Politecnico di Milano

Software Engineering 2

Requirements Analysis and Specifications Document

Pietro Melzi, Alessandro Pina, Salvadore Matteo

AA 2017-2018

Contents

Chapter 1

INTRODUCTION

1.1 Purpose

This document provides a baseline for the project planning of Travlendar+, the system we want to develop. It analyses the environment in which the system is intended to be used and contains information about the offered functions.

Travlendar+ is a service based on mobile application and web application that helps people to organize daily appointments and travels between them. The application wants to be a useful tool that can be used both in work time and in free time. In Travlendar+ the user simply has to create an event; the application will organize the travel in order to reach the location in the best way and will add the new event within the daily schedule. The user can modify the schedule with any feasible combination of added events and can specify different types of preferences on travel means, so the system can plan the trip according to personal needs.

People currently use different services, provided by calendar and maps applications, in order to organize meetings and events. Travlendar+ includes all the helpful functionalities you need to plan the day in the same application.

1.2 Scope

TODO

1.2.1 Goals

Visitor should be able to:

[**G1**] sign up into the system;

User should be able to:

[**G2**] log into the system;

[**G3**] create event specifying location, date, starting and ending time;

[**G4**] obtain the best path according to his preferences and the list of eventual alternative paths to reach a location;

[**G5**] change the selected path with an alternative path;

[**G6**] obtain a daily schedule that allows to attend to every event in program;

[**G7**] apply constraints on travel means;

- [**G7.1**] related to the length of travel;
- [**G7.2**] related to the period of day;

[**G8**] deactivate one or more travel means;

[**G9**] select combinations of transportation means that minimize carbon footprint;

[**G10**] reserve time for lunch or break events;

[**G11**] arrange trips;

- [**G11.1**] buy needed tickets;
- [**G11.2**] find available sharing vehicle.

1.2.2 Domain properties

[**D1**] username must be unique;

[**D2**] every event is related to a location;

[**D3**] events must happen in an existing place;

[**D4**] events cannot be in the past;

[**D5**] starting time is always specified;

[**D6**] ending time is not mandatory;

[**D7**] a person can travel only on one travel mean at once;

[**D8**] allowed travel means are cars, trains, metro, on foot, trams, bicycles, taxis, car sharing, bike sharing;

[**D9**] every travel is programmed with a combination of one or more travel means;

[**D10**] a person can be only in one place at once;

[**D11**] all travel means are related to information about average carbon footprints;

[**D12**] flexible timeslots can have a daily or periodical validity;

[**D13**] every flexible timeslot has a minimum amount of time that must be reserved;

[**D14**] to use a public transport a ticket is required;

[**D15**] a ticket may have a daily validity or a different periodicity;

[**D16**] user can own day/week/season passes;

[**D17**] to use a sharing vehicle a payment is required;

[**D18**] a sharing vehicle must be parked in an allowed position.

1.3 Definitions, Acronyms, Abbreviations

TODO

1.3.1 Acronyms

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

1.3.2 Abbreviations

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

1.4 Revision history

TODO

1.5 Reference Documents

- Specification Document: "Mandatory Project Assignments.pdf".

1.6 Document Structure

This RASD is composed by six parts:

1. The first part of the document contains an introduction to the problem, where goals are identified and basic information is provided in order to better understand the rest of the document.
2. The second part of the document describes generally the system, identifying its boundaries and the actors involved in the life-cycle of the system. Boundaries are defined providing all the necessary assumptions: both the ones required in order to adapt to the user's specifications and the ones that will hold and from now on will be considered as true.
3. The third part of the document is composed by:
   - functional and non functional specific requirements defined;
   - a list of eight scenarios is provided: each of them describes a particular situation that the system might have to cope with;
   - UML diagrams that model in detail the system behavior.
4. The fourth part of the document contains the Alloy model of the system, including all the relevant details. It also provides a proof of consistency and an example of the world generated.
5. The fifth part of the document contains a report of the hours spent to write this document.
6. The sixth part of the document contains references to external documents used in this document.

