POLITECNICO
MILANO 1863

Requirements analysis and specification document (RASD)

Davide Rossetto 894029, Alessandro Tatti 883861

Delivery date: 2017 Oct 29

v1.2

Contents

Section 1

Introduction

1.1 Purpose

The Requirements Analysis and Specification Document for the Travlendar+ system management is intended

- to describe the system itself,
- its functional and non-functional requirements,
- the components,
- its constraints,
- the relationship with real world and users, providing several use cases and scenarios.

A part of the documentation uses Alloy, a language to describe structures and a tool to explore and provide a formal specification of some features of the system.

This document is addressed primarily

- to developers and programmers who must meet the requirements,
- testers who need to determine if the requirements are met,
- project managers, who control the development process,
- and users who validate the goals of the system.

1.2 Scope

The product is a digital management system to support the creation of a calendar-based application which automatically computes and accounts for travel time between appointments in order to make sure that users are not late for appointments and supports users in their travels.

The system consists of two back – end server applications:

- one handles requests for entering appointments or trips, managing schedules and routes;
- the other communicates with the systems of transport companies.

And two front – end applications:

- The web-based application which provides the end user with a friendly interface to take advantage of services of Travlendar+;
- A mobile application that allows the user to easily access the service wherever he/she needs.

The system is intended for users who must be allowed to register and access the system via username and password, to make the appointment and manage the process easier and quicker.

Users can create meetings, and when meetings are created in locations that are unreachable at the allotted time, a warning is created; the application must also take into account possible issues related to the request (e.g. public service strikes on the day scheduled for the meeting).

The system should allow users to define various kinds of user preferences: users can activate or deactivate each travel means, should be able to provide constraints on different travel means and select combinations of transportation means that minimize carbon footprint. In addition, the user must specify a flexible break: the system must manage this aspect, grant 30 minutes to the users in order to have break within the set time interval.

1.3 Goal

The goals of Travlendar+ are the followings:

1. Let the user register to the service and login via provided credentials;
2. Let the user manage his/her own profile;
3. Let the user delete his/her own profile;
4. Let the user insert his/her meeting in the schedule application;
5. Let the user modify his/her meetings in the schedule application;
6. Let the user delete his/her meetings in the schedule application;
7. Let the system work efficiently by generating an alarm when a meeting is not possible within the specified time range;
8. Let the system indicate which is the best travel means to be used for a given meeting;
9. Let the user indicate his/her preferences on the travel means;
10. Automatically the system searches for the shortest path to reach the meeting site;
11. Automatically the system searches the cheapest means of transport to reach the meeting site;
12. Let the user specify its intent to minimize his carbon footprint;
13. Let the user specify a flexible break;
14. Let the user specify an interval time for a break;
15. Automatically the system must find the specify interval time to have break.

1.4 Definitions, Acronyms, Abbreviations

Actor: Specifies a role played by a user or any other system that interacts with the system.

API: Application Programming Interfaces.

Availability: the amount of time that the above remains online with respect to the total time.

Back – end application: Computer program that remains in the background, or resides on a server located in a back room. A user, generally, interfaces only with a front – end application.

Front – end application: Any application the users interact with directly. It provides the so called presentation layer.

GPS: Global Positioning System

Guest: Any person who is not registered or logged in to the Travlendar+ service.

JEE: Java Enterprise Edition.

Maintainability: determines the probability that a failed machine can be restored to its normal operable state within a given timeframe, using the prescribed practices and procedures.

Mobile application: Computer program designed to run on a mobile device such as smartphone or tablet.

OS: Operative system.

Portability: measurement of how easily an application can be transferred from one computer environment to another.

RASD: Requirements Analysis and Specification Document.

Reliability: Ability of a computer program to perform its intended functions and operations in a system's environment, without experiencing failure.

Security: Prevention of access to information by unauthorized recipients, and intentional but unauthorized destruction or alteration of that information.

System: Hardware and software components that run a computer or computers.

User: Any person subscribed and logged in to the service who hence can insert a meeting using Travlendar+.

User Interface: It is the way through which a user interacts with an application or a website.

Web application: Client – server application accessible by a user through a browser.

1.5 Revision history

v0.1 Construct basic document's structure.

v0.2 Add *Purpose*, *Scope*, *Goal*, *Definitions, Acronyms, Abbreviations*, *Reference documents*, *Document structure*.

v0.3 Add *Product Perspective*, *Product Functions*, *User Characteristics*, *Assumptions and Dependencies*, *Constraints*, *World and Machine model interpretation*. Modify *Definition, Acronyms, Abbreviations* and document structure. Create a simple Appendix, which will be completed at the end.

v0.4 Add *External Interface Requirements*. Modify *Goal*, *Definitions, Acronyms, Abbreviations*, *Product Functions*, *Use Case Model*, *Assumption and Dependences*, *World and Machine model*.

v0.5 Add *Functional Requirements*.

v0.6 Add *Performance Requirements* and *Software System Attributes*.

v0.7 Add Activity diagrams.

v0.8 Change part of LunchTime in Break: add and modify last three goals, modify use case part.

v0.9 Correct the form and sintax of the document.

v0.10 Add mockups.

v0.11 Add *Formal Analysis Using Alloy*.

v1.0 First delivery.

v1.1 Add assumption on *Travel information provider*, other minor fixes.

v1.2 Syntactic revision.

1.6 Reference documents

This document is based on the specifications concerning the RASD assignment for the Software Engineering II project, part of the course held by professors Elisabetta Di Nitto and Matteo Giovanni Rossi at the Politecnico di Milano, A.Y. 2017/2018.

1.7 Document structure

This document consists of three sections:

Section 1: Introduction A general introduction and overview of the system-to-be purpose, scope and goals, along with some important information about this document.

Section 2: Overall description It describes the general factors that affects the product and its requirements. The section provides a background for those requirements which are defined in detail in Section 3 and makes them easier to figure out.

Section 3: Specific Requirements All the software requirements are sufficiently specified in order to allow the designers to use them. Both functional and non-functional requirements are mentioned.

There are two additional parts, Appendix and Bibliography that provide another information about the sections of this document.

Section 2

Overall Description

2.1 Product Perspective

2.1.1 User interfaces

The user has two main ways to access the system:

- via a web application accessible from any modern browser
- via a mobile application that can run on any modern smartphone

Although they are two different platforms, the user interface must be unified and intuitive, allowing anyone to use it without any specific training.

2.1.2 Hardware interfaces

The web application can be executed on any modern computer that meets the minimum hardware requirements (2.5.2).

The mobile application can run on any modern mobile device (i.e. smartphone, tablet) with mobile data connectivity, GPS and meets the minimum hardware requirements (2.5.2).

2.1.3 Software interfaces

The web application must support most of the modern browsers e.g., IE, Firefox, Chrome, Safari.

The mobile application must be supported by the most widespread mobile OSs such as iOS and Android that meets the minimum hardware requirements (2.5.2).

The backend application must rely on a commercial DBMS to store data and must be implemented in Java. The backend also have to interface with the APIs of a public transportation and traffic information provider.

2.2 Product Functions

The system allows the users to create meetings and helps them to reach the location.
The users can:

- register to the service;
- login in to the service;
- manage personal information and delete their accounts;
- create meetings;
- manage their preferences, such as activate/deactivate travel mean, provide constraints on travel mean, specify their intent to minimize his carbon footprint;
- specify break time.

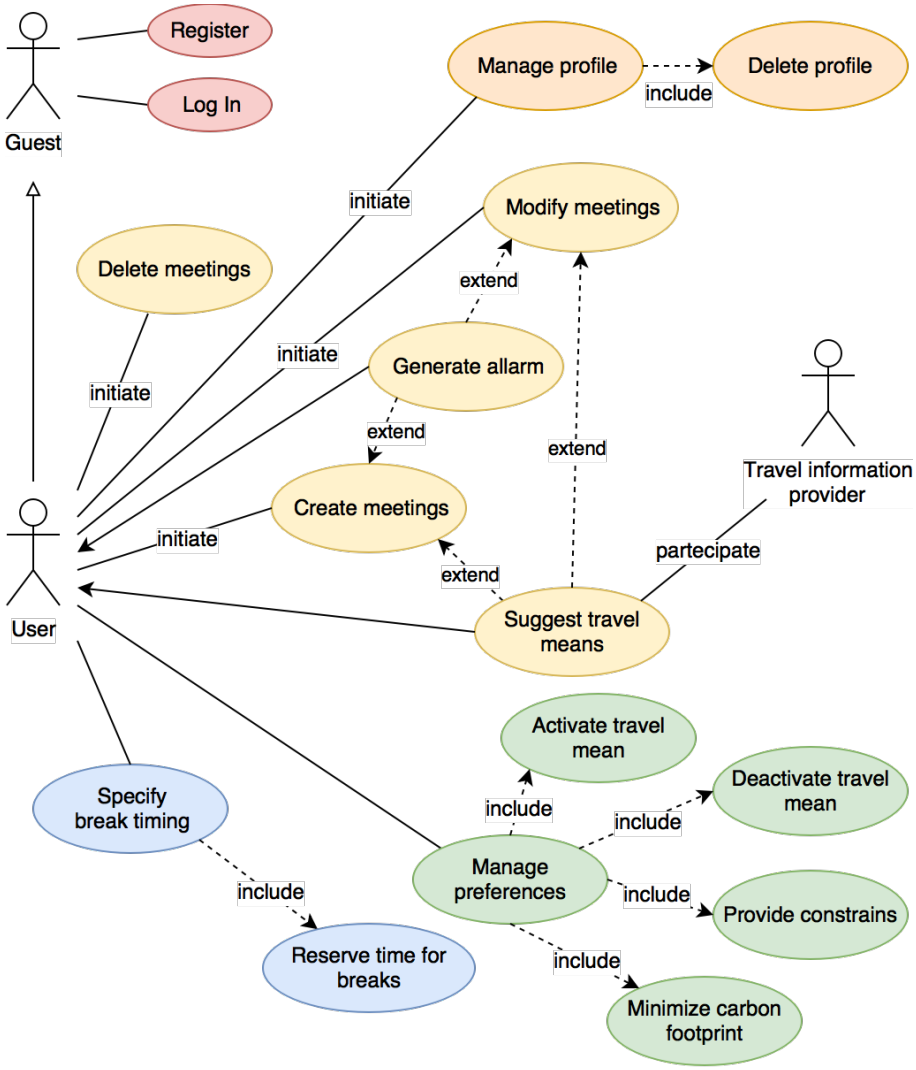

Figure 2.1: Use Case Model. Explains actors and functionalities offered by the system.

2.3 User Characteristics

The target user is someone who needs an automated system to manage appointments and schedules the travels.

The meeting should not overlap with any pre - existing one and the app should allow the user to move from his/her present position to the meeting site, using the means of transport preferred and according to traffic conditions.

2.4 Assumptions and Dependences

The following assumptions are given for granted:

- Transport means comply with users requests.
- The device is always connected to the server.
- All users provide correct and valid data at time of registration.
- GPS shows the actual position of the owner.
- The system can rely on a *Travel information provider* that gives correct and updated information about travel means paths and timetables.
- The event, when inserted, modified or deleted, must not be in the past.

2.5 Constraints

2.5.1 Regulatory policies

The application must be allowed by the user to collect his/her position, through GPS.

