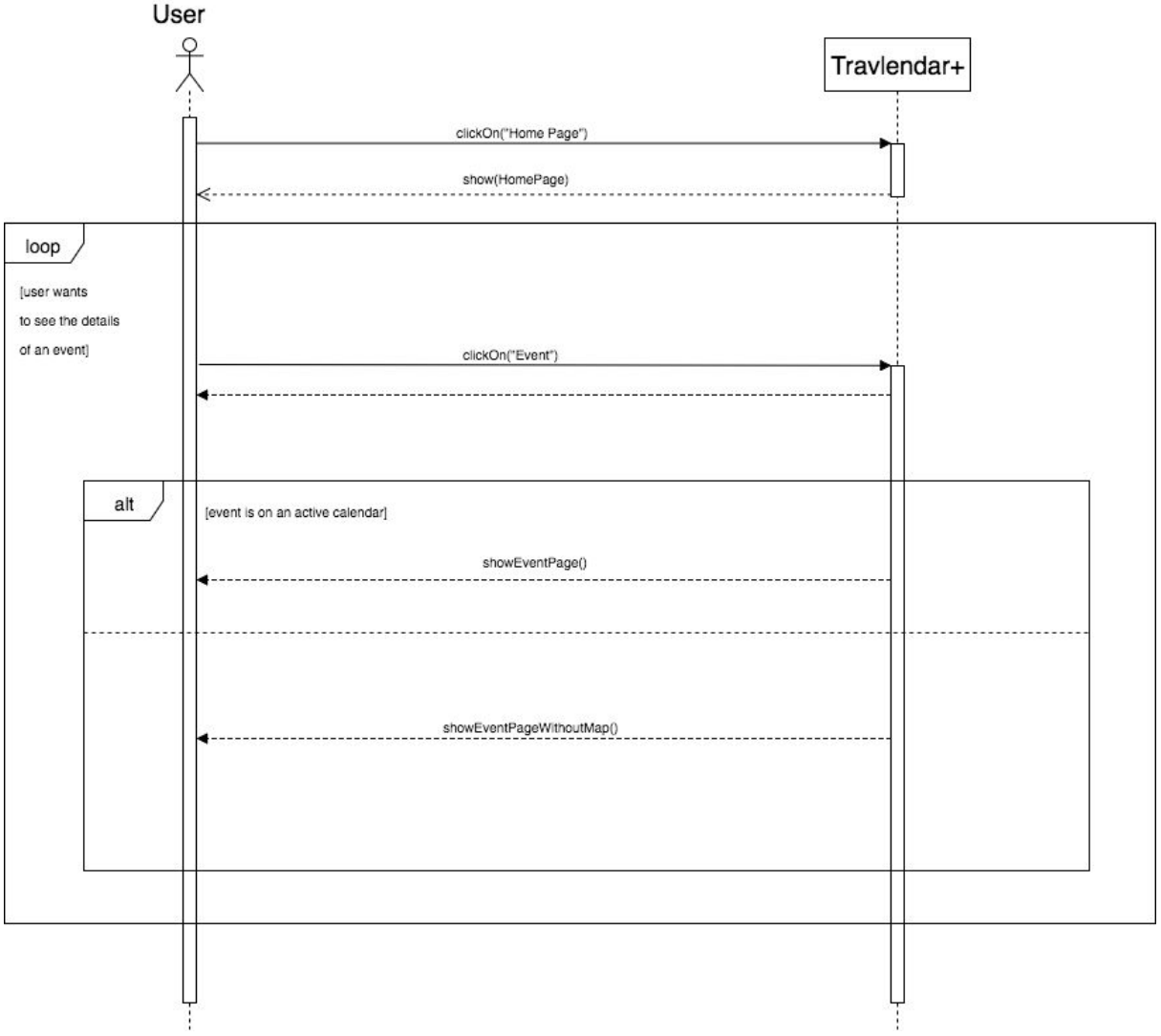
User modifies an event in a calendar



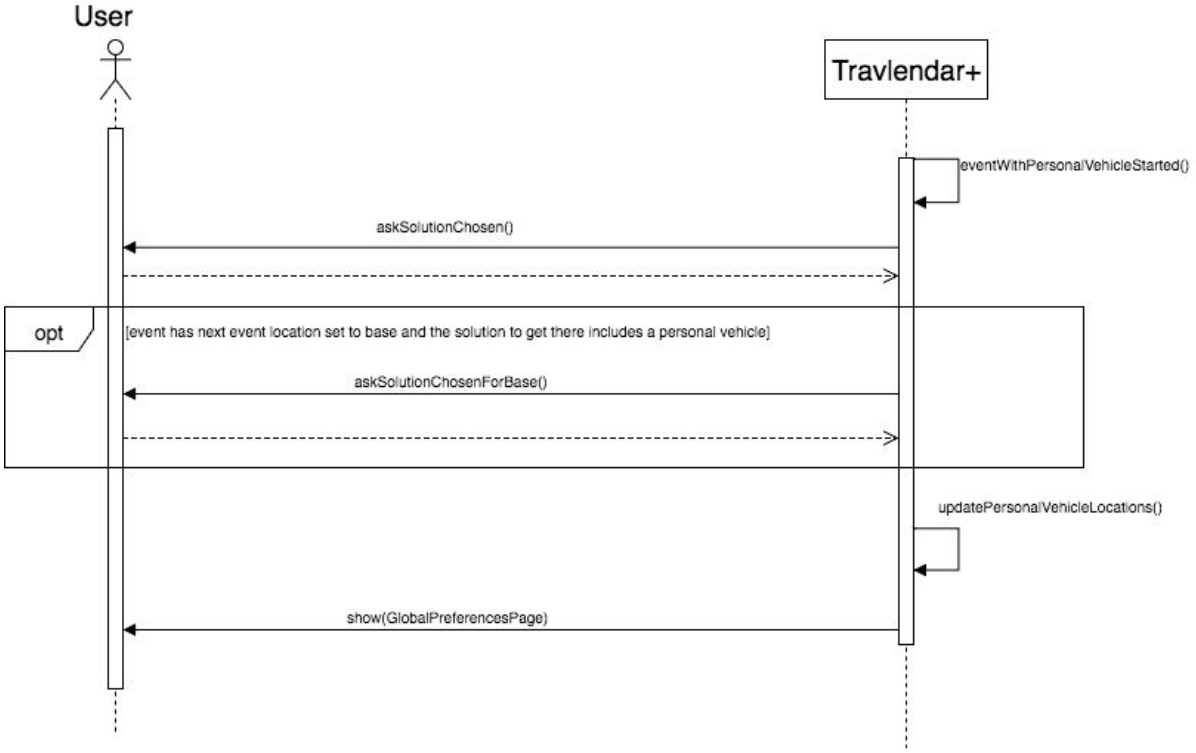
User deletes an event in a calendar



User sees all the events set



User tells the system his/her personal vehicles movements



6 Formal Analysis Using Alloy

6.1 Purpose of the model

The Alloy modelisation is useful, aside to give a visualization of the generated world, to understand how the entities operating inside the application world interact with each other and what types of constraints and facts exist between them.
What it is analysed:

- Chronological order of the events
- Travel solutions suggested by the system for the events in the schedule
- Feasibility of the travel solutions
- Absence of overlappings
- Travel solutions suggested by the system in relation to weather forecast
- Travel solutions suggested by the system in relation to shared travel options
- Calendar and global preferences and how they influence the system