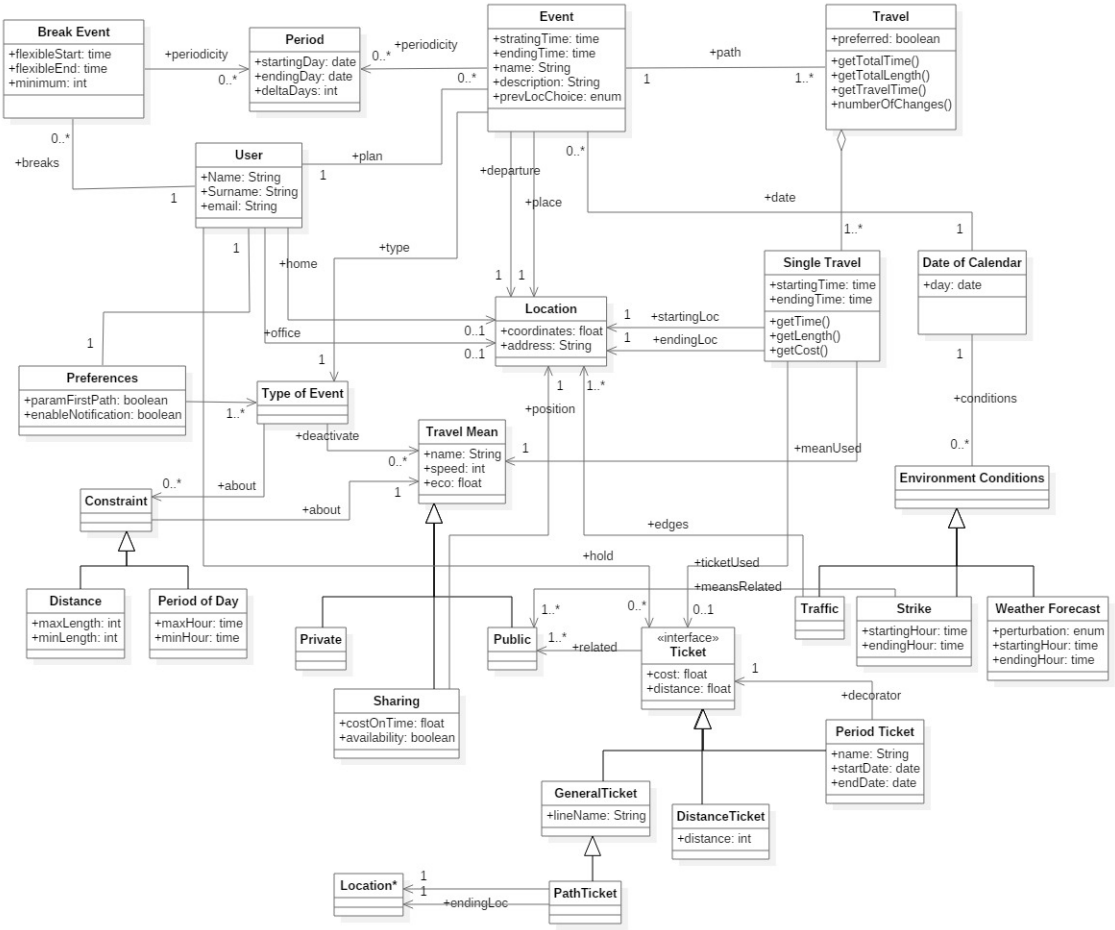
Chapter 2

OVERALL DESCRIPTION

2.1 Product perspective

TODO

2.1.1 Class Diagram



An *Event* is associated to one or more *Travels*, a *Travel* proposed requires the use of one or more *Travel means*. The path of a *Travel* that include different *Travel Means* needs a structure where every *Single Travel* (the path that can be performed with the same *Travel Mean*), is memorized. Through a vector, an object of the *Travel* class contains each *Single Travel*.

Travel class has the attribute *preferred*: a Boolean value that identifies the path showed in the daily schedule for an inserted *Event*. It must be checked that, for each *Event*, only one *Travel* is marked as preferred.
If a proposed path has different *Single Travels*, information about time and length are provided by methods of the *Travel* class that combine the values found in each related object of the*Single Travel* class.

To represent *Tickets* we use a hierarchy of classes: *Distance Ticket* is used for tickets whose validity is indicated with a length, *General Ticket* is used for particular tickets that allow the user to travel without constraints in a specified area, *Path Ticket* is used when departure and arrival locations are indicated on the ticket.
A *Ticket* can be valid into specified dates or after the validation, for this reason the management of the time of validity is done with *pattern Decorator*: it is used to specify the validity of an existing ticket.

2.2 Product functions

TODO

2.2.1 Requirements

Visitor should be able to:

G1] sign up into the system:

[**D1**] username must be unique;

[**R1**] the system checks if the email inserted is real;
[**R2**] user cannot sign up with the same mail twice.

User should be able to:

[**G2**] log into the system:

[**D1**] username must be unique;

[**R3**] mail and password inserted must be correct;
[**R4**] incorrect credentials prevent the user to log in.

[**G3**] create events specifying location, date, starting and ending time:

[**D2**] every event is related to a location;
[**D3**] events must happen in an existing place;
[**D4**] events cannot be in the past;
[**D5**] starting time is always specified;
[**D6**] ending time is not mandatory;

[**R5**] user must specify all mandatory fields to add the new event;
[**R6**] the system reserves the specified time for the event;

[**R7**] the system warns the user if the inserted event overlaps with an existing one;

[**R8**] if ending time is not specified, the systems considers as ending time the hour of departure for the next event;

[**R9**] when an event is inserted after an event without a specified ending time, its ending time is anticipated as stated in [[R8]].

[**G4**] obtain the best path according to his preferences and the list of eventual alternative paths to reach a location:

[**D7**] a person can travel only on one travel mean at once;

[**D8**] allowed travel means are cars, trains, metro, on foot, trams, bicycles, taxis, car sharing, bike sharing;

[**D9**] every travel is programmed with a combination of one or more travel means;

[**R10**] every proposed path must be feasible in the available time (the interval between two consecutive events);

[**R11**] if the travel involves more than one travel mean, starting location of the first proposed path and ending location of the last proposed path must coincide with starting and ending location of the whole planned travel;

[**R12**] the system does not consider paths that violate constraints on travel means defined by the user;

[**R13**] the system checks user preferences to decide which is the best path;

[**R14**] the system warns the user if it isnt possible arrive at the location before the event to attend starts;

[**R15**] appropriate travel means must be suggested according to the type of event that they are related to;

[**R16**] if a strike occurs the system wont consider involved travel means;

[**R17**] if it rains the system wont consider paths involving the bicycle.

[**G5**] change the selected path with an alternative path:

[**D7**] a person can travel only on one travel mean at once;

[**D8**] allowed travel means are cars, trains, metro, on foot, trams, bicycles, taxis, car sharing, bike sharing;

[**D9**] every travel is programmed with a combination of one or more travel means;

[**R18**] the system must show to the user all possibilities to reach a location in according with the requirements of [[G4]];

[**R19**] the system allows the user to change the path only if doesnt generate overlapping with events added later to the involved one.

[**G6**] obtain a daily schedule that allows to attend to every event in program:

[**D10**] a person can be only in one place at once;

[**R20**] the combination of the paths proposed for the day must be feasible in the allotted time;

[**R21**] if there are multiple events at the same time the system will propose in the schedule only the first event added.

[**R22**] if the user forces into the schedule an event that overlaps with events present in the schedule, these are removed from the schedule.

