Travlendar+ Requirement Analysis and Specification Document

Sinico Matteo, Taglia Andrea

Version 1.0

October 2017

Table of Contents

1. INTRODUCTION **3**
A. Purpose: 3
A.1 Goals: 3
B. Scope: 3
C. Definitions, Acronyms, Abbreviations 3
D. Revision history 4
F. Document Structure 4

2. OVERALL DESCRIPTION **5**
A. Product perspective: 5
B. Product functions: 5
C. User Characteristics 7
D. Assumptions, dependencies and constraints: 7

3. SPECIFIC REQUIREMENTS: **8**
A. External Interface Requirements 8
A.1 User Interfaces 8
A.2 Hardware Interfaces 9
A.3 Software Interfaces 9
A.4 Communication Interfaces 9
B. Functional Requirements: 10
B.1. Scenario 10
B.2. Use case diagram 11
B.3. Use case Tables 11
B.4. Sequence diagrams 21
C. Performance Requirements 38
D. Design Constraints 38
D.1 Standards compliance 38
D.2 Hardware limitations 38
E. Software System Attributes 38
E.1 Reliability 38
E.2 Availability 38
E.3 Security 39
E.4 Maintainability 39
E.5 Portability 39

4. FORMAL ANALYSIS USING ALLOY: **39**

5. EFFORT SPENT: **48**

6. REFERENCES **48**
A. Bibliography 48
B. Used Tools 48

1. INTRODUCTION

A. Purpose:

This document addresses the requirement analysis of the application Travlendar+, which will be developed in order to automatize the process of scheduling meetings at various location all across a city. No previous version of this application have been developed. The intended audience is formed by the developers of the software, the financial stakeholders and the end users.

A.1 Goals:

[G1]: Allow the user to reach his appointments on time;

[G2]: Allow the user to reach his appointments by means of travel based on his preferences;

[G3]: Allow the user to reach his appointments using a suboptimal path;

[G4]: Warning the user when he would try to schedule a new appointment at a location which is not reachable on time.

[G5]: Allow the users to have a quick overview about the organization of transports during the day.

B. Scope:

Travlendar+ is an application which will automatize the process of scheduling the user's meeting during the day. Every day we spend a lot of time thinking and planning how to reach a location at a certain time, Travlendar+ allows users to save all this time. Users will be able to manage their own calendar by adding, updating or deleting personal appointments at a certain location and time during the day, and they will also be able to edit their preference of travel (use their own car, avoid public transportation, use bike sharing…). Then the application will compute a personal schedule taking into account the user's calendar, user's preferences and the weather forecast.
In order to accomplish this task, the application will receive from the external world the following input: weather forecast from a third-party application, personal users appointment from the user, transportation issues, like strikes, accidents or constructions, and also urban setup (streets, indications, estimated times), from another third party application.

C. Definitions, Acronyms, Abbreviations

C.1. Definitions

- Preferences of travel: the set of preference chosen by the user, about what kind of means of transport prefer to use between: car, bike, by foot, public means of transport; what kind of service would like to use between: car sharing, bike sharing;

carbon footprint leverage and flexible break preferences, when he wants to have a break and how much does it takes.
- Day-travel Proposal: the plan of the day computed by the application for the user, which include the user's appointment, the means of transport suggested to move from an appointment to the other and the estimated time of the movement, it could be made of more than one travel-piece.
- Travel Piece: it is intended as a single atomic unit as part of a more complex route made by different transport means and different times, all represented by travel pieces. In details a travel piece it's composed by the following field:
    - From: the starting location;
    - By: the means of transport chosen;
    - Leave by: the time at which is suggested to leave, in order to be on time.
- Transportation issues: Public, personal or shared transports break down, strikes of the public worker, accident.
- Application: by this term we refer to the web-app application;
- Web-app: a software program that runs on a web server
- Appointment and Event are used as synonymous.

C.2. Acronyms

• API: application program interface.

C.3. Abbreviations

[Gn] The n-th goal.
[Wn] The n-th wireframe.
[An] The n-th dependency.
[Dn] The n-th domain assumption.
[Rm.n] The n-th functional requirement related to the m-th goal.
[NFRn] The n-th non-functional requirement.
[UCn] The n-th use case diagram.
[SDn] The n-th sequence diagram.
[SCn] The n-th scenario.
[An] The n-th assumption.
[Dn] The n-th dependencies
[Cn] The n-th constrain.

D. Revision history

1. 27/10/2017 Document v1.0 ready

F. Document Structure

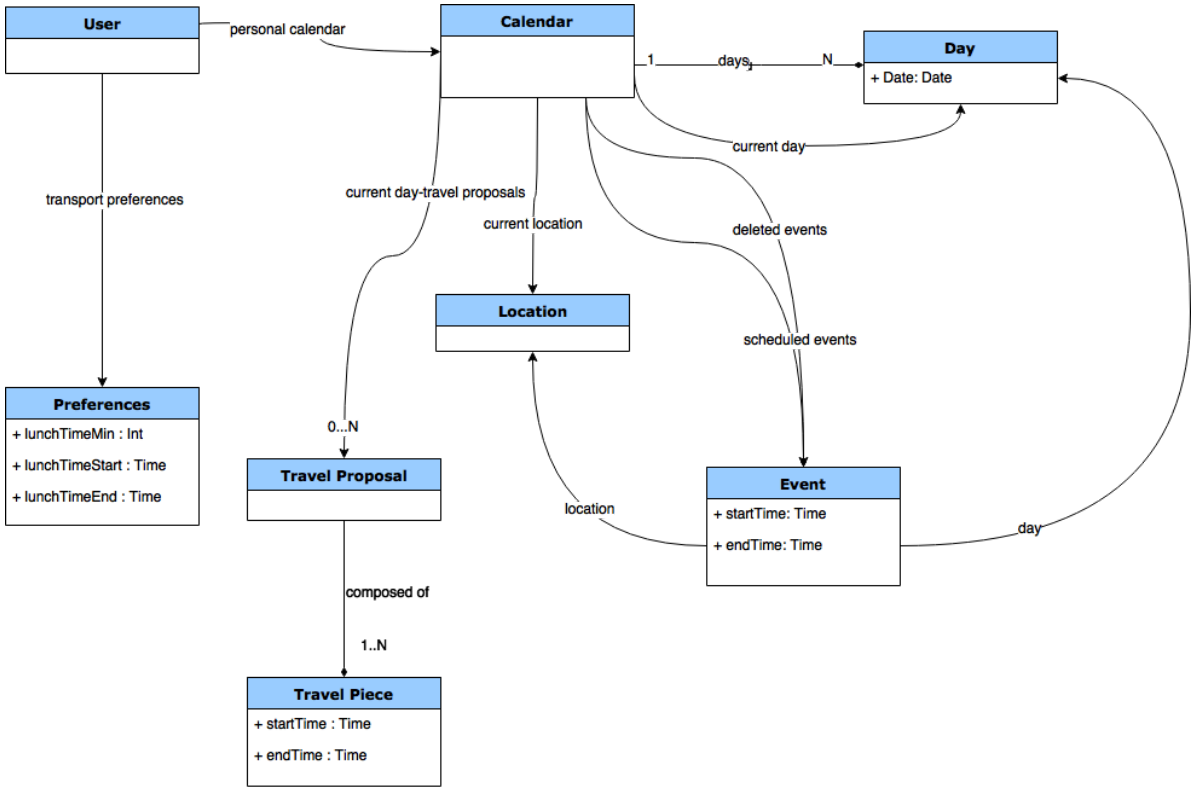
- Introduction: this section gives a brief explanation of the system to be. The introduction also provides both the basic knowledge and the referenced documents to properly keep on reading the document itself.

- Overall Description: in this section it will be provided a solid background for the requirement so that they become easier to understand.
- Specific Requirements: the section contains the software requirements described with UML diagrams, giving an easy and formal overview to software requirements.
- Formal Analysis Using Alloy
- Effort Spent: working time spent by each team member.
- References

2. OVERALL DESCRIPTION

A. Product perspective:

It is provided the class diagram modelling the world Travlendar+ has to deal with. The following class diagram is intended for high level usage. Further details will be presented in the Design Document.



B. Product functions:

[G1]: Allow the user to reach his appointments on time;

- [R1.1] A log in functionality must be provided in order to authenticate the user.
- [R1.2] User's data (calendar appointments, preferences, personal data) must be kept secret to the user himself and to anybody else.

- [R1.3] A password recovery functionality must be available in order for the user not to permanently lose access to the service.
- [R1.4] Users must have a personal calendar.
- [R1.5] Users must be able to edit their personal data.

[G2]: Allow the user to reach his appointments by means of travel based on his preferences;

- [R2.1] Users must be able to add an event to their personal calendar giving a title, a day, a time at which the event should be scheduled in the calendar.
- [R2.2] Users must be able to edit an existing event changing the title.
- [R2.3] Users must be able to delete an already existing event in their personal calendar.
- [R2.4] Users must be able to change their preference about car usage.
- [R2.5] Users must be able to change their preference about bike usage.
- [R2.6] Users must be able to change their preference about walking.
- [R2.7] Users must be able to change their preference about carbon footprint.
- [R2.8] Users must be able to change their preference about public transports.
- [R2.9] Users must be able to edit preferences on break times choosing a lower and upper bound for lunch and dinner times and a minimum break time.

[G3]: Allow the user to reach his appointments using a suboptimal path;

- [R3.1] Computation of optimal route must take into account user preferences.
- [R3.2] Computation of optimal route must take into account weather forecast.
- [R3.3] Computation of optimal route should be made more times and each with different weight on preferences in order to give different day-travel proposals.
- [R3.4] There must be a way to let the user switch among different proposal.
- [R3.5] No appointment can be successfully scheduled after another if it would not respect flexible break times.
- [R3.6] Change in break times preferences are not taken into account for previously scheduled appointments.

[G4]: Warning the user when he would try to schedule a new appointment at a location which is not reachable on time.

- [R4.1] No overlapping appointments are allowed. A warning would show up and the user may not be able to schedule the event.
- [R4.2] Transport time estimation between appointment location being added and previous location on schedule must be done based on public transports only.
- [R4.3] The application must always be aware of the previous user location with respect to the newly added event.

[G5]: Allow the users to have a quick overview about the organization of transports during the day.

- [R5.1] The application must compute a travel proposal to next event location to show to the user.
- [R5.2] A brief description of each Travel Piece of a day Travel Proposal should be shown to the user.
- [R5.3] Allow users to check any day scheduled appointments;
- [R5.4] There must be a comprehensive view over any given day.
- [R5.5] Each appointment should have an estimated time of arrival (computed based on public transports) attached.

C. User Characteristics

Software must really take care of Usability standards and comply with the Material Design UI standards. The application must be easy to use from the user point of view. During the development of the user interface it will be necessary to take into accounts the following:

- Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design;
- Efficiency: Once users have learned the design, how quickly can they perform tasks;
- Memorability: When users return to the design after a period of not using it, how easily can they reestablish proficiency;
- Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors;
- Satisfaction: How pleasant is it to use the design.

D. Assumptions, dependencies and constraints:

[A1] Every appointment is inserted correctly by the user at the right location and at the right time;

[A2] The weather forecast offered by a forecast third party application are considered to be right in a certain probability ensured by the application itself

[A3] Default estimated time between a place and another is computed based on public transport times.

> **[A3.1]** Users should be able to use public transports, otherwise the default computation process on time between a place and another would make no sense to them.

[A4] The directions and the estimated time offered by a third-party application are sufficiently accurate.

[A5] Transportation issues may take place with no previous notice.

[A6] New user calendar starts empty.

[A7] The application must be aware of the current time of the day and the current date.

[A8] Events cannot last longer than 20 hours.

[D1] Default estimated time between a place and another is based on a third party application.

[D2] The application will rely on car and bike sharing applications in order to use those services.

[D3] Car/bike sharing services should always be up and running.

[D4] Weather forecast are offered by a third party application.

[D4] The application must be aware of the next user's scheduled appointment.

[C1] A login functionality is required because the application is web-based, so has to store the personal data of the user in an external storage.

[C2] Car or bike sharing entity should always be located up to one kilometer from any point in Milan.

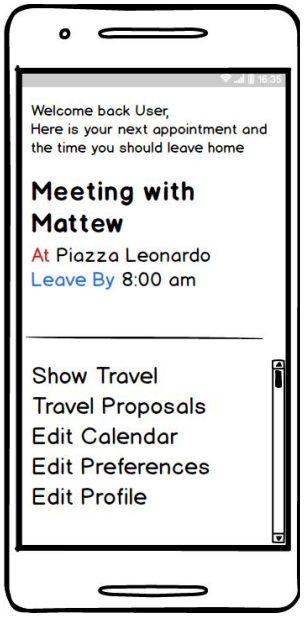
3. SPECIFIC REQUIREMENTS:

A. External Interface Requirements

A.1 User Interfaces

Every software has its own way to be used. Users always need to learn. Bots speak users' language, an interface that they already know. However, too much typing is something we believe users do not appreciate, especially in mobile environments.
For the reasons above it is required that the users can navigate the application just by choosing one of a few options given in the input area of the application user interface.
The other part of the screen has to be devoted to read-only output view. It would make user experience neater by getting output and input views radically separated.
A breadcrumbs-like function must always be available in order for the users not to get lost among the tabs.
Below a few UI wireframes based on scenario **[SC1]** are represented:

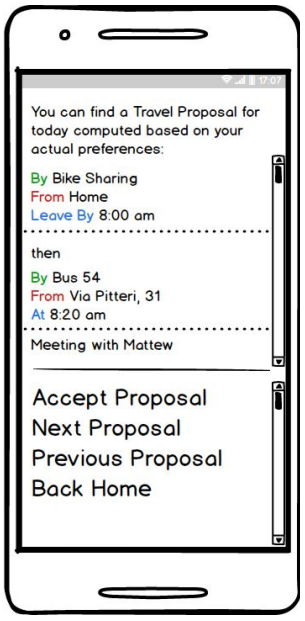
[W1] Home Screen



The home screen wireframe on the left shows how the required UI for Travlendar+ should always have an introductory text at the top, the information to show to the user (output panel) in the middle and the input panel at the button, where the user can choose among the options just by tapping.

The example scenario comes from **[SC1]**

[W2] Travel Proposal Screen



The User gets to this screen by tapping on the Travel Proposals choice in the input panel from the Home Screen. Here a first day travel proposal gets computed and is shown to the user which can choose to accept it or have a look at others by tapping next or previous.
Once the proposal is accepted it becomes the one that the Show Travel option from the home screen will show.

The example scenario comes from **[SC1]**

A.2 Hardware Interfaces

As the application is intended for a web oriented architecture, a web browser will abstract from the hardware interfaces.

A.3 Software Interfaces

- Google Maps APIs are needed to compute the estimated time it would take to get from one place to another by different means of transportation.
- Bike sharing services APIs are necessary to estimate the feasibility of alternative cycling routes
- Car sharing services APIs are necessary to estimate the feasibility of alternative driving routes
- Weather forecast services API is needed to taking in account also the weather condition as a variable in the algorithm to estimate the feasible schedule.

A.4 Communication Interfaces

TCP protocol served on a HTTPS at default port 443 for external APIs requests.

B. Functional Requirements:

B.1. Scenario

[SC1] Andrew is a user of Travlendar+. He has scheduled an event on the October 17th which he named as “Meeting with Mattew” a few days before. The event will be at 9:00 am in Piazza Leonardo. On the day of the event he opened the application and tapped on Travel Proposal from the input panel. Travlendar+ computed a few options for the day. After switching among a few options he choose the one with a 20 minutes bicycling (with a bike sharing service) to via pitteri 31 where he would take the bus 54 and be on time for the event. After the meeting other travel pieces are provided which won’t be described in this scenario for brevity. As he gets back to the home screen he can have a quick look at the next appointment and the time to leave home.