2.5.2 Hardware limitations

- Web application:
  - Internet connection;
  - 800x600 screen resolution;
  - JavaScript enabled.
- Mobile application:
  - Internet connection;
  - 50 MB of available storage space;
  - 1GB of RAM;
  - GPS module.

2.5.3 Reliability requirements

The system reliability, that is the probability to operate without a failure for a specific period of time, must be at least 99%.

2.6 World and Machine model interpretation

In this part of RASD, a description of the system-to-be is provided following the World and Machine model introduced by Jackson and Zave.

They indicate as the Machine the portion of the system-to-be developed, typically software-to-be plus hardware. The Machine domain is the set of phenomena located entirely in the machine and that the machine control (e.g., machine algorithms, controlled device, ...)

Opposite, the World domain is a set of phenomena that the machine cannot observe.

The World is connected with the Machine through Shared Phenomena part of it both observed by the Machine and by the World themselves. The Shared Phenomena can be controlled by the World and observed by the Machine or controlled by the Machine and observed by the World.

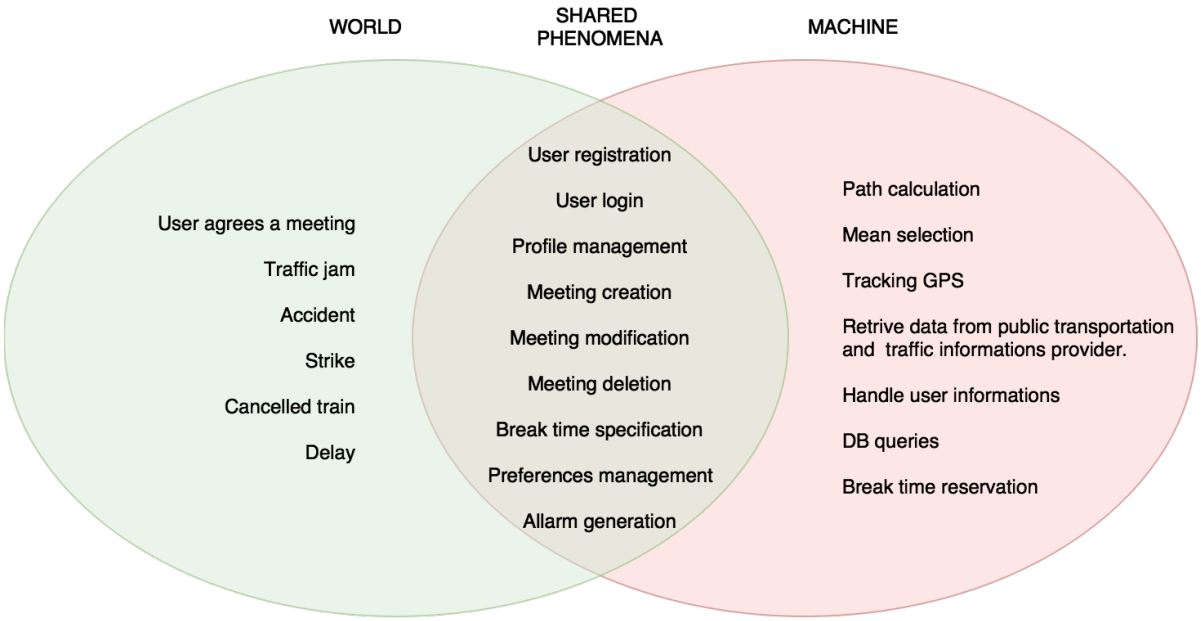

Figure 2.2: World and Machine model for the main functionalities offered by the system.

Section 3

Specific Requirements

3.1 External Interface Reqirements

3.1.1 User Interfaces

The user interface must be intuitive and unified, granting to the user a pleasant experience. In order to make it possible both web and mobile application must satisfy the following requirements:

- If the session is not already active the user must be redirected to a Login page. If he/she is not already registered he/she can create a new account within the same page. Moreover, the user must be able to Sign In/Sign Up not only with username and password but also with his/her social accounts (e.g. Google, Facebook, Twitter);
- Once logged, the user must be redirected to his/her personal page;
- A toolbar must allow the user to navigate through the pages described below;
- The user's personal page must display an overview of user's future events (display them in a list, in a calendar or in a map) and must also offer the possibility to insert new ones;
- Clicking on an event the user must be redirected to the event's detail page;
- The event's detail page must display detailed information such as location of the event, time and suggested mean (including the path) needed to reach it;
- The event's detail page must allow the user to modify or delete the event;
- If an user tries to insert or modified an event which location is unreachable in the remaining time an alert should be displayed;
- The Account settings page must allow the user to modify its/her personal informations or delete the account;
- The user must be allowed to select between different languages (en, it, de, fr, es, ru, zh, ja, ar);
- The UI must comply the Flat Design principles;
- Both web and mobile applications must use the same graphic objects for the same interface elements.

Specific constraints must be satisfied by specific application:

- Web application:
  - The user interface must be responsive i.e. adapt to screen size;
  - All pages must comply W3C standards.
- Mobile application:
  - Must run on iOS 9.3 or greater and Android 5.0 or greater.

3.1.2 Hardware Interfaces

As already mentioned in section 2.1.2, the web application can be executed on any computer that meets the basic requirements described in the "Hardware Limitation" section.

The mobile application must exchange data with the GPS module located on any type of smartphone or tablet. You must also have an internet connection to communicate with the main system server.

3.1.3 Software Interfaces

The backend application requires the following software products:

- Java EE 7 - http://www.oracle.com/technetwork/java/javaee/overview/index.html
- MySQL 5.7 - http://dev.mysql.com/

As mentioned in Section 2.1.3, the backend must interfaced with the APIs of a public transportation and traffic informations provider, to have the information useful to plan the path.

The mobile applications requires the following software products:

- (iOS) Swift 4 - https://developer.apple.com/swift/
- (Android) Java SE 7 - http://www.oracle.com/technetwork/java/javase/overview/index.html

3.1.4 Communications Interfaces

Every communication between application server and client must comply the HTTPS protocol.

If the back-end application and the DBMS runs on different servers the communication between them must be SSL/TLS encrypted.

3.2 Functional requirements

3.2.1 Register

Purpose
Anyone who want to subscribe to Travlendar + can make the registration process through the mobile app or the web site. The guest must first complete the registration form with personal data (i.e., first name, last name, birth, sex, e-mail, password) and accept terms on data protection and conditions.

Once the information are submitted, the system checks them and sends a confirmation email. If the operation is successful, the guest becomes a user and may start using the application.

Scenario 1
Allyson wants to register Travlendar + using the web page. She opens the home page of a search engine and types "travlendar +". When you reach home page of Travlendar, click on Sign up and the registration form appears. She types in her data, such as first name, last name, birthdate, sex, email and password, and accepts the terms of data protection. At the end she clicks on the "Done" button, and after a few seconds she will receives an email confirming her registration.

Scenario 2
Benjamin has just moved to London and his friend Carl has spoken to him about Travlendar +, so he decides to sign up in order to manage better his new meetings. He downloads the app on his mobile phone and opens it. The home page appears and he clicks on "Sign up". The registration form appears. He inserts types in his personal data and clicks on Done. Immediately after his click, a message informs him that his password has been already used.

Use cases

Table 3.1: Register use case.

Actor	Guest
Goal	Goal 1
Input Condition	The person wants to subscribe to the service
Event flow	1. The guest opens the home page or mobile application of "Travlendar +" and clicks the "Sign up" button; 2. The registration form appears and the guest completes the mandatory fields; 3. The guest allows the personal data processing to be marked on the corresponding box; 4. The guest clicks on "Done" button; 5. The system saves the information in the database and sends an email to the guest confirming his/her subscription and that he/she has become a user of the service.
Output Condition	The system informs the user that he/she has registred successfully.
Exception	The system can report alert messages when a non valid or already used mail address is typed in. Another exception can be reported if the license number does not exist or when the terms and conditions of use are not accepted.

Functional requirements

- The system must not accept an email address already used during the registration procedure.
- The system requires the following mandatory personal information:
  - First name
  - Last name
  - Sex
  - Date of birth
  - Address
  - E-mail address
  - Password
  - Driving license number (optional)
- If the terms and conditions of use and data protection are not accepted, the system must not complete the registration procedure.
- As soon as the guest clicks on "Done", the system sends automatically an email of confirmation.
- The guest can leave the registration procedure at any time.

Activity diagram

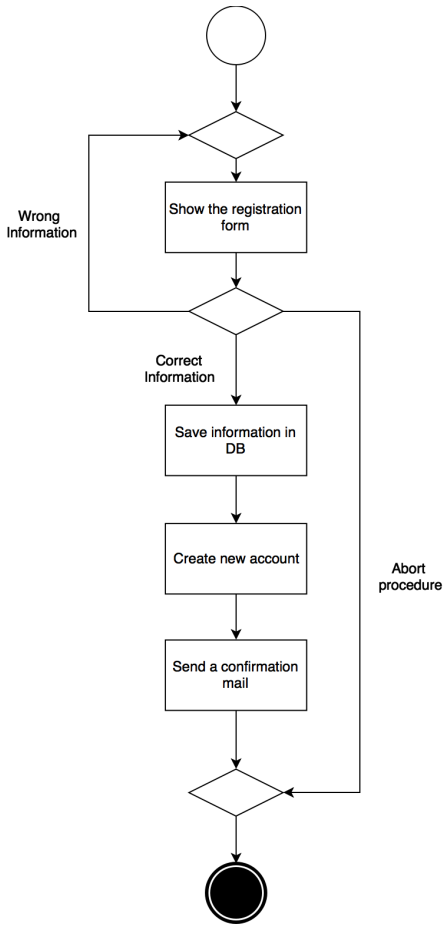

Figure 1: Activity diagram for Register.

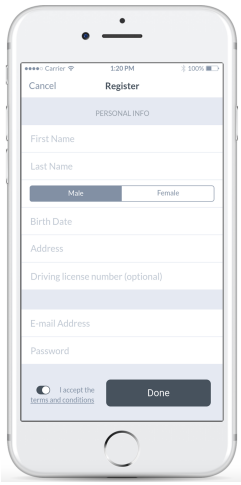

Figure 2: Mockup for registration on a mobile device.

3.2.2 Login

Purpose
The Login phase aims to guarantee the access to authorized users. Access can be made by inserting mail and password, or via a social network account (i.e., *Facebook*, *Instagram*, *Linkedin*).

If the user forgets his/her password, it is also possible to retrieve it clicking on "Forgot Password?". Once the button is clicked, an email with credentials is sent.

Scenario 1
Catherine wants to enter a new meeting in her calendar. She accesses the "Travlendar +" home page on her laptop entering her personal email and password (registered when she signed up). She clicks then on "Sign in" and gets in as authorized user because everything is correct.

Scenario 2
Dominique wants to insert his check up to the dentist on next Tuesday. He opens the app on his mobile phone and types his credentials. However there is a problem now: he can't remember the password used when authenticated himself. Dominique clicks therefore on "Forgot password?" button. The system forwards to the email address recorded a new password.

Use cases

Table 3.2: Login use case.

Actor	User
Goal	Goal 1
Input Condition	A user already registered, wants to login.
Event flow	1. The user opens the home page or the mobile application and the system shows the login page; 2. The user inserts his/her email address and password; 3. The user clicks on "Sign in" button.
Output Condition	The system connects the user to his / her personal page.
Exception	The exception that may arise is the inputting of an insertion mail or an incorrect password. The system sends the user an error message asking him/her to verify the address and/or the password re-enter them.