[**G7**] apply constraints on travel means:

[**D7**] a person can travel only on one travel mean at once;

[**D8**] allowed travel means are cars, trains, metro, on foot, trams, bicycles, taxis, car sharing, bike sharing;

[**D9**] every travel is programmed with a combination of one or more travel means;

[**R23**] the system requires min/max length allowed for a path to impose a constraint on a travel mean;

[**R24**] the system requires an interval of time allowed to impose a constraint on a travel mean;

[**R25**] the system doesnt consider solutions that violate constrains.

[**G8**] deactivate one or more travel means:

[**D7**] a person can travel only on one travel mean at once;

[**D8**] allowed travel means are cars, trains, metro, on foot, trams, bicycles, taxis, car sharing, bike sharing;

[**D9**] every travel is programmed with a combination of one or more travel means;

[**R26**] the system allows the user to specify one or more travel means that cant be used.

[**R27**] the system doesnt consider solutions that include deactivated travel means.

[**G9**] select combinations of transportation means that minimize carbon footprint:

[**D11**] all travel means are related to information about average carbon footprints;

[**R28**] for each path, the system estimates carbon footprint produced.

[**G10**] reserve time for lunch or break events:

[**D12**] flexible timeslots can have a daily or periodical validity;

[**D13**] every flexible timeslot has a minimum amount of time that must be reserved;

[**R29**] the system allows the user to specify a flexible interval and a minimum amount of time to schedule a break;

[**R30**] if there is enough time for a break, the system reserves it within the specified flexible interval;

[**R31**] if there isnt enough time into the flexible interval specified a warning is thrown.

[**G11.1**] arrange trips/buy needed tickets:

[**D14**] to use a public transport a ticket is required;

[**D15**] a ticket may have a daily validity or a different periodicity;

[**D16**] user can own day/week/season passes;

[**R32**] the system allows the user to specify all the ticket he owns;

[**R33**] the system shows to the user if he holds a ticket for a proposed travel;

[**R34**] the system allows the user to buy public transportation tickets according to proposed travel;

[**R35**] the system provides information about time of departure and arrival of the proposed travels.

[**G11.2**] arrange trips/find available sharing vehicle:

[**D17**] to use a sharing vehicle a payment is required;

[**D18**] a sharing vehicle must be parked in an allowed position;

[**R36**] the system shows to the user where sharing vehicles are located;

[**R37**] the system provides information about travel time with shared vehicles.

