

Software Engineering 2

Travlendar RASD

Requirement Analysis and Specification Document

Version 1.0 - 15/10/2017

Plamen Pasliev – 898793

Victor Álvaro Gonzalez -

Table of contents

[1. Introduction 4](#_Toc495847700)

[1.1 Objectives 4](#_Toc495847701)

[1.2 Scope 4](#_Toc495847702)

[1.2.1 Description of the given problem 4](#_Toc495847703)

[1.2.2 Current System 4](#_Toc495847704)

[1.2.3 Goals 4](#_Toc495847705)

[1.3 Definitions, acronyms and abbreviations 5](#_Toc495847706)

[1.3.1 Definitions 5](#_Toc495847707)

[1.3.2 Acronyms 5](#_Toc495847708)

[1.3.3 Abbreviations 5](#_Toc495847709)

[1.4 Reference documents 6](#_Toc495847710)

[1.5 Document Structure 6](#_Toc495847711)

[2. Overall description 7](#_Toc495847712)

[2.1 Product perspective 7](#_Toc495847713)

[2.2 Product functions 7](#_Toc495847714)

[2.3 User characteristics 7](#_Toc495847715)

[2.4 Constraints 7](#_Toc495847716)

[2.4.1. Regulatory policies 7](#_Toc495847717)

[2.4.2 Hardware constraints 8](#_Toc495847718)

[2.5 Assumptions and dependencies 8](#_Toc495847719)

[Specific requirements 9](#_Toc495847720)

[External interface requirements 9](#_Toc495847721)

[User Interfaces 9](#_Toc495847722)

[Hardware Interfaces 9](#_Toc495847723)

[Software Interfaces 9](#_Toc495847724)

[Communication Interfaces 9](#_Toc495847725)

[Functional requirements 9](#_Toc495847726)

[Performance requirements 9](#_Toc495847727)

[Design Constraints 9](#_Toc495847728)

[Standards compliance 9](#_Toc495847729)

[Hardware limitations 9](#_Toc495847730)

[Any other constraint 9](#_Toc495847731)

[Software System Attributes 9](#_Toc495847732)

[Reliability 9](#_Toc495847733)

[Availability 9](#_Toc495847734)

[Security 9](#_Toc495847735)

[Maintainability 9](#_Toc495847736)

[Portability 9](#_Toc495847737)

[Formal analysis using Alloy 10](#_Toc495847738)

[Effort spenT 11](#_Toc495847739)

[References 12](#_Toc495847740)

# 1. Introduction

## 1.1 Objectives

This document represents the Requirement Analysis and Specification Document (RASD). The main goals of this document are to analyze the users in order to model a system that meet their needs, describe the system in terms of functional and non- functional requirements, specify the constraints and the limits of the software and define the main typical use cases and user’s behaviors. This document is addressed to the developers who have to implement the requirements and could be used as a contractual basis.

## 1.2 Scope

### 1.2.1 Description of the given problem

In this project we are going to develop and implement an application called Travlendar. It is a calendar-based application which allows you to create a calendar according to the events you have such as meetings related to work or personal reasons. On top of calculating the time that the user has between appointments so that he does not arrive late, the application will suggest the best mobility option between events and will alert him when it is impossible to reach a specific event on time. In addition to these functionalities, the system will allow the user to buy public transportation tickets and locating the nearest point to hire another type of service (bike of a bike sharing system, car of a car-sharing service, etc.). Users could define their transportation preferences, they can activate or deactivate any kind of transportation (including walking). The application will also take into account the weather in the location of the user. If it is raining at the time the user has to move to another event, the system will take this into account and will change the way of transport if it is necessary. The application will also allow the user to define breaks to eat or to develop other types of activities. In this way the system will organize the appointments of the user according to their breaks and the time they need to do these activities. Finally, users should also be able, if they wish to, to select combinations of transportation means that minimize carbon footprint.

### 1.2.2 Current System

Even though there are already applications in the market in charge of the planning of meetings, none of these have the functionalities of Travlendar. These are limited to organize the events and notify the participants through the application and that these participants can be put in contact with other users. By this we can say that there is no application in the market with the qualities of Travlendar and therefore it has no competitors until now.