→ UC17, UC18, UC19, UC20, UC21

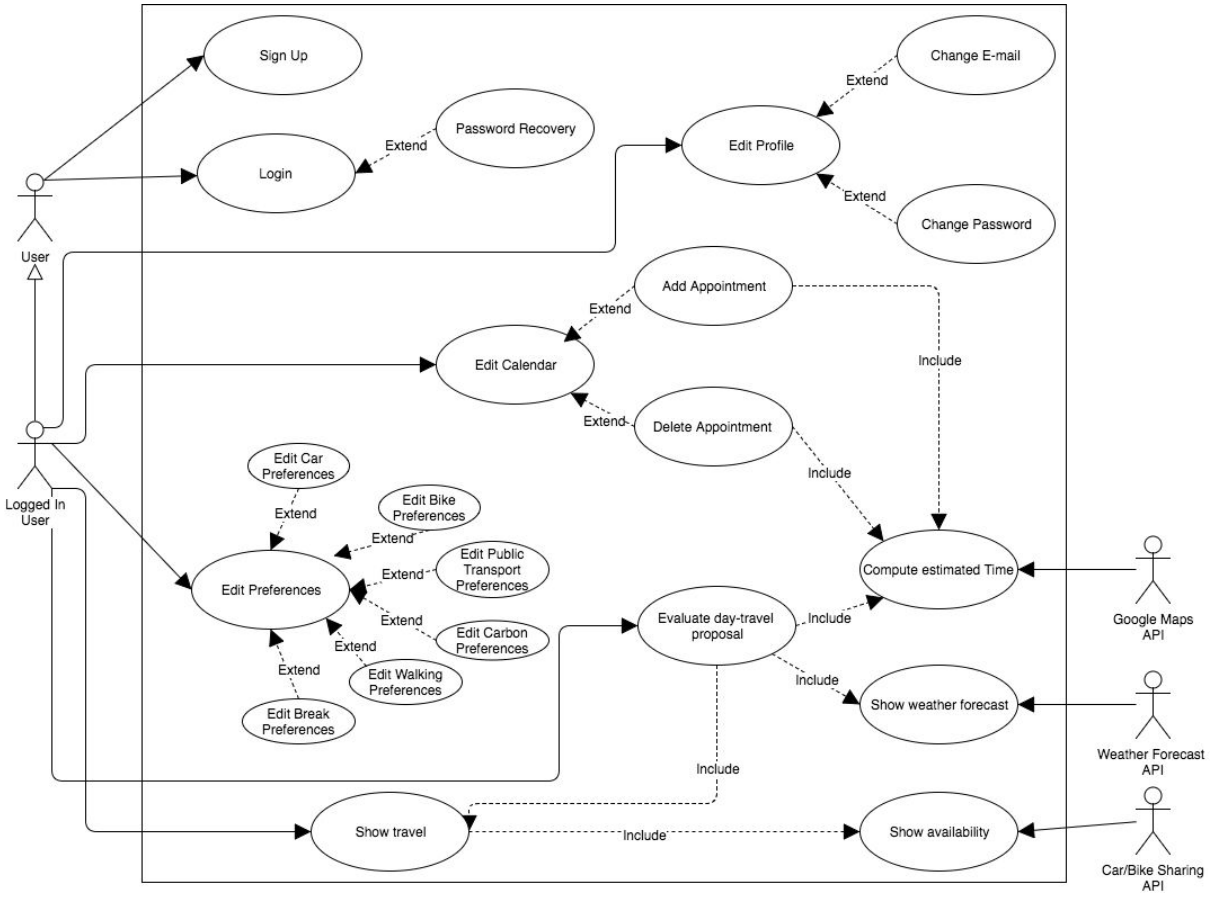
[SC2] Mattew is an user of Travlendar+. He has just met with Andrew, who has invited him to see together to his house the football match Inter-Catania scheduled on the October 15 at 20:45. So Mattew wants to add this new appointment to the Travlendar+ calendar, he opens the application, he logs in by clicking the log in button and fulfilling the records. Then he clicks on the edit calendar button, then on the add appointment button and finally he can fulfill the records with the info about the appointment, title: Big Match, location: Andrew’s house, start time: 20:45, end time: 23:00. Few hours late he receives a call by Andrew who says him that his house is occupied and they will see the match to Henry’s house. So now Mattew has to modify the appointment. In order to do so, he clicks on the edit calendar button, then on delete appointment button and finally he choose the “Big Match” appointment. Then he returns on the edit calendar page and clicks to add appointment and create the new appointment with the correct information.

→ UC2, UC7, UC8, UC9

[SC3] Alex is an user of Travlendar+. He has recently managed to get his drive license. Therefore he wants to enable his car preferences, which was previously set to disabled, and instead disable the public transport preferences in order to practice his driving ability. So he clicks on the edit preference button, then on edit car preferences. Here he clicks on the enable preference button, on use car sharing service button, on use car for short movement button and on avoid traffic button. Then he returns to the edit preferences page and clicks on the edit Public transport preferences, and on the disable button, because he wants to practice with car for a little time.

→ UC10, UC 11, UC 13

B.2. Use case diagram



B.3. Use case Tables

[UC1]	Sign Up
Description	This use case allow the user to sign up to the system.
Actors	User
Entry conditions	The User isn't registered to the Travlendar+ application

Flow of events	• The user open the application
	• The system shows him the home page
	• He clicks on the sign up button
	• The system shows him the sign up page
	• The user fulfills the sign up form which requires the following information to be inserted:
	  ○ First name
	  ○ Last name
	  ○ Username
	  ○ Password
	  ○ Sex
	  ○ E-mail
	  ○ Date of birth

	• He submits the form • He confirms his account by clicking a link in a confirmation e-mail previously sent automatically by the system.
Exit Conditions	Account confirmation successfully done
Exceptions	In case of missing mandatory information or username already in use or incorrect data then the system shows the user an error message highlighting the incorrect records

[UC2]	Login
Description	This use case allow the user to log in the system.
Actors	User
Entry conditions	The User is already registered to the Travlendar+ application
Flow of events	• The user open the application • The system shows the home page • The user clicks on the log in button • The system shows the log in page • The user insert his username and password • The user click on the log in button
Exit Conditions	The user successfully access the application
Exceptions	The user has forgotten his password so the system ask him if he wants to recover it by starting the **password recovery** use case

[UC3]	Password Recovery
Description	This use case allow the user to recover his password in the case he has forgotten it
Actors	User
Entry conditions	The User has forgotten his password and has clicked on the password recovery button
Flow of events	• The system shows the user the password recovery page • The user fulfills the email record • The user clicks on the send code button • The user receives the security code on his e-mail account • The system shows the user the code checking page to allow him to change his password • The user fulfills the code record with the received code • The user submits it • The system generates a new password and replaces the old one • The system sends the new password to the user e-mail
Exit Conditions	The system send the new password to the user

Exceptions	If the user enters a non-registered mail or an invalid security code then the recovery procedure must be aborted by the system showing an error message. If there is a connection loss then the procedure is also aborted

[UC4]	Edit Profile
Description	This use case allow the user to edit the main field of his profile
Actors	Logged in User
Entry conditions	The logged in User needs to change some information about his profile and clicks the edit profile button
Flow of events	• If the user needs to change his account password or his e-mail, then he respectively clicks on the change password button which starts the **Change Password** use case or on the change e-mail button which starts the **Change E-mail** use case. • Otherwise, the user simply changes the information to be modified • The user submits the updated profile
Exit Conditions	The system successfully stores the information the user has just changed
Exceptions	If the changed data are not valid the system shows the user an error message highlighting the invalid records

[UC5]	Change E-mail
Description	This use case allow the user to change the e-mail on his account
Actors	Logged in User
Entry conditions	While the logged in User is editing his profile he clicks on the change e-mail button
Flow of events	• The system shows the user the change e-mail page • The user fulfills the record with his new e-mail • He clicks over the submit button to send the new information • The system sends him a confirmation e-mail
Exit Conditions	The user confirms his new e-mail
Exceptions	If the new e-mail is not valid the system shows an error message

[UC6]	Change Password
Description	This use case allow the user to change his password
Actors	Logged in User

Entry conditions	While the logged in User is editing his profile he clicks on the change password button
Flow of events	• The system shows the user the change password page • The user fulfills the records, the first one related to the old password and the second related to the new one • He clicks over the submit button to send the new information
Exit Conditions	The system stores the new password by replacing the old one
Exceptions	If the old password is wrong or the new one is invalid the system shows the user an error message highlighting the wrong record

[UC7]	Edit calendar
Description	This use case allow the user to view his personal calendar
Actors	Logged in User
Entry conditions	The logged in User wants to check or to edit his personal calendar, so he clicks on the button edit calendar
Flow of events	• The system show the personal calendar page • The user select a specific day among the calendar • The system show the default day-proposal computed based on public transport times and the appointment already inserted. • If the user wants either to add an appointment or to delete an appointment or to edit an appointment already added, then he respectively clicks on the Add Appointment button which starts the **Add Appointment** use case, the Delete Appointment button which start the **Delete Appointment** use case, or he selects the appointment and he can only change the title of the appointment.
Exit Conditions	The system successfully stores the information the user has just changed
Exceptions	

[UC8]	Add Appointment
Description	This use case allow the user to add an appointment to his calendar
Actors	Logged in User
Entry conditions	The User has already clicked the Add Appointment button

Flow of events	• The system show the add appointment page
	• The system ask the user to insert these following data:
	  ○ Appointment title
	  ○ Data
	  ○ Start Time

	○ End time ○ Location address ● The system asks a third party application to compute the estimated time based on the public transport time. This request starts the **Compute estimated time** use case.
Exit Conditions	The user successfully add an appointment to his personal calendar
Exceptions	If one of the following conditions happen: ● the just added appointment isn't reachable in the estimated time ● the following appointment isn't reachable in the estimated time ● two or more appointment overlap then a warning is generated by the system.

[UC9]	Delete Appointment
Description	This use case allow the user to delete an appointment from his personal calendar
Actors	Logged in User
Entry conditions	The User has already clicked the delete appointment button
Flow of events	● The system shows the delete appointment page. ● The system asks the user to choose the appointment to delete. ● The system shows the details of the appointment to delete and asks the user if is it sure to delete the following appointment. ● The user clicks on the button to confirm the appointment. ● The system remove the appointment ● The system asks a third party application to compute the estimated time based on the public transport time. This request starts the **Compute estimated time** use case.
Exit Conditions	The user successfully delete the selected appointment.
Exceptions	

[UC10]	Edit Preferences
Description	This use case allow the user to edit his travel preferences.
Actors	Logged in User
Entry conditions	The logged in User wants to check or to edit his travel preferences, so he clicks on the button edit preferences
Flow of events	● The system shows the edit preferences page ● The user could modify the following preferences by clicking the respectively button, then the respectively use case starts: ○ Car preferences ○ Bike preferences ○ Public transport preferences ○ Walking preferences

	○ Carbon footprint preferences ○ Break preferences
Exit Conditions	The user successfully modify his travel preferences.
Exceptions	

[UC11]	Edit Car Preferences
Description	This use case allow the user to edit his car preferences.
Actors	Logged in User
Entry conditions	The logged in User has clicked on the edit Car preferences button.
Flow of events	● The system shows the edit Car preferences page ● The user could check and modify these preferences that are yes or no question. ○ Disable* ○ Use your own car ○ Use car sharing service ○ Use car for (long/short) movement** ○ Avoid traffic ○ Avoid motorway ○ Avoid toll ● Then the system updates the car travel preferences
Exit Conditions	The user successfully modify his travel preferences, the system shows the edit preferences page
Exceptions	

[UC12]	Edit Bike Preferences
Description	This use case allow the user to edit his bike preferences.
Actors	Logged in User
Entry conditions	The logged in User has clicked on the edit Bike preferences button.
Flow of events	● The system shows the edit Bike preferences page ● The user could check and modify these preferences that are yes or no question: ○ Disable* ○ Use your own bike ○ Use bike sharing service ○ Use bike for (long/short) movement** ● Then the system updates the bike travel preferences
Exit Conditions	The user successfully modify his travel preferences, the system shows the edit preferences page

Exceptions	

[UC13]	Edit Public transport Preferences
Description	This use case allow the user to edit his public transport preferences.
Actors	Logged in User
Entry conditions	The logged in User has clicked on the edit Public transport Preferences button.

Flow of events	• The system shows the edit Public transport references page
	• The user could check and modify these preferences that are yes or no question:
	  ○ Disable*
	  ○ Use public transports for (long/short) movement**
	  ○ avoid underground
	  ○ avoid tram
	  ○ avoid buses
	  ○ avoid trains
	• Then the system updates the public transports travel preferences

Exit Conditions	The user successfully modify his travel preferences, the system shows the edit preferences page
Exceptions	

[UC14]	Edit Walking Preferences
Description	This use case allow the user to edit his walking preferences.
Actors	Logged in User
Entry conditions	The logged in User has clicked on the edit walking Preferences button.

Flow of events	• The system shows the edit walking preferences page
	• The user could check and modify these preferences that are yes or no question:
	  ○ Disable*
	  ○ Walking for (long/short) movement**
	  ○ Avoid congested streets
	• Then the system updates the walking travel preferences

Exit Conditions	The user successfully modify his travel preferences, the system shows the edit preferences page
Exceptions	

[UC15]	Edit Carbon footprint Preferences

Description	This use case allow the user to edit his Carbon footprint references.
Actors	Logged in User
Entry conditions	The logged in User has clicked on the edit Carboon footprint preferences button.
Flow of events	• The system shows the edit Carboon footprint preferences page • The user could choose among two level (low, high) of priority to associate to the importance of minimize the carbon footprint. Or could also disable the preferences clicking on the disable* button. • Then the system updates the carbon footprint travel preferences
Exit Conditions	The user successfully modify his travel preferences, the system shows the edit preferences page
Exceptions	

[UC16]	Edit Break Preferences
Description	This use case allow the user to edit his break references.
Actors	Logged in User
Entry conditions	The logged in User has clicked on the edit break preferences button.

Flow of events	• The system shows the edit break preferences page
	• The user could set the following data to edit his break preferences:
	  ○ Start time (lower bound)
	  ○ End time (upper bound)
	  ○ minimum time
	or could click the disable button to disable* the preferences.
	• Then the system updates the bike travel preferences

Exit Conditions	The user successfully modify his travel preferences, the system shows the edit preferences page
Exceptions	

[UC17]	Evaluate day-travel proposal
Description	This use case allow the user to view a few options of the day-travel proposal, and choose one of them
Actors	Logged in User
Entry conditions	The User has clicked on the Travel Proposal Button
Flow of events	• The system request the weather forecast API the weather forecast for each travel piece in the user schedule. Each request

	starts the **Show weather forecast** user case • Then the system, based on the user preferences, create a few day-travel proposal, for the current day. • The system, request the google maps API to compute each travel piece that is in the schedule options. Each request starts the **Compute estimated time** use case. • Finally the system discard the solution that are not possible and choose the best ones to present to the user. • The system shows the Travel page. • The user could scroll the day-travel proposal to check the solution computed by the application, or could switch to the next proposal by clicking the next button. • When the user finds the best proposal for him,he can accept the proposal by clicking the Accept button. • The system records the day-travel proposal accepted.
Exit Conditions	The system starts the **show travel** use case.
Exceptions	

[UC18]	Compute Estimated Time
Description	This use case allow the google maps API to communicate with the google maps service to compute the estimated time of a travel piece.
Actors	google Maps API
Entry conditions	One of the following conditions has just happened: • the user has added an appointment • the user has deleted an appointment • the evaluate day-travel proposal use case request to compute a travel piece.

