POLITECNICO
MILANO 1863

Software Engineering 2 Project - Travlendar+

Requirement Analysis and Specification Document - V1

October 28, 2017

Authors:

Francisco Cristóvão
Samsom Tsegay Beyene

Contents

1 Introduction

1.1 Purpose

The main goal of this project is to create a calendar-based system which provides the user a flexible and fully-featured calendar support that considers the travel time between meetings. With this in mind, the application will:

- Compute and account for travel time between appointments, and prevent conflicts between them.
- Support the user in his/her travels, adding automatically the travel time to the calendar between meetings, and suggesting the best travel option based on the available time.
- Plan the User schedule in the best and most efficient way.

1.2 Scope

Travlendar+ is a calendar-based application that provides the user a convenient way of organizing his/her daily schedule, maximizing its productiveness and minimizing the worthless time of his/her day. This application was not only thought for the regular businessman/businesswoman, who travel in between meetings the whole day and have no time to spare, but also for the parents with a more regular daily schedule, who just want to get the best out of their time while being able to pick their kids from school and take them to other activities, always being on time. Of course the system will fully support the features of a regular calendar application (booking of appointments in a specific time and location), but in a "smart" way, being able to detect and warn the user if a new appointment is not feasible because it has a conflict (the start of it doesn't allow the needed travel time after the end of the last appointment) and arranging all of the appointments in the best possible way. The application is meant to be used in the City of Milan, and so it will take advantage of the wide range of travel means and services already existing in the city, from public transports to shared bikes and cars. With the information gathered from those services, it will be able to suggest the best travel mean for the user to move between appointments, based on the available travel time, total cost, current weather and even user preferences.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Visitor: A person who uses Travlendar+ for the first time, and is not yet registered.
User: A person who uses Travlendar+.
Home Screen: User interface screen that shows the current appointments.
Flexible Appointment: An appointment which the start and end date may change between a given time range (are flexible), as long as its minimum duration is ensured.
System: defines the overall set of software components that implement the required functionality.
Local Time: time of the system.

1.3.2 Acronyms

API: Application Programming Interface
RASD: Requirements Analysis and Specification Document
ATM: Azienda Trasporti Milanesi
SQL: Structured Query Language
BPMN: Business Process Model and Notation

1.3.3 Abbreviations

Gn: Goal defined with index n.
Dn: Domain Assumption defined with index n.
Rn: Functional Requirement defined with index n.

1.4 Revision History

Version 1.0: Initial Release

1.5 Reference Documents

- Assignment document: Mandatory Project Assignments.pdf
- IEEE Recommended Practice for Software Requirements Specifica- tions (Std 830-1998)

1.6 Document Structure

Other than this introductory chapter, this RASD is organized in six more chapters. Chapter two is meant to provide an overview of the systems functionality, the type of users it is meant for and the different kinds of interactions it contemplates, not only with the users themselves, but also with other systems. Some of the systems requirements are also slightly discussed in this chapter, even though they'll be analysed in the following chapter. In the third chapter (as mentioned) the system requirements, attributes and constraints are analysed and discussed with the appropriate detail and depth, specifying exactly how they should be. The fourth chapter deals with the formal analysis of the system using an Alloy model. It includes the Alloy model of the system with a brief discussion on its purpose and on the relevance of using Alloy as a tool to validate our solution, given the problem we had to solve. In the fifth chapter the effort spent by each of the group members is described by specifying the number of hours each member of the group worked on the development of this document, and on the final chapter the tools we used to develop this RASD are specified.

2 Overall Description

2.1 Product Perspective

The system will need to communicate with both public transport systems, bike or car sharing system of the city and Google maps API to find the exact status of available (active) types of transportation with their corresponding locations. These information helps the System to exactly locate the position of the user and the appointments place so that it can assign the best travel means to get to the referred appointment. In addition, the System also retrieves the weather conditions and takes them into consideration when choosing the travel mean. Since this system relies on data storage to work, it will need some alternative to store this data. So, the system will communicate with the database (probably distributed) to add or modify the appointments, as well as the user details and preferences. All the database communication will go over the Internet.