### 1.2.3 Goals

* [G1] Allow a User to manage appointments.
* [G2] Allow a User to specify their own travel preferences.
* [G3] Allow a User to introduce the breaks that he requires during the day and the temporal range in which he wants to do the rest.
* [G4] User should receive alerts.
* [G5] The user should receive directions from his current location to the location of the event.
* [G6] The system must consider days in which the public transport or the transport chosen by the user is not available or delayed.
* [G7] The system must consider the weather when an event takes place.

## 1.3 Definitions, acronyms and abbreviations

### 1.3.1 Definitions

* *Meeting:* appointment of any kind whether related to work or personal.
* *User*: a user of the Travlendar system
* *Calendar:* a timetable containing meetings sorted by date

### 1.3.2 Acronyms

* *RASD*: Requirement Analysis and Specification Document.
* *API*: Application Programming Interface

### 1.3.3 Abbreviations

* [Gn]: n-goal.
* [Dn]: n-domain assumption.
* [Rn]: n-functional requirement.

## 1.4 Reference documents

## 1.5 Document Structure

1. In the first part of the document the objectives as well as the main goals of the project are defined. In the same way is explained, without going into much detail, how the application works. Finally, is given some information about definitions and abbreviations to better understand the rest of the document.
2. In the second part it is given an overall description of the system including the functions of the application, clarifying some concepts of the system. Also list the actors who are going to take part of the system. In the same way the constraints and limitations of the system will be defined. Finally, are specified text and domain assumptions to resolve certain types of doubts that may arise by reading the document.
3. The third part of the document refers to specific requirements. We have defined both functional and non-functional requirements. In this part of the document we will go into more detail in the aspects mentioned in section 2.

# 2. Overall description

## 2.1 Product perspective

In present times, our lives are full of events. Work, family and friend events fill our daily schedules and it is hard to keep up with everything. Everyone needs to remember where, at what time and with who they need to be at a certain time of the day. To solve this problem the mobile app Travlendar comes in place. It helps the user to manage his events, how to go reach them and how to pay for them.

## 2.2 Product functions

The full list of functional and non-functional requirements as well as constraints, assumptions and dependencies will be discussed in the section 3. The main requirements are that the user can add events to his agenda. Additionally, the app must provide optimal suggestions of the different ways the user can reach his destination including travelling time, minimizing carbon emissions, event overlaps, etc. The app should assist the user in paying for tickets, bike and car rent subscriptions.

## 2.3 User characteristics

The only user that is affiliated with the application is the smartphone user himself.

User: A person who has downloaded, installed and authorized the app to use the phone’s native functions such as geolocation, calendar, etc.

## 2.4 Constraints

The system shall adhere to certain requirements regarding reliability and performance. Those will be discussed in greater detail in section 3.

### 2.4.1. Regulatory policies

Travlendar does not use a database and customers do not need to deliver sensitive information. Travlendar simply provides the means customers use to manage their events locally on their device.

### 2.4.2 Hardware constraints

- Android 4.4+ or iOS 9.0+ smartphone

- Internet connection

- Geolocation

## 2.5 Assumptions and dependencies

For Travlendar to function in a correct manner, these assumptions and dependencies must be met:

[D1] The smartphone device has constant internet connectivity.

[D2] The smartphone device has geolocation turned on at all times.

[D3] The phone is synchronized correctly with the current time of the user's region via mobile link or manually.

[D4] If for any reason the user has not attended an appointment, the system will give indications to the next meeting from the current location of the user.