2.3 User characteristics

TODO

2.4 Assumptions, dependencies and constraints

TODO

Chapter 3

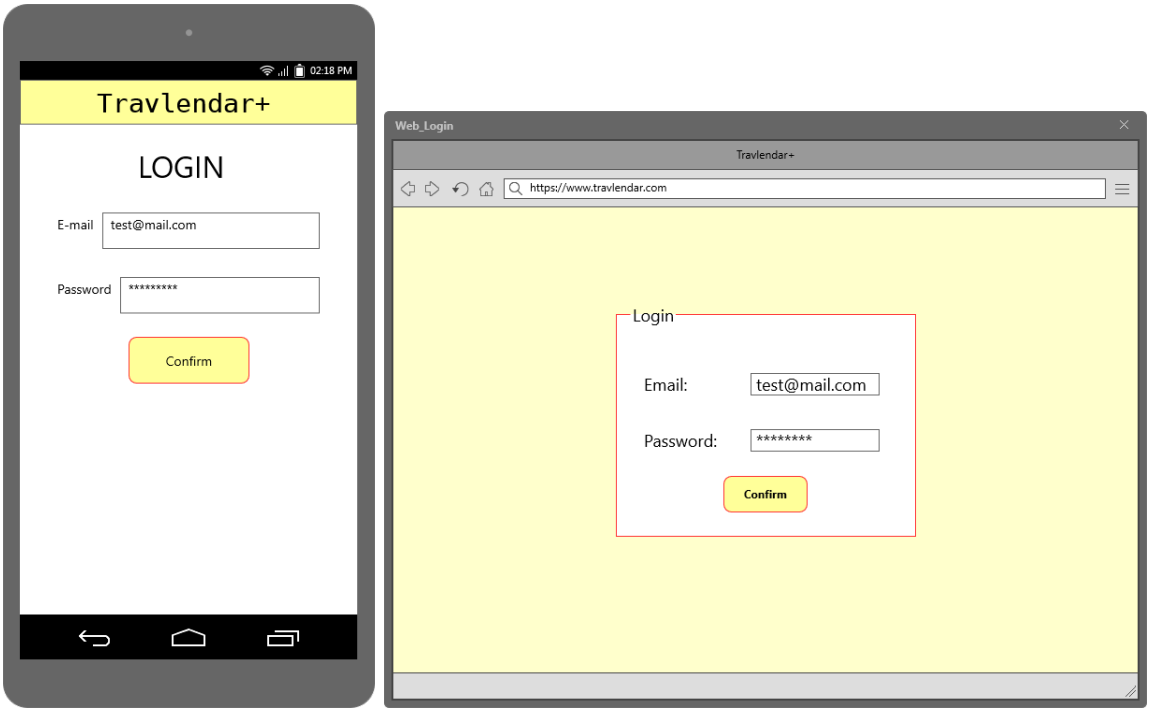
SPECIFIC REQUIREMENTS

3.1 External Interface Requirements

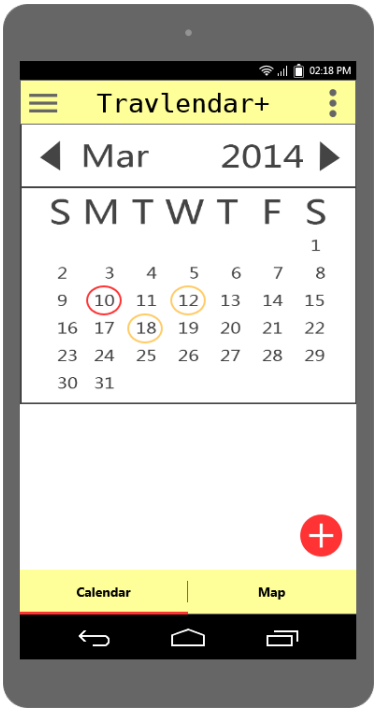
3.1.1 User Interfaces

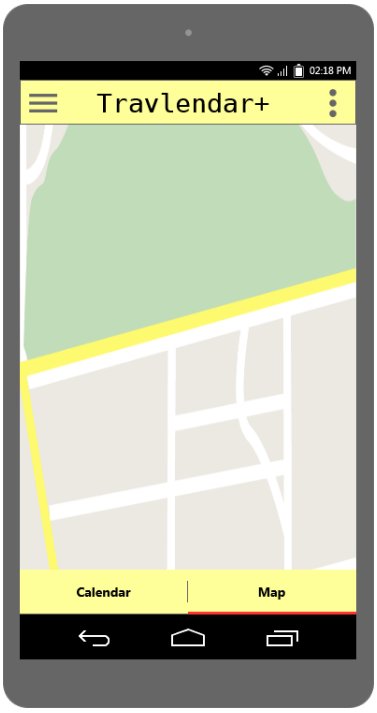
The following mockups are a representation of the look of the app in its first release.