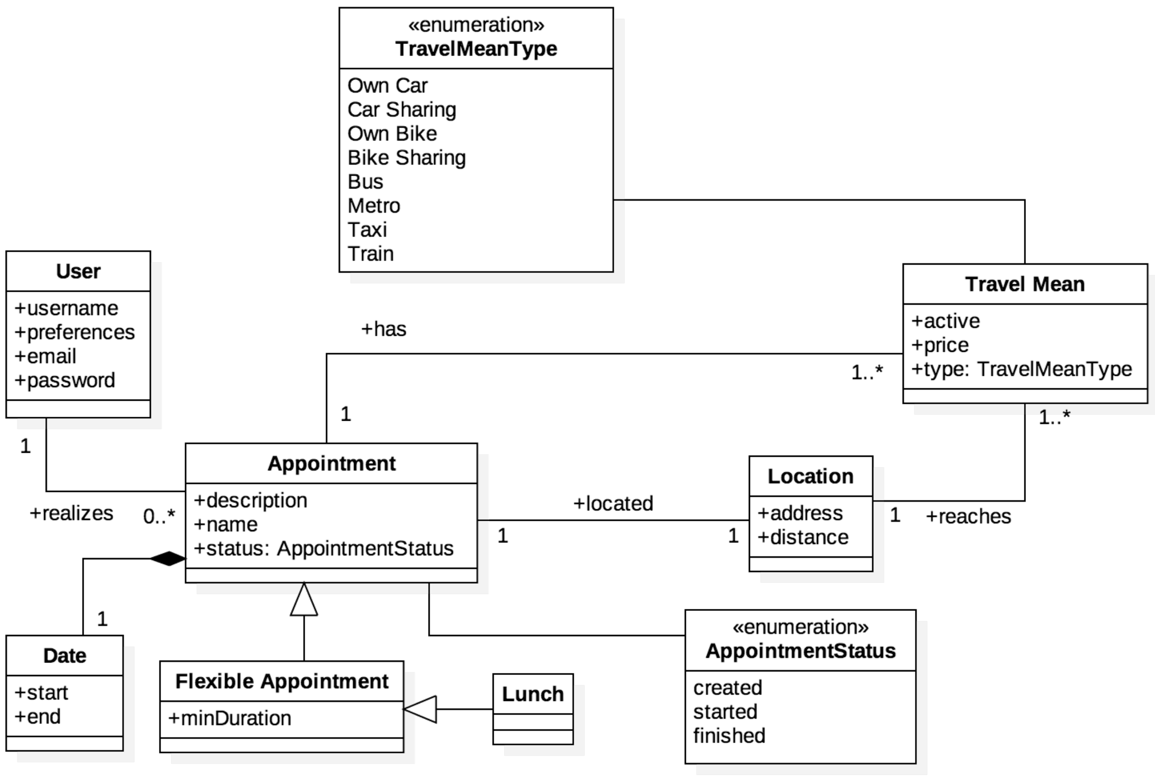


Figure 1: Class Diagram of the domain

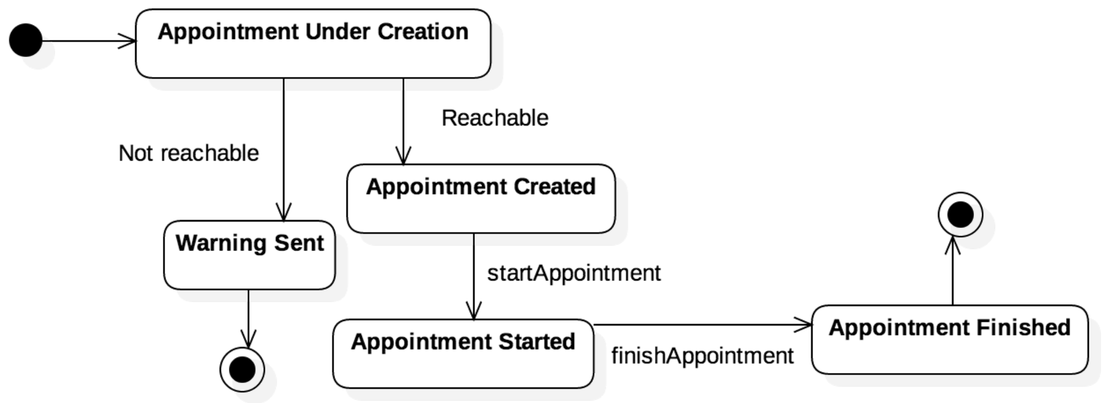


Figure 2: Statechart of an Appointment

2.2 Product Functions

The main functions supported by the system will be:

- Allow the user to book an appointment with a given time, location and duration.
- Plan the user's schedule in the best possible way, choosing for him where to place flexible appointments and suggesting which travel means to use in between appointments, based on his preferences, the time available and the weather.

2.3 User Characteristics

The system aims at satisfying the needs of many different types of users. The first type is the "businessman" user, the user that is always in a tight schedule and needs a tool like our system to simplify and improve the way he organizes his schedule and the way turns up to the appointments. The second type is the opposite of the "businessman" user. It's a user that has a less busy life and only needs the system to remind him of the appointments without having to worry about how to get there or when to leave to arrive on time. There's also a third type of user which is placed in between the two described previously: also has a tight office schedule but still has time for personal activities, and needs the system to help him reach this balance in the most efficient way. All of these types of users are of working age (20 to 70 years old, being this a small deviation of the official definition, which is from 15 to 64 years old)[1] and are expected to know how to use a smartphone with reasonable proficiency. It's easy to see that the needs of the users will map directly to the goals of the system, most precisely to G5, which is the more general and "different" from the similar applications already on the market, and the one the users are looking more forward to.

