POLITECNICO
MILANO 1863

Software Engineering II

Travlendar+

Requirements Analysis
and
Specifications Document

Authors:

Edoardo D'Amico
Gabbolini Giovanni
Parroni Federico

1st October 2017

Indice

Chapter 1

Introduction

1.1 Purpose

Our team will develop Travlendar+, a calendar-based application that aims to provide a schedule of user appointments, giving a plan to organize his daily life. The main goals the app must fulfill are:

G1 Allow the user to insert an appointments according to his necessities and his preferences;

G2 The system S.P.W. to modify an inserted appointment

G3 The system S.P.W. to create a valid schedule of the user appointments when requested **fare ref alla def**

G4 The system should let the user create multiple valid schedule and decide which one is chosen the current day

G5 The system should offer the possibility to create a new account

G6 The system should be able to handle a login phase

G7 The system S.B.A to book the travel means involved in the current schedule under user approval

G8 The system S.B.A to display the scheduling result **(fare riferimento alla definizione)**

G9 The system should offer a GUI that handles All the interactions with the user.

G10 The system S.B.A. to display in real time user position and the directions to be followed in order to arrive to the next appointment on a map dinamically updated

G11 The system should re

1.2 Scope

Here we provide a brief description of the aspects of the reality of interest which the application is going to interact with.

User can receive an appointment on a certain date, time and location (over a region), that can be reached using different available travel means. The appointment can be held either at a specific time or in a time interval and lasts for a certain amount of time. An appointment can be recurrent, in other words, it repeats regularly over time (e.g., lunch, training, etc.). User can travel with someone else and can pick up or leave off these people during the day.

User can have his own travel means and a pass for public transportation. The travel means considered in this scenario can be grouped in three categories: public, shared or private.

- Public travel means: these include trains, buses, underground, taxis, trams. They have to be taken in their **appositi** stops. User must have a valid ticket in order to get on a public travel means (except for taxis, that pick up the user wherever he wants upon a call and do not require any ticket);
- Shared travel means: these include car and bike. They are located in specific places and require a reservation in order to be used by the user;
- Private travel means: vehicles owned by the user. They can be cars, bikes, motorbikes.

Weather conditions can change during the day affecting usable travel means. At the beginning of the day, or on demand, user can request a schedule of his daily appointments, following some criteria evaluated according to their assigned priority and satisfying some constraints imposed by the user. When a new appointment is received, user creates a new item in the application and saves it in the appointment list. User can request a

reschedule to the application due to unexpected changes of his plan (e.g. a cancelled appointment).

1.2.1 World Phenomena

- User receives a new appointment;
- User picks up a person;
- User owns private travel means and/or passes for public transportation;
- User wakes up;
- User pass expires.

1.2.2 Shared Phenomena

- Shared travel mean moves;
- Shared travel mean its not available anymore;
- Wheather condition changes;
- Public travel means reach a stop-place;
- Public travel means are late;
- Public travel means are not available due to a strike day;
- User requests a schedule to the machine;
- User inserts a new appointment into the application;
- User requests to book rides;
- User moves.

1.3 Definitions, Acronyms, Abbreviations

sinonimi: Appointment/meeting Schedule/Scheduler System/Application preferences/-constraint

def:

preferences: constraints on appointments or schedules

la toglierei questa in quanto abbiamo definito i optimization criteria nella sezione schedule model Opt Criteria: criteria followed by the scheduler in order to optimize

travel option: the combination of travel path and travel means that allow to reach one spot from another.

travel option data: additional information about a travel option:

- Cost;
- Traveling time;
- Carbon emission;

Schedule: a set of time-ordered and not overlapping appointments where their starting times are fixed and they're linked each other by a path travelled with a specific transportation mean

Valid Schedule: a Schedule which:

- is optimized according to the criteria chosen by the user;
- ensures that the user will be on time for all his appointments;
- respects the constraints imposed by the user

relative path: portion of a path travelled by the same travel means

Scheduling result: the set of:

- Graphical representation of the path that will be travelled by the user
- money spent for each relative path
- total money spent
- Length of the path expressed in KM
- Length of relative path
- Carbon footprint emission
- Exstimated travel time of each relative path

- Total exstimated travel time

convenzioni: variables are italic states are bold

abbr: GPS: global positioning system GUI: graphic user interface ETA: estimated time of arrival S.P.W.: should provide a way S.B.A.: should be able API: application programming interface CRUD: create/read/update/delete

1.4 Revision history

1.5 Reference documents

1.6 Document structure

Chapter 2

Overall Description

2.1 Product Perspective

2.1.1 User Model

A user is represented within the application by his password and his e-mail. Some important informations about him are held by the following parameters:

- *travelPass*: indicates if the user has a pass for public transportation;
- *hasBike*;
- *hasCar*.