Flow of events	• The google maps API create a new request to be sent to the google maps services with the following data:
	  ○ Location of the appointment A
	  ○ End time of the appointment A
	  ○ Location of the appointment B
	  ○ Means of transport selected X
	• The API send the request to the google maps services, and wait for an answer.
	• The services compute the estimated time to reach B from A, starting at the end time of A, with X, and answer the API.

Exit Conditions	The google maps API successfully receives the answer from the google maps service, and makes it available to the system.
Exceptions	If the API cannot receive an answer because of connection loss, sends back an error message

[UC19]	Show travel

Description	This use case allow the user to see the day-travel schedule previously accepted.
Actors	Logged in User
Entry conditions	The user has already accepted one of the day-travel proposal offered by the system in the Evaluate day-travel proposal
Flow of events	• The user can scroll the day-travel schedule to see his next appointments and all the directions and informations that needs to reach them • If the day-travel schedule suggest to use a shared bike or car the user can click on the show availability button which starts the **Show availability** use case.
Exit Conditions	The user successfully see his day-travel schedule.
Exceptions	

[UC20]	Show weather forecast
Description	This use case allow the weather forecast API to communicate with the weather forecast service.
Actors	Weather forecast services API
Entry conditions	The system has requested the weather forecast API the weather forecast for the current day

Flow of events	• The API request the weather forecast to the third party weather forecast services, giving it:
	  ○ the location of the appointment A
	  ○ the start time of A
	  ○ the end time of A
	  ○ the current date
	• Then waits for an answer from the weather forecast service.

Exit Conditions	The API receive an answer from the weather forecast service, and makes it available to the system.
Exceptions	If the API cannot receive an answer because of connection loss, sends back an error message

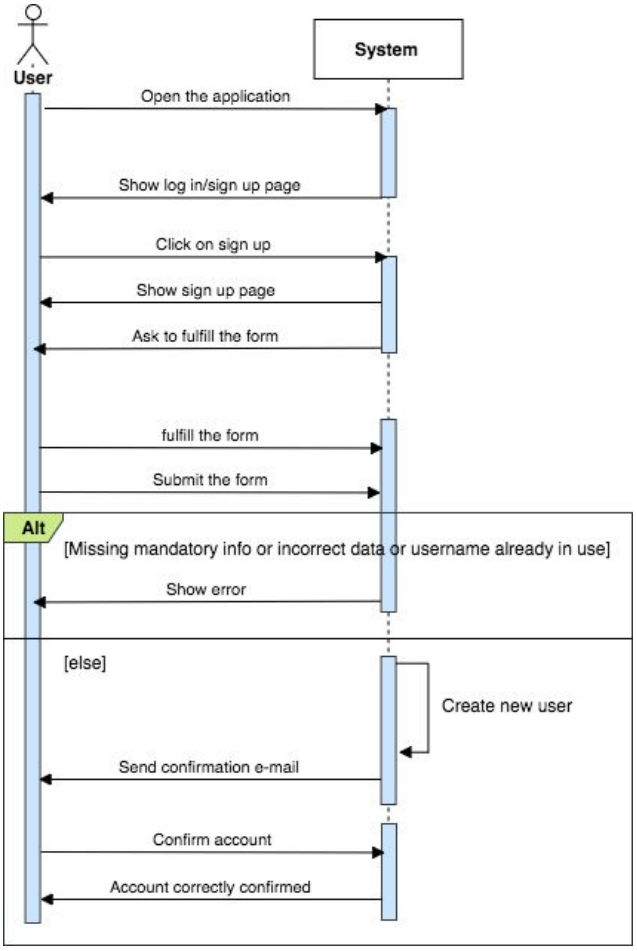
[UC21]	Show availability
Description	This use case allow the car/bike sharing API to communicate with the car/bike sharing services to show the car/bike available
Actors	Car sharing API, Bike sharing API
Entry conditions	The user has clicked on the show availability button
Flow of events	• The sharing APIs request to the sharing services to show the

	current available car/bike giving it the current location, and wait for an answer.
Exit Conditions	The sharing APIs receive an answer from the sharing services, and make it available to the system that shows to the user the available car/bike
Exceptions	If the APIs cannot receive an answer because of connection loss, sends back an error message

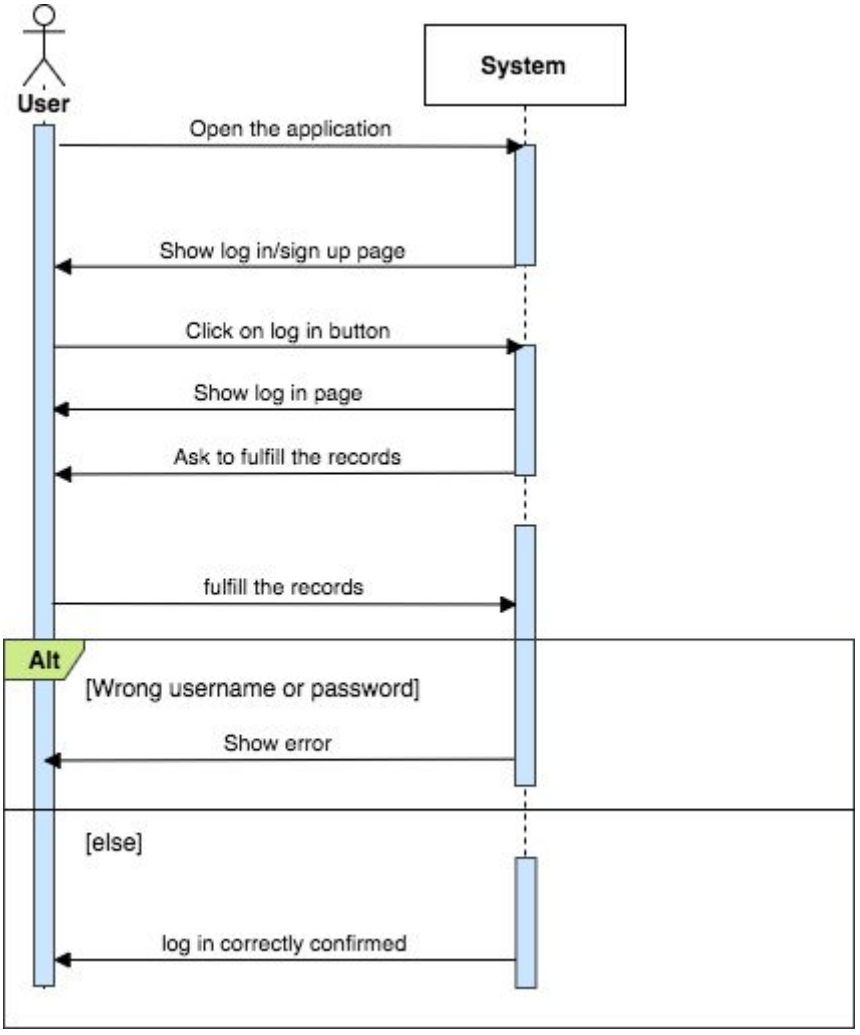
*If the user set the disable preferences to true, then all the other preferences are disabled, and the algorithm doesn't take in account this category of preference to compute the day-travel proposal.
**The User can only choose among two option offered by the system (long, short) in term of distance.