2.4 Assumptions, dependencies and constraints

[D1] The username and email must be unique.
[D2] The system will have access to the phone GPS functionality.
[D3] The phone will always have a connection to the internet (WiFi or mobile data).
[D4] The best route and travel mean to get to an appointment will always be computed correctly.
[D5] The users authorize the system to provide user data (when needed) to the third-party applications it interacts with.

[1]Definition from OECD: https://data.oecd.org/pop/working-age-population.htm

[D6] The external services used by the system are assumed to be always available and reachable.
[D7] Changes on the user preferences only apply to the appointments created after the change. The ones already created stay the same.
[D8] Two different users are not allowed to interact with each other.
[D9] The best travel means to get to an appointment won't change if the appointments don't change.
[D10] Users don't need to confirm if they started or finished the appointment.
[D11] An Appointment maps to only one User.
[D12] The location of the users private car and bike is always known by the system.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

These Mockups represent a basic idea of how the systems user interface will look like on the first release (mobile application):

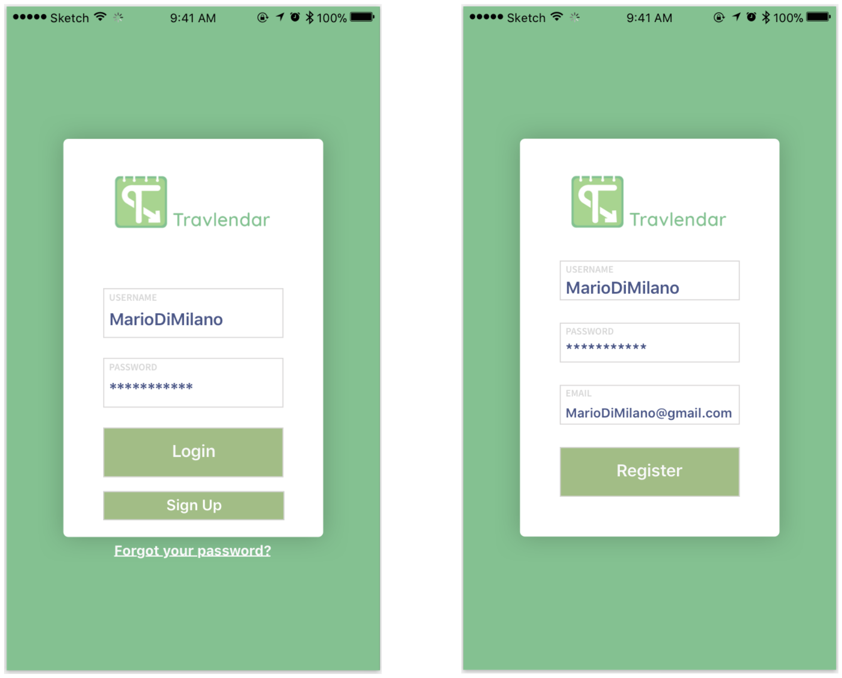
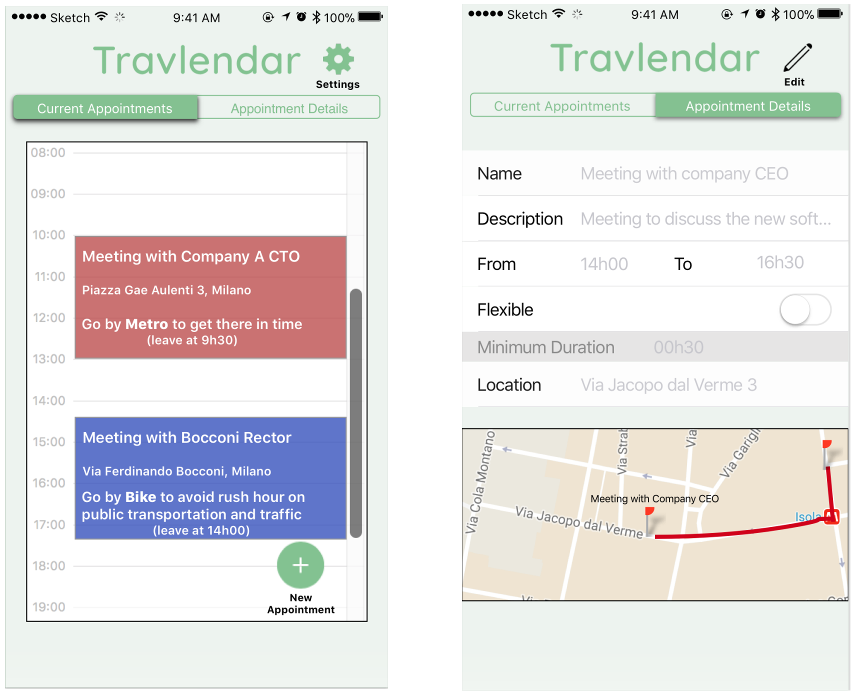


Figure 3: Login Screen (left) and Register Screen (right)