Functional requirements

- The user must have been already registered with the system for successful login.
- The user must have correct email and password available to successfully login.
- The user must use a password connected to a specific email address.
- Bad credentials must not allow the user authentication.
- If the user clicks on "Forgot password?" the system must send a link where the user can reset his/her password.
- When a new password has been set, the system must allow the user to authenticate only with new credentials.

Activity diagram

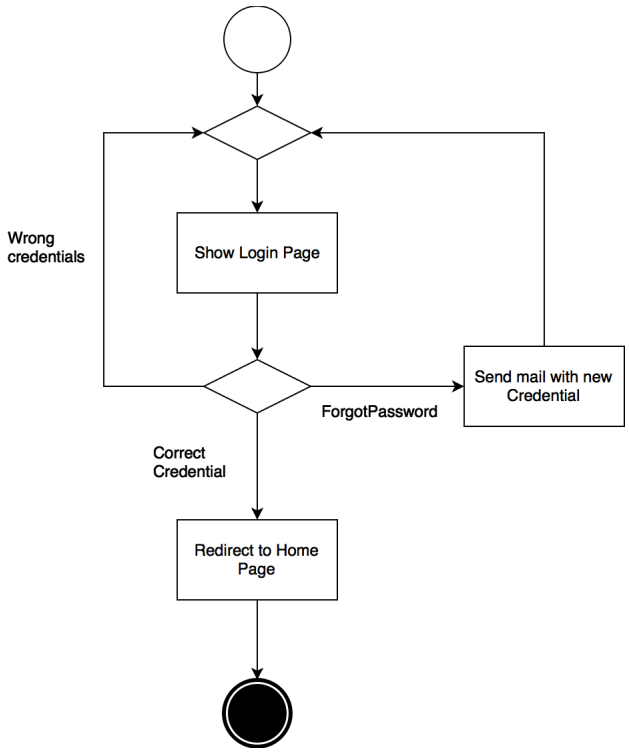

Figure 3: Activity diagram for Login.

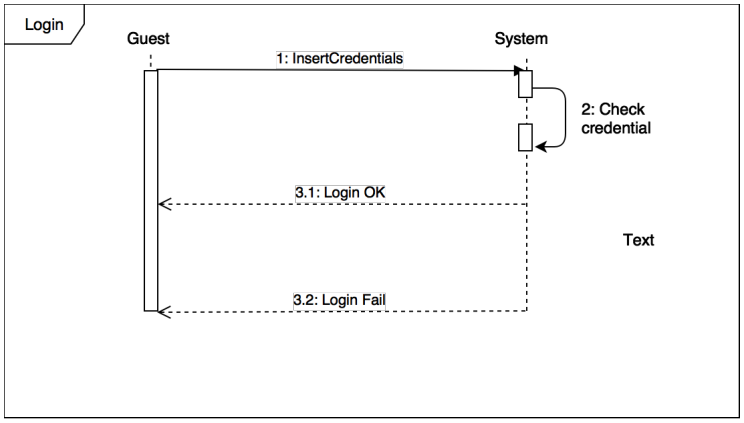

Figure 4: Login sequence diagram.

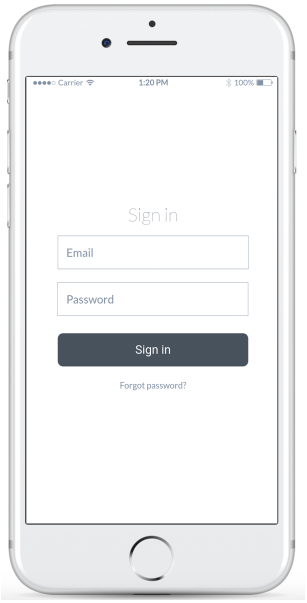

Figure 5: Mockup for Login on a mobile device.

3.2.3 Create meetings

Purpose
The purpose is to allow the authorized user to create a new meeting on the calendar by entering the meeting name, date, time, place, and also specifying how to reach the site. The task of the system is to check this information and, if incorrect, alert the user to modify them.

Scenario 1
Emma has to enter a driving lesson for tomorrow. Then she logs in her "Travlendar +" personal page as authorized user and press the "New event" button to create the appointment. Then the system open a form where she must enter the name of the event, date, time and place. Once completed, she press "Save" button. The system verifies the correctness of the information, calculates the most appropriate way to reach it and creates the event on the calendar.

Scenario 2
Franklin just took an appointment from the hairdresser in half an hour. He has decided afterward to create an appointment on her mobile app "Travelendar +". Log in with your smartphone and press the "+" button. A form appears to be completed with the event, date, time and place. Once the form is completed, submit it. The system checks the information and issues an error message because the site can not be reached in half an hour by bike (Franklins only active travel mean is bike sharing service).

Use cases

Table 3.3: Create meetings use case table.

Actor	User
Goal	Goal 4
Input Condition	An user wants to create a new meeting on his/her calendar.
Event flow	1. The user must login by entering his/her mail and password; 2. The user clicks the "+" / "New event" button to create a new meeting; 3. The system shows the form to be filled; 4. The user fulfills the form and clicks on "Save"; 5. The system checks the information, calculates the most appropriate way to reach the meeting's location and saves them in the database; in the case of incorrect information, the user alerts you with an error message.
Output Condition	The system creates a new meeting on the user's calendar in the home page.
Exception	The exception might be generated when the mean to reach the site is not valid (e.g., if you have to drive on a motorway, you cannot use the bicycle), or your downtime is less than the time you need. Another exception is raised if the date creating the meeting is later than the date of the meeting.

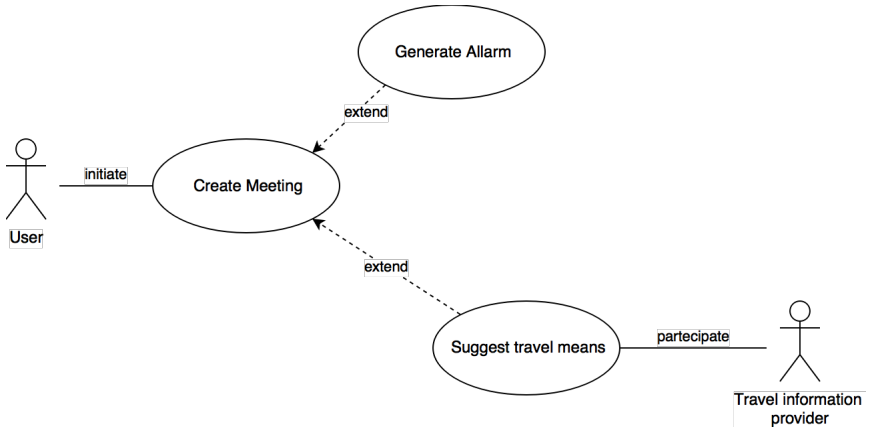

Figure 6: Create meeting Use case diagram.

Functional requirements

- The user must log in successfully;
- The user must insert the following information in the form:
  - Name
  - Start time
  - End time
  - Meeting location
  - People number
- The system must verify that the meeting's date is later than the creation date;
- The system must check that the way to reach is compatible with the path;
- The system must verify that the time available is sufficient to reach the appointment site;
- After verification of the correctness of the information, the system must create the meeting on the personal calendar;
- The system must send an error message in case of inappropriate data or mean.

Activity diagram

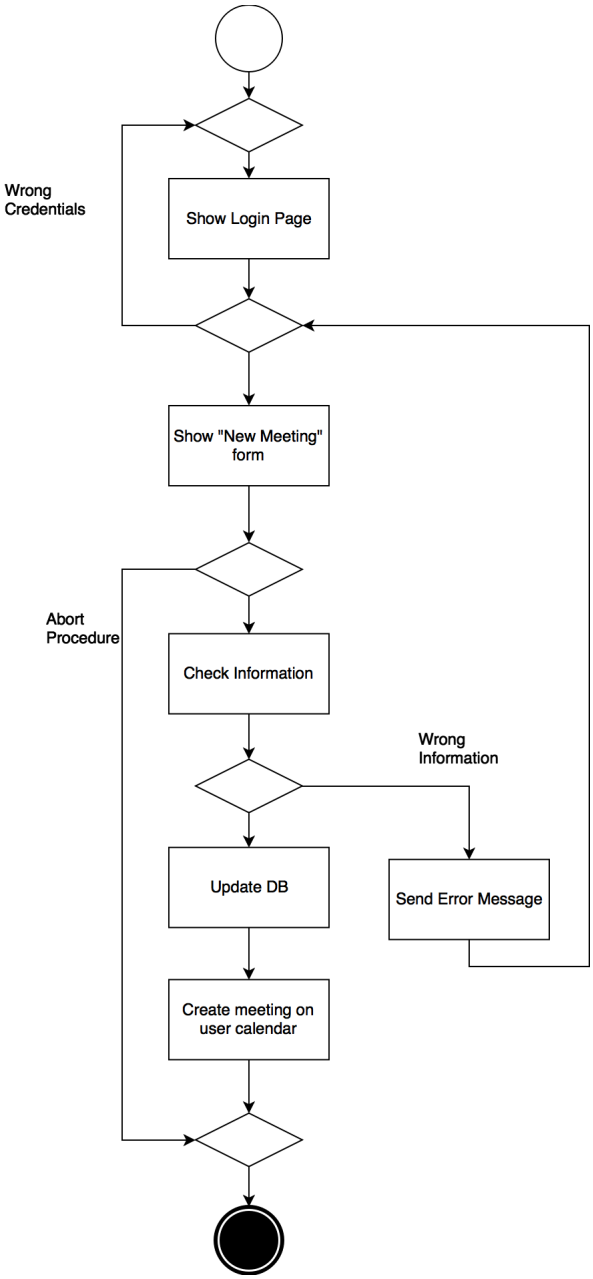

Figure 7: Activity diagram for Create meetings.

3.2.4 Modify meetings

Purpose
The purpose of this part is to allow the user, after saving a meeting on the calendar, to be able to modify some information (such as date, time, number of people, place, means of transport). The system must verify the compatibility of the modification.

Scenario 1

Gordon was just called by the school secretary and was advised that he had a lesson in IIIA class at 10 a.m. instead of at 9 a.m. on Thursday. Enter the personal page of "Travlendar +" and select the appointment. Press the "Modify" button next to the selected event. The system shows the form already completed and Gordon changes the time. Press "Save" button. The system checks the information and updates successfully the appointment.

Scenario 2
Helen has to hold a conference in Paris. Now she is at Milan Malpensa Airport and she has just been notified that the plane has 120 minutes delay. He decides to change the meeting to control if she can reach the conference place in time. Enter your personal page through the mobile application. Helen selects the meeting and presses the "Modify" button. Then, she changes the times in the form and presses "Save". The system verifies the information and sends an error message because changing the time the plane can't reach Paris in time for the conference.

Use cases

Table 3.4: Modify meetings use case.

Actor	User
Goal	Goal 5
Input Condition	The user wants to modify a meeting on his/her calendar.
Event flow	1. The user must login entering his/her email and password; 2. The user selects the meeting he/she wants to modify with one click; 3. The system shows a dialog box with "Modify" and "Delete" buttons; 4. The user presses the "Modify" button; 5. The system shows the form with the meeting information; 6. The user modifies the form and presses the "Save" button; 7. The system verifies the information and saves it in the database; in the case of incorrect information, the system alerts the user with an error message.
Output Condition	The system modifies a meeting on the calendar on the user's home.
Exception	The exceptions that can be raised are the same as those for creating the event. The exception that may be raised is that the way to reach the site is not valid (for example, if you have to drive a motorway, you can not use the bicycle), or your downtime is less than the time you need. Another exception is raised if the meeting change date is later than the date of the meeting.