B.4. Sequence diagrams

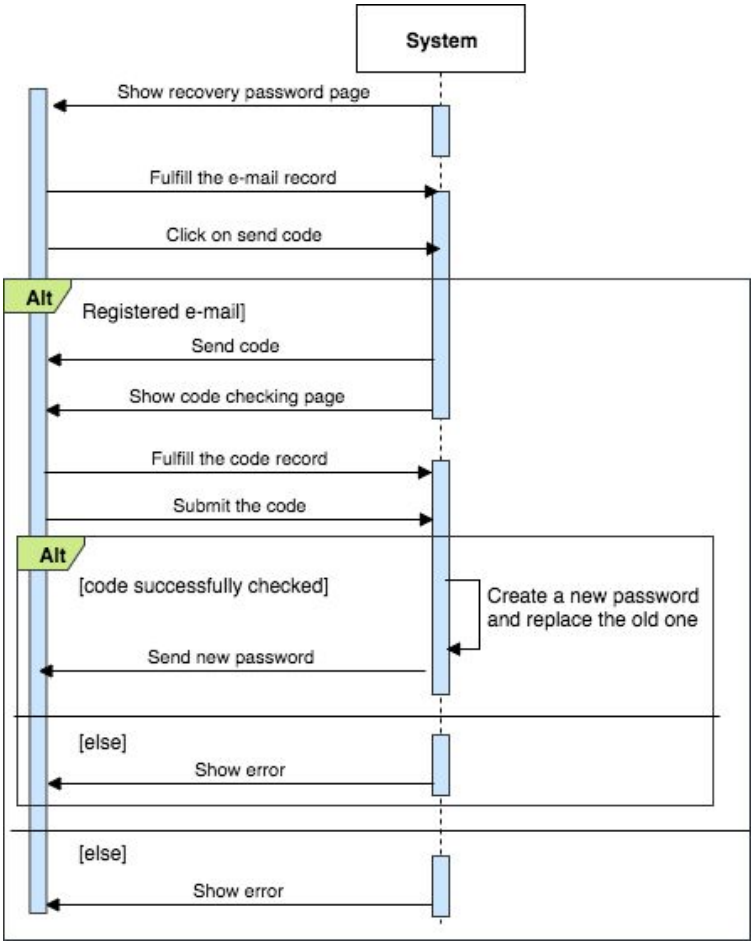
[SD1] : Sign Up



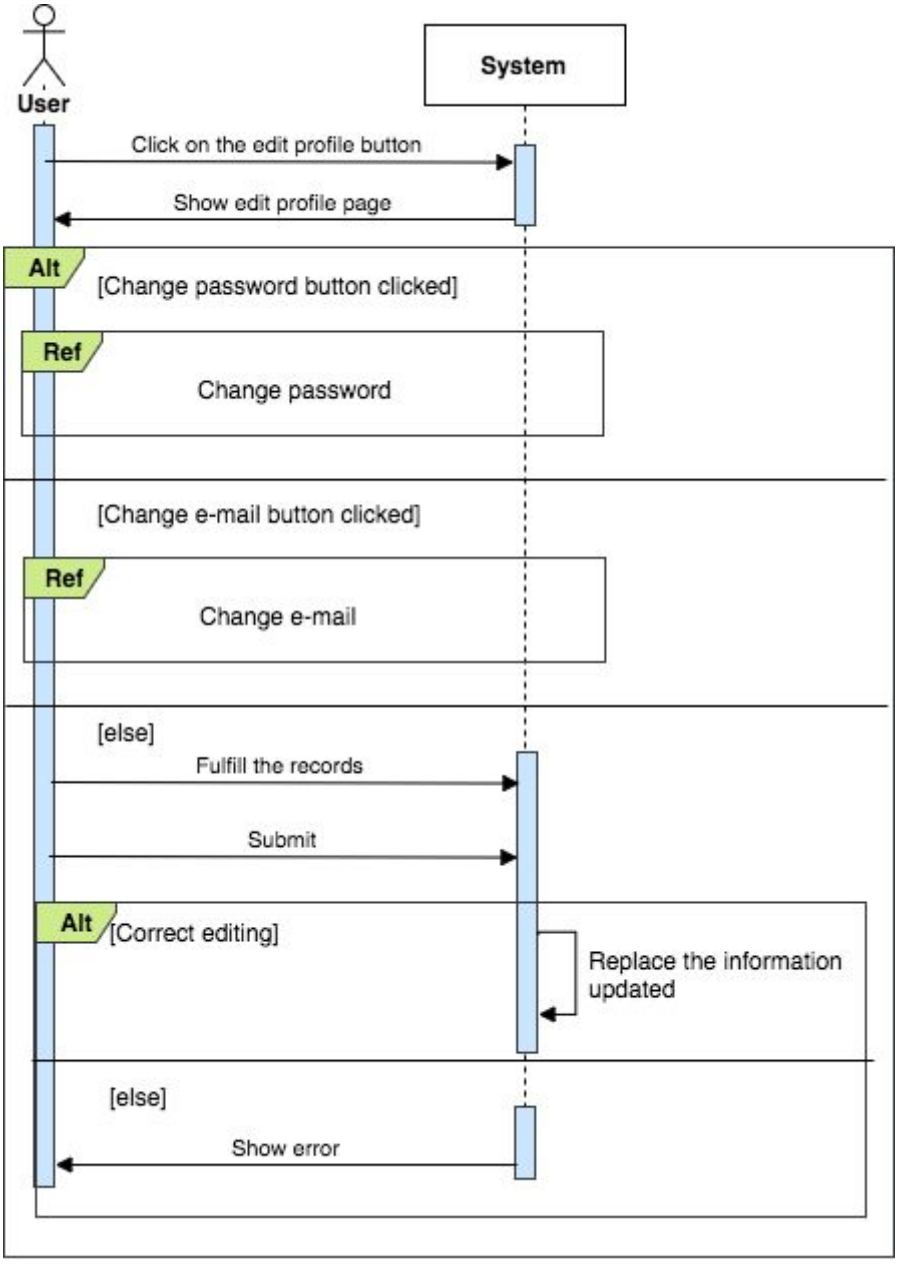
[SD2]: Log In



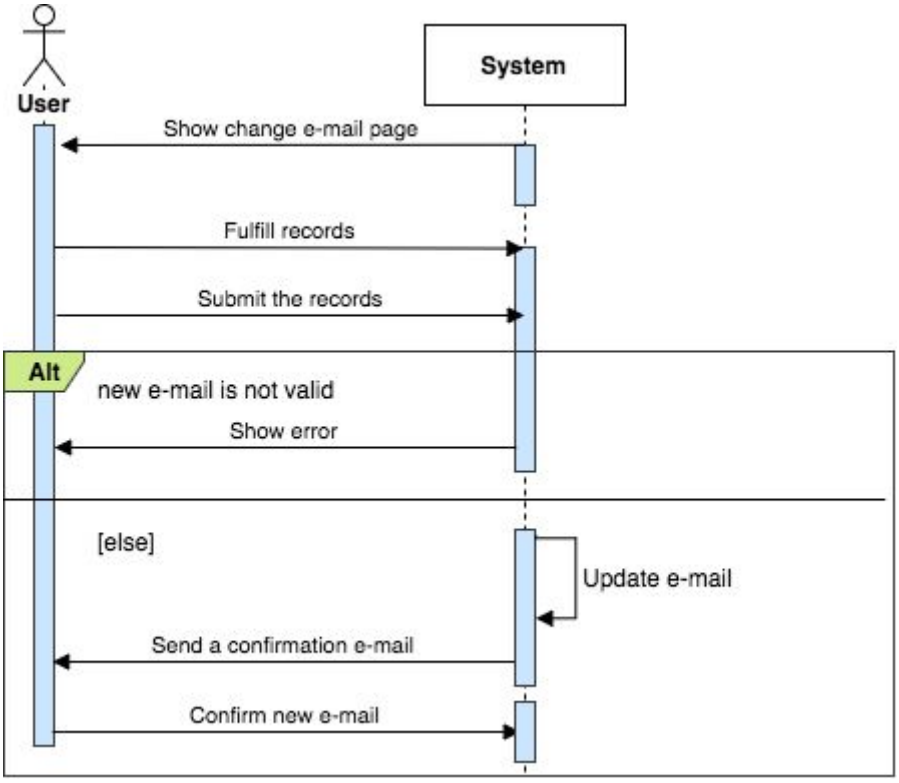
[SD3]: Password Recovery



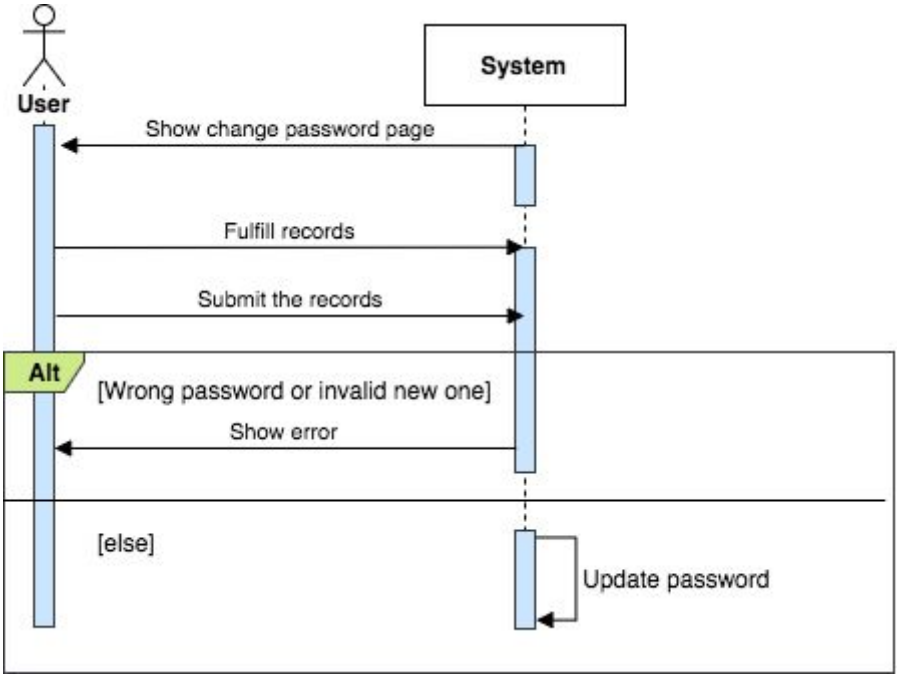
[SD4]: Edit Profile



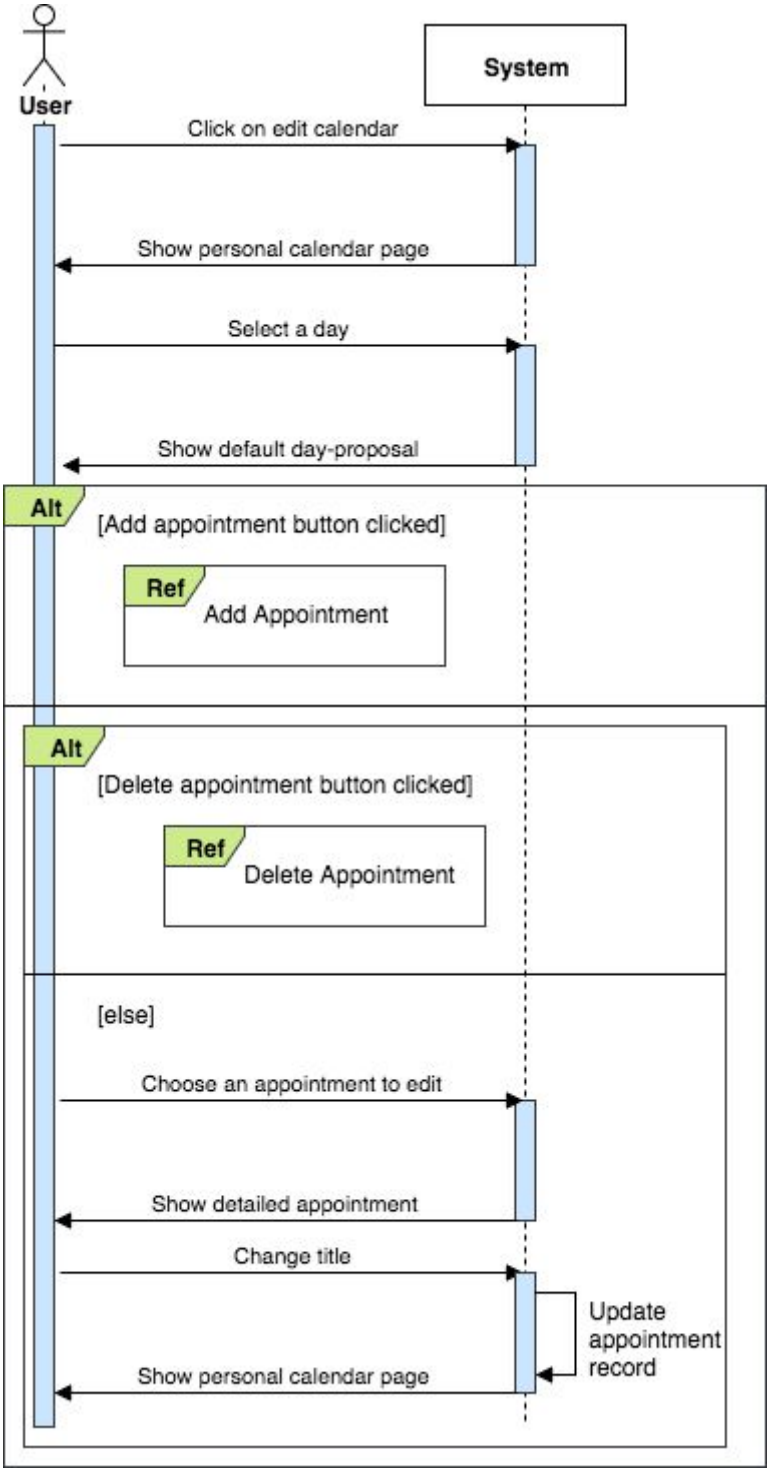
[SD5]: Change e-mail