6.2 Alloy code

```
open util/ordering[Time] as Times

-- SIGNATURES

sig Time {}

abstract sig WeatherForecast {}
one sig RAINY extends WeatherForecast {}
one sig NOTRAINY extends WeatherForecast {}

abstract sig TravelOption {
  -- this is the status in the global preferences
  status: one TravelOptionStatus
}
abstract sig PersonalVehicle extends TravelOption {}
one sig Walking extends TravelOption {}
sig Bike extends PersonalVehicle {}
sig Car extends PersonalVehicle {}
sig MotorBike extends PersonalVehicle {}
one sig Bus extends TravelOption {}
one sig Subway extends TravelOption {}
one sig Train extends TravelOption {}
one sig Tram extends TravelOption {}
one sig LightRail extends TravelOption {}
one sig Taxi extends TravelOption {}
one sig Mobike extends TravelOption {}
one sig Enjoy extends TravelOption {}

abstract sig TravelOptionStatus {}
one sig ACTIVATED extends TravelOptionStatus {}
one sig NONACTIVATED extends TravelOptionStatus {}
```

```
sig FeasibleTravelSolution {
  travelTrips: set TravelOption,
  startTime: one Time,
  endTime: one Time
} {
  #travelTrips > 0
  endTime in nexts[startTime]
}

sig Event {
  startTime: one Time,
  endTime: one Time,

  -- this set contains the best feasible travel solution and eventually an
  -- alternative
  travelInstructions: set FeasibleTravelSolution,

  weatherForecast: one WeatherForecast
}{
  endTime in nexts[startTime]
}

sig Calendar {
  events: set Event,
  status: one CalendarStatus,
  travelOptionPreferences: set TravelOptionPreference
}

abstract sig CalendarStatus {}
one sig ACTIVE extends CalendarStatus {}
one sig INACTIVE extends CalendarStatus {}

sig TravelOptionPreference {
  option: one TravelOption
}

-- PREDICATES

-- checks if for the event the best feasible travel solution includes
-- bike/walking/mobike with rain forecast
pred feasibleAndRain [event: Event] {
  (one feasibleTravelSolution1: event.travelInstructions |
    ((Walking in feasibleTravelSolution1.travelTrips ||
      Bike infeasibleTravelSolution1.travelTrips ||
      Mobike in feasibleTravelSolution1.travelTrips) &&
      (event.weatherForecast = RAINY)))
}
```

```
-- checks if for the event the best feasible travel solution includes
-- shared options
pred feasibleAndShared [event: Event] {
  (one feasibleTravelSolution1: event.travelInstructions |
    Mobike in feasibleTravelSolution1.travelTrips ||
    Enjoy in feasibleTravelSolution1.travelTrips)
}

-- checks if for the event the best feasible travel solution includes
-- shared options and bike/walking/mobike with rain forecast
pred feasibleAndRainAndShared [event: Event] {
  (one feasibleTravelSolution1: event.travelInstructions |
    ((Walking in feasibleTravelSolution1.travelTrips ||
      Bike in feasibleTravelSolution1.travelTrips ||
      Mobike in feasibleTravelSolution1.travelTrips) &&
      (event.weatherForecast = RAINY)) &&
      (Mobike in feasibleTravelSolution1.travelTrips ||
      Enjoy in feasibleTravelSolution1.travelTrips))
}

-- checks if the best feasible travel solution includes bike/walking/mobike
-- with rain forecast
-- and only one alternative feasible solution is shown
pred isThereAlternativeForRainWithBikeOrWalking [event: Event] {
  feasibleAndRain[event]
  implies
  (one feasibleTravelSolution2: event.travelInstructions |
    Walking not in feasibleTravelSolution2.travelTrips &&
    Bike not in feasibleTravelSolution2.travelTrips &&
    Mobike not in feasibleTravelSolution2.travelTrips)
}

-- checks if the best feasible travel solution includes shared options
-- and only one alternative feasible solution is shown
pred isThereOnlyOneAlternativeForShared [event: Event] {
  feasibleAndRainAndShared[event]
  implies
  (one feasibleTravelSolution2: event.travelInstructions |
    Mobike not in feasibleTravelSolution2.travelTrips &&
    Enjoy not in feasibleTravelSolution2.travelTrips)
}
```

```
-- checks if the best feasible travel solution includes shared options and
-- bike/walking/mobike with rain forecast
-- and only one alternative feasible solution is shown
pred isThereOnlyOneAlternativeForSharedAndRainForecast [event: Event] {
  feasibleAndRainAndShared[event]
  implies
  (one feasibleTravelSolution2: event.travelInstructions |
    Walking not in feasibleTravelSolution2.travelTrips &&
    Bike not in feasibleTravelSolution2.travelTrips &&
    Mobike not in feasibleTravelSolution2.travelTrips &&
    Enjoy not in feasibleTravelSolution2.travelTrips)
}

-- FACTS

-- Only the best feasible solution is shown unless an alternative is shown
-- because of
-- weather forecast and shared options.
fact onlyTheBestSolutionIsShownIfThereAreNoAlternatives {
  all calendar: Calendar |
    all event: calendar.events |
      calendar.status = ACTIVE implies
      (not feasibleAndRain[event] iff #event.travelInstructions = 1 &&
      not feasibleAndShared[event] iff #event.travelInstructions = 1 &&
      not feasibleAndRainAndShared[event] iff #event.travelInstructions=1)
}

-- A feasible travel solution of an event let the user reach the event
before the event start.
-- The user travels in the interval of time after the previous event end
time and before the start time of the event to reach.
-- The user does not travel in interval of time occupied by events on
active calendars
fact noTravelSolutionsOverlapping {
  all event: Event |
    all feasibleTravelSolution: event.travelInstructions |
      feasibleTravelSolution.endTime in prevs[next[event.startTime]]
                  &&
      (no event1: Event |
        event1 != event &&
        event1.endTime in prevs[next[event.startTime]] &&