3.1.1.1 Login

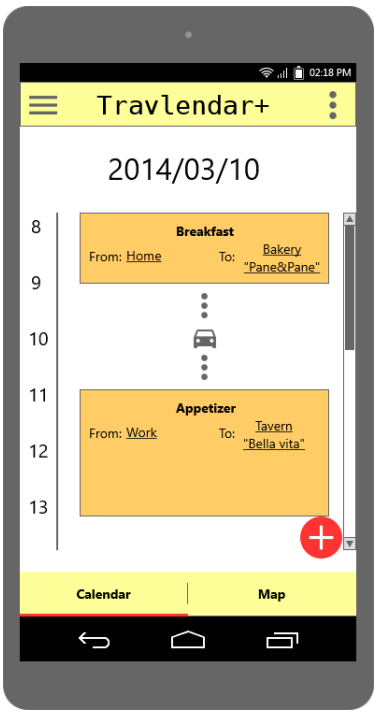


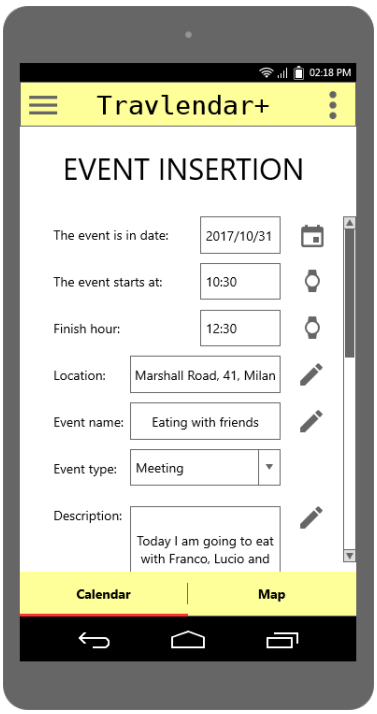
3.1.1.2 Calendar and Map views



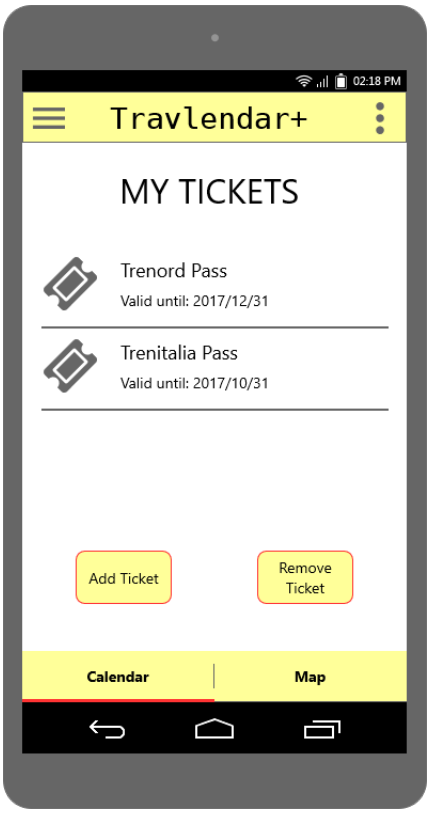


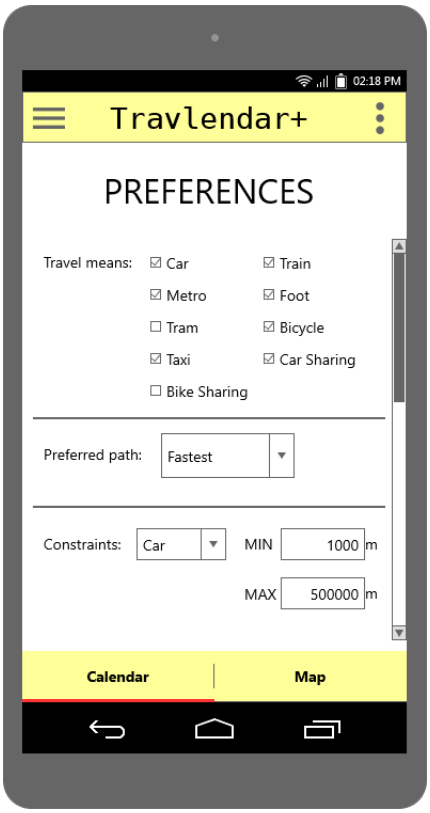
3.1.1.3 Day Viewer and Event Creation



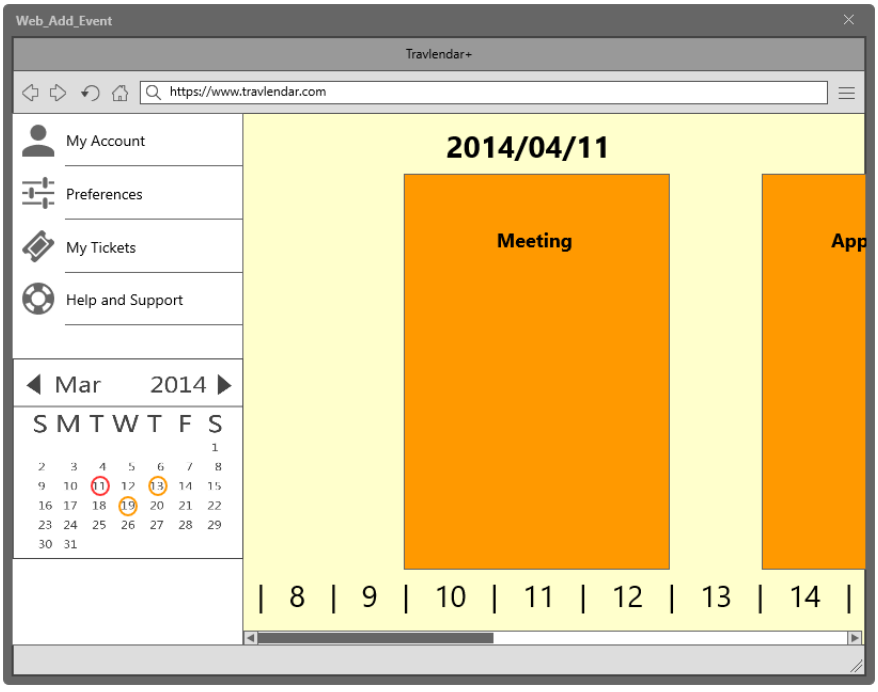


3.1.1.4 My Tickets and Preferences





3.1.1.5 WebApp View



3.1.2 Hardware Interfaces

TODO

3.1.3 Software Interfaces

TODO

3.1.4 Communication Interfaces

TODO

3.2 Functional Requirements

3.2.1 Scenarios

3.2.1.1 Scenario 1

Oscar is a businessman who travels a lot and he would like to organize his travels quickly and precisely. A friend advises him to try Travlendar+. Oscar enters the Travlendar+ website and loads the registration page by clicking on the 'signup' button, he fills all the mandatory fields and then he receives a confirmation email and he clicks on the confirmation link inside. Then Oscar logs in and starts using Travlendar+.

3.2.1.2 Scenario 2

Nigel lives in Milan and tomorrow hes going to have an appointment in Lecco, so he has to decide how to reach his appointment location; Nigel accesses the login page from his personal computer by using his browser, fills the username and password fields, clicks on the 'confirm' button and logs in. After he clicks on the dedicated button that adds a new event, he fills all the requested fields, setting the event type as 'work meeting' in order to obtain proper suggested travel means, then he confirms the event creation. On the next day, Nigel travels to Lecco, following the travel schedule proposed by his Travlendar+ app, reaching his appointment right on time.

3.2.1.3 Scenario 3

Jasper inserts an event into his Travlendar+ calendar, but the location inserted is too far away from his previous event, therefore a feasible path to reach the location of the next event is not available in the allotted time. Jasper is notified by a warning that he will not be able to reach his appointment in time.

3.2.1.4 Scenario 4

Ophelia inserts a new event into his Travlendar+ calendar, but that event overlaps with another already existing event. Ophelia is notified by her app and the system asks Ophelia to choose which one of the overlapping events she wants to attend. She chooses the last event inserted, then she reads the travel means proposed to respect her new time schedule.

3.2.1.5 Scenario 5

Henrietta wants to personalize her Travlendar+ experience, so she clicks on the menu icon and then the voice preferences. She selects her preferred travel means, she chooses to always obtain the fastest path and then, since she walks with a limp, she inserts a maximum walking distance of 500 meters. Before clicking on the 'confirm' button, she sees a polluting truck outside her window, so she decides to always follow travel paths that minimize her carbon footprint. She

selects the relative option in her preferences. Finally she clicks on the confirm button and she observes that her suggested travels have changed according to her new preferences.

3.2.1.6 Scenario 6

Arthur cant concentrate on his work if he doesn't eat his lunch between 12.00 and 13.00. He also needs at least 25 minutes to consume a proper lunch. For this reason Arthur inserts a flexible break event into his Travlendar+ calendar, in order to be sure that every day his app will reserve a time slot for his lunch. A week later, Arthur inserts a new event whose travel overlaps with his flexible lunch event and his app notifies him. Arthur decides to ignore the warning, since he'll s going to eat during his travel.

3.2.1.7 Scenario 7

Gwendolyn looks at her Travlendar+ app and discovers that on the next day she has to travel first by train to Milan and then take a bike of MoBikes sharing system in Milan. She clicks on the train travel slot in order to buy the proper ticket and the apps redirects her in the right Trenords online shop webpage. When she is about to buy the ticket she suddenly remembers that she has a weekly Trenord pass, therefore she cancels the transaction and, in order to avoid making the same mistake twice, she adds her pass information into her Travlendar+ app. The next day Gwendolyn takes the train to Milan and when she arrives she opens her Travlendar+ app in order to find the nearest bike of MoBikes. Gwendoline arrives in time at her appointment.

3.2.1.8 Scenario 8

Harvey has to reach an appointment near his home next week. He adds an event in his Travlendar+ calendar. He observes that the app suggests him to use a bike to reach the location of this event. The day before his appointment the weather forecasts rain for the successive day, so Harvey is notified by his app that, due to the forecast, he should avoid taking the bike, suggesting instead to reach his appointment by car.