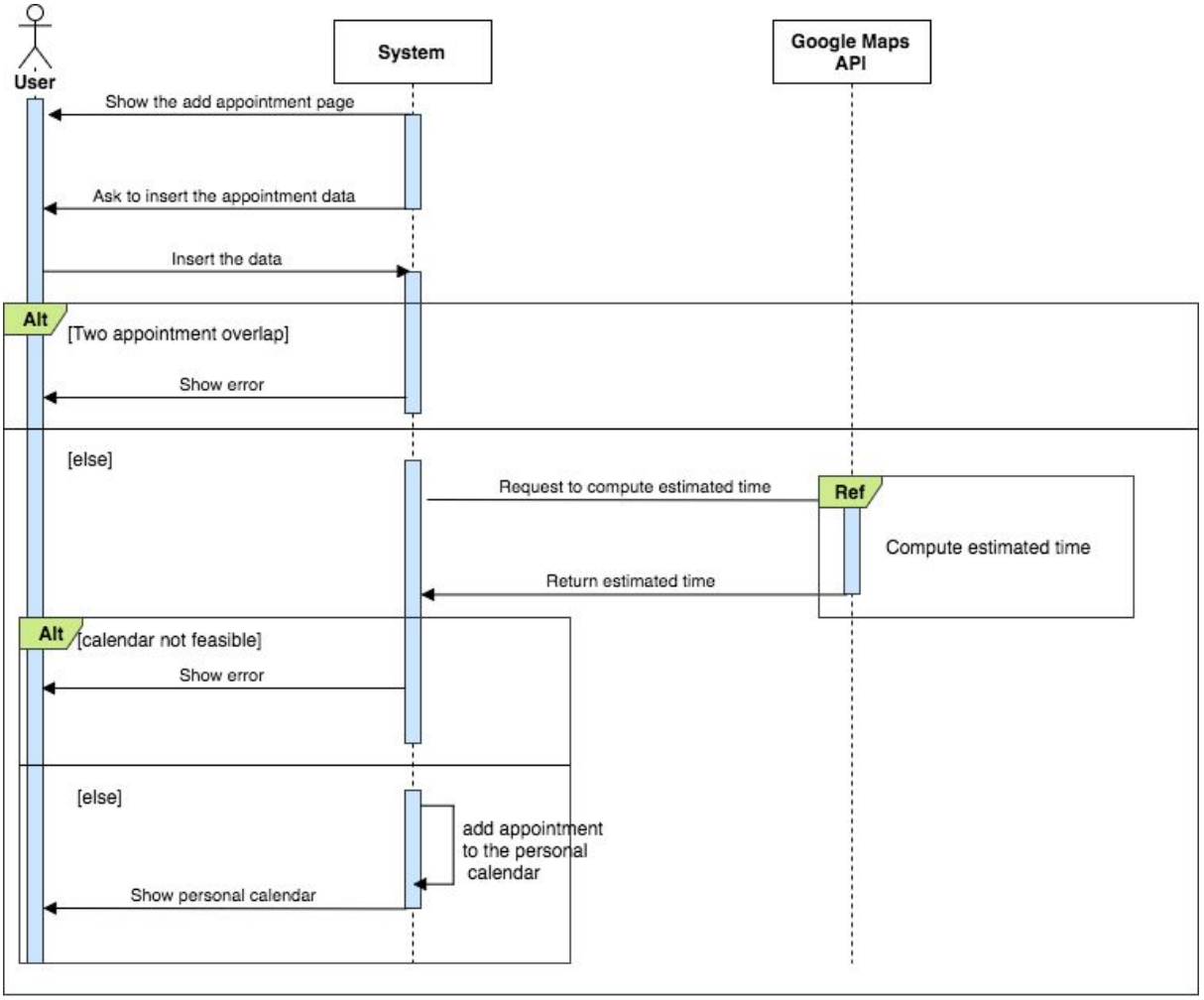
[SD6]: Change Password



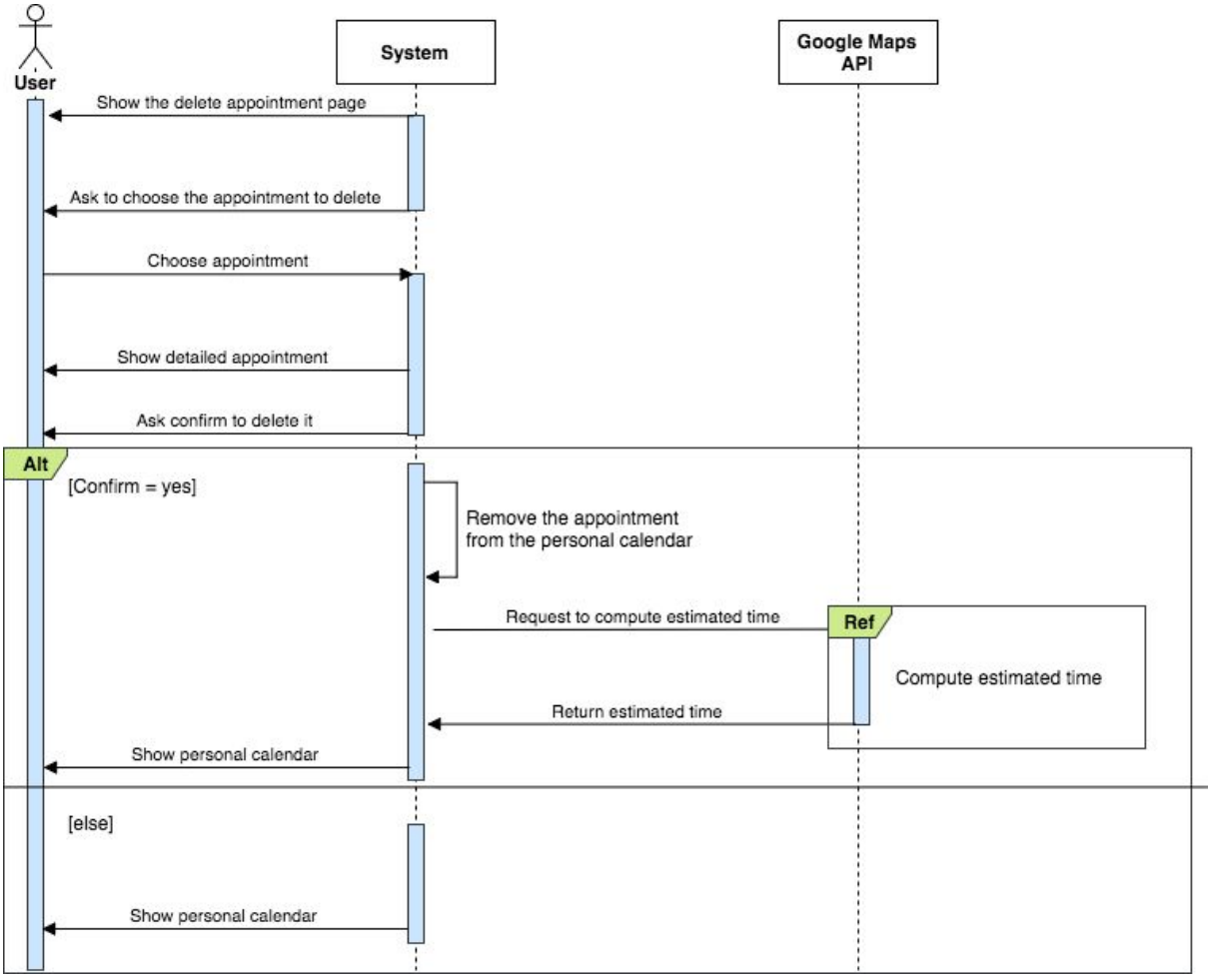
[SD7]: Edit Calendar



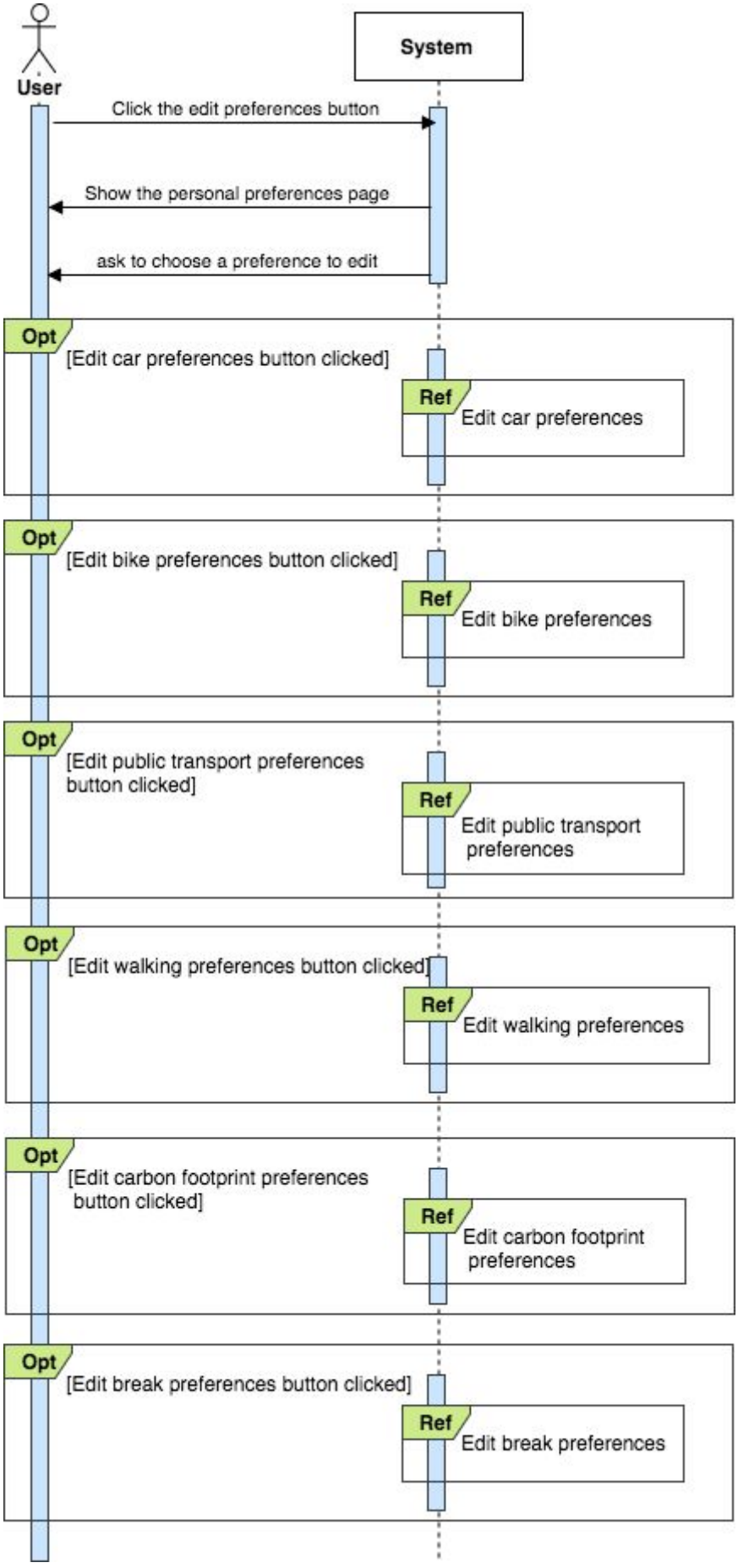
[SD8]: Add Appointment



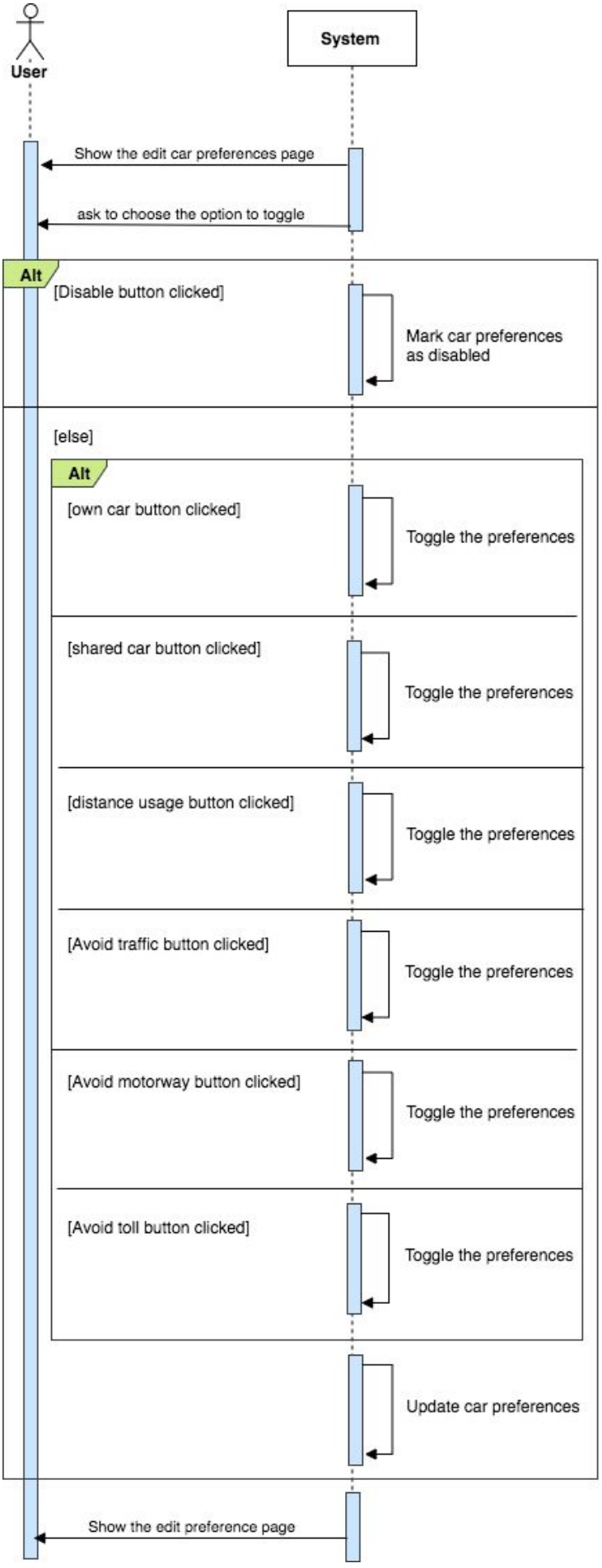
[SD9]: Delete Appointment



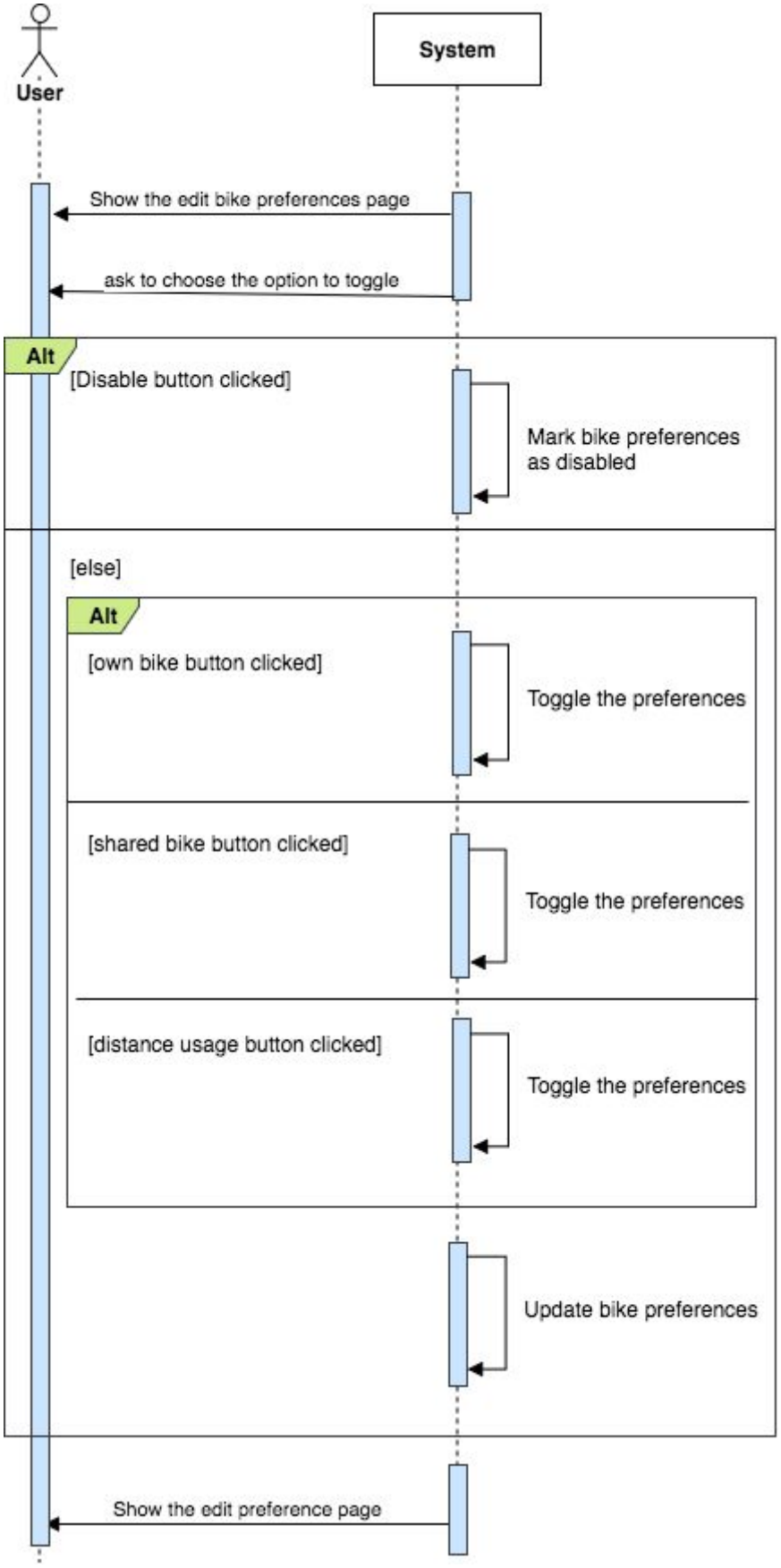
[SD10]: Edit Preferences



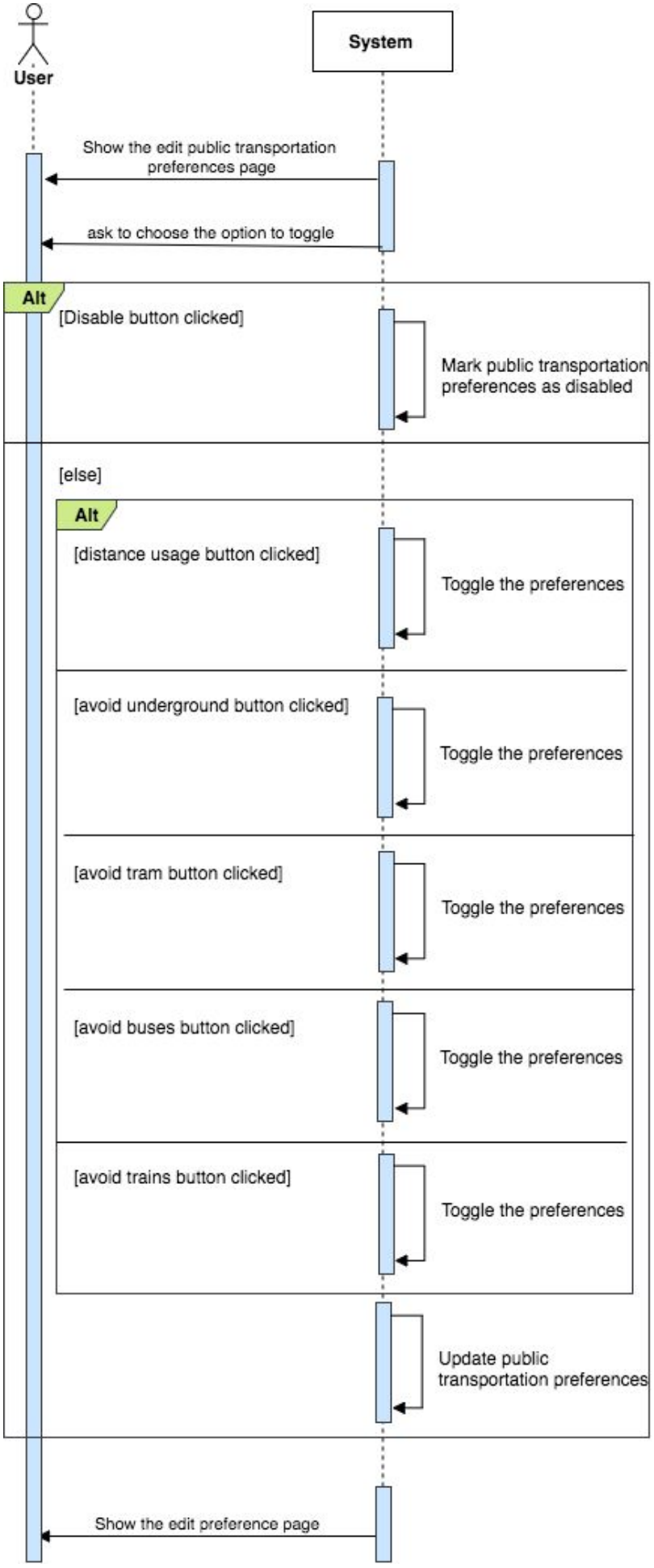