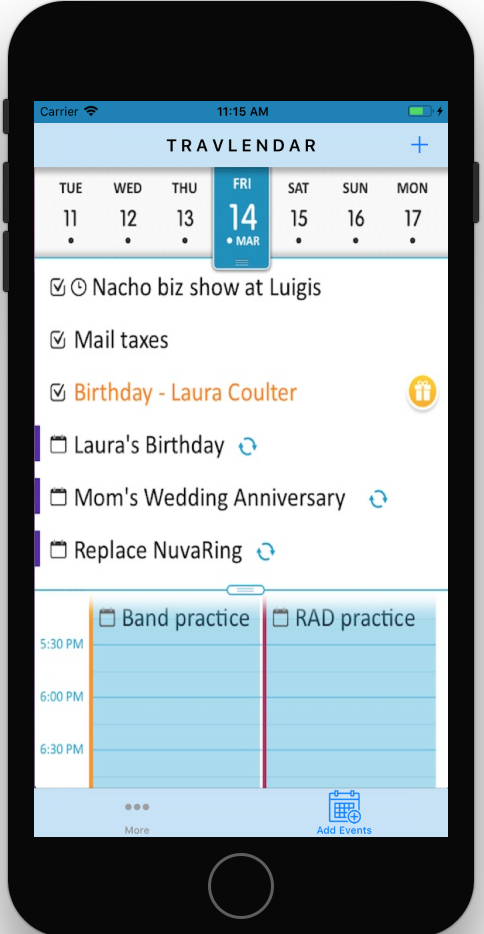
[D5] User correctly enters meeting address when created.

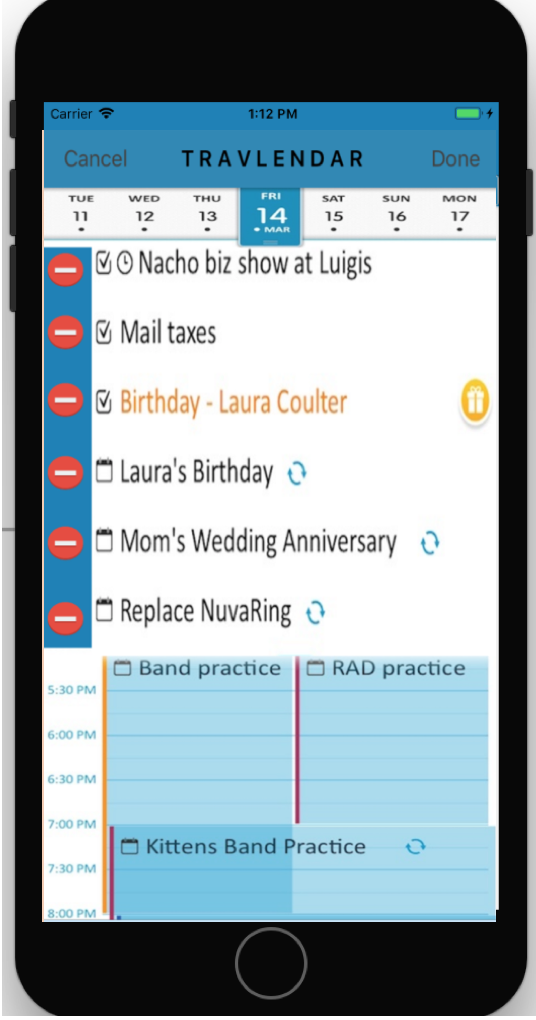
# Specific requirements

**3. Specific Requirements**

3.1 User Interfaces

This part of the document presents the application interface. The application is divided mainly in two views by a tap bar controller. In the main one the user activates or deactivates the means of transport they want. The second view presents the list with the events we have on the selected day. The user can change the day by browsing the calendar at the top of the view. Also in this view there is a button in the top right to add an appointment.





3.2 Functional Requirements

This section specifies the requirements of each goal previously named.

* [G1] Allow an User to create a calendar with meetings at one time in a specific location:

- [R1] In order to create a meeting the user should introduce a definition or a title to identify the appointment, a valid date (date later than or equal to current day), time it took to develop the activity and a valid event localization.

- [R2] A user must be able to add multiple appointments in one day.

- [R3] The system has to calculate the time between the previous meeting (temporarily) to the newly entered and the new one and verify if it is possible at that time to arrive punctual to the new event.

- [R4] If the system detects that it is impossible to arrive on time to the event, it should send an alert to the user.

- [R5] The system must be able to obtain different routes between the events in different means of transport in such a way that the user has several options to choose from.