3.2.1.9 Scenario 9

Sara inserts an event in her Travlendar+ calendar, then she looks at the proposed travel path. Since she did not like the proposed travel, she clicks on the proposed path and selects another feasible alternative. The next day Sara travels according to the travel path she likes more.

3.2.2 Use case descriptions

3.2.2.1 Registration

Participating actors	Generic visitor
Entry Condition	There are no entry conditions.
Event Flow	1. The visitor clicks on the Register button displayed onto the homepage; 2. The visitor fills all the mandatory fields shown required by the system including his email, his password (twice) and a captcha; 3. The visitor clicks on Confirm button; 4. The visitor receives a confirmation email and clicks on the confirmation link; 5. The system saves all user data inserted.
Exit Condition	The visitors registration is completed successfully, so the visitor is registered as an user of Travlendar+ and he can log in into the system as a registered user.
Exception	If: • The visitor insert an email already connected to an existing account; • Insert invalid infos into in some mandatory field; • Leave empty a mandatory field; Then the system will request the visitor to complete/ revise all uncorrected field, highlighting them. if the visitor doesnt activate the account, after a month the activation link will expire and all user data will be deleted.

Table 3.1: registration use-case

3.2.2.2 Login

Participating actors	Unauthenticated User
Entry Condition	There are no entry conditions.
Event Flow	1. The visitor clicks on the 'Login' button displayed on the home-page; 2. The visitor inserts the email and the password previously used for registration; 3. The visitor clicks on 'Confirm' button; 4. The system redirect the user to the main view of Travlendar+.
Exit Condition	The Visitors login is completed successfully, so the visitor can use all the Travlendar+ functions.
Exception	If: • The email inserted is not one of the emails previously used by an user to sign up; • The password inserted by the visitor is not the one associated with the email inserted; • At least one of the field is left empty; Then the system will notify the visitor to complete/ revise all un-corrected field, highlighting them.

Table 3.2: Login use-case

3.2.2.3 Create event

Participating actors	User
Entry Condition	The user must be logged/registered in Travlendar+.
Event Flow	1. The user clicks on the dedicated button to add a new event; 2. The user insert: date, starting time, eventually ending time, location, name of the event, type of event (predefined or personalized), description, starting location (previous one or others); 3. The user confirms the events creation; 4. The system computes the best possible path according to users preferences Describe the Exit Condition;
Exit Condition	The system redirects the user to the calendar and add the travel time slot required to reach that event (comprehensive of travel description).
Exception	The inserted event overlaps with one or more previously added events (also the travel is considered in the eventual overlap). The user is notified with a warning message and the overlapping event isnt considered in the user travel planning schedule (but remain saved into the calendar), the system have to choose which one of the overlapped event he want to attend.

Table 3.3: Create event use-case

3.2.2.4 Define preferences

Participating actors	User
Entry Condition	The user must be registered and logged in Travlendar+.
Event Flow	1. The user opens the menu; 2. The user selects the tab 'preferences'; 3. The system shows a page containing fields to fill; 4. The user defines his preferences by filling the fields on the page; 5. The user clicks on the 'save' button.
Exit Condition	The user has selected his preferences, which have been saved correctly.
Exception	If the user exits the page without clicking the 'save' button, then the system will not save the preferences modified by the user.

Table 3.4: Define preferences use-case

3.2.2.5 Define flexible breaks

Participating actors	User
Entry Condition	The user must be logged/registered in Travlendar+.
Event Flow	1. The user clicks on the dedicated button to add breaks into the schedule; 2. The user inserts a flexible period of time (specifying start and end times) into which the break must happen and the minimum amount of time that must be dedicated to the break; 3. The user also specifies the return period of the break (daily, weekly, monthly or until a specified date) or if it refers only to certain days of the week (until a specified date);
Exit Condition	The system notifies the user that the info about break are correctly saved.
Exception	It is not possible to dedicate the required time into the schedule of one or more days because of previous added events. The system asks to the user to change the length of the break.

Table 3.5: Define flexible breaks use-case

3.2.2.6 Arrange trips

Participating actors	User, Transport service provider
Entry Condition	The user must be logged in Travlendar+.
Event Flow	1. The user opens his calendar 2. The user clicks on the trip he want to arrange; 3. The system shows all tickets to be buyed and the tickets already bought; 4. The user clicks on the ticket he want to buy. 5. The system redirect the user to the right website (of the right Transport service provider) in order to buy the tickets.
Exit Condition	The user has successfully arranged his travel.
Exception	There are no exceptions.

Table 3.6: Arrange trips use-case

3.2.2.7 Locate the nearest sharing vehicle

Participating actors	User, Transport service provider
Entry Condition	The user has opened the arrange trip tab and is about to travel.
Event Flow	1. The user open the map; 2. The system shows in the map the nearest vehicle of car sharing according to the chosen path and the infos provided by the transport service provider;
Exit Condition	The user take and use the suggested vehicle.
Exception	• If no near vehicle is found the system re-compute another path and show it to the user; • if the user take a shared vehicle, but not the suggested one nothing happen, just as he has taken the suggested vehicle;

Table 3.7: Locate nearest sharing vehicle use-case

3.2.2.8 Add ticket possessed

Participating actors	User
Entry Condition	The user has opened the arrange trip tab.
Event Flow	1. The user selects the tab 'my tickets'; 2. The system show a page containing all the tickets and passes possessed by the user; 3. The user clicks on the 'add ticket' button; 4. The system show a page containing fields to fill; 5. The user inserts info regarding the ticket/pass possessed; 6. The user clicks on the 'save' button; 7. The system adds the ticket/pass to those already present in the user account.
Exit Condition	The user has successfully inserted his ticket/pass in the system.
Exception	If the user exits the page without clicking the save button, then the system will not save the ticket added by the user.