[SD11]: Edit Car Preferences



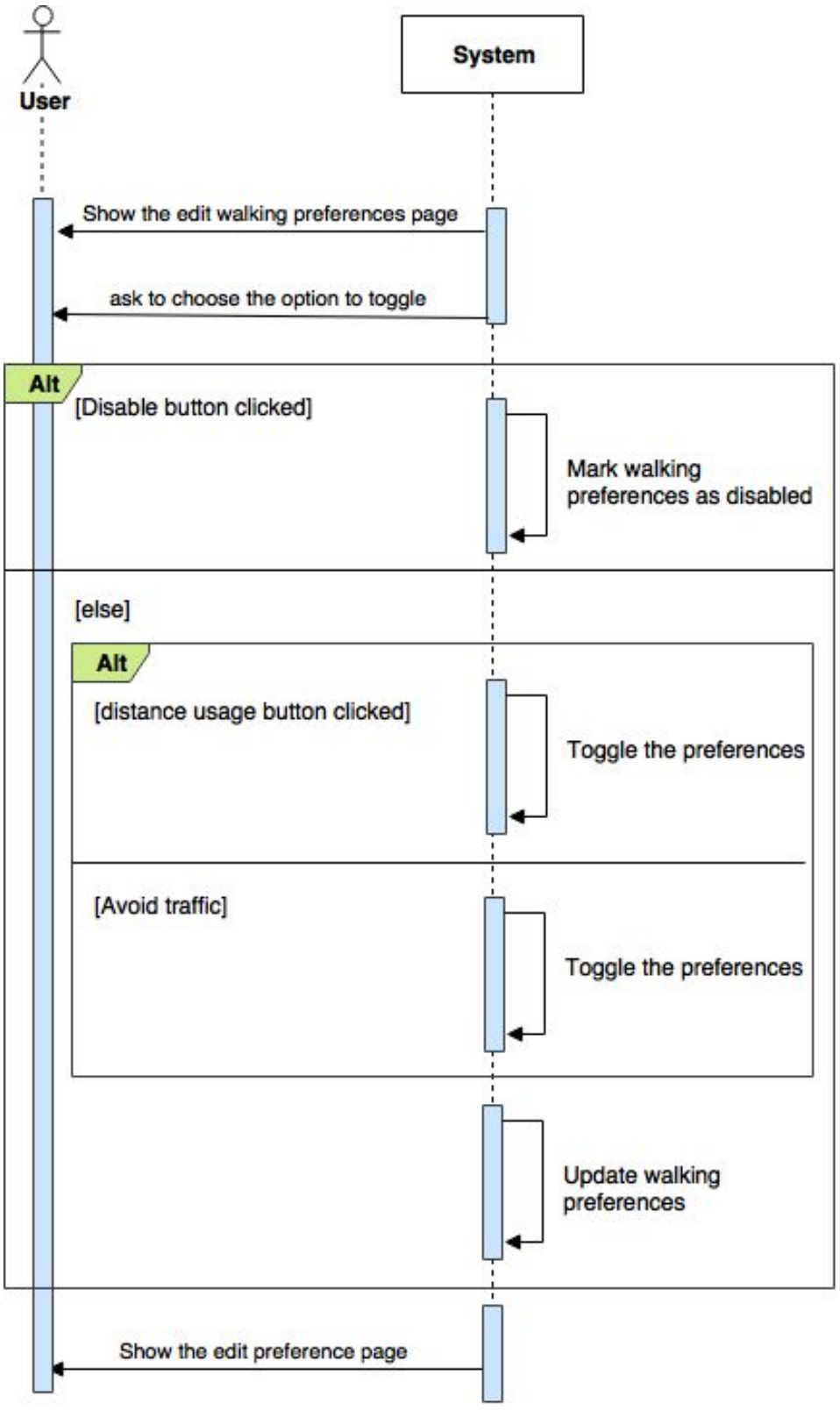
[SD12]: Edit Bike Preferences



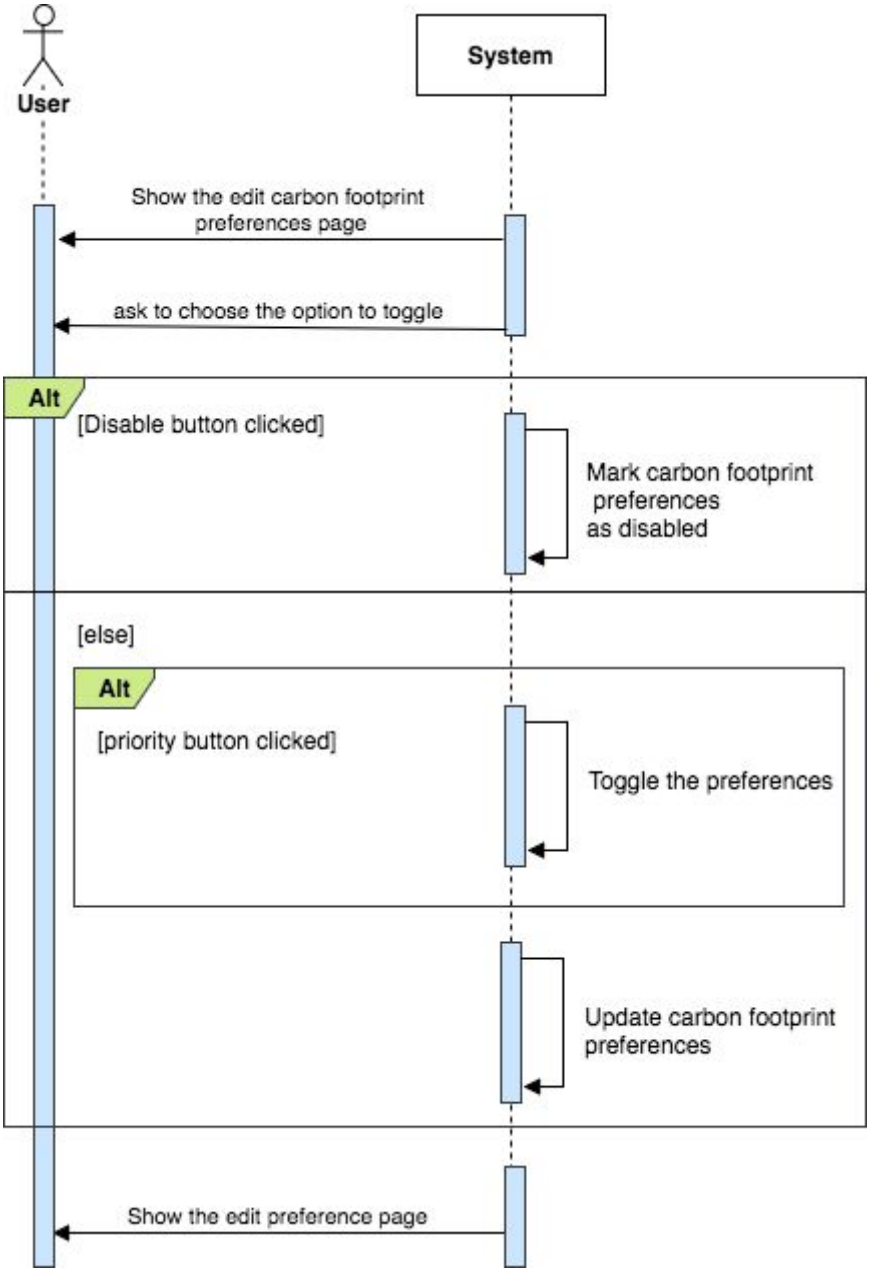
[SD13]:Edit Public Transportation Preferences



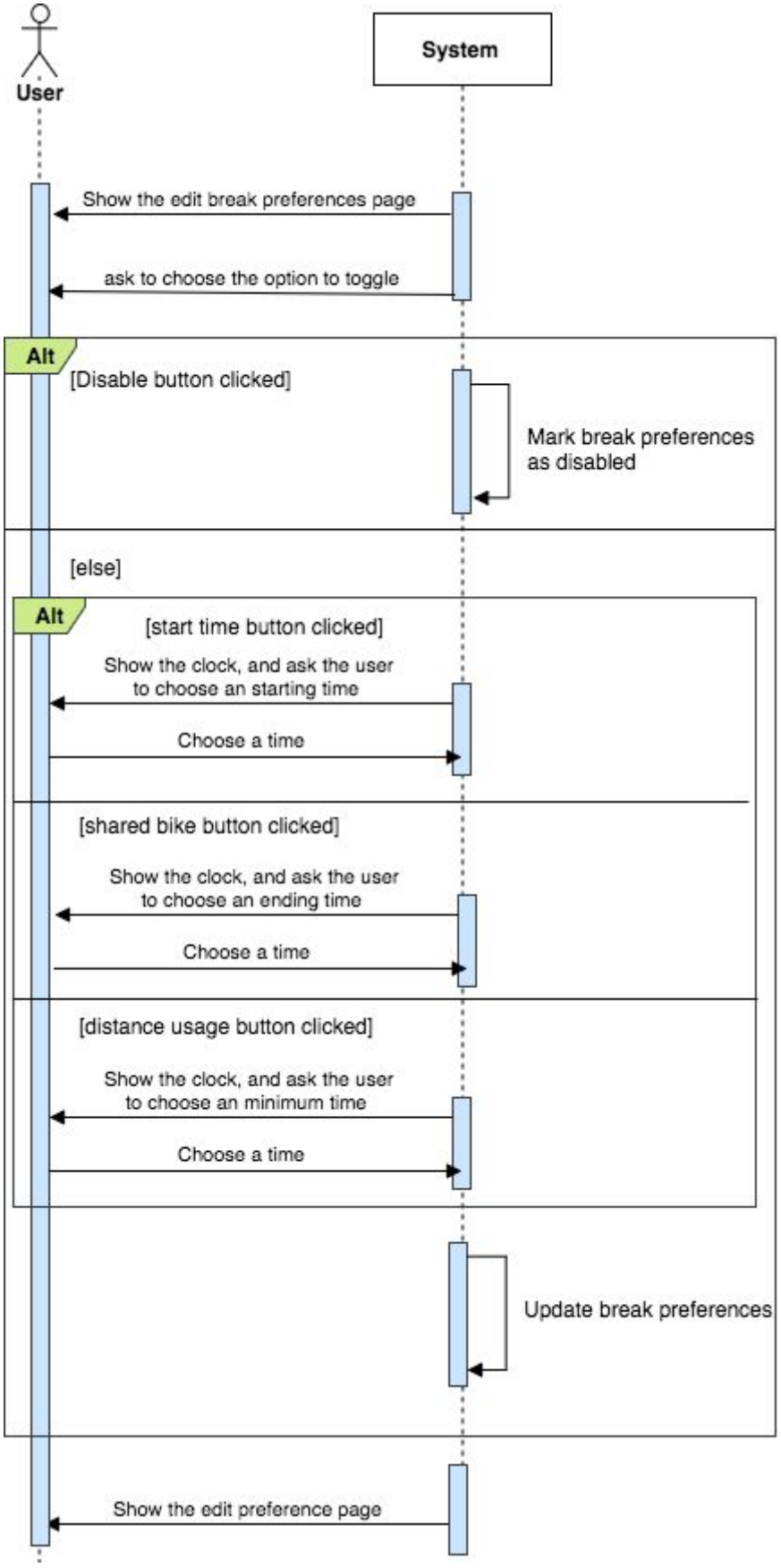
[SD14]: Edit Walking Preferences



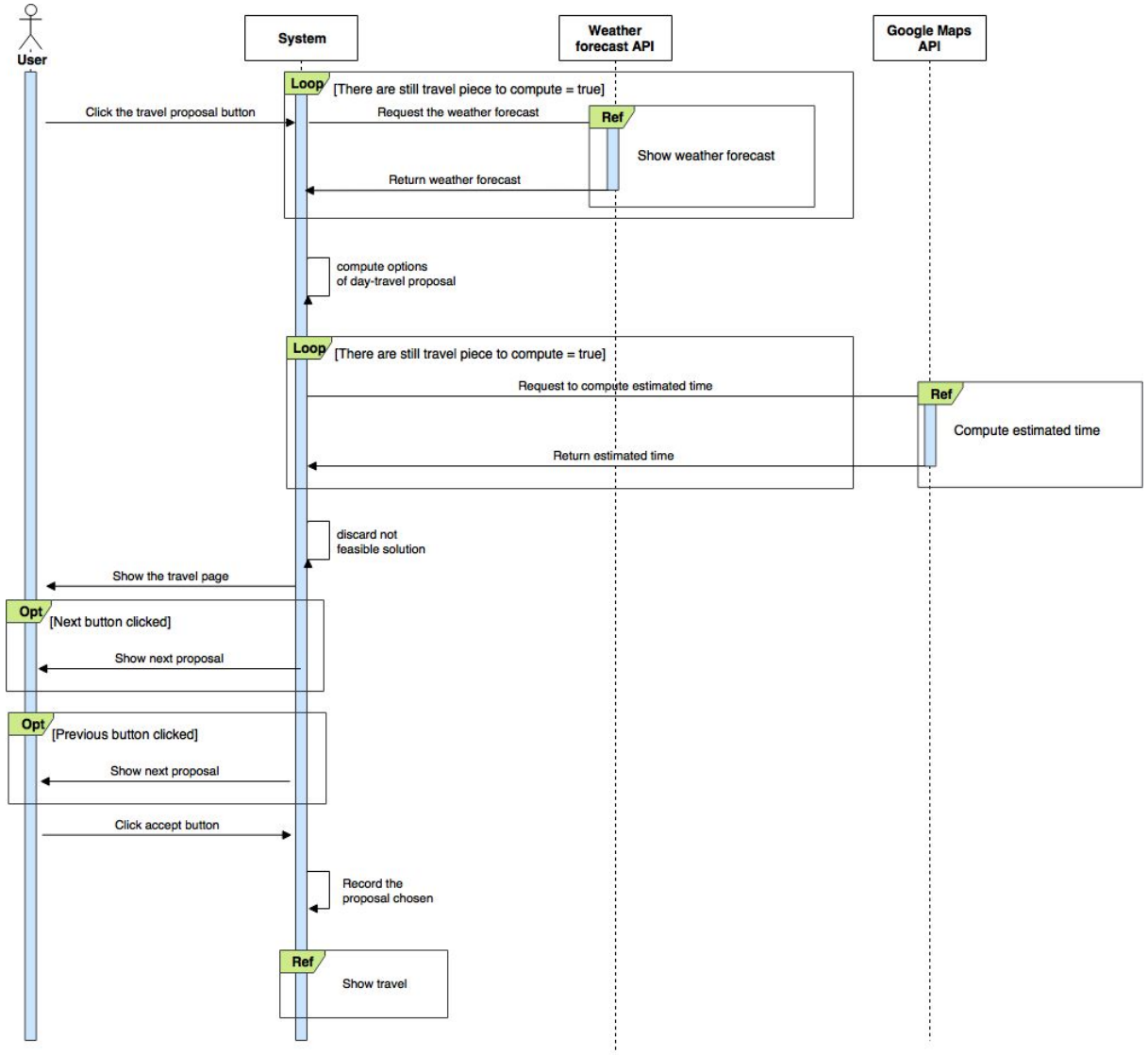
[SD15]: Edit Carbon Footprint Preferences