Functional requirements

- The user must log in successfully;
- The user must have already created an event;
- The user must change one or more of the following information in the form:
  - Name
  - Start time
  - End time
  - Meeting location
  - People number
- The system must verify that the date of the event is after the creation date;
- The system must check that the way to reach is compatible with the path;
- The system must verify that the time range is sufficient to reach the appointment place;
- Once the information is verified, the system has to change the appointment on the personal calendar;
- The system must send an error message in case of inappropriate date or time.

Activity diagram

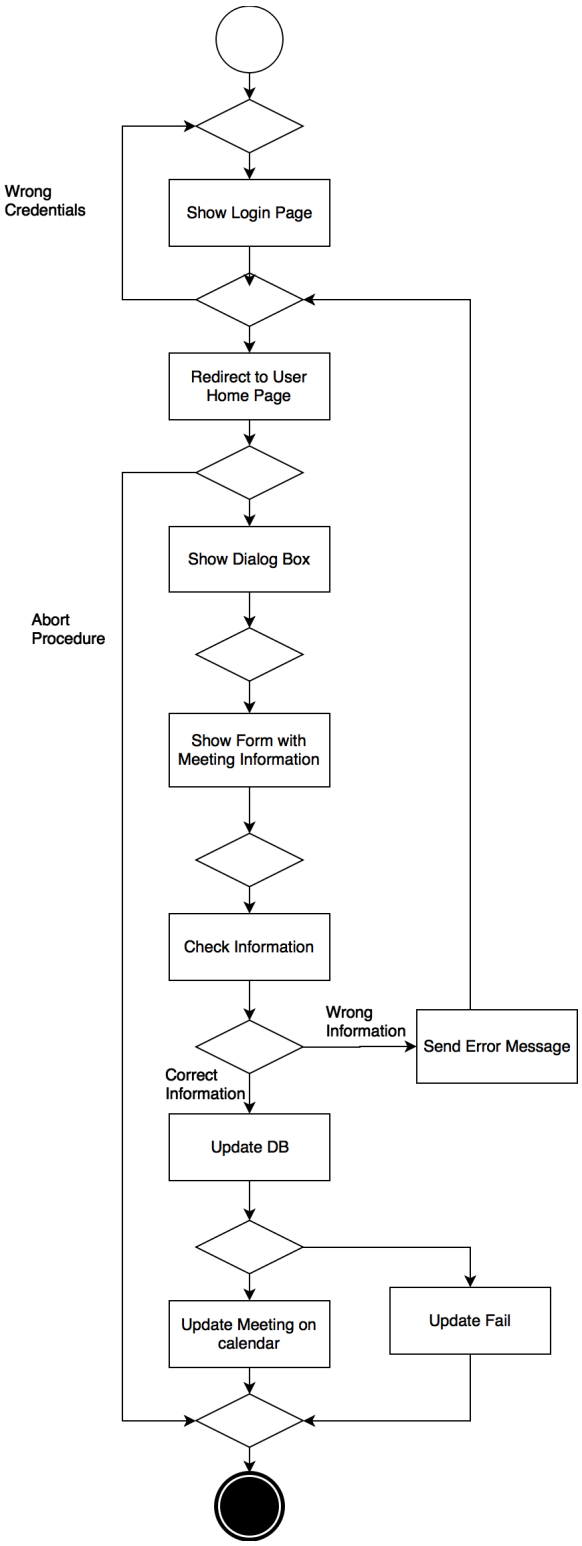

Figure 8: Activity diagram for Modify meetings.

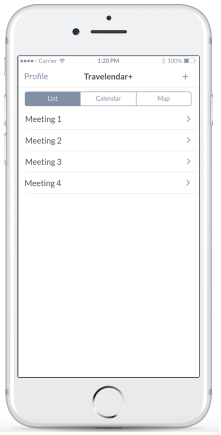

Figure 9: Meetings list view on mobile.

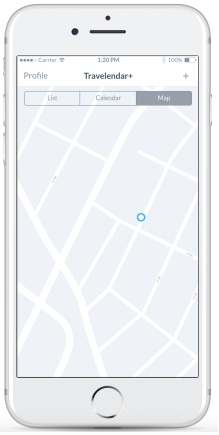

Figure 10: Meetings map view on mobile.

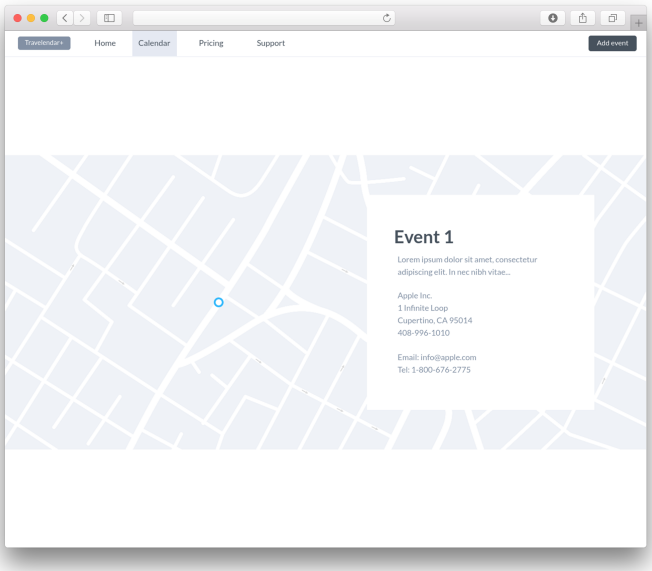

Figure 11: Mockup for meetings map view on a browser.

3.2.5 Delete meetings

Purpose
The purpose of this part is to allow the user, after creating or modifying an meeting on the calendar, to delete it.

Scenario 1
Igor is a computer science student. He attends on Monday and Tuesday the course Database 2. The professor has just written an email, signaling that on next Monday there will be no lesson. Igor opens the "Travlendar +" application on his smartphone and he logs in. He selects the "DB2 lesson" meeting and clicks on Delete button. The system asks whether he is sure to cancel the meeting. Igor confirms. The system processes the information and updates the database.

Use cases

Table 3.5: Delete meetings use case.

Actor	User
Goal	Goal 6
Input Condition	The user wants to delete a meeting on his/her calendar.
Event flow	1. The user must login entering your email and password; 2. The user selects the meeting he/she wants to select with one click; 3. The system shows a dialog box with two buttons: "Modify" and "Delete"; 4. The user clicks on "Delete" button; 5. The system prompts the user if he/she is sure and wants to delete the meeting with a dialog box; 6. The user presses on "Yes" button; 7. The system processes the information, updates the database, and deletes the event from the calendar.
Output Condition	The system deletes a meeting on the user's home calendar.
Exception	The delete fails and the user is notified.

Functional requirements

- The user must log in successfully;
- The user must have already created a meeting;
- The user must select the meeting he/she wants to delete;
- The system must ask the user whether or not he/she wants to delete the meeting;
- The system must update the DB and delete the meeting from the calendar.

Activity diagram

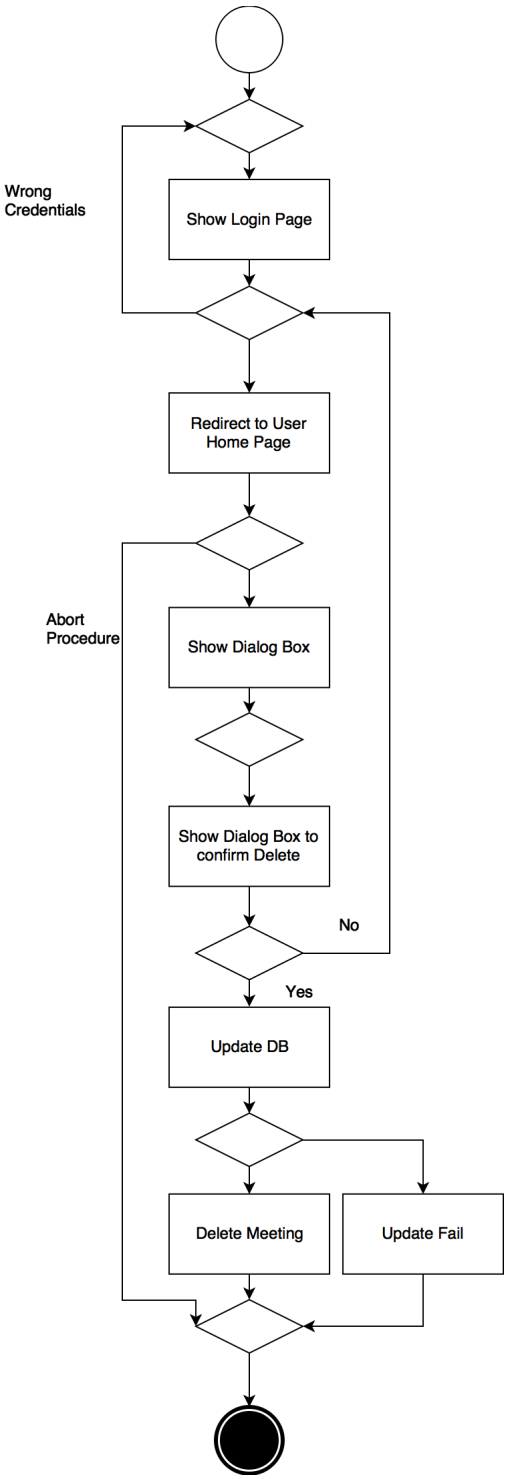

Figure 12: Activity diagram for Delete meetings.

3.2.6 Manage profile informations

Purpose
Both web and mobile applications must allow the user to view and update some personal informations coherently with the provided constraints.

Scenario 1
Karlie, due to a problem with her e-mail services provider, wants to change her email address. Karlie opens a browser window, reaches the Travelendar+ homepage and logs in. Then she clicks on her profile picture in the right of the navigation bar, a dropdown menu appears and clicking "Edit Profile" she reaches a page showing her profile information. She changes her email address and clicks then on "Save Changes" button. The system notifies Karlie that her email address has been correctly updated.

Use cases

Table 3.6: Manage profile informations use case.

Actor	User
Goal	Goal 2
Input Condition	A user wants to view or update his/her personal informations.
Event flow	1. The user opens a web browser page or the mobile application and, if he/she has not already done it, authenticates to the service; 2. The user reaches his/her profile settings page; 3. The user updates his/her personal information.
Output Condition	Information concerning the user is successfully updated and the user is notified.
Exception	The update fails and the user is notified.

Functional requirements

- The user must be already logged in;
- The system must display to the user his/her own personal informations;
- The system must allow the user to change any information provided during the registration phase, only if the new piece of information does not get in conflict with the registration constrains;
- Either a modification succeeds or it fails, the user must be notified.