2.1.2 Appointment Model

An appointment is represented within the application by a set of parameters:

- *duration*: the time extension of the appointment
- *startingTime* or *timeInterval*: the first should be given if the starting hour is well-known (deterministic), otherwise a time interval in which the appointment will be held it's provided.
- *location*: identifies the coordinates of the place where the appointment will be held;
- *recurrent*: specifies if the appointment will be repeated over a fixed period of time;
- *peopleTravelling*: represents a variation occourring when the user picks up or leaves off someone.

The life cycle of an appointment can be represented by the following statechart:

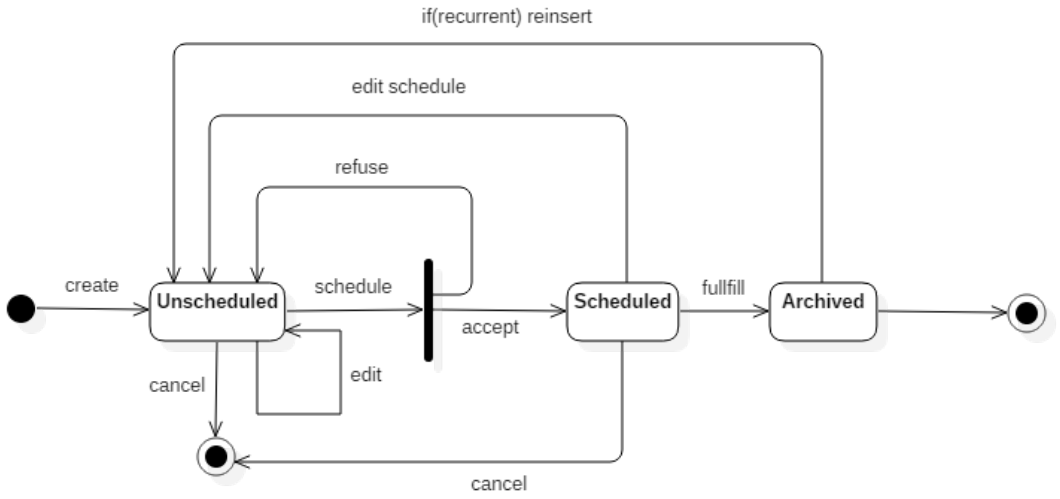


Figure 2.1: Appointment statechart

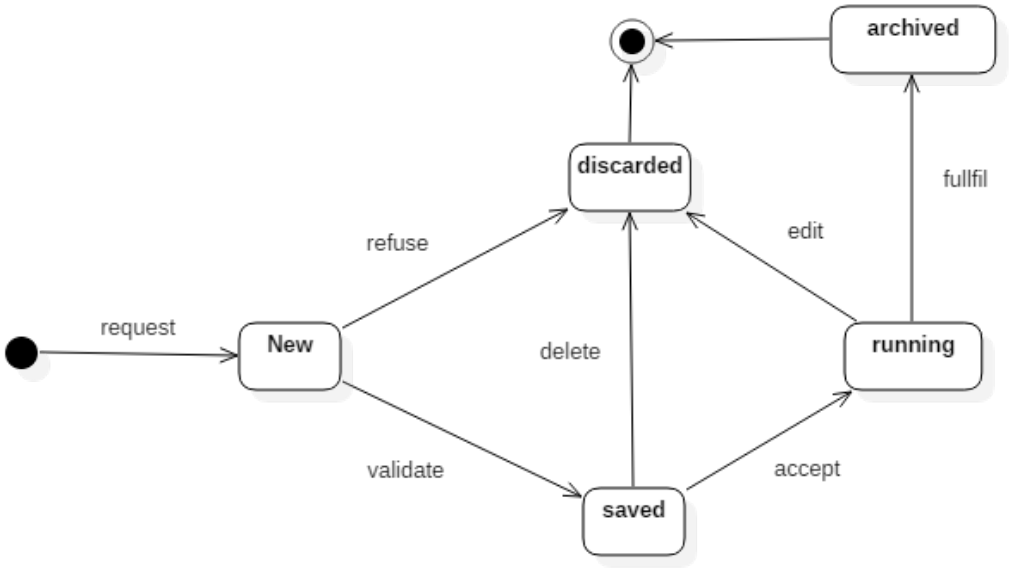
A newly-created appointment is **Unscheduled**. It could remain **Unscheduled** either when edited or there isn't a possible arrangement when a schedule is performed. Otherwise it becomes **Scheduled** if there's a feasible way to arrange it. When a scheduled appointment is edited all the appointments in that schedule return to be **Unscheduled**, because it's possible that they bring to a different schedule. When a scheduled appointment is fullfilled it becomes **Archived** and stored in the schedule history. If this last one is a recurrent appointment it must be reinsert in the list of unscheduled appointment so it will become **Unscheduled** again. The user can cancel an appointment in every moment.