* [G2] Allow an User to specify their own preferences according to the availability of that means of transport and to their pleasures:

- [R6] The system must allow the user to activate or deactivate a means of transport at any time.

- [R5] The system must be able to obtain different routes between the events in different means of transport in such a way that the user has several options to choose from.

* [G3] Allow an User introduce the breaks that he requires during the day and the temporal range in which he wants to do the rest:

- [R7] Users must be able to specify the number of rest they want in a day and the time they need to do the rest.

- [R8] The system should be able to calculate if it is possible to perform these breaks in the specified time range.

- [R4] If the system detects that it is impossible to arrive on time to the event, it should send an alert to the user.

* [G4] Users should receive alerts

- [R3] The system has to calculate the time between the previous meeting (temporarily) to the newly entered and the new one and verify if it is possible at that time to arrive punctual to the new event.

- [R4] If the system detects that it is impossible to arrive on time to the event, it should send an alert to the user.

- [R9] Users should receive an alert if, when entering the location of an event, the system notifies that there is no time to arrive on time.

-[R10] Users should receive an alert if the weather conditions are not pleasant or if there is some alteration in the route to their next appointment.

- [R11] The user must accept the request of the system to access its location when opening the application for the first time.

- [R12] The system should be able to check the weather at the location of the user / location of the events.

- [R13] The system must offer new routes between events to improve the quality of the user's journey.

* [G5] The user must receive directions from their current location to reach the meeting:

- [R11] The user must accept the request of the system to access its location when opening the application for the first time.

- [R14] Using an external application (Google maps, Apple maps, etc) the system should present the user the route to his next destination.

- [R15] The system has to be prepared to warn the user at what time he should leave his location to arrive on time.

* [G6] The system must take into account in addition to the bad weather, days of strike or in general days in which the public transport or the transport chosen by the user is reduced:

- [R9] The user must accept the request of the system to access its location when opening the application for the first time.

- [R14] The system has to be able to analyze the transport status in the location of the user and the events of the day.

- [R11] The system must offer new routes between events to improve the quality of the user's journey.

3.3 Software System Attributes

* Reliability:

The system must guarantee service 24/7. This will require updates to improve the services of the application and correct possible errors of the early versions.

* Availability:

The system must have a wide range of possibilities to offer the user. In such a way that the user can choose between all means of transport and the routes offered by the application.

* Maintainability:

It is one of the most important points that represents the majority of the expenses of the project. There must be continuous maintenance of the application to make it work in the best possible way.

* Accuracy:

The data that the application provides to the user must be as accurate as possible, so that the user can perform all the appointments in the estimated time. This way the GPS location is going to be one of the fundamental points.

* Security:

The security does not take a very important value since the data is stored in the application itself so it is very difficult to occur information theft to users.

## External interface requirements

### User Interfaces

### Hardware Interfaces

### Software Interfaces

### Communication Interfaces

## Functional requirements

## Performance requirements

## Design Constraints

### Standards compliance

### Hardware limitations

### Any other constraint

## Software System Attributes

### Reliability

### Availability

### Security

### Maintainability

### Portability

# Formal analysis using Alloy

User creating a new appointment

|  |  |
| --- | --- |
| **ACTORS** | User |
| **GOALS** | [G1] |
| **INPUT CONDITIONS** | There are no entry conditions. |
| **EVENTS FLOW** | * The user should press the “Calendar” bottom. * The user should press the “+” button. * The user should enter a title for the event. * The user should enter a start time for the event. * The user should enter a finishing time for the event. * The user should enter his location before the event (start). * The user should enter the location of the event (end). * The user should press the “Save” button. |
| **OUTPUT CONDITIONS** | The appointment is added to the calendar. |
| **EXCEPTIONS** | 1. The user inserts a non-existing location.  This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 6.  2. The user inserts an event that overlaps with another one.  This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 4. |