Activity diagram

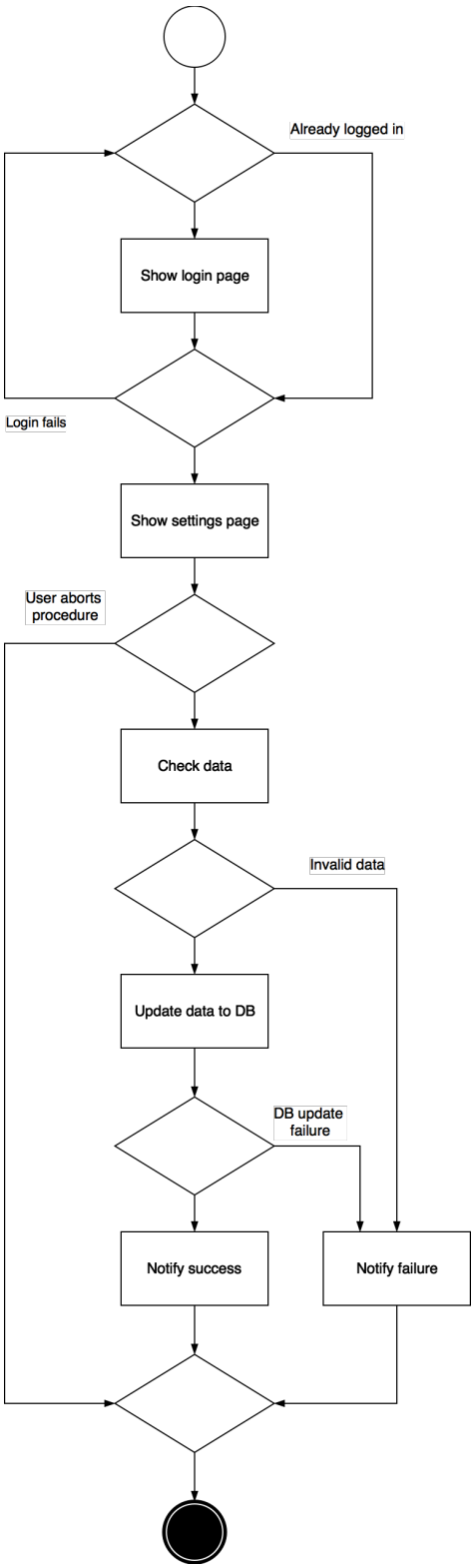

Figure 13: Activity diagram for Manage profile informations.

3.2.7 Delete profile

Purpose
Both web and mobile applications must allow the user to delete his/her account if does no longer want to use the service.

Scenario 1
Luca decides that he no longer wants to use Travelendar+ and wants to delete his account. Luca opens the mobile application and clicks the profile button on the navigation bar, the profile informations page is now displayed. He clicks on the red button "Delete Account". As soon as and after he confirms his intentions, Luca is no longer a Travelendar+ registered user and his data are no longer stored in the system.

Use cases

Table 3.7: Delete profile use case.

Actor	User
Goal	Goal 3
Input Condition	An user wants to delete his/her account.
Event flow	1. The user opens a web browser page or the mobile application and, if he/she has not already done it, authenticates to the service; 2. The user reaches his/her profile settings page; 3. The user clicks on "Delete account" button; 4. The user confirms his/her intent to delete the account.
Output Condition	The users account is deleted, his/her data are erased from the system and of course he/she can no longer login with his/her credentials, unless he/she registers into the service again.
Exception	The account's deletion fails and the user is notified.

Functional requirements

- The user must be already logged in;
- The system must allow the user to delete his/her profile;
- The user must confirm his/her intention to delete his/her profile;
- Once the user's profile has been deleted, the system must no longer store the user's personal informations.
- Once the users profile has been deleted, the user can no longer login with his/her credentials unless he/she re-register to the service;
- Either a modification succeeds or it fails, the user must be notified.

Activity diagram

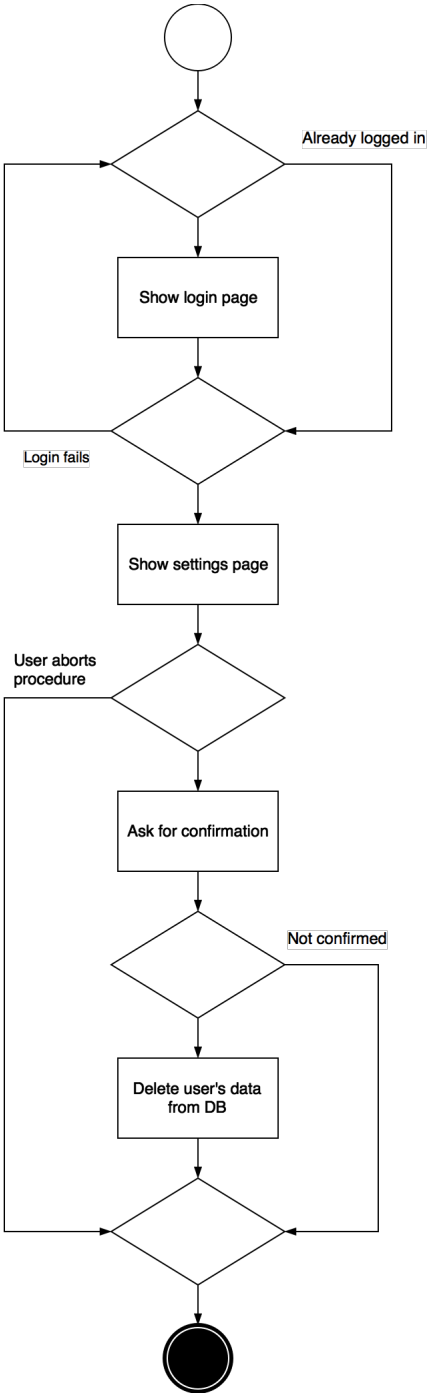

Figure 14: Activity diagram for Delete profile.

3.2.8 Activate travel mean(s)

Purpose
Both web and mobile applications must allow the user to specify his/her intention to use a specific travel mean to move from an event to another.

Scenario 1
Its almost summer. Mario decides that the weather is good enough to allow him to use the bike. Mario opens the Travelendar+ app on his Android smartphone and navigates to the settings page, scrolls down until he reaches "Bike" in the travel menas list and toggles the checkbox near to it. From now on the system will also suggest him the bike as an appropriate mean of transport, if it thinks it is appropriate.

Use cases

Table 3.8: Activate travel mean(s) use case.

Actor	User
Goal	Goal 9
Input Condition	The user wants to activate a travel mean.
Event flow	1. The user opens a web browser page or the mobile application and, if he/she has not already done it, authenticates to the service; 2. The user reaches his/her profile settings page; 3. The user checks the checkbox corresponding to the travel mean he/she wants to activate.
Output Condition	The travel mean(s) is/are now activated as desired; the user is notified about it.
Exception	The travel mean(s) activation fails and the user is notified.

Functional requirements

- The user must be already logged in;
- The system must display to the user which travel mean is already selected;
- The system must allow the user to select a travel mean if it was not already selected;
- If a new travel mean has been selected, from now on it must the system take it into consideration when it calculates the more convenient travel between the two locations;
- Either a modification succeeds or it fails, the user must be notified.

Activity diagram

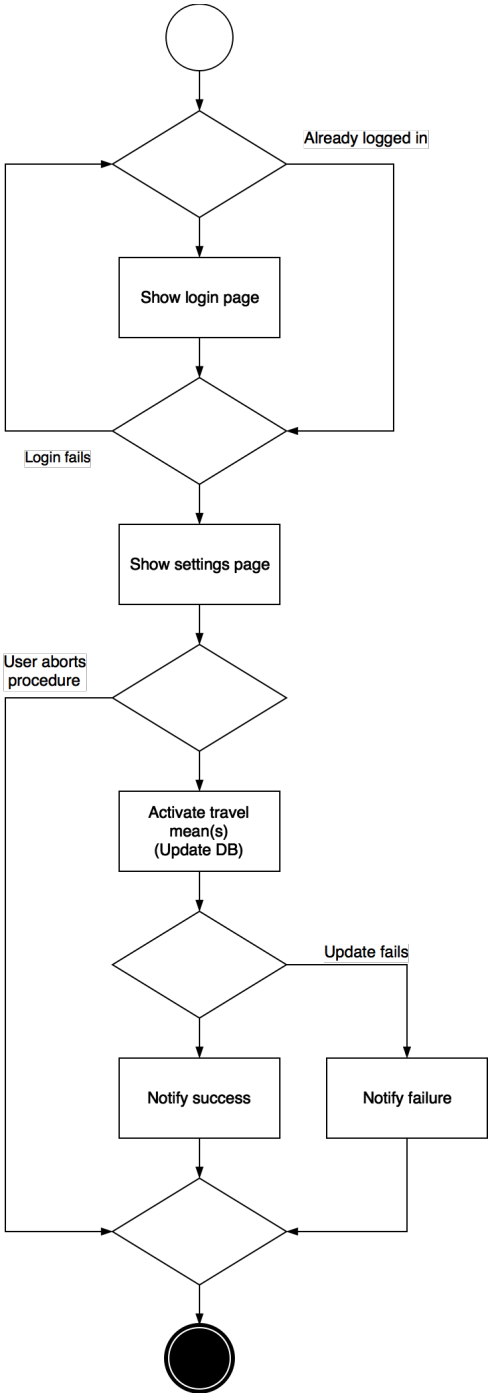

Figure 15: Activity diagram for Activate travel mean(s).

3.2.9 Deactivate travel mean(s)

Purpose
Both web and mobile applications must allow the user to specify his/her intention to do not use a specific travel mean to move from an event to another.

Scenario 1
Otis, after breaking his leg, realizes that the only suitable means for his travels are taxi and Uber. Otis opens a browser window, reaches the Travelendar+ homepage and logs in. Then he clicks on his profile picture in the right of navigation bar, a dropdown menu appears and clicking "Edit Profile" he reaches a page showing his profile information and unchecks all the checkboxes near to the travel means excepted, either "Taxi" or "Uber". From now on the system could only rely on one other travel mean, either Taxi or Uber.

Use cases

Table 3.9: Deactivate travel mean(s) use case.

Actor	User
Goal	Goal 9
Input Condition	The user wants to deactivate a travel mean.
Event flow	1. The user opens a web browser page or the mobile application and, if he/she has not already done, authenticates to the service; 2. The user reaches his/her profile settings page; 3. The user unchecks the checkbox corresponding to the travel mean he/she wants to deactivate.
Output Condition	The travel mean(s) is/are now deactivated as desired and the user is notified.
Exception	The travel mean(s) deactivation fails and the user is notified.

Functional requirements

- The user must be already logged in;
- The system must display the user travel mean already selected;
- The system must allow the user to deselect a travel mean if it was not previously selected;
- If a travel mean has been deselected, from now on it can not be taken in consideration when the system calculates the optimal travel between two locations;
- Either a modification succeeds or it fails, the user must be notified.