Table 3.8: Add ticket possessed use-case

3.2.2.9 Obtain feasible travel paths

Participating actors	User, google maps APIs
Entry Condition	The user must be registered and logged in Travlendar+.
Event Flow	1. The user open the calendar tab; 2. The user selects a day in the calendar; 3. The system shows the proposed feasible paths (shown as travel events) between the events; 4. If the user want to select an alternative travel path he can click on the travel event in order to choose among the proposed feasible alternatives.
Exit Condition	The user has seen the possible travel paths that can be used to reach his meetings.
Exception	If no feasible travel path exist between two events, the system shows a warning in the calendar.

Table 3.9: Obtain feasible travel paths use-case

3.2.2.10 Create personalized event profiles

Participating actors	User
Entry Condition	The user must be registered and logged in Travlendar+.
Event Flow	1. The user opens the menu; 2. The system shows the default event's profile 3. The user selects the tab 'create personalized type of event'; 4. The system shows a page containing a text field to fill; 5. The user inserts the name of the personalized type of event that he wants to create; 6. The user inserts the constraints on travel means related to that type of event; 7. The user clicks on the 'save' button; 8. The system adds the new type of events to those already existing.
Exit Condition	The user has successfully created a new personalized type of event in the system.
Exception	If the user exits the page without clicking the 'save' button, then the system will not save the new personalized type of event created by the user.

Table 3.10: Create personalized event profiles use-case

3.2.2.11 View calendar

Participating actors	User
Entry Condition	The user must be logged in Travlendar+.
Event Flow	1. The user clicks on the calendar button; 2. The system shows a calendar including all inserted events, and the travel paths related.
Exit Condition	The system let the user check his calendar.
Exception	There are no exceptions.

Table 3.11: View calendar use-case

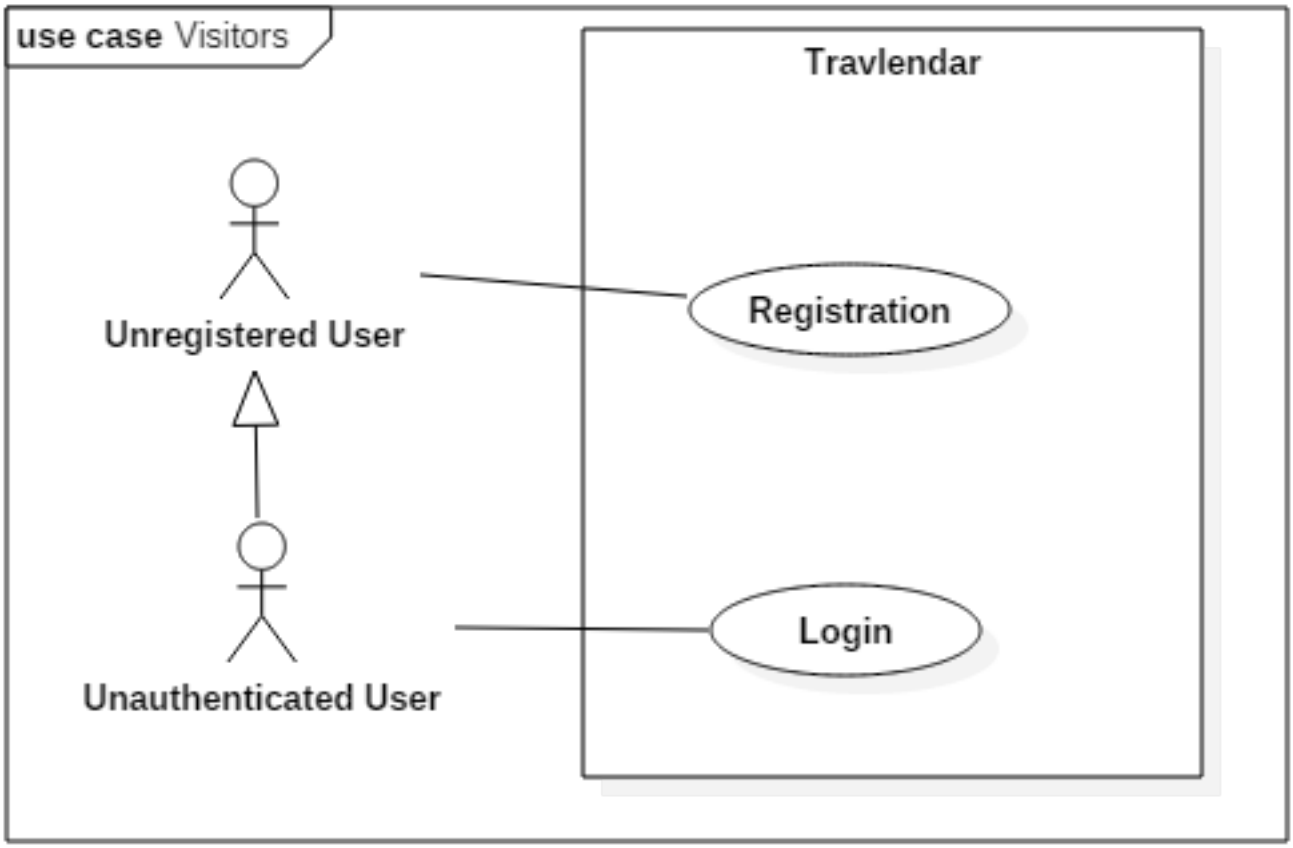
3.2.2.12 Choose between overlapping events

Participating actors	User
Entry Condition	The user must be logged in Travlendar+, at least two events are overlapped.
Event Flow	1. The user clicks on the calendar button; 2. The system shows a calendar including all inserted events, and the travel paths related, the overlapping events are displayed in a separate way in respect to the actual day schedule; 3. The user drag the chosen overlapping event into his day schedule; 4. The system remove the precedent event (in conflict) and put it into the overlapping event list; 5. The system shows the new day schedule, with updated travel paths.
Exit Condition	The system let the user check his calendar.
Exception	There are no exceptions.