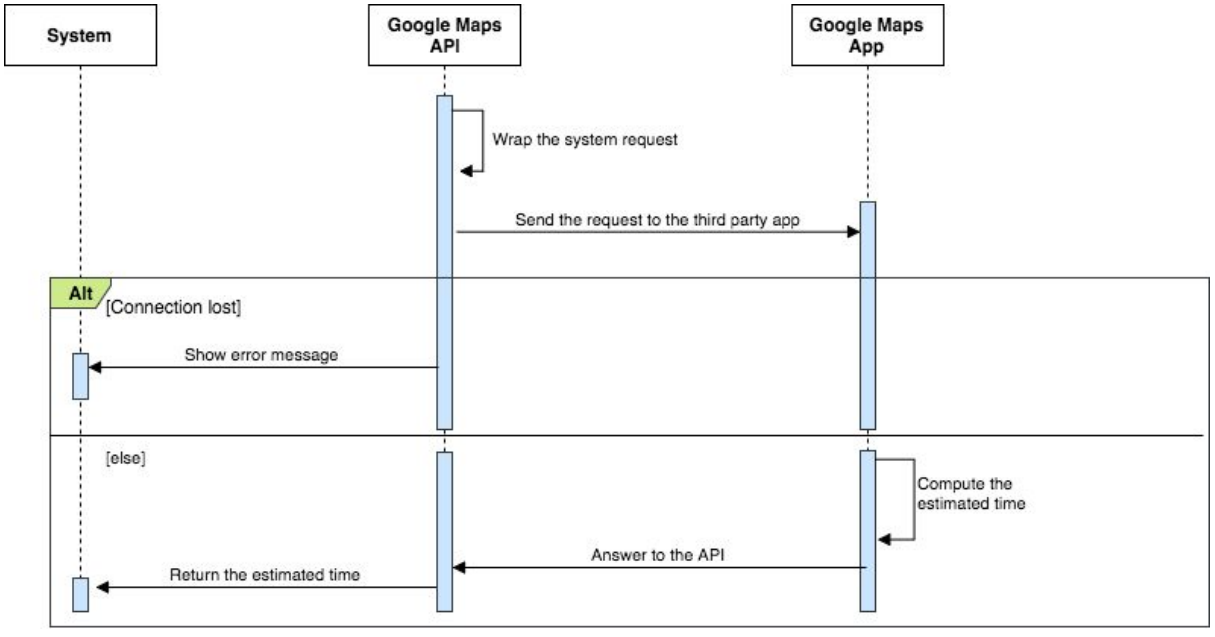
[SD16]: Edit Break Preferences



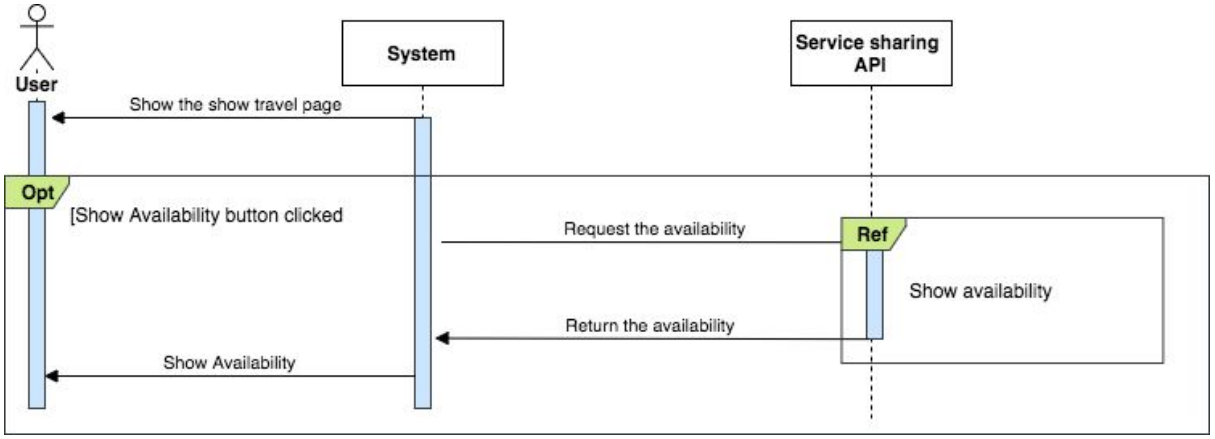
[SD17]: Evaluate Day-Travel Proposal



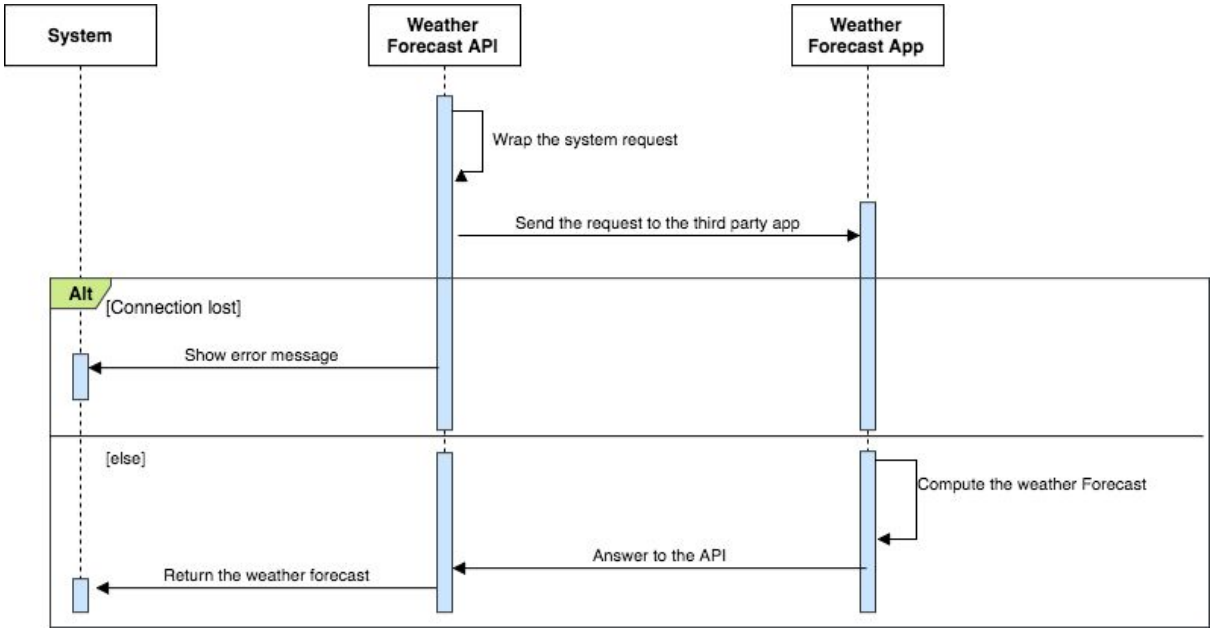
[SD18]: Compute Estimated Time



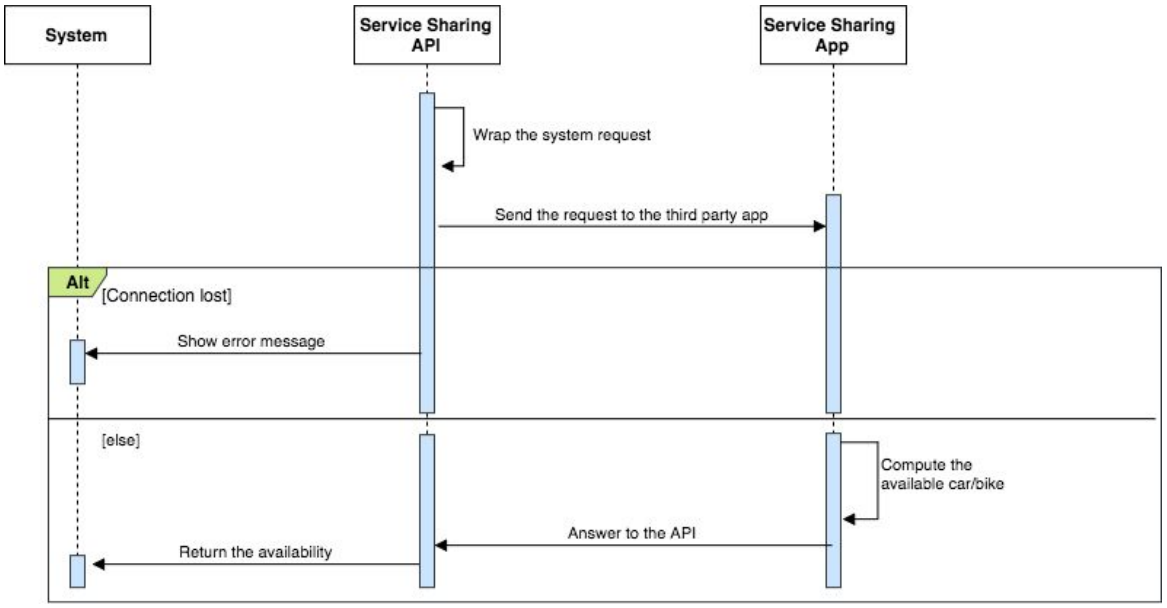
[SD19]: Show Travel



[SD20]: Show Weather Forecast



[SD21]: Show Availability



C. Performance Requirements

- **[NFR1]** The web application must run at least on the following browsers: Chrome, Safari, Android Web Browser, Opera, Firefox, Internet Explorer.
- **[NFR2]** The mobile application must run at least on the following OS: iOS, Android.
- **[NFR3]** Travel proposal computation must be completed in less than 30 seconds.
- **[NFR4]** The code sent by the system during the password recovery procedure is valid for 2 hours starting from the moment in which the code is generated.
- **[NFR5]** The system should be available 99.9% of time over a year.

D. Design Constraints

D.1 Standards compliance

Software must adhere to any common practices and standards for the programming language that will be used to code the application.

D.2 Hardware limitations

Each user should operate through a device that has these characteristics:

- Network connection
- Geolocation system capable to give a precision of a few metres
- Enough free memory to run the application

E. Software System Attributes

E.1 Reliability

Reliability: at least one month of failure-free period in between two consecutive failures.
Fault Tolerance: the probability of a failure that makes the system unusable must be less than 0.001%.
Disaster Recovery: All Informations about users must be recoverable after natural disaster event with a probability of 99.999%.

E.2 Availability

Every user should be able to use the application in every moment of the day. Ideally the whole infrastructure must be able to serve registered user 24/7. The desired availability complies the “five nine” standard which many software architectures manage to achieve. It is 99.999% which means the system can only afford a 5.26 minutes down time over a year.

E.3 Security

The system security provides secure communication among users and serves and guarantees the protection of data from malicious agents. In order to guarantee security the system must be conformed to the ISO 27001

E.4 Maintainability

Data Structures and Code Complexities should kept as down as possible. Lines of code must be less than 15 thousands.
A versioning system must expose the whole Change History Documentation.
Each class file must be have a general explaination and the main methods should also be documented.

E.5 Portability

The software should work on every Android (6+) or iOS (10+) commercial device and should also be available for Windows (7+) and Mac OS (10+). Furthermore the system source code must be portable from a platform to another for at least the 70%.

4. FORMAL ANALYSIS USING ALLOY:

The model is intended to show correctness of the high level class diagram for a world made of a single logged in user and events scheduled on a single day.
Correctness of the evolving model is proved by mean of the dynamic analysis.
No useless attribute has been taken into account.

Below is presented the Alloy code that has been computed by the official Analyzer from MIT. Right after the code a few predicates have been ran and a resulting world example are shown alongside the proof of instance (medium verbosity).

```
open util/integer

//-------------------------------------------------
// Entities
//-------------------------------------------------

//Travlendar+ user
one sig User {
transportPref: some Preferences,
calendar: some Calendar
}

//User's personal calendar
sig Calendar {
days: some Day,
currentDay: one Day,
currentLocation: one Location,
currentProposals: set TravelProposal, //current day travel proposals - if any calculated
scheduledEvents: set Event,
deletedEvents: set Event //set of events that were previously in the scheduleded Events set
}

//Any day of a given year
some sig Day {
}{
all c : Calendar | this in c.days //all days belong to every calendar
}
```

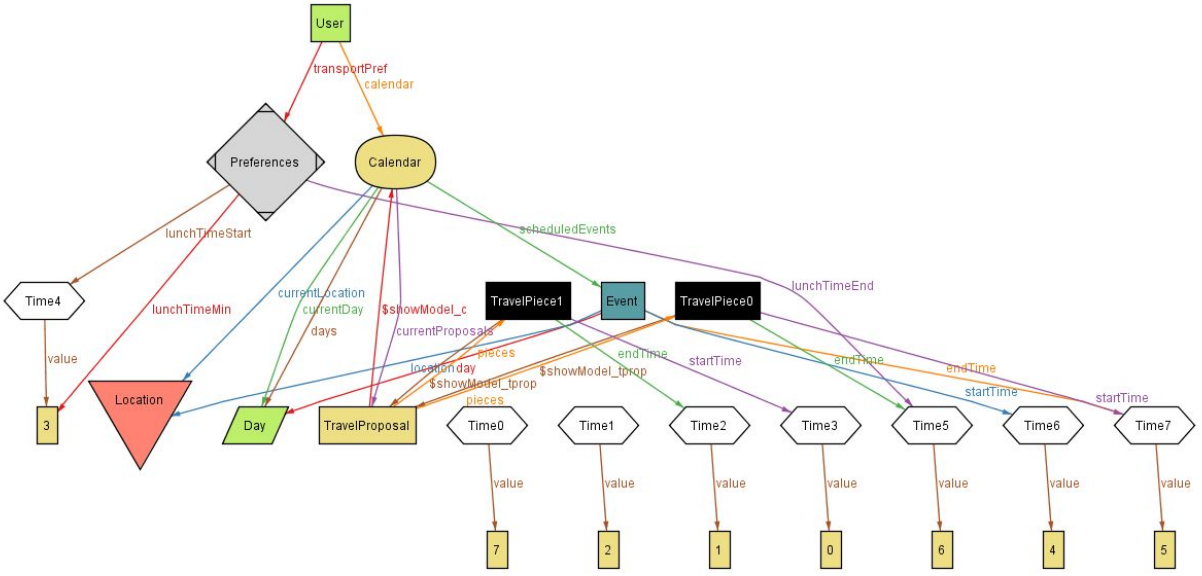
```
//User's event/appointment
some sig Event {
startTime: one Time,
endTime: one Time,
day: one Day,
location: one Location
}{
startTime != endTime
startTime.value < endTime.value
}

//Time data-type is used to simulate day times (i.e. 0 maps to 10:00am...7 maps to 10:00pm)
sig Time{
value: Int
}{
value>=0
}

//User's travel preferences
sig Preferences {
lunchTimeMin: Int, //minimum time to devote to lunch
lunchTimeStart: one Time, //lunch lower time boundry
lunchTimeEnd: one Time //lunch upper time boundry
}{
lunchTimeMin=1 //in terms of unit of Time entity
lunchTimeStart.value<lunchTimeEnd.value
lunchTimeEnd.value.sub[lunchTimeStart.value].sub[lunchTimeMin] >= 0 //must ensure a feasible lunch break
}

//Travel proposal entity containg the pieces of travel composing an entire day of movements
sig TravelProposal {
pieces: some TravelPiece
}{
some c : Calendar | this in c.currentProposals
}

//Single travel piece
sig TravelPiece {
startTime: one Time,
endTime: one Time
}{
startTime.value<endTime.value
}

//Generic Location
sig Location {}

//-------------------------------------------------------
// Sanity Constrains
//-------------------------------------------------------

//cannot have preferences not belonging to any user
fact preferencesAreBindToUsers{
#Preferences = #User
all disj u1,u2 : User | u1.transportPref != u2.transportPref
}

//cannot have calendars not belonging to any user
fact calendarsAreBindToUsers{
all c : Calendar, u : User | c in u.calendar
}

//ensures there is only one day per event
fact justOneDayPerEvent{
all disj u1,u2 : User | u1.calendar != u2.calendar
}

//cannot have timePiece not belonging to any TravelPorposal
fact travlPiecesBindToTravelProposal{
all tpiece : TravelPiece | some tprop : TravelProposal | tpiece in tprop.pieces
}
```

```
//model timeline ensuring that each Time instance has a different value
fact uniqueTimes{
all disj t1,t2 : Time | t1.value != t2.value
}

//call predicate to ensure minimum break times
fact ensureMinBreakTime{
lunchTimePreference
}

//ensures a single event doesn't go over the minum break time
fact eventsBreakPreference{
all e : Event, p : Preferences | (e.endTime.value >= p.lunchTimeStart.value =>
 e.endTime.value.sub[e.startTime.value].sub[p.lunchTimeMin] < 0)
and (e.startTime.value <= p.lunchTimeEnd.value => p.lunchTimeEnd.value.sub[e.startTime.value].sub[p.lunchTimeMin] < 0)
}

//no overlapping allowed between travel pieces and events
fact noTravelPieceOverlapsEvents{
all disj e : Event, t : TravelPiece | ascOrdered[e.endTime, t.startTime] or ascOrdered[t.endTime, e.startTime]
}

//cannot have events belonging to no calendars and cannot have overlapping events
fact eventsAreBindToCalendar{
all e : Event | one c : Calendar | e in c.scheduledEvents
and noOverlappingEvents
}

//------------------------------------------------
// Dynamic Analysis
//------------------------------------------------

//no overlapping events allowed
pred noOverlappingEvents{
all disj e1,e2 : Event | ascOrdered[e1.endTime, e2.startTime] or ascOrdered[e2.endTime, e1.startTime]
}

//ensures there is a minum lunch time between two different events
pred lunchTimePreference{
all e : Event, p : Preferences | ascOrdered[p.lunchTimeStart, e.startTime] =>
 e.startTime.value.sub[p.lunchTimeStart.value].sub[p.lunchTimeMin] >= 0
and ascOrdered[e.endTime, p.lunchTimeEnd] or ascOrdered[e.startTime, p.lunchTimeStart] =>
 p.lunchTimeEnd.value.sub[e.endTime.value].sub[p.lunchTimeMin] >= 0
all e : TravelPiece, p : Preferences | ascOrdered[p.lunchTimeStart, e.startTime] =>
 e.startTime.value.sub[p.lunchTimeStart.value].sub[p.lunchTimeMin] >= 0
and ascOrdered[e.endTime, p.lunchTimeEnd] or ascOrdered[e.startTime, p.lunchTimeStart] =>
 p.lunchTimeEnd.value.sub[e.endTime.value].sub[p.lunchTimeMin] >= 0
}

//TravelPieces of same proposal cannot overlap
pred noOverlappingTravelPieces{
all tp : TravelProposal |all disj t1,t2 : TravelPiece | (t1 in tp.pieces and t2 in tp.pieces) =>
ascOrdered[t1.endTime, t2.startTime] or ascOrdered[t2.endTime, t1.startTime]
}

//TravelPieces cannot go over an event
pred noTravelPieceOverEvents{
all t : TravelPiece, e : Event | ascOrdered[t.endTime, e.startTime] or ascOrdered[e.endTime, e.startTime]
}

//time ordering utility predicate
pred ascOrdered[t1,t2 : Time]{
t1.value <= t2.value
}
```

```
//a new Event is added by the user's calendar
pred addEvent[e : Event, c,c' : Calendar]{
e not in c.scheduledEvents and e not in c.deletedEvents
c'.scheduledEvents = (c.scheduledEvents + e)
c'.currentLocation = c.currentLocation
c'.currentProposals = c.currentProposals
noOverlappingEvents and noOverlappingTravelPieces and lunchTimePreference
}

//an existing Event is deleted from the user's calendar
pred deleteEvent[e : one Event, c,c' : one Calendar]{
e not in c.deletedEvents and e in c.scheduledEvents
c'.currentLocation = c.currentLocation
c'.currentProposals = c.currentProposals
c'.scheduledEvents = (c.scheduledEvents - e) and c'.deletedEvents = (c.deletedEvents + e)
noOverlappingEvents and noOverlappingTravelPieces and lunchTimePreference
}

//computation of a new Day - TravelProposal
pred computeTravelProposal[tp,tp' : TravelProposal, piece : TravelPiece, c,c' : one Calendar]{
tp not in c.currentProposals and tp' in c'.currentProposals
piece in tp'.pieces and noTravelPieceOverEvents and noOverlappingTravelPieces and lunchTimePreference
}

//simply show the Travlendar+ world model
pred showModel {
}
```

- **run showModel**:

Note: the lunchtime user's preference is respected by the model in terms of minimum break time and day time boundaries. Time has been modelled by using a custom entity which relies on Integer type. Hours of the day pretend to be mapped to those integers giving a sortable and easy quantifiable timeline.