Activity diagram

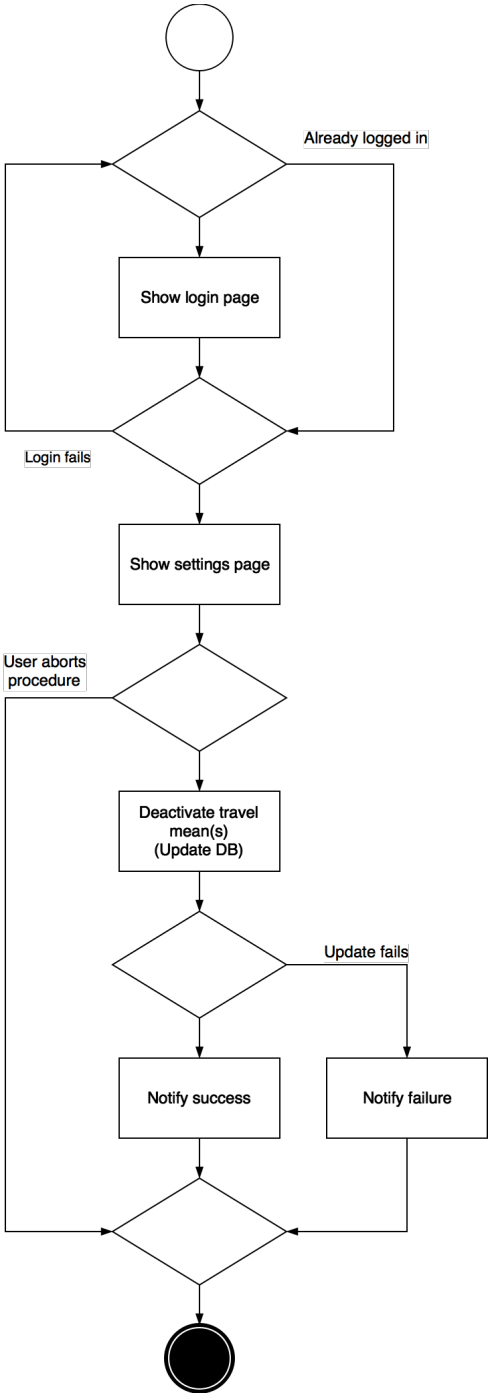

Figure 16: Activity diagram for Deactivate travel mean(s).

3.2.10 Provide constraints

Purpose
Both web and mobile applications must allow the user to select a specific travel mean for a specific travel.

Scenario 1
Quentin has a meeting scheduled for tomorrow morning at 9:00AM not far away from his house. The system suggests him to take the bus, but Quentin needs to take to the meeting a cumbersome model of the building he will present to his boss. Due to his particular need he decides that the most appropriate transport mean is a taxi. He opens the Travelendar+ app on his iPhone. In the page detailing events, he selects the taxi as the preferred mean of transport and the system re-calculates the travel time and eventually reserves a vehicle for the next morning.

Use cases

Table 3.10: Provide constraints use case.

Actor	User
Goal	Goal 8
Input Condition	The user wants to select a specific travel mean for a specific travel.
Event flow	1. The user opens a web browser page or the mobile application and, if he/she has not already done it, authenticates to the service; 2. The user opens the detail page of a certain event; 3. The user selects a certain travel mean, the one he/she prefers for travelling.
Output Condition	The change succeeded and the mean for that travel is updated.
Exception	The change fails and the user is notified.

Functional requirements

- The user must be already logged in;
- The system must allow the user to use a specific travel mean to reach a given event (only if the mean can reach the event's location);
- Either a modification succeeds or it fails, the user must be notified.

Activity diagram

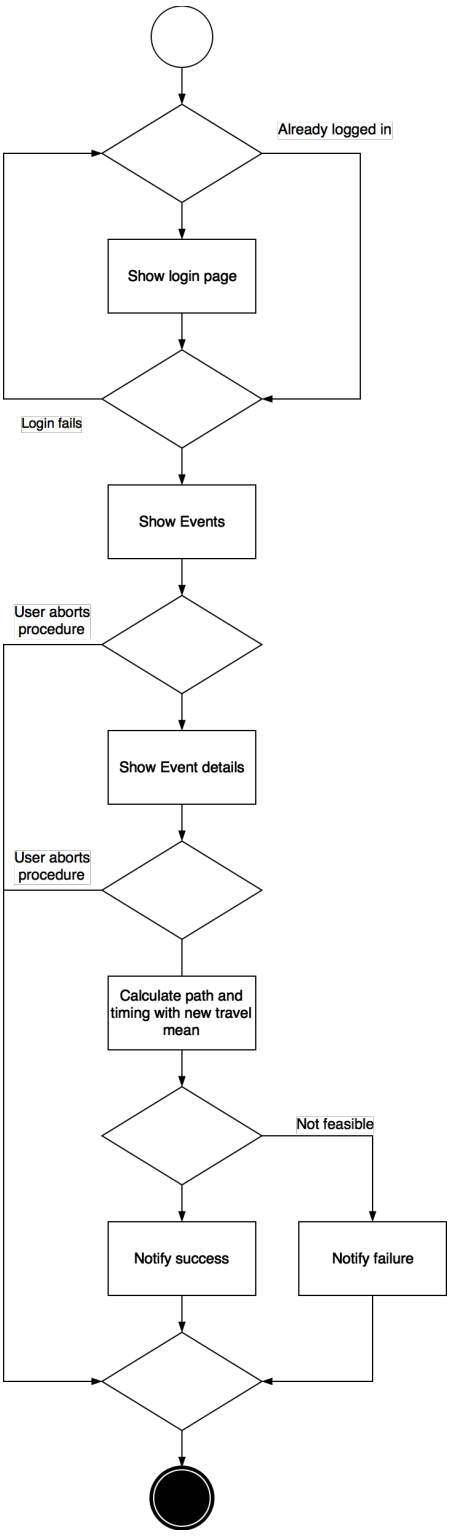

Figure 17: Activity diagram for Provide constraints.

3.2.11 Minimize carbon footprint

Purpose
Both web and mobile applications must allow the user to specify his/her intention to move from an appointment to another with the less polluting mean(s).

Scenario 1
Serena, while she was watching a National Geographic documentary, realized that climate change is a real thing and everyone should do his best to limit pollution. For this reason she wants to minimize her carbon footprint during her travels. Serena opens a browser window, reaches the Travelendar+ homepage and logs in. Then she clicks on her profile picture in the right of navigation bar, a dropdown menu appears and by clicking "Edit Profile" she reaches a page showing her profile information and toggles the checkbox near to "Minimize carbon footprint". From now on the suggested travel mean is the less polluting one.

Use cases

Table 3.11: Minimize carbon footprint use case.

Actor	User
Goal	Goal 12
Input Condition	The user wants to move from an appointment to another with the less polluting mean(s).
Event flow	1. The user opens a web browser page or the mobile application and, if he/she has not already done it, authenticates to the service; 2. The user reaches his/her profile settings page; 3. The user checks the checkbox corresponding to "Minimize carbon footprint".
Output Condition	The preference update succeeds and the user is notified.
Exception	The preference update fails and the user is notified.

Functional requirements

- The user must be already logged in;
- The system must allow the user to express his/her intention to minimize his/her carbon footprint or not;
- If the user has expressed the intention to minimize his/her carbon footprint, the suggested travel mean must be the less polluting one;
- Either a modification succeeds or it fails, the user must be notified.

Activity diagram

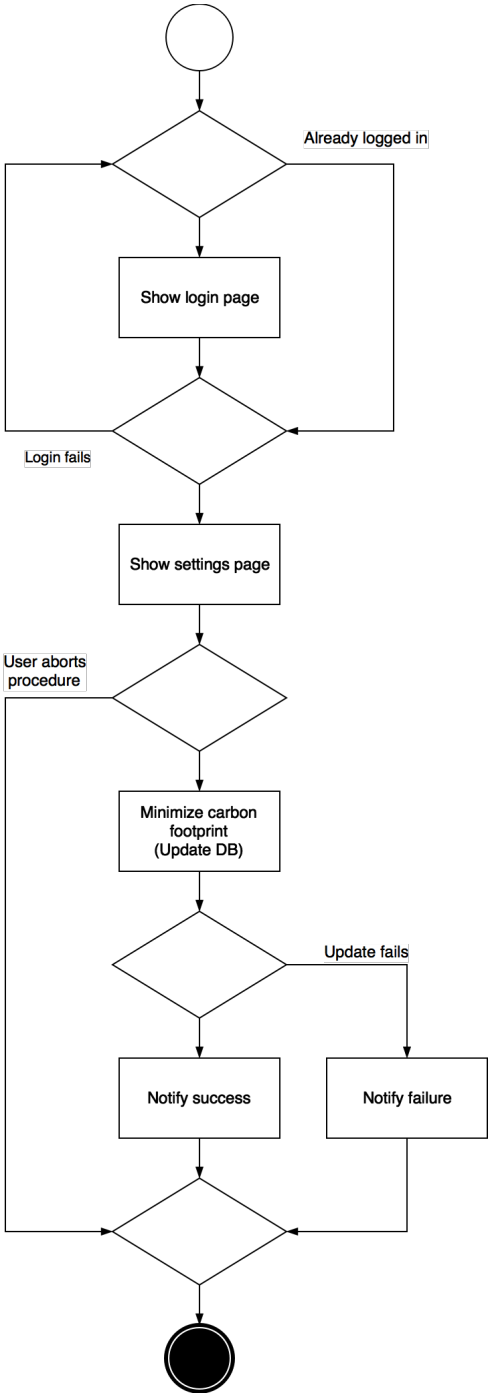

Figure 18: Activity diagram for Minimize carbon footprint.

3.2.12 Specify break timing

Purpose
The purpose is to allow the user to enter a time interval where the system must ensure at least half an hour for the break, based on the present appointments.

Scenario 1
Ubald is a university professor and on Wednesday he has two hours lesson in the morning (10 a.m. – 12 a.m.) and two hours lesson in the afternoon (1 p.m. – 3 p.m.). Ubald connects to the "Travlendar +" home page and selects the "BreakTime" entry. The system shows you a form to be filled in with the date and time interval for the break. Ubald inserts the break from 12.30 to 15.30 and submits it by pressing on "Save". The system verifies availability and saves it successfully in the database.

Scenario 2
Vichy is a manager who has to go to Naples for a meeting at 3:15 p.m.. She bought a train ticket from Milan at 10.30 a.m.. The journey lasts 270 minutes. Vichy opens the mobile application of "Travlendar +" and selects the entry "BreakTime". The system shows a form to be filled in with the date and duration for the break. Vichy inserts as interval 1 p.m. – 3 p.m. and submits it pressing on"Save" button. The system checks the information and sends an error message.

Use cases

Table 3.12: Specify lunch timing use case.

Actor	User
Goal	Goal 13 and Goal 14
Input Condition	The user specifies a time interval for break.
Event flow	1. The user must login entering his/her email and password.; 2. The user selects "Break Time" button; 3. The system opens a form to be completed indicating date and duration of the interval; 4. The user inserts information regarding the date and the duration of break; 5. The user saves the information pressing on "Save" button; 6. The system processes the information, updates the database and notifies the user that the operation has been successfully concluded (specifying the start and end time for break). If not, it generates an error message.
Output Condition	The system selects within the time interval, time of specify duration for the break.
Exception	The system issues an error message if there is no duration time for the break during the specified time interval.

Functional requirements

- The user must log in successfully.
- The user must enter information in the form:

  - Start at least
  - End at most
  - Duration
- The system must verify that the date is behind the insertion date.
- The system must check that the available time (duration time) is sufficient for the break.
- Once the information are verified, the system must update the database.
- The system must alert the user about the selected time interval.
- The system must send an error message if an incorrect date or time space is not available.