        event1.endTime in nexts[feasibleTravelSolution.startTime] &&
        one calendar: Calendar |
          event1 in calendar.events && calendar.status = ACTIVE)
}
```

```
-- There are no events overlapping between all the calendars
fact noEventsOverlapping {
  all event: Event |
    no event1: Event |
      event != event1 && (
      (event1.startTime = event.startTime ||
      event1.endTime = event.endTime) ||
      (event1.startTime in prevs[event.endTime] &&
      event1.endTime in nexts[event.endTime]) ||
      (event1.startTime in nexts[event.startTime] &&
      event1.startTime in prevs[event.endTime]))
}

-- Each event is inserted in a calendar
fact eventsBelongToASingleCalendar {
  all event: Event |
    one calendar: Calendar |
      event in calendar.events
}

-- Each solution belongs to an event
fact travelSolutionsBelongToAnEvent {
      all feasibleTravelSolution: FeasibleTravelSolution |
            one event: Event |
                  feasibleTravelSolution in event.travelInstructions
}

-- A travel option preference is created for a calendar
fact travelOptionPreferencesBelongToASingleCalendar {
  all travelOptionPreference: TravelOptionPreference |
    one calendar: Calendar |
      travelOptionPreference in calendar.travelOptionPreferences
}

-- Only active calendars have events with feasible travel solutions
fact onlyActiveCalendarEventsHaveSolutions {
  all calendar: Calendar|
    all event: calendar.events |
      (calendar.status = INACTIVE implies event.travelInstructions = none)
            &&
      (calendar.status = ACTIVE implies event.travelInstructions != none)
}

-- Only globally activated travel option are used in calendars
fact onlyActivatedTravelOptionAreUsed {
  all feasibleTravelSolution: FeasibleTravelSolution |
    all travelOption: feasibleTravelSolution.travelTrips |
      travelOption.status = ACTIVATED
}
```

```
-- Only travel options globally active can be used on the events of a
-- calendar
fact calendarPreferencesAreRestrictedByGlobal {
  all calendar: Calendar |
    no travelOption: TravelOption |
      travelOption.status = NONACTIVATED &&
      travelOption in calendar.travelOptionPreferences.option
}

-- Only one preference for each travel option
fact {
  all calendar: Calendar |
    all travelOptionPreference1, travelOptionPreference2:
      calendar.travelOptionPreferences |
      travelOptionPreference1 != travelOptionPreference2
      implies
     travelOptionPreference1.option != travelOptionPreference2.option
}

-- Only travel options present in calendar's preferences are used
fact onlyActivatedCalendarTravelOptionAreUsed {
  all calendar: Calendar |
    all event: calendar.events |
      all travelOption: event.travelInstructions.travelTrips |
        travelOption in calendar.travelOptionPreferences.option
}

-- There is only one alternative feasible solution if the best feasible
-- travel solution includes bike/walking/mobike with rain forecast
fact rainWithBikeOrWalkingAndAlternative {
  all event: Event |
    isThereAlternativeForRainWithBikeOrWalking[event]
}

-- There is only one alternative feasible solution if the best feasible
-- travel solution includes shared options
fact sharedAndAlternative {
  all event: Event |
    isThereOnlyOneAlternativeForShared[event]
}

-- There is only one alternative feasible solution if the best feasible
-- travel solution includes shared options and bike/walking/mobike with
-- rain forecast
fact onlyOneAlternativeIsProvided {
  all event: Event |
    isThereOnlyOneAlternativeForSharedAndRainForecast[event]
}
```

```
-- ASSERTS

-- To check that if the best feasible travel solution includes
-- bike/walking/mobike with rain forecast
-- an alternative feasible travel solution is shown
assert alternativeIsProvidedForRain {
  no event: Event |
    (one feasibleTravelSolution1: event.travelInstructions |
      ((Walking in feasibleTravelSolution1.travelTrips ||
      Bike in feasibleTravelSolution1.travelTrips ||
      Mobike in feasibleTravelSolution1.travelTrips) &&
      (event.weatherForecast = RAINY))) &&
      #event.travelInstructions = 1

  all event: Event |
      isThereAlternativeForRainWithBikeOrWalking[event]
}

-- To check that if the best feasible travel solution includes shared
-- options
-- only one alternative feasible travel solution is shown
assert onlyOneAlternativeIsProvidedForShared {
  no event: Event |
    (one feasibleTravelSolution1: event.travelInstructions |
      Mobike in feasibleTravelSolution1.travelTrips ||
      Enjoy in feasibleTravelSolution1.travelTrips) &&
      #event.travelInstructions = 1

  all event: Event |
    isThereOnlyOneAlternativeForShared[event]
}

-- To check that if the best feasible travel solution includes shared
-- options and bike/walking/mobike with rain forecast
-- only one alternative feasible travel solution is shown
assert onlyOneAlternativeForSharedAndRainForecast {
  no event: Event |
    (one feasibleTravelSolution1: event.travelInstructions |
      ((Walking in feasibleTravelSolution1.travelTrips ||
      Bike in feasibleTravelSolution1.travelTrips ||
      Mobike in feasibleTravelSolution1.travelTrips) &&
      (event.weatherForecast = RAINY)) &&
      (Mobike in feasibleTravelSolution1.travelTrips ||
      Enjoy in feasibleTravelSolution1.travelTrips)) &&
      #event.travelInstructions = 1

  all event: Event |
    isThereOnlyOneAlternativeForSharedAndRainForecast[event]
}
```

```
-- To check that at max two travel solutions are shown, the best one and
-- eventually an alternative
assert atMaxTwoTravelSolution {
  all event: Event |
    #event.travelInstructions <= 2
}
```