User editing an appointment

|  |  |
| --- | --- |
| **ACTORS** | User |
| **GOALS** | [G1] |
| **INPUT CONDITIONS** | The user must have created an event. |
| **EVENTS FLOW** | * The user should press the “Calendar” bottom. * The user should press the “Edit” bottom. * The user should press on the event he wishes to edit. * The user should edit the event. * The user should press the “Save” button. |
| **OUTPUT CONDITIONS** | The appointment is added to the calendar. |
| **EXCEPTIONS** | 1. The user inserts a non-existing location.  This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 4.  2. The user inserts an event that overlaps with another one.  This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 4. |

User deleting an appointment

|  |  |
| --- | --- |
| **ACTORS** | User |
| **GOALS** | [G1] |
| **INPUT CONDITIONS** | The user must have created an event. |
| **EVENTS FLOW** | * The user should press the “Calendar” bottom. * The user should press the “Edit” bottom. * The user should press on the red circle at the left of the event he wishes to delete. |
| **OUTPUT CONDITIONS** | The appointment is removed from the calendar. |
| **EXCEPTIONS** |  |

User creating a new break

|  |  |
| --- | --- |
| **ACTORS** | User |
| **GOALS** | [G3] |
| **INPUT CONDITIONS** | There are no entry conditions. |
| **EVENTS FLOW** | * The user should press the “Calendar”” bottom. * The user should press the “+” button. * The user should press the “New break” button * The user should enter a title of the break. * The user should enter the interval time where he/she wants to do the break. * The user should enter the duration of the break. * The user should press the “Save” button. |
| **OUTPUT CONDITIONS** | The break is added to the calendar. |
| **EXCEPTIONS** | 1. The user inserts a break that overlaps with another event.  This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 5. |

User editing a break

|  |  |
| --- | --- |
| **ACTORS** | User |
| **GOALS** | [G3] |
| **INPUT CONDITIONS** | The user should have created a break. |
| **EVENTS FLOW** | * The user should press the “Calendar”” bottom. * The user should press the “Edit” button. * The user should press on a break. * The user should edit the break. * The user should press the “Save” button. |
| **OUTPUT CONDITIONS** | The break is added to the calendar. |
| **EXCEPTIONS** | 1. The user inserts a break that overlaps with another event.  This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 4. |

User deleting a break

|  |  |
| --- | --- |
| **ACTORS** | User |
| **GOALS** | [G3] |
| **INPUT CONDITIONS** | The user should have created a break. |
| **EVENTS FLOW** | * The user should press the “Calendar”” bottom. * The user should pressed the “Edit” button. * The user should press on the red circle at the left of the event he wishes to delete. |
| **OUTPUT CONDITIONS** | The break is deleted from the calendar. |
| **EXCEPTIONS** |  |

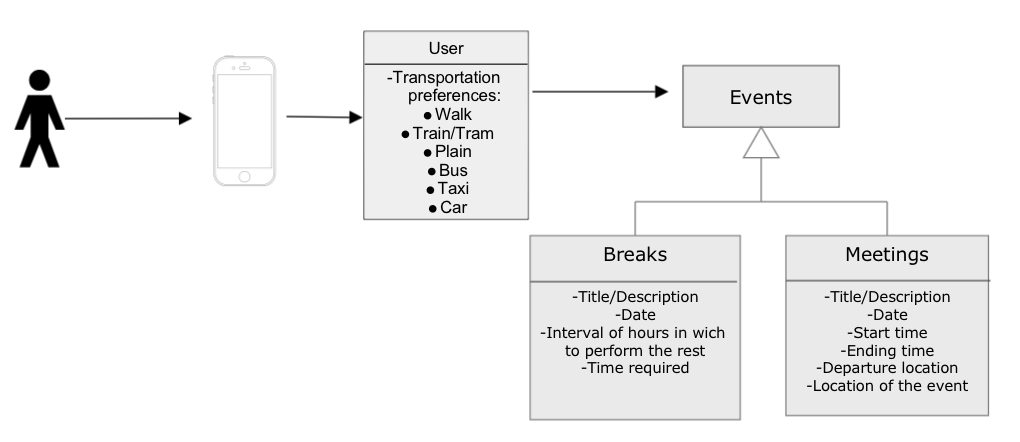
User specifies travel preferences

|  |  |
| --- | --- |
| **ACTORS** | User |
| **GOALS** | [G2] |
| **INPUT CONDITIONS** | There are no entry conditions. |
| **EVENTS FLOW** | * The user should press the “More”” bottom. * The user should choose his preferences by using the radio buttons mapped to the different means of travel. |
| **OUTPUT CONDITIONS** | The travel preferences have been chosen. |
| **EXCEPTIONS** | 1. The user deselects all travel preferences.  This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 2. |