Activity diagram

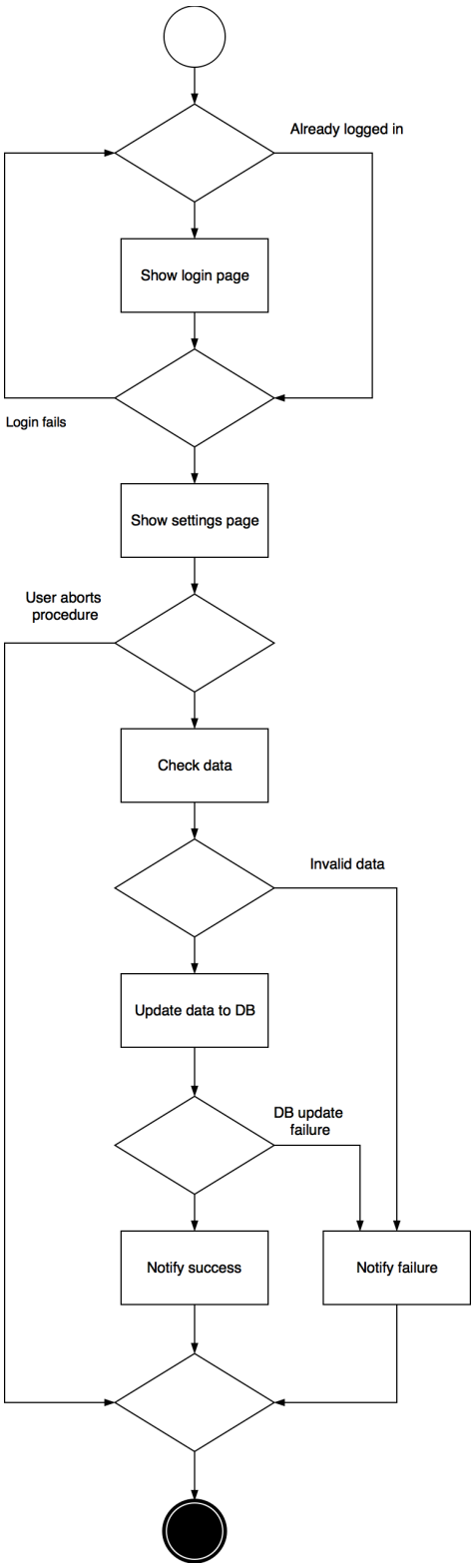

Figure 19: Activity diagram for Specify lunch timing.

3.3 Performance requiremens

In order to guarantee the best user experience, the following performance requirements must be satisfied by the system:

- The system must be operative 24 hours a day, 7 days a week with an uptime of at least 99%;
- There must not be an upper bound to the number of users registered to the system;
- There must not be an upper bound to the number of meetings that each user can insert;
- The system must handle at least 1000 logged users at the same time;
- 95% of the requests must be handled in at least 1 second; [1]
- 99% of the requests must be handled in at least 3 seconds. [1]

[1] *not taking in count the time spent to communicate with other actors (e.g., Traffic info provider).*

3.4 Software System Attributes

3.4.1 Reliability

Reliability can be expressed as a percentage on the total operations that succeeds, then we can write the following relation:

$$Reliability = 100\% - (probability\ of\ Failure * 100)$$

The system must have a reliability greater or equal than 99.9%, that means that failing the operations are less than 0.1%.

3.4.2 Availability

The system must be up at least 99% of the time with the exception of ordinary maintenance work.

3.4.3 Security

The security attribute depends on the following factors:

- Every communication between application server and client must comply the HTTPS protocol.
- Communication between different servers must be SSL/TLS encrypted.
- Sensitive informations (i.e. password) must be properly stored (i.e., key-hashed salted hash).

3.4.4 Mantainability

Code and the documentation must match in order to make the code easy to understand for future maintainers.

3.4.5 Portability

The back-end application must be written in Java EE7 in order to be able to run on every server that runs JEE7. The web application must support at least the main modern browser (e.g., IE, Firefox, Chrome, Safari). The mobile application must be developed both for IOS and Android.

3.4.6 Class Diagram

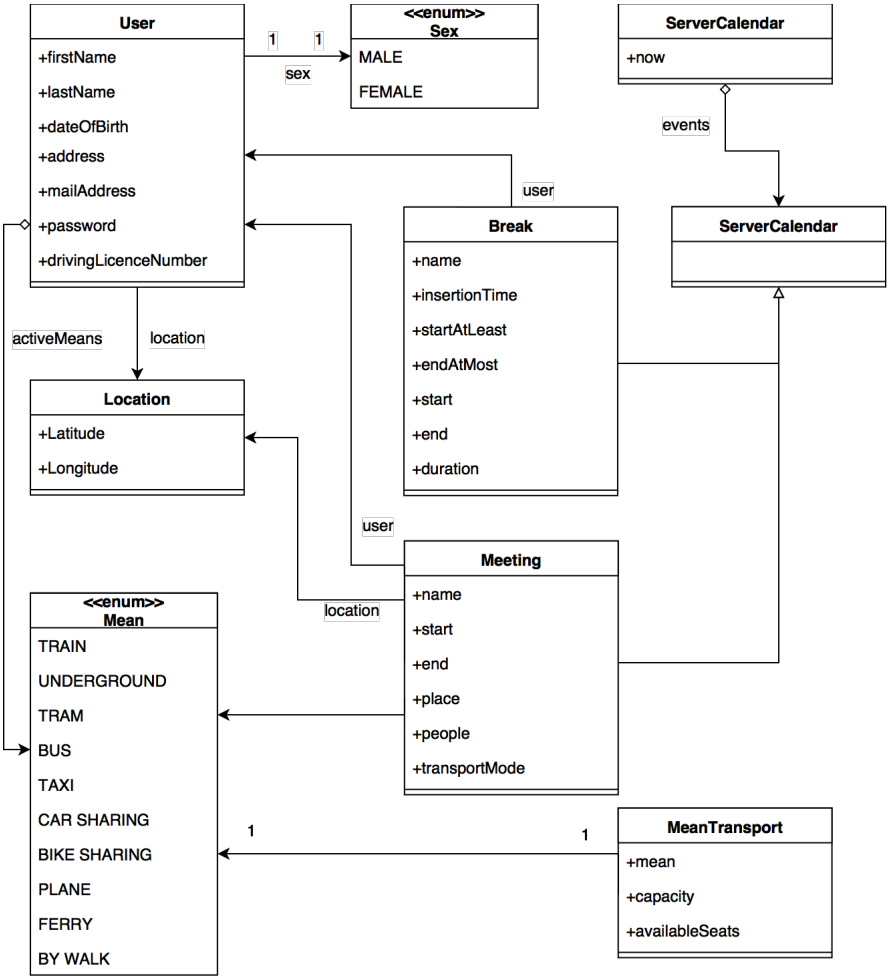

Figure 20: Class Diagram.

Section 4

Formal Analysis Using Alloy

```
1
2       //SIGNATURES
3
4       one sig ServerCalendar{
5           events: set Event,
6           now: one Int //current date
7       }
8       {
9           #events>=0
10          now     >=0
11      }
12
13      sig User{
14          firstName:              one StringLetter,
15          lastName:               one StringLetter,
16          dateOfBirth:            one Int,
17          sex:                    one Sex,
18          mailAddress:            one StringLetter,
19          password:               one StringLetter,
20          drivingLicenceNumber:   lone Int,
21          userMeans:              some Mean
22      }
23      {
24          drivingLicenceNumber > 0
25      }
26
27      abstract sig Event {}
28
29      sig Meeting extends Event{
30          user:           one User,
31          name:           one StringLetter,
32          insertionTime:  one Int,
33          start:          one Int,
34          end:            one Int,
35          place:          one Location,
36          people:         one Int
37      }
38      {
39          insertionTime>=0
40          start        >=0
41          end           >  0
42          end           >  start
43      }
44
45      //Supposing that duration is an integer attribute
46      sig Break extends Event{
47          user:           one User,
48          name:           one StringLetter,
49          insertionTime:  one Int, //date of insertion the break into the calendar
50          startAtLeast:   one Int,
51          endAtMost:      one Int,
52          start:          one Int,
53          end:            one Int,
54          duration:       one Int
55      }
56      {
57          insertionTime>=0
58          start         <  end
```

```
59          startAtLeast  <  endAtMost
60          start         >=0
61          end           >  0
62          startAtLeast  >=0
63          endAtMost     >  0
64          duration      >  0
65          duration      =  end - start
66          start         >=startAtLeast
67          end           <=endAtMost
68      }
69
70      abstract sig Mean{
71          capacity:        one Int,
72          available_seats: one Int
73      }
74      {
75          capacity>0
76          available_seats>=0
77          available_seats<=capacity
78      }
79
80      sig TRAIN        extends Mean {}
81      sig UNDERGROUND  extends Mean {}
82      sig TRAM         extends Mean {}
83      sig BUS          extends Mean {}
84      sig TAXI         extends Mean {}
85      sig CARSHARING   extends Mean {}
86      sig BIKESHARING  extends Mean {}
87      sig PLANE        extends Mean {}
88      sig FERRY        extends Mean {}
89      sig BYWALK       extends Mean {}
90
91      sig Location{
92          latitude:  one Int,
93          longitude: one Int
94      }
95      {
96          latitude  >=-90
97          latitude  <=90
98          longitude>=-180
99          longitude<=180
100     }
101
102     abstract sig Sex {}
103     one sig Male   extends Sex{}
104     one sig Female extends Sex{}
105
106     sig StringLetter{}
107
108
109     // PREDICATES
110
111     pred insertMeeting [s, s': ServerCalendar, m: Meeting] {
112         s'.events = s.events + m
113         m.insertionTime>=s.now
114         m.insertionTime<=s'.now
115     }
116
117     pred insertBreak [s, s': ServerCalendar,b: Break] {
118         s'.events = s.events + b
119         b.insertionTime>=s.now
120         b.insertionTime<=s'.now
121     }
122
123     pred removeMeeting [s, s': ServerCalendar, m: Meeting] {
124         s'.events = s.events - m
125     }
126
127     pred removeBreak [s, s': ServerCalendar, b: Break] {
128         s'.events= s.events - b
```

```
129     }
130
131     pred overlapsMM[m1, m2: Meeting] {
132         m1.user = m2.user implies
133         ((m1.start < m2.start and m1.end  >m2.start) or
134         (m1.start>=m2.start and m1.start < m2.end))
135     }
136
137     pred overlapsMB[m: Meeting, b: Break] {
138         m.user = b.user implies
139         ((m.start < b.start and m.end > b.start) or
140         (m.start>=b.start and m.start < b.end))
141     }
142
143     pred overlapsBB[b1, b2: Break] {
144         b1.user = b2.user implies
145         ((b1.start < b2.start and b1.end  >b2.start) or
146         (b1.start>=b2.start and b1.start < b2.end))
147     }
148
149
150     //FACTS
151
152     //No two distinct overlapping position
153     fact NoPositionOverlapping{
154         no disj pos1, pos2: Location|
155         (pos1.latitude = pos2.latitude and pos1.longitude = pos2.longitude)
156     }
157
158     //No two distinct coinciding users
159     fact UniqueUser{
160         no disj u1, u2: User|
161         (u1!=u2 and
162         (u1.mailAddress = u2.mailAddress or
163         u1.drivingLicenceNumber = u2.drivingLicenceNumber))
164     }
165
166     //No two distinct overlapping meeting
167     fact UniqueMeeting{
168         no disj m1, m2: Meeting|
169         (m1!=m2 and
170         (m1.user = m2.user or
171         m1.start = m2.start or
172         m1.end = m2.end))
173     }
174
175     //No two distinct overlapping break
176     fact UniqueBreak{
177         no disj b1, b2: Break|
178         (b1!=b2 and
179         (b1.user = b2.user or
180         b1.startAtLeast = b2.startAtLeast or
181         b1.endAtMost = b2. endAtMost ))
182     }
183
184     //No meeting overlapping meeting
185     fact NoMeetingOverlappingMeeting {
186         no disj m1, m2:Meeting | overlapsMM[m1, m2]
187     }
188
189     //No meeting overlapping break
190     fact NoMeetingOverlappingBreak {
191         no disj m:Meeting, b: Break | overlapsMB[m, b]
192     }
193
194     //No break overlapping break
195     fact NoBreakOverlappingBreak {
196         no disj b1, b2: Break | overlapsBB[b1, b2]
197     }
198
```

```
199     //time for break
200     fact TimeBreak{
201         all b:Break | ((b.endAtMost - b.startAtLeast)>=b.duration)
202     }
203
204     //date of meeting is after date of insertion
205     fact meetingMustBeInFuture{
206         all m: Meeting | (m.insertionTime<=m.start)
207     }
208
209     //date of meeting is after date of insertion
210     fact breakMustBeInFuture{
211         all b: Break | (b.insertionTime<=b.start)
212     }
213
214     //Server Calendar contains all breaks
215     fact containAllMeetings {
216         all m: Meeting, c: ServerCalendar | (m in c.events)
217     }
218
219     //Server Calendar contains all breaks
220     fact containAllBreaks {
221         all b: Break, c: ServerCalendar | (b in c.events)
222     }
223
224     //duration break
225     fact breakDuration{
226         all b: Break | (b.end - b.start = b.duration)
227     }
228
229     //The number of people must be lower than the capacity of mean
230     fact ContainPeople{
231         all m: Meeting, t: Mean | (m.people<=t.capacity)
232     }
233
234     //The number of people must be lower than the available seats
235     fact AvailablePeople{
236         all m:Meeting, t: Mean| (m.people<=t.available_seats)
237     }
238
239
240     //ASSERTIONS
241
242     //No overlps
243     assert noOverlapping {
244       no disj m1, m2: Meeting | overlapsMM[m1, m2]
245       no disj m: Meeting, b: Break | overlapsMB[m, b]
246       no disj b1, b2: Break | overlapsBB[b1, b2]
247     }
248     //Available seats imply higher capacity of the means of transport
249     assert available {
250         all m:Meeting, t: Mean | ((m.people<=t.available_seats) implies (m.people<=t.
            ↪ capacity))
251     }
252
253     //Duration of break is equal to difference between end and start time
254     assert breakBuration{
255         all b: Break | (b.duration = (b.end - b.start))
256     }
257
258     pred show() {}
259
260     run show for 15
```