2.1.3 Schedule Model

A schedule is a set of Appointments in a given day, ordered by the scheduler following the criteria described below. A schedule is characterized by the following variables:

- *date*;
- *startingPosition*: is the starting location of the user (e.g. user's home);

- *startingNumberOfPeople*: the number of people that must reach the first appointment.
- *wakeUpTime*: it is the starting time from which the schedule should start arranging appointments.



The optimization criteria

The criteria that can be chosen with priority for a schedule by the user for the optimization are the following:

- *Minimize carbon footprint*: the scheduler will try to minimize the amount of kilometers travelled in polluting means;
- *Minimize money spent*: the scheduler will try to avoid expensive means and to exploit the public ones (especially if the user has a pass) or going by bike or on foot;
- *Minimize travelling time*: the scheduler will compute the quickest possible path reaching all the appointments locations.

2.1.4 Constraints

Constraints are impositions on some parameters managed by the system during the process of scheduling the appointments. We can distinguish between constraints on

schedule and contraints on the single appointment. These can be selected by the user when he inserts an appointment or when he requests a schedule, otherwise the constraints are initialized to default values.

Constraints on schedule

- *Maximum travelling distance with a specific travel mean*: the user can set a maximum amount of km to travel with a travel mean;
- *Travel means time slots*: user can specify a time interval in which a travel mean can be used;
- *User can deactivate a particular travel means*;
- *User can select which travel means uses under certain weather condition.*