Travlendar

Settings

Current Appointments | Appointment Details

08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00

Meeting with Company A CTO
Piazza Gae Aulenti 3, Milano
Go by **Metro** to get there in time
(leave at 9h30)

Meeting with Bocconi Rector
Via Ferdinando Bocconi, Milano
Go by **Bike** to avoid rush hour on public transportation and traffic
(leave at 14h00)

New Appointment

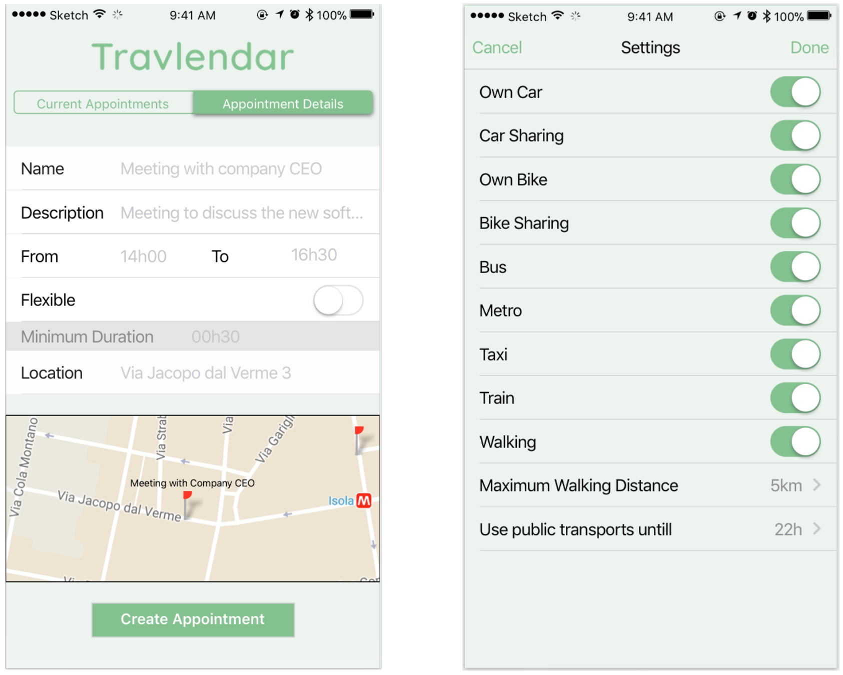
Travlendar

Edit

Current Appointments | Appointment Details

Name	Meeting with company CEO		
Description	Meeting to discuss the new soft...		
From	14h00	To	16h30
Flexible			
Minimum Duration	00h30		
Location	Via Jacopo dal Verme 3		

Figure 4: Home Screen (left) and Appointment Details Screen (right)


Travlendar

Current Appointments | Appointment Details

Name	Meeting with company CEO		
Description	Meeting to discuss the new soft...		
From	14h00	To	16h30
Flexible			
Minimum Duration	00h30		
Location	Via Jacopo dal Verme 3		

Create Appointment

Cancel | Settings | Done

Own Car	✓
Car Sharing	✓
Own Bike	✓
Bike Sharing	✓
Bus	✓
Metro	✓
Taxi	✓
Train	✓
Walking	✓
Maximum Walking Distance	5km
Use public transports untill	22h

Figure 5: New Appointment Screen (left) and Settings Screen (right)

3.1.2 Hardware Interfaces

In the first release no Hardware Interfaces will be necessary, since the system doesn't need to interact physically with other systems.

3.1.3 Software Interfaces

Given the wide range of features the system offers, it will need to have several software interfaces. There has to be a way of storing all the user-related data (mainly login, preferences and appointments data). In that sense, the system will use a **MySQL API** to connect with a MySQL database server, where all the app data will be stored.
The system will also need different kinds of information about the users location and its surrounding, in order to compute the travel time between appointments. To support this functionality, the system will use **Google Maps Geolocation API** and **Google Maps API**. The first one will provide information about the user exact location and the second one will provide the real-time information about maps and traffic, and allow it to calculate the best route between two different appointments. To have access to the current weather, the system will use **AccuWeather API**, which will provide information about the current weather on both where the user is and where the appointment is located. At last, the system will also need to get all sorts of information from many different travel means. In order to get these information, it will have a software interface with **ATM Milano** (public transport schedule and routes), **Drive Now API** (for the car sharing system) and **Mobike API** (for the bike sharing system).

3.1.4 Communication Interfaces

Communication interfaces ensure the communication between the system and the other software interfaces (third party service providers). The communication between the system and those service providers is crucial because the system depends on those services to perform its functions. Given this, the systems communication interface must be either WiFi or Mobile Data (2G, 3G or 4G), and the service providers communication interface can be of any type, as long as it ensures that they are connected to the internet. The protocol used shall be HTTPS, in order to keep the communications secure.

3.2 Functional Requirements

[G1] Allow a visitor to become a user after providing the required information and credentials.

- [R1] The system shall allow the visitor to begin the registration process when he accesses the system.
- [R2] The system shall ask the visitor for the required information and credentials during the registration process.
- [R4] The system shall register the visitor as a user when the registration process ends, if all the data inserted by the user is valid.