Here is the execution of checks on the assertions:

```
Executing "Check noOverlapping for 25"
   Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
   408265 vars. 13116 primary vars. 1170897 clauses. 9215ms.
   No counterexample found. Assertion may be valid. 146ms.

Executing "Check available for 25"
   Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
   408796 vars. 13166 primary vars. 921430 clauses. 4645ms.
   No counterexample found. Assertion may be valid. 1133ms.

Executing "Check reakBuration for 25"
   Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
   408224 vars. 13141 primary vars. 920158 clauses. 3455ms.
   No counterexample found. Assertion may be valid. 387ms.
```

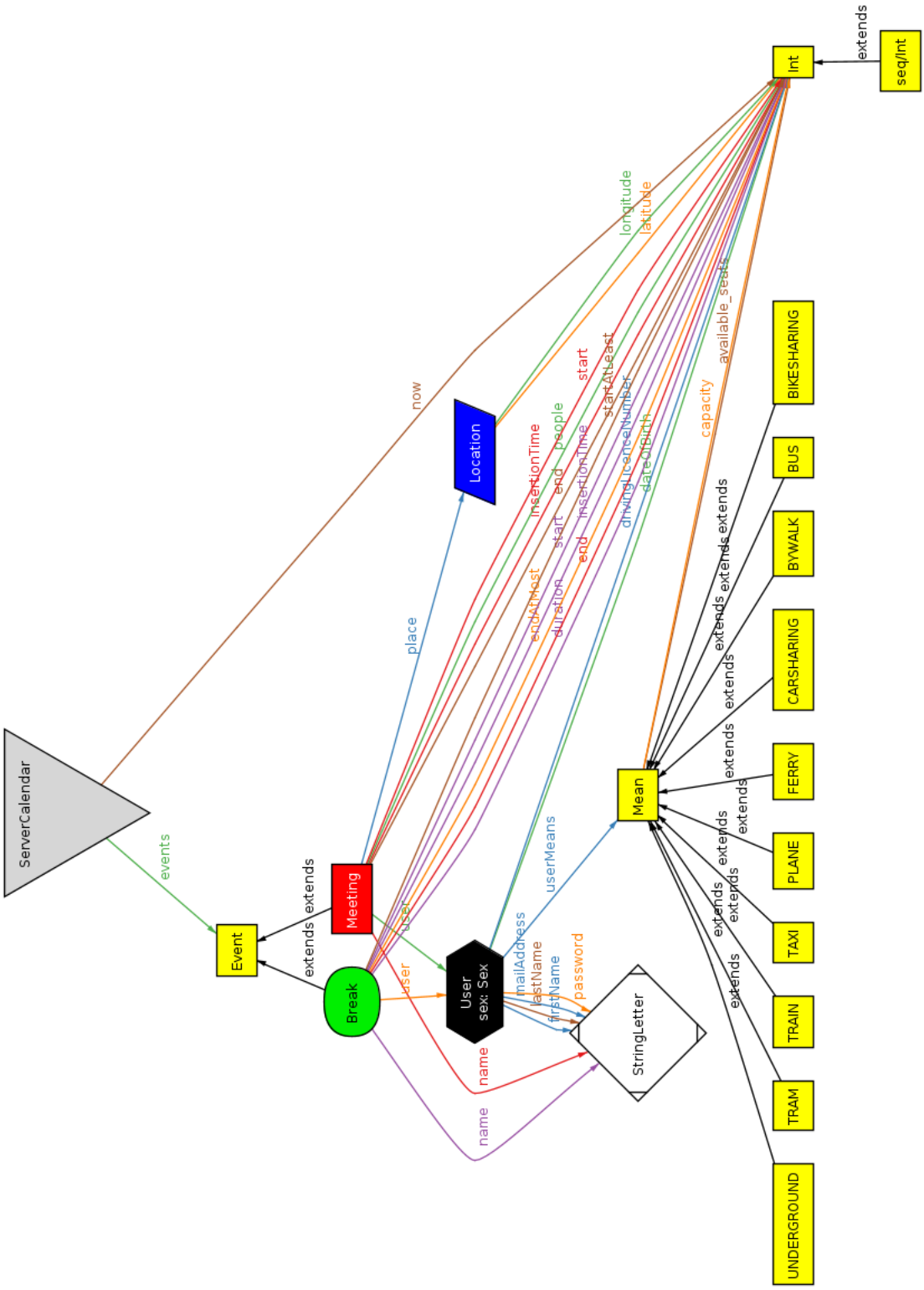

Figure 21: Metamodel.

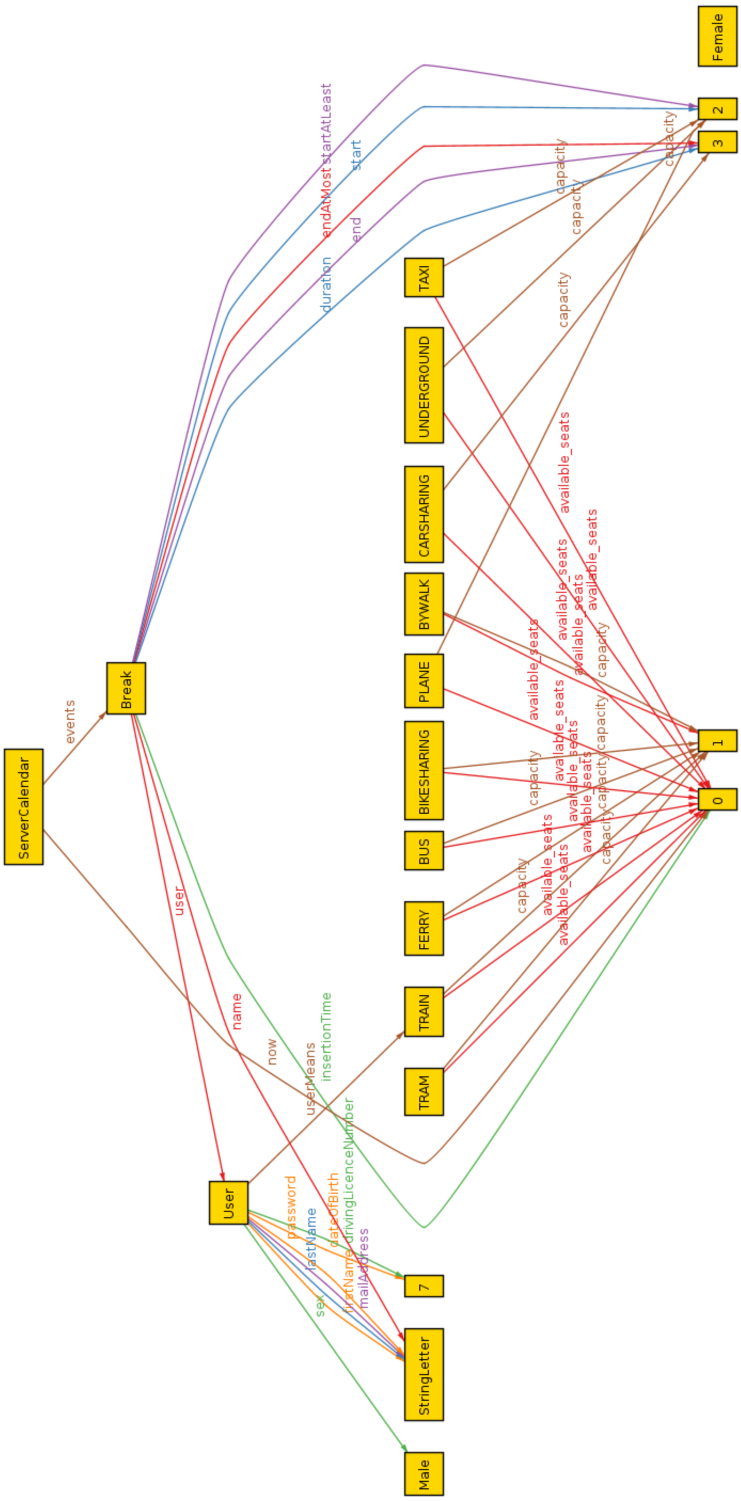

Figure 22: An Alloy instance.

Section A

Appendix

A.1 Software and tools used

The software and tools used during the drawing of this document are:

LaTex: used to build this document.

Google Drive / Google Documents: used to always update and share the documents.

draw.io: used to draw diagrams (https://www.draw.io).

Marvel: used to build the mockups (https://marvelapp.com).

GitHub: used as version control and to keep it shared between group members and teachers (https://www.github.com).

Alloy Analyzer: used to analyze specifications and requirements.

A.2 Hours of work

Most of the work on this document was made in the presence of both members of the group. The approximate number of hours worked by each member of the group is as follow (including hours spent in group work):

Davide Rossetto: about 40 hours
Alessandro Tatti: about 40 hours

Section B

Bibliography

[1] AA 2017/2018 Software Engineering 2 - Project goal, schedule and rules
[2] Jackson, Zave - The World and the Machine