6.3 World generated

Note how in the random world generated the feasible travel solution for the Event1 is suggested in the interval of time 8-9 where does also take place the event 3, but this event is on an inactive calendar.

```
pred showWorld {
  #Calendar = 3
  #Event = 5
}
run showWorld for 5 but 15 Time
check atMaxTwoTravelSolution for 10
check onlyOneAlternativeForSharedAndRainForecast for 10
check alternativeIsProvidedForShared for 10
check alternativeIsProvidedForRain for 10
```

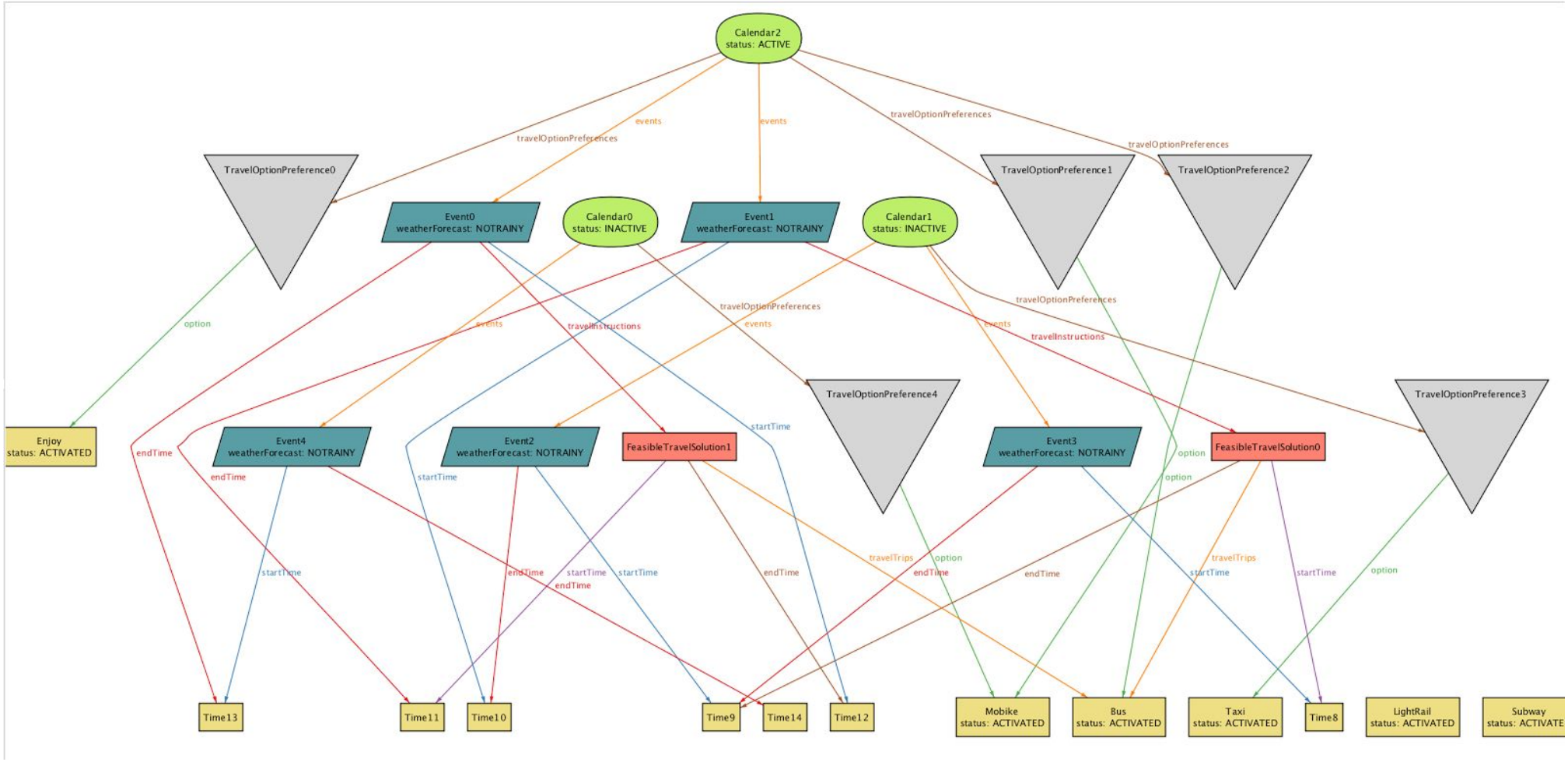
```
Executing "Run showWorld for 5 but 15 Time"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  8783 vars. 523 primary vars. 18820 clauses. 76ms.
  Instance found. Predicate is consistent. 59ms.
```

```
Executing "Check atMaxTwoTravelSolution for 10"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  28013 vars. 1013 primary vars. 57820 clauses. 953ms.
  No counterexample found. Assertion may be valid. 45094ms.
```

```
Executing "Check alternativeIsProvidedForRain for 10"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  27788 vars. 1003 primary vars. 57343 clauses. 435ms.
  No counterexample found. Assertion may be valid. 542ms.
```

```
Executing "Check onlyOneAlternativeIsProvidedForShared for 10"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  27788 vars. 1003 primary vars. 57343 clauses. 401ms.
  No counterexample found. Assertion may be valid. 165101ms.
```

```
Executing "Check onlyOneAlternativeForSharedAndRainForecast for 10"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  27788 vars. 1003 primary vars. 57343 clauses. 290ms.