[G2] Allow a logged user to create an appointment.

- [R5] The system shall allow the user to fill in all the needed information about the appointment.
- [R6] The system shall not allow two appointments to overlap.
- [R7] The system shall create a warning if the appointments location is unreachable.
- [R8] The system shall not allow the user to create an appointment if the location is unreachable.
- [R9] The system shall allow the user to select if the appointment is flexible or not.
- [R10] The system shall create the appointment if all the data inserted by the user is valid.

[G3] Allow a logged user to edit an appointment.

- [R11] The system shall allow the user to edit all the provided information about the appointment.
- [R6] The system shall not allow two appointments to overlap.
- [R7] The system shall create a warning if the appointments new location is unreachable.
- [R12] The system shall not allow the user to submit the changes if the new location is unreachable.
- [R13] The system shall submit the changes to the appointment if all the data inserted by the user is valid.

[G4] Allow a logged user to delete an appointment.

- [R14] The system shall allow the user to delete all the data related to the appointment that the user wants to delete.

[G5] Allow a logged user to have his schedule planned in the best possible way.

- [R15] The system shall compute the best possible arrangement of the users appointments that fulfills his/her preferences.
- [R6] The system shall not allow two appointments to overlap.
- [R16] The system shall compute the best travel mean to get to each appointment.
- [R17] The system shall re-plan the users schedule every time an appointment is inserted, deleted or its location and/or date is changed.
- [R18] The system shall re-compute the best travel mean to get to the other appointments in case one of the appointment gets deleted or its location is edited.
- [R19] The system shall display the best travel mean to get to each appointment, along with the best arrangement of appointments.

[G6] Allow a logged user to define preferences related to travel means.

- [R20] The system shall allow the user to enable all the travel means he wants the system to consider when computing the best way to move in between appointments.
- [R21] The system shall allow the user to disable all the travel means he doesn't want the system to consider when computing the best way to move in between appointments.
- [R22] The system shall allow the user to select the maximum distance he is willing to walk from one appointment to another.
- [R23] The system shall allow the user to select an hour of the day from when public transportation shall not be used.
- [R24] The system shall allow the user to select a combination of public transportation in order to minimize his carbon footprint (includes only walking, bike or electric travel means).