Constraints on appointment

- *User can deactivate a particular travel means.*

2.1.5 Class Diagram

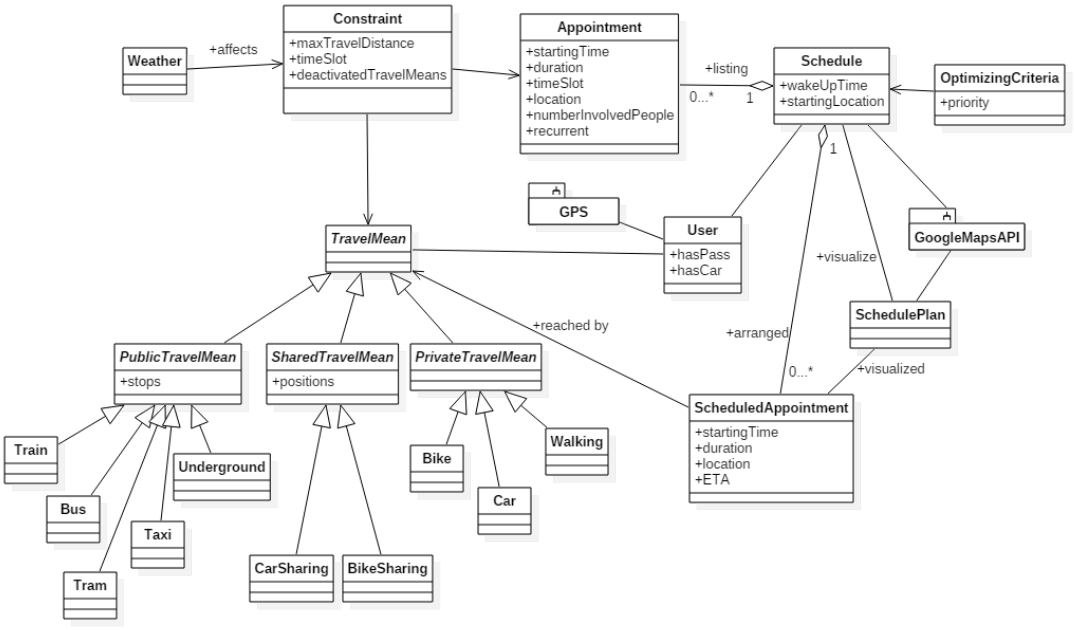


Figure 2.2: System Class Diagram

2.2 Product Functions

The following requirements are derived in order to fullfill the specified goals.

Requirements for **G1**:

R1 The system S.B.A. to retrieve information from the user about his appointments;

R2 The system S.B.A. to store an appointment in his memory

Requirements for **G3**:

R3 Allow the user to set the parameters of the schedule (2.1.3), or to accept the default values;

R4 Allow the user to select the optimization criteria (2.1.3) for the schedule;

R5 The system S.B.A. to gather information from external APIs about:

- travel options with related travel option data;
- weather forecast;
- strike days;

R6 The system S.B.A. to select the best travel option according to the optimization criteria taking into account:

- user constraint
- user parameters **(aggiungere riferimento a user model)**
- travel option data
- weather forecast
- information about strike day

Requirements for **G5**:

R7 The system S.B.A to handle a registration phase in which the user will provide an e-mail and a password

R8 The system S.B.A. to verify the e-mail given by the user

Requirements for **G6**:

R9 The system S.B.A to recognize a registered user given an e-mail and a password

R10 The system S.B.A. to retrieve information from the user about his registration informations, i.e. his e-mail and password (**da verificare se serve o meno**);

Requirements for **G7**:

R11 The system should offer to the user a way to link all his travel service accounts into the Travlendar+ account;

R12 The system S.B.A to book a travel mean through external API offered by third part application in which the user is signed

Requirements for **G8**:

R13 The system S.B.A to retrive the graphical representation of a path from an external API

R14 The stystem S.B.A to retrive the travel option from an external API **(questa che già si è detta ma serve anche per questo goal come facciamo?)**

R15 The S.B.A to retrive the length of a path from an external API

Requirments for **G4**:

R16 The system S.B.A to store valid schedules requested by the user

R17 The system should let the user accept a schedule from the saved ones

2.2.1 User characteristics

Users can use our system when they want something that let them able to schedule theire meetings according to theire necessities and constraints. Necessary conditions for the users in order to use the system are:

- He must have an **(specificare meglio in seguito in quale dispositivo occorre farla girare)** connected to the internet in which run the application

This is the only requirement that is needed. Anyway additional characteristics of the user lead to the exploitment of all the system features. In fact some of them are guaranteed only after having submitted to the application some information. In this sense, wellcomed user's characteristics are:

- The ownership of some travel means
- The ownership of travel passes
- The registration to sharing services

Beside these, an obviously tacit assumption that is made it's that the user has an age in which is able to move where he wants with autonomy.

2.2.2 Assumptions, dependencies and constraints

- The system should be able to retrieve information about public travel means. In particular:

  - information about delays;
  - information on possible strike days.

- The system should be able to retrieve information about shared means. In particular:
  - position of the available ones;
  - prices per time unit;
- There exist external APIs that allow to:
  - retrieve all travel options and travel option data
  - signed user to book and pay for all travel services
  - retrive information about weather forecast
  - retrive a graphical representation of a path
  - retrive the lenght of a path
- The device on which the application runs is connected to the internet;
- every user has at least one personal e-mail

Chapter 3

Specific requirements

3.1 External Interface Requirements

The application shows its best potential when run in a mobile device, for instance a smartphone or a tablet. This permits to extend the features and the automatic tasks of the application, thanks to the built-in device functionalities. However, a computer client version of the application can be installed, too.