  No counterexample found. Assertion may be valid. 406ms.
```


Calendar2
status: ACTIVE
travelOptionPreferences
events
events
travelOptionPreferences
travelOptionPreferences
TravelOptionPreference0
TravelOptionPreference1
TravelOptionPreference2
Event0
weatherForecast: NOTRAINY
Calendar0
status: INACTIVE
Event1
weatherForecast: NOTRAINY
Calendar1
status: INACTIVE
option
events
travelInstructions
travelOptionPreferences
events
travelOptionPreferences
events
travelInstructions
TravelOptionPreference4
TravelOptionPreference3
Enjoy
status: ACTIVATED
endTime
Event4
weatherForecast: NOTRAINY
Event2
weatherForecast: NOTRAINY
FeasibleTravelSolution1
startTime
Event3
weatherForecast: NOTRAINY
option
FeasibleTravelSolution0
endTime
option
startTime
startTime
endTime
endTime
startTime
startTime
endTime
endTime
endTime
travelTrips
option
endTime
travelTrips
startTime
startTime
option
Time13
Time11
Time10
Time9
Time14
Time12
Mobike
status: ACTIVATED
Bus
status: ACTIVATED
Taxi
status: ACTIVATED
Time8
LightRail
status: ACTIVATED
Subway
status: ACTIVATE

7 Effort Spent

Lorenzo Pratissoli
10/10/17: 2 hours
11/10/17: 4 hours
15/10/17: 4 hours
17/10/17: 2 hours
21/10/17: 5 hours
22/10/17: 7 hours
23/10/17: 5 hours
25/10/17: 7 hours
27/10/17: 5 hours
28/10/17: 3 hours

Riccardo Novic
11/10/17: 2 hour
13/10/17: 4 hours
14/10/17: 2 hours
15/10/17: 3 hours
16/10/17: 3 hours
17/10/17: 3 hours
24/10/17: 3 hours
25/10/17: 4 hours
27/10/17: 3 hours
28/10/17: 3 hours