3.2.1 Use Case Diagrams

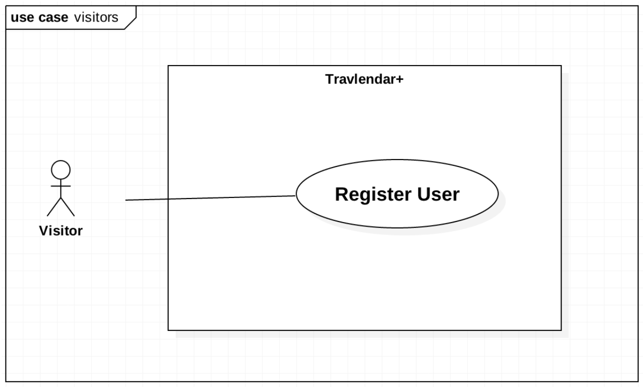


Figure 6: Visitor Use Case

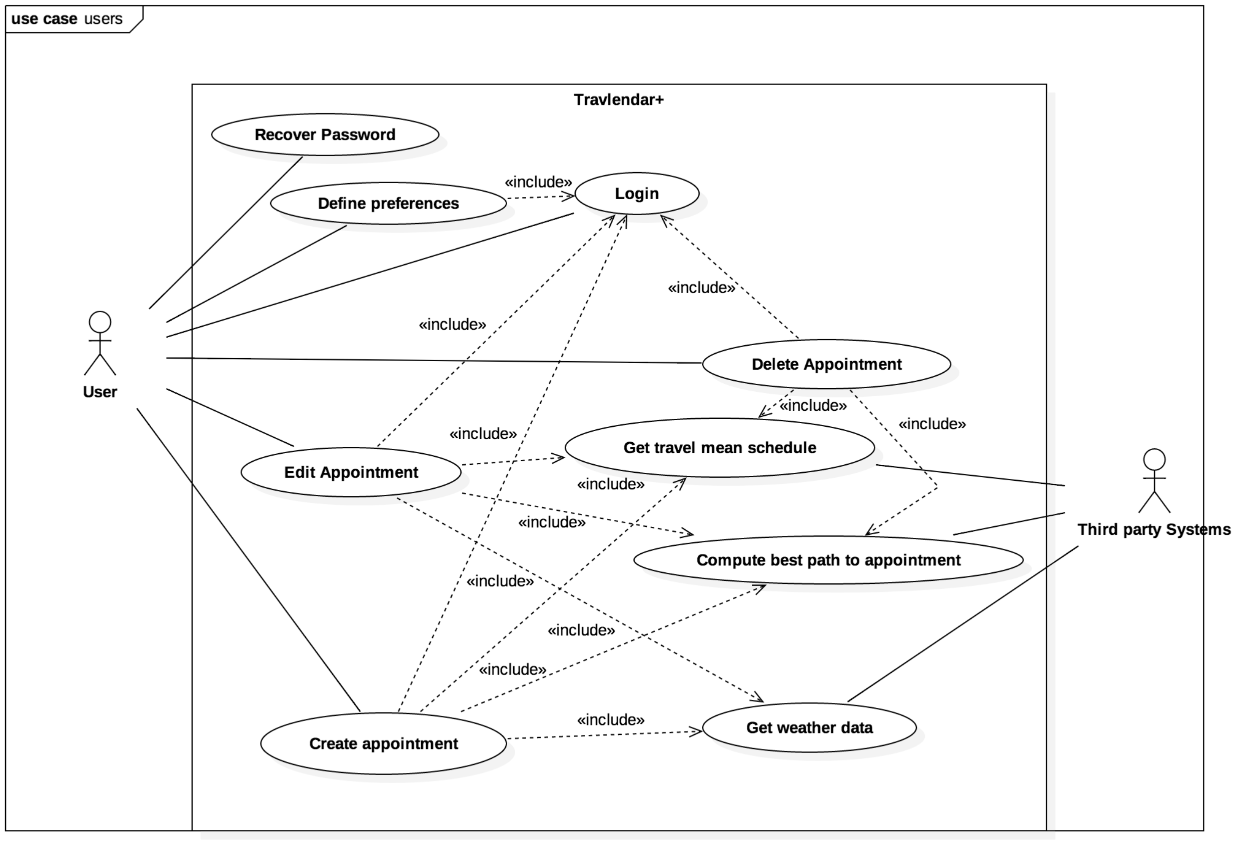


Figure 7: User Use Case

3.2.2 Use Case Description

Visitor Registration

Actors	Visitor
Goals	[G1]
Preconditions	There are no preconditions.
Events Flow	1. The visitor opens Travlendar+ application. 2. The visitor clicks on the "Sign up" button to start the registration process. 3. The system provides the Visitor a form with the required data fields which need to be filled in order for him to be registered as a user. 4. The visitor fills the fields with his data. 5. The visitor clicks on the "Register Button", submitting the data to the system. 6. The system verifies if all the data are valid. 7. The system saves the data. 8. The system notifies the Visitor that his registration as a user has been completed with success, and redirects him to the Home Screen
Postconditions	The visitor ends the registration process successfully and from now on is a user of the system.
Exceptions	1. The visitor is already a user. 2. The visitor doesn't fill all the mandatory fields or fills them with invalid data. 3. The visitor chooses a username or email which is already associated with another user. All exceptions are handled notifying the visitor about the issue and going back to the applications initial screen.

User Login

Actors	User
Goals	All goals except [G1]
Preconditions	There are no preconditions.
Events Flow	1. The user opens Travlendar+ application. 2. The user inserts his username and password on the respective fields. 3. The user clicks on the ”Log In” button. 4. The system validates the user credentials. 5. The system redirects the user to the Home Screen.
Postconditions	The user is successfully logged in to the system and redirected to the Home Screen.
Exceptions	1. The user inserts an invalid username or email. All exceptions are handled notifying the User about the issue and going back to event 2 of the Event Flow described above.

User creates an appointment

Actors	User
Goals	[G2 and G5]
Preconditions	The User must be already logged in.
Events Flow	1. The user clicks on the "New Appointment" button 2. The system provides the user a form with the required data fields which need to be filled in order for the appointment to be created. 3. The user fills the form. 4. The user clicks on the "Create Appointment" button. 5. The system validates the data. 6. The system computes the best new possible arrangement of appointments. 7. The system computes the best travel means to get to this new appointment. 8. The system saves the data. 9. The system redirects the user to the Home Screen.
Postconditions	The user successfully creates an appointment and continues having the best possible appointment arrangement.
Exceptions	1. The user inserts an invalid Location. 2. The user creates an appointment which is unreachable. The exceptions are handled notifying the user about the issue and going back to event 2 of the Event Flow described above.

User edits an appointment

Actors	User
Goals	[G3 and G5]
Preconditions	The User must have created at least one appointment.
Events Flow	1. The user clicks on the appointment he wants to edit. 2. The system shows the same form as the one showed when creating an appointment, but with the fields filled with the appointments current data. 3. The user clicks on the "Edit" button. 4. The user changes the location of the appointment. 5. The user clicks on the "Done" button. 6. The system validates the data. 7. The system computes the best new possible arrangement of appointments. 8. The system computes the best travel means to get to the other created appointments, to see if any of those changed with this change. 9. The system saves the data. 10. The system shows the new arrangement of appointments.
Postconditions	The user successfully edits an appointment and continues having the best possible appointment arrangement.
Exceptions	1. The user inserts an invalid Location. 2. The editing of the location of this appointment makes this or some of the other appointments unreachable. The exceptions are handled notifying the user about the issue (and about which appointments are in conflict if that's the case) and going back to event 2 of the Event Flow described above.