3.1.1 User interfaces

The user can interact with the application through several graphical interfaces:

1. Registration/login interface: allows the user to insert credentials in order to registering or logging into the system;

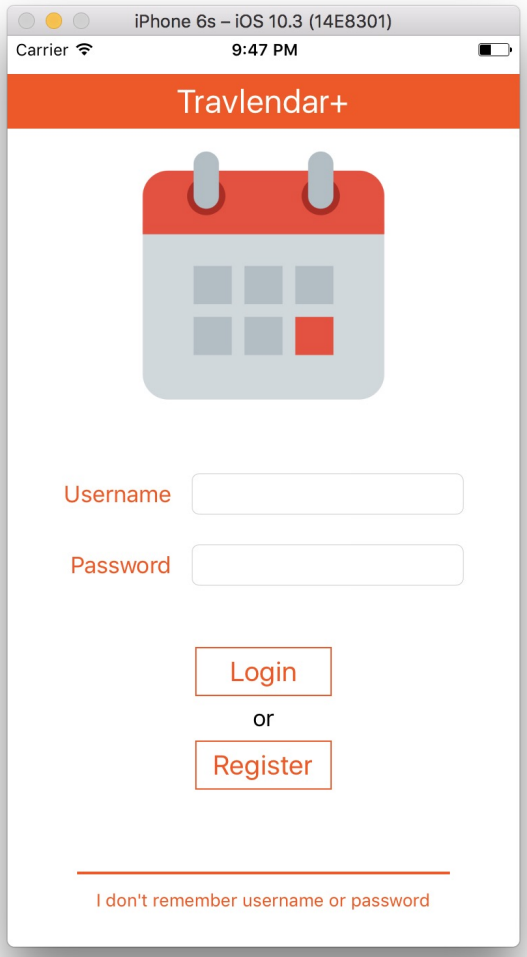


Figure 3.1: Registration/login interface

2. User account interface: user can specify his profile characteristics, such as his passes, car and/or bike ownership;

3. Appointment CRUD interface: allows creating, showing and editing appointment parameters and related constraints;

4. Non-scheduled appointments interface: provides a list of all inserted appointments, but not **already** scheduled (includes the possibility to delete an item of the list);

5. Schedule interface: user can set parameters, contraints, optimization criteria and request a schedule creation for a given date;

6. Schedules result interface: shows the computation of the requested schedules for a given date and asks the user to select one, then waits for confirmations for that;

7. Schedule progress interface: permits to keep track of the completeness percentage, indicating the directions to be followed by the user in a map, in order to arrive to the next appointments;

8. Tickets/rides reservation interface: allows user to buy tickets for public travel means and/or reserve a ride for the shared travel means;

9. Appointments history interface: shows a list of archived appointments;

3.1.2 Hardware interfaces

Hardware interfaces are physical linking across which two or more separate components of a system exchange information. A hardware interface is described by the mechanical and electrical signals at the interface and the protocol for sequencing them. There are no interesting hardware interfaces in our scope.

3.1.3 Software interfaces

Software interfaces are logical linking across which two or more separate applications running on a system exchange information.

- API: information can be exchanged through HTTP request/response;

3.1.4 Communications interfaces

Communication interfaces allows two different architectures of the system to exchange information through communication channel. These non-homogeneous component of the system can communicate thanks to the following software interfaces and protocols:

- Cellular connectivity: mobile devices can connect to the internet thanks to LTE standard;

- GPS: cellular can retrieve his coordinates position through NMEA protocol;

- QRCode: present in most of the shared means, allows to quick identify the transportation which we are near;

3.2 Functional requirements

3.2.1 Scenarios

Here are some scenarios that describe the usage of the system.

Scenario 1

Giovanni will start the fourth year of his Master's degree. Surfing the internet, he finds out that his lesson schedule for the first semester it has been published. Giovanni decides to fill in the application with his new appointments concerning the attendance of lessons. In fact he knows where to go, at which time and day and for which amount of time. Since he knows that these events will going to happen for 3 months, he sets that they should be recurrent.