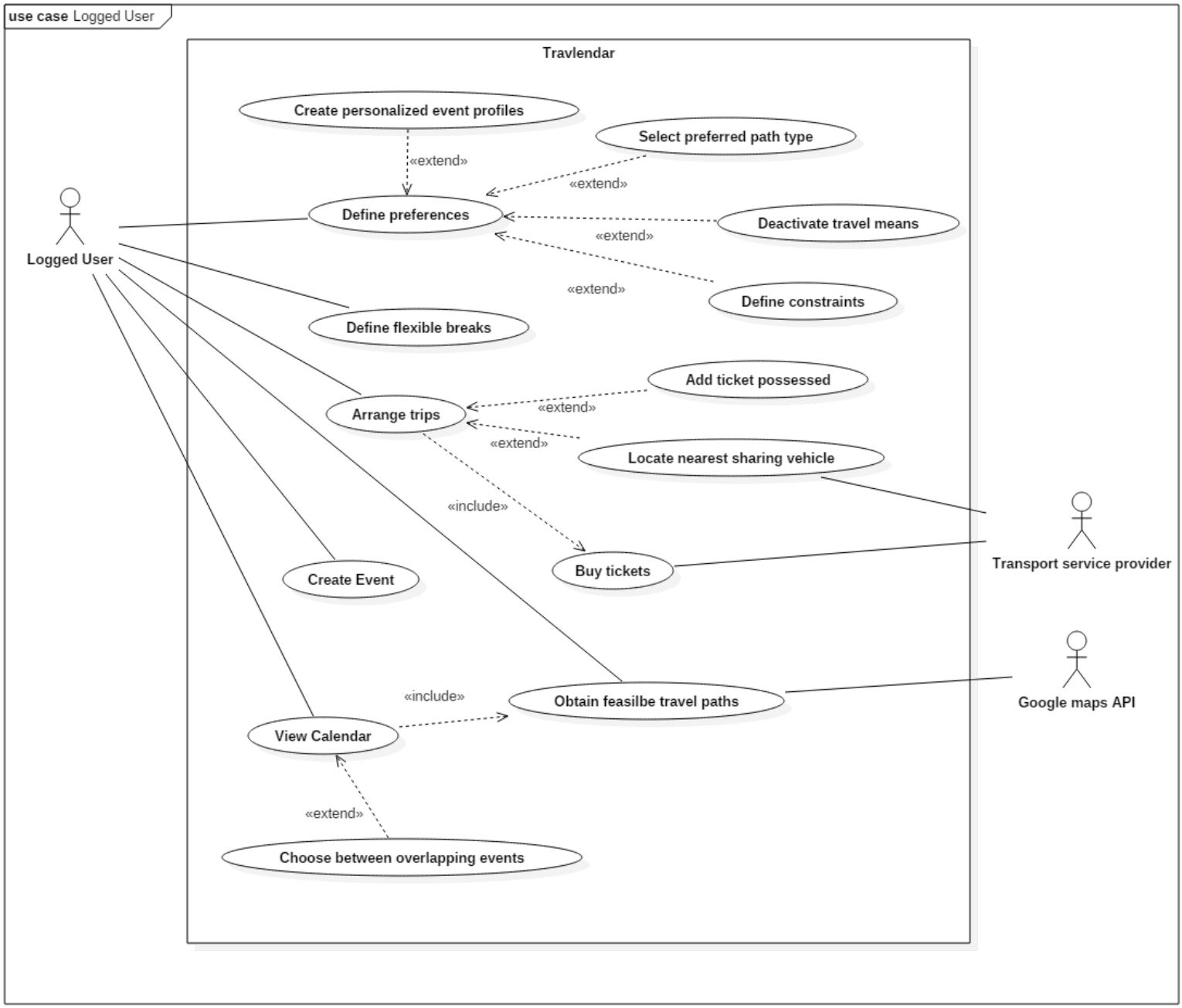
Table 3.12: Choose between overlapping events use-case

3.2.3 Use case diagrams

3.2.3.1 Visitors



3.2.3.2 User



3.3 Performance Requirements

TODO

3.4 Design Constraints

3.4.1 Standards compliance

TODO

3.4.2 Hardware limitations

TODO

3.4.3 Any other constraint

TODO

3.5 Software System Attributes

3.5.1 Reliability

The reliability (Nb = 1 - Probability of failure) requested is related to many factors: the software developed must be stable and efficient in both server and client side, the user must satisfy the minimum requirement specifications specified in this document otherwise the reliability of the client app or the browser interface are not guaranteed; The reliability of the servers must be also taken into account and guaranteed by a proper maintenance .

3.5.2 Availability

The system must offer an availability of 99% (an overall 24/7 service). The remaining 1% (at maximum) shall take into account the time spent for ordinary maintenance sessions. A backup server will be provided in order to maintain the availability of the service even after the main server failure.

3.5.3 Security

All user's data will be stored and encrypted using a proper hashing mechanism, even system administrators must not been able to access personal user's data. The GPS location of the users, detected by the mobile app, during the user's travels, is not to be memorized. All the connections established between users and server must use the HTTPS protocol. Every communication between server and client will be encrypted.

3.5.4 Maintainability

A version control system will be used in order to manage and organize all the different code revisions. In the development phase the entire source code will be properly documented and commented in order to ease the effort of possible future developers of understanding how the system has been designed and how it works. The documentation will also facilitate the system's maintenance.

3.5.5 Portability

The web application must support main browsers (such as Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) latest version.
The mobile application will be developed at first for Android architecture and then for iOS architecture.

The server-side of the application is written in Java, therefore it can be executed on any machine that runs a Java Virtual Machine.

Chapter 4

FORMAL ANALYSIS USING ALLOY

TODO

Chapter 5

EFFORT SPENT

TODO

Chapter 6

REFERENCES

TODO