User deactivates Bike Sharing as a travel mean

Actors	User
Goals	[G6]
Preconditions	The User must be already logged in.
Events Flow	1. The user clicks on the ”Settings” button 2. The system shows a slide button form, one for each mean of transportation. 3. The user sets the Bike Sharing slide button to deactivated. 4. The User clicks on the ”Done” button. 5. The System saves the data.
Postconditions	The user successfully deactivated Bike Sharing as a travel mean and is redirected to the Home Screen.
Exceptions	There are no exceptions.

3.2.3 BPMN Diagrams

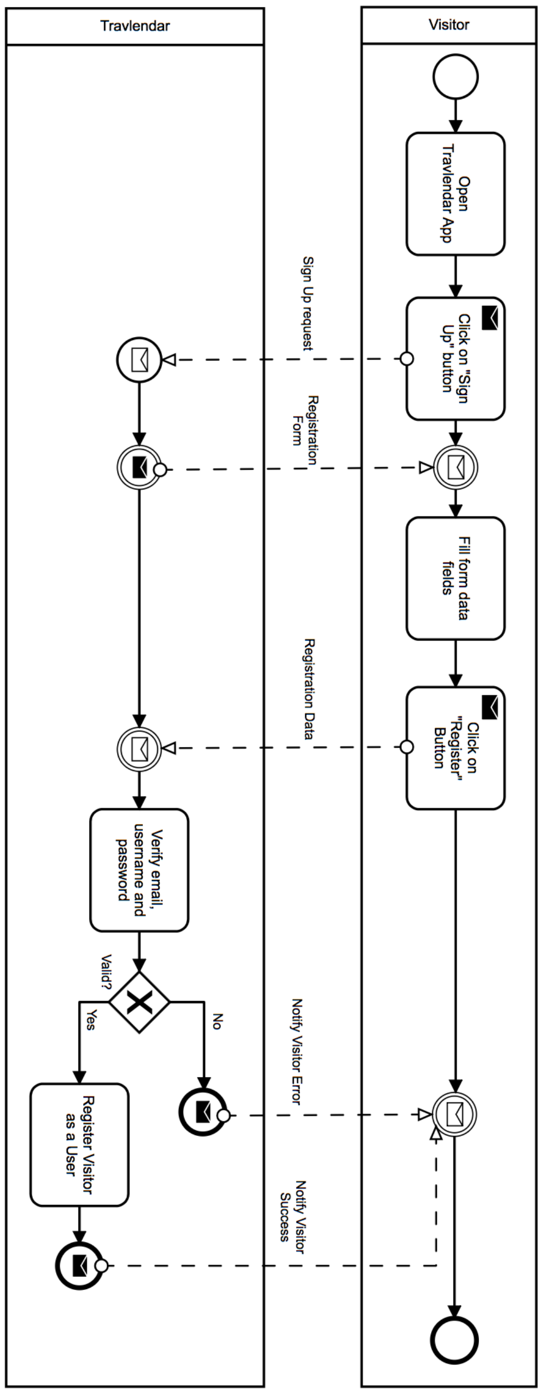


Figure 8: Visitor Registration

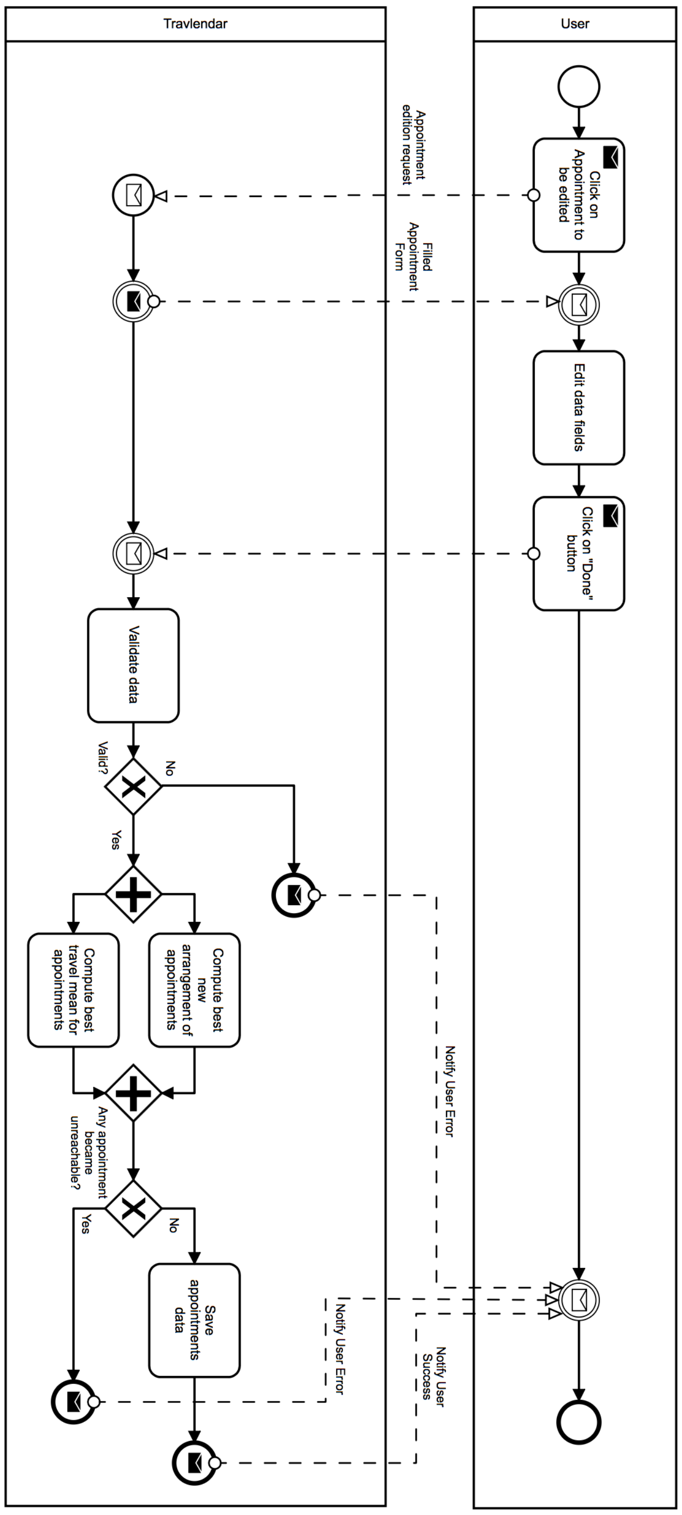

Travlendar
User
Click on Appointment to be edited
Appointment edition request
Filled Appointment Form
Edit data fields
Click on "Done" button
Validate data
Valid?
No
Yes
Compute best new arrangement of appointments
Compute best travel mean for appointments
Notify User Error
Any appointment became unreachable?
Yes
No
Save appointments data
Notify User Error
Notify User Success

Figure 9: User Edits Appointment

3.3 Performance Requirements

The system has to be able to respond to a possibly large number of requests (depending on the amount of users the system will have). Taking into account other (less powerful but widely used) calendar and transport applications, an infrastructure that supports a maximum workload on the order of hundreds of thousands of requests shall be enough to satisfy all the possible simultaneous requests after the release. The infrastructure must have a good scalability in order to increase the maximum workload in case the user base increases too. The response times of the system shall be[2]:

- 0.1 second for the tasks that require almost no computing power (changing menus in the systems case), in order for the user to feel that the system is reacting instantaneously
- 1 second for the tasks that require a normal amount of computing power, in order for the user's flow of thought to stay uninterrupted (deleting an appointment in the systems case)
- 10 second for the tasks that require significant computing power, in order to keep the user's attention (computing the best travel route and consequent travel mean to get to an appointment in the systems case). The users must also receive some sort of feedback during the wait, so they can know what to expect.

N.B. A user is assumed to have a stable internet connection in order to achieve these response times.