3.2.2 Use cases

User log-in

Name: User log-in
Actors: User
Goals: **G6**
Input Condition: The user is registered to the system
Event Flow: 1. The user needs to log-in the application, so he runs it; 2. The user requests to log-in to the system; 3. The system provides to the user a form to fill; 4. The user fills up the form with the his e-mail and his password (as said in 2.1.1) 5. The user submits the form to the system; 6. The system checks the user identity and provides to the user the main application page (**reference to the main application page**)
Output Condition: The user submits the form after having filled it with a matching e-mail and password.
Exceptions: The user submits the form after having filled it with a wrong email or password.
Mapping on requirements: • Events from 3 through 5 granted by **R10**; • Event 6 grandet by **R9**;

Appointment creation

Name: Appointment creation
Actors: User
Goals: G1
Input Condition: The user is registered to the system
Event Flow: 1. The user wants to add a new appointment to his schedule, so he runs the application; 2. The user logs-in in the application (**ref to the log-in use case**); 3. The user requests the creation of a new appointment to the application; 4. The system provides to the user a form to fill; 5. The user fills up the form with the parameters (specified in 2.1.2) and constraints (specified in 2.1.4 about the new appointment; 6. The user submit the form to the system; 7. The system allocates the new appointment as Unscheduled (referring to statechart in figure;) 8. The system sends a confirmation to the user.
Output Condition: The user submits the form after having filled it with all the parameters needed. The fields about constraints could also be left blank (they are optional)
Exceptions: 1. Some fields of the form referring to parameters are left blank; 2. The *location* field doesn't belong to the domain area of the application (**riferimento alla domain assumption della regione**)
Mapping on Requirements: • Events 4 through 7 are granted by **R2** • Event 7 is granted by **R1**

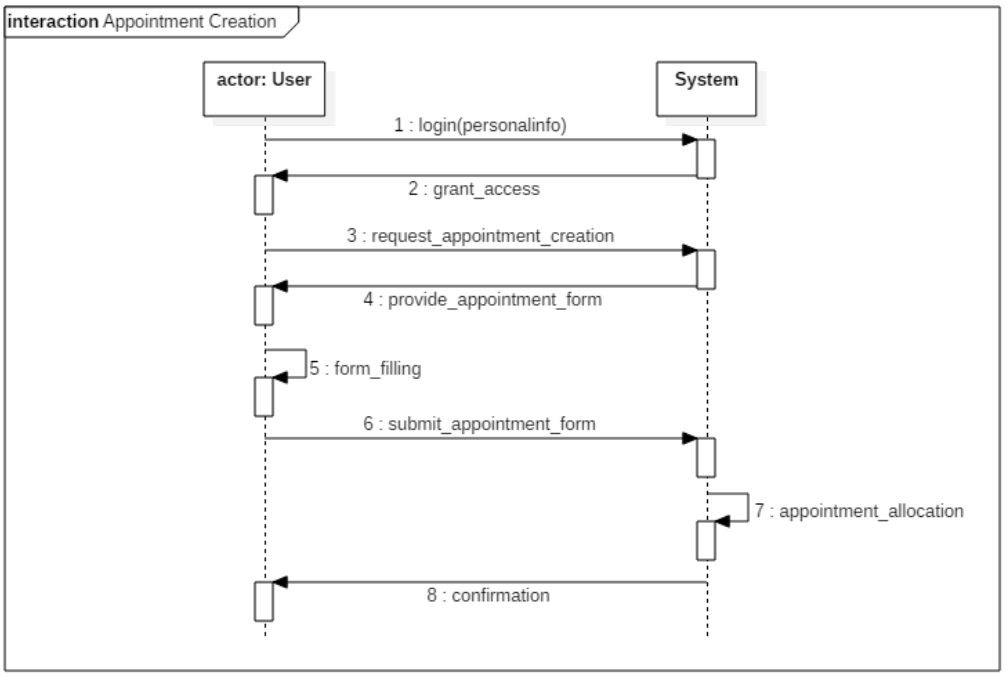


Figure 3.2: Appointment creation sequence diagram

In the sequence diagram there's the assumption that the log-in proceeds successfully. The log-in procedure referenced is the one explained in (**use case del log-in**.)

3.2.3 Definition of use case diagrams

eventuali informzioni che specificano meglio come leggere i deagrammi che si producono