User chooses route

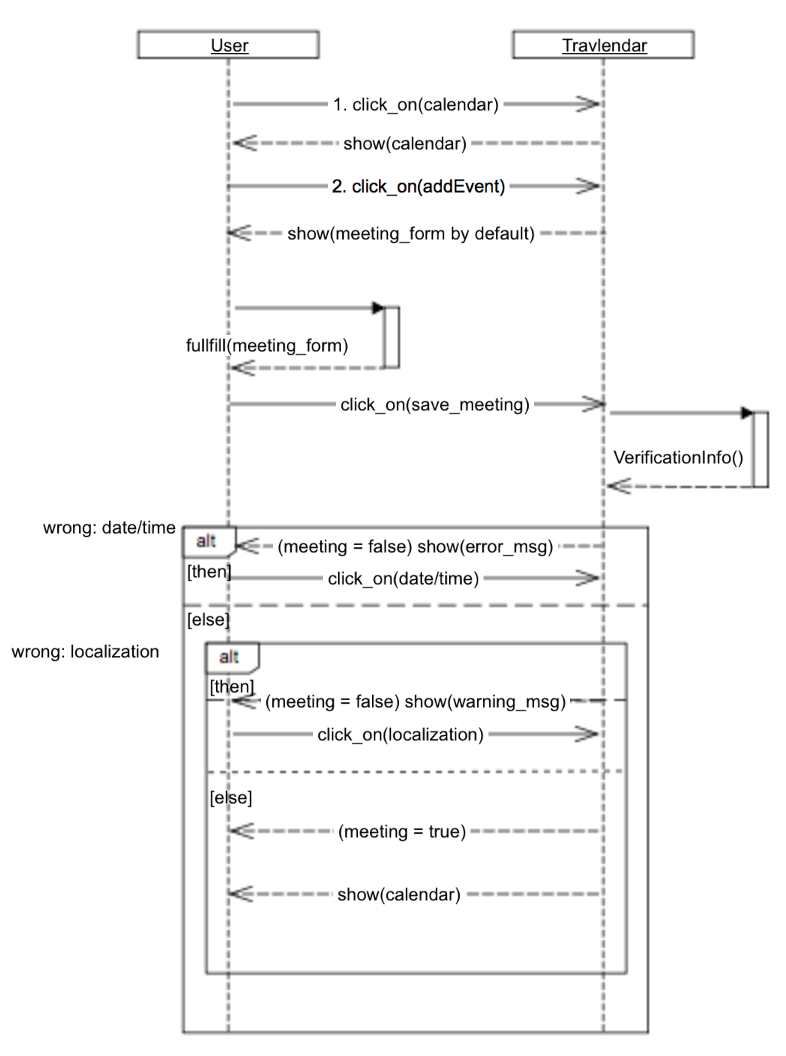
|  |  |
| --- | --- |
| **ACTORS** | User |
| **GOALS** | [G2] [G6] [G7] |
| **INPUT CONDITIONS** | The user should have created an appointment. |
| **EVENTS FLOW** | * The user should press the “Calendar”” bottom. * The user should press on an appointment. * The user should press the “Travel” button. * The user should choose one of the suggested routes * The system will display a map with a marker which represents the user while he reaches his destination. |
| **OUTPUT CONDITIONS** | The route has been chosen. |
| **EXCEPTIONS** | 1. There is no route available.  This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 3. |

Class diagram:

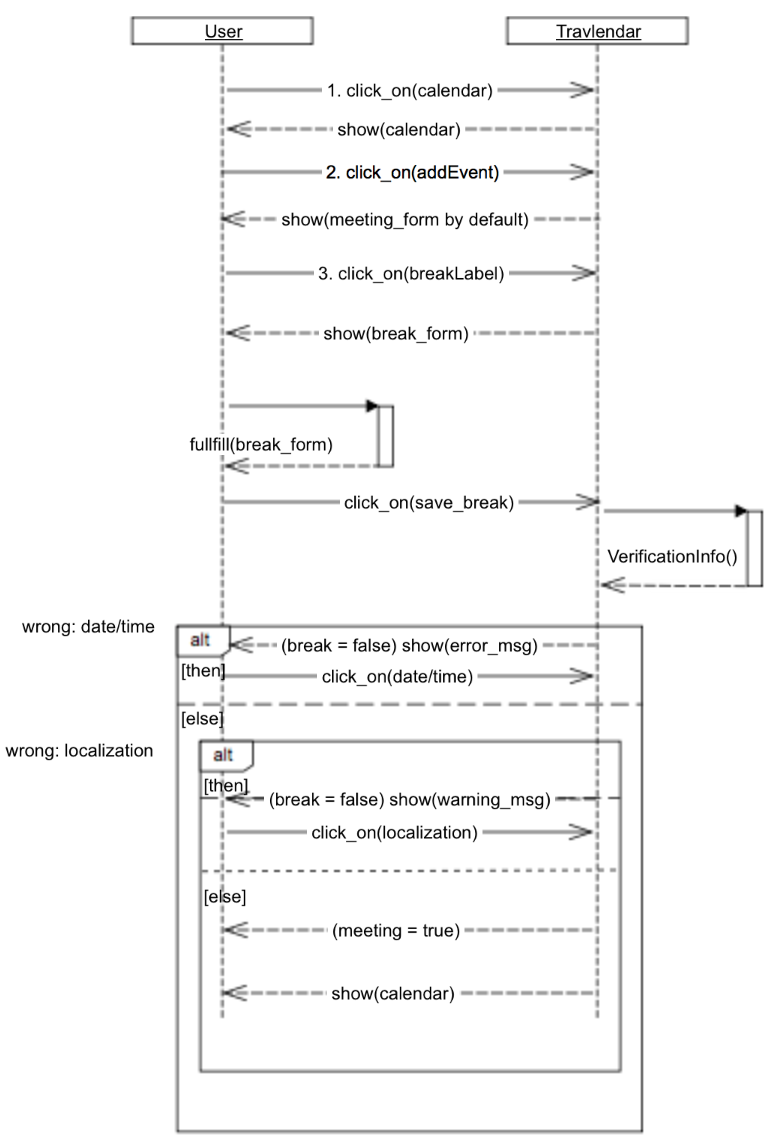


Sequence diagrams:

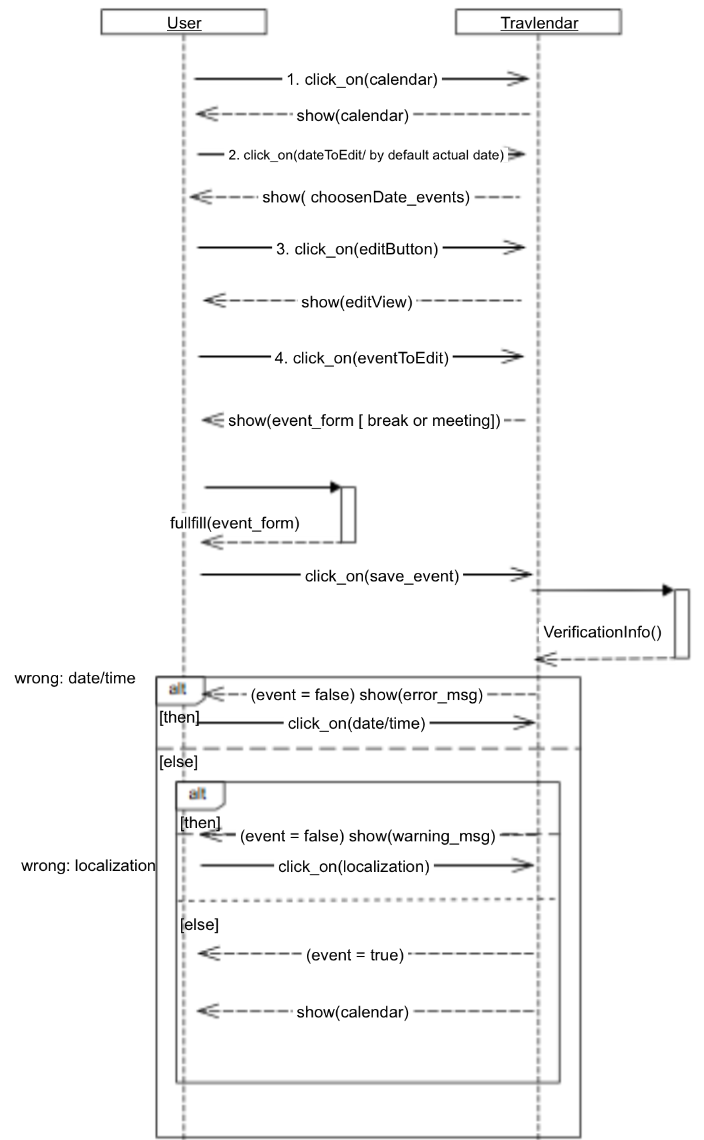
1. *Creating a meeting*



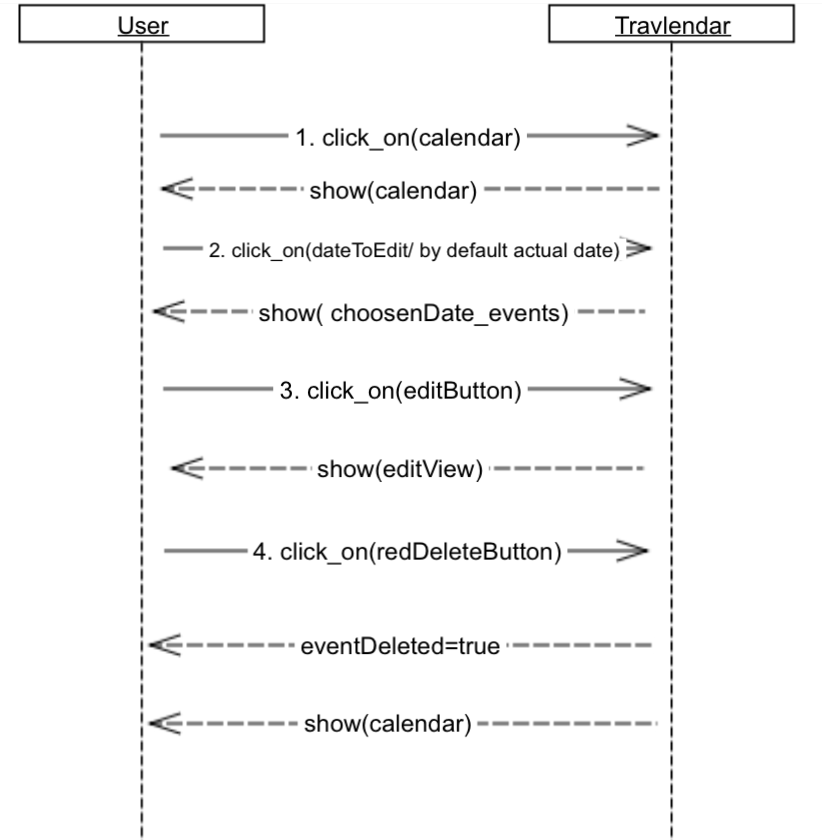
1. *Creating a break*

**

1. *Editing event*



1. *Deleting an event*

**

# Effort spenT

|  |  |  |
| --- | --- | --- |
| Name | Date | Hours spend |
| Plamen | 12/10/17 | 2 |
| Plamen | 15/10/17 | 5.50 |
| Plamen | 16/10/17 | 4 |
| Plamen | 17/10/17 | 1.5 |
|  |  |  |

# References