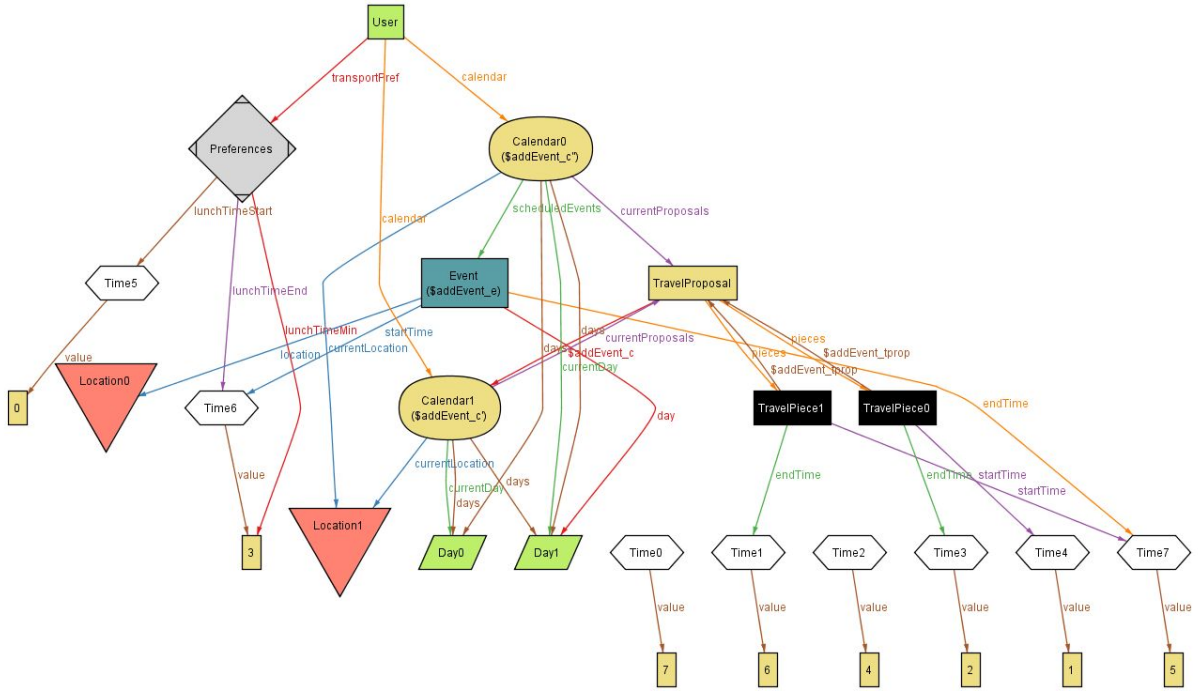


Executing "Run showModel for 3 but exactly 8 Time, 1 User"

```
Sig this/Time scope <= 8
Sig this/User scope <= 1
Sig this/Calendar scope <= 3
Sig this/Day scope <= 3
Sig this/Event scope <= 3
Sig this/Preferences scope <= 3
Sig this/TravelProposal scope <= 3
Sig this/TravelPiece scope <= 3
Sig this/Location scope <= 3
Sig this/User == [[User$0]]
Sig this/Calendar in [[Calendar$0], [Calendar$1], [Calendar$2]]
Sig this/Day in [[Day$0], [Day$1], [Day$2]]
Sig this/Event in [[Event$0], [Event$1], [Event$2]]
Sig this/Time == [[Time$0], [Time$1], [Time$2], [Time$3], [Time$4], [Time$5], [Time$6], [Time$7]]
Sig this/Preferences in [[Preferences$0], [Preferences$1], [Preferences$2]]
Sig this/TravelProposal in [[TravelProposal$0], [TravelProposal$1], [TravelProposal$2]]
Sig this/TravelPiece in [[TravelPiece$0], [TravelPiece$1], [TravelPiece$2]]
Sig this/Location in [[Location$0], [Location$1], [Location$2]]
Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
15066 vars. 446 primary vars. 38617 clauses. 134ms.
Instance found. Predicate is consistent. 225ms.
```

- **run addEvent**:

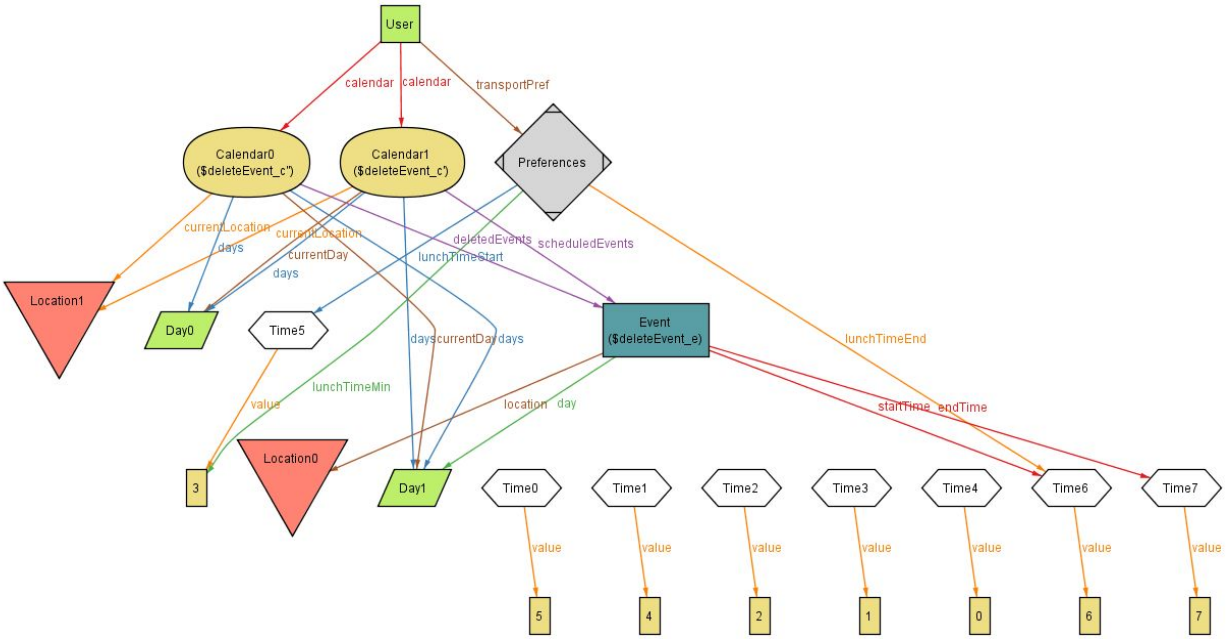
Note: the two Calendars before and after the Event has been added by the user.




```
Executing "Run addEvent for 3 but exactly 8 Time, 1 User"
  Sig this/Time scope <= 8
  Sig this/User scope <= 1
  Sig this/Calendar scope <= 3
  Sig this/Day scope <= 3
  Sig this/Event scope <= 3
  Sig this/Preferences scope <= 3
  Sig this/TravelProposal scope <= 3
  Sig this/TravelPiece scope <= 3
  Sig this/Location scope <= 3
  Sig this/User == [[User$0]]
  Sig this/Calendar in [[Calendar$0], [Calendar$1], [Calendar$2]]
  Sig this/Day in [[Day$0], [Day$1], [Day$2]]
  Sig this/Event in [[Event$0], [Event$1], [Event$2]]
  Sig this/Time == [[Time$0], [Time$1], [Time$2], [Time$3], [Time$4], [Time$5], [Time$6], [Time$7]]
  Sig this/Preferences in [[Preferences$0], [Preferences$1], [Preferences$2]]
  Sig this/TravelProposal in [[TravelProposal$0], [TravelProposal$1], [TravelProposal$2]]
  Sig this/TravelPiece in [[TravelPiece$0], [TravelPiece$1], [TravelPiece$2]]
  Sig this/Location in [[Location$0], [Location$1], [Location$2]]
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  15458 vars. 455 primary vars. 39234 clauses. 193ms.
  Instance found. Predicate is consistent. 450ms.
```

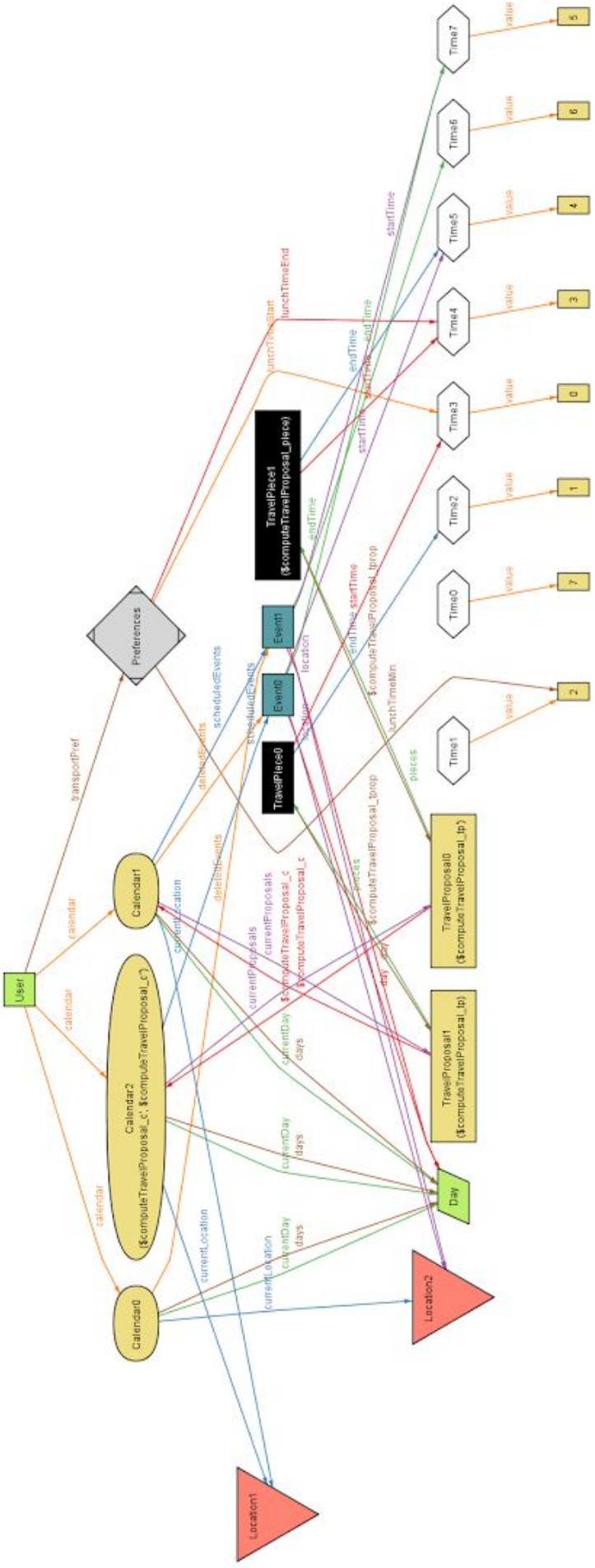
- **run deleteEvent**:

Note: the two Calendars before and after the Event has been deleted by the user.




```
Executing "Run deleteEvent for 3 but exactly 8 Time, 1 User"
  Sig this/Time scope <= 8
  Sig this/User scope <= 1
  Sig this/Calendar scope <= 3
  Sig this/Day scope <= 3
  Sig this/Event scope <= 3
  Sig this/Preferences scope <= 3
  Sig this/TravelProposal scope <= 3
  Sig this/TravelPiece scope <= 3
  Sig this/Location scope <= 3
  Sig this/User == [[User$0]]
  Sig this/Calendar in [[Calendar$0], [Calendar$1], [Calendar$2]]
  Sig this/Day in [[Day$0], [Day$1], [Day$2]]
  Sig this/Event in [[Event$0], [Event$1], [Event$2]]
  Sig this/Time == [[Time$0], [Time$1], [Time$2], [Time$3], [Time$4], [Time$5], [Time$6], [Time$7]]
  Sig this/Preferences in [[Preferences$0], [Preferences$1], [Preferences$2]]
  Sig this/TravelProposal in [[TravelProposal$0], [TravelProposal$1], [TravelProposal$2]]
  Sig this/TravelPiece in [[TravelPiece$0], [TravelPiece$1], [TravelPiece$2]]
  Sig this/Location in [[Location$0], [Location$1], [Location$2]]
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  15483 vars. 455 primary vars. 39318 clauses. 94ms.
  Instance found. Predicate is consistent. 174ms.
```

- **run showTravelProposal**:



Note: a feasible day Travel Proposal made of some Travel Pieces is proposed to the user. It respects the user preferences (break preferences visible) and it travels the user to the scheduled events.

```
Executing "Run computeTravelProposal for 3 but exactly 8 Time, 1 User, 2 TravelProposal"
  Sig this/Time scope <= 8
  Sig this/User scope <= 1
  Sig this/TravelProposal scope <= 2
  Sig this/Calendar scope <= 3
  Sig this/Day scope <= 3
  Sig this/Event scope <= 3
  Sig this/Preferences scope <= 3
  Sig this/TravelPiece scope <= 3
  Sig this/Location scope <= 3
  Sig this/User == [[User$0]]
  Sig this/Calendar in [[Calendar$0], [Calendar$1], [Calendar$2]]
  Sig this/Day in [[Day$0], [Day$1], [Day$2]]
  Sig this/Event in [[Event$0], [Event$1], [Event$2]]
  Sig this/Time == [[Time$0], [Time$1], [Time$2], [Time$3], [Time$4], [Time$5], [Time$6], [Time$7]]
  Sig this/Preferences in [[Preferences$0], [Preferences$1], [Preferences$2]]
  Sig this/TravelProposal in [[TravelProposal$0], [TravelProposal$1]]
  Sig this/TravelPiece in [[TravelPiece$0], [TravelPiece$1], [TravelPiece$2]]
  Sig this/Location in [[Location$0], [Location$1], [Location$2]]
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  15087 vars. 446 primary vars. 38612 clauses. 63ms.
  Instance found. Predicate is consistent. 281ms.
```

5. EFFORT SPENT:

The effort spent by the team is of 33 hours for each group member.

6. REFERENCES

A. Bibliography

- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, which can be retrieved on the beep page of the course.

B. Used Tools

- MIT Alloy Analyzer
- Balsamiq Mockups 3
- Draw.io
- Google Docs