3.4 Design Constraints

3.4.1 Standard Compliance

The system must require the users permission to provide their location, preferences and schedule to the external services it uses (transport services and weather services), in order to manage sensible data always respecting the privacy law. For this, the system has to be compliant with the regulation on Software in Data Security and Privacy, international treaty obligations and Italian laws, as well as with any other regulation the third party systems it interacts with comply.

3.4.2 Hardware Limitations

The user will need a mobile phone with GPS and Wi-fi/Mobile Data always on when using the app, other than the obvious space for the application package.

[2]According to Jakon Nielsen book on Usability

3.5 Software System Attributes

3.5.1 Reliability

In order to keep maintenance costs low, the system should offer a high level of reliability (no key component of the system may fail more than once a year).

3.5.2 Availability

The system is required to have an availability of 99.99%. This means that it's acceptable for the system to have a downtime of up to 1 hour every year caused by a system malfunction.

3.5.3 Security

Users credentials and payment information are critical data that will be stored in the systems database. Since the system communicates with external systems, the security and inaccessibility (when not needed) of this data is a big concern that has to be assured. All of the data related to the appointments has to be secured too since it can be sensible information.

3.5.4 Maintainability

To measure the systems maintainability a metric shall be used. This metric needs to be tracked during the development and kept as low as possible, in order to avoid high maintainability costs subsequently. It's obvious that the system must have a high maintainability, which means that the probability of performing a successful repair action within a given (necessary) time must be high (higher than 95%).

3.5.5 Portability

The first release of the system will be available for the mobile platform our target users use the most, IOS. Nonetheless, an Android version shall be released soon, and even a new platform may arise. So, it's crucial that the system is built in a way which makes it easy to deploy in different platforms. The portability measure consists on the cost that deploying the system in a new platform implies, and in this case, this cost should be smaller than the cost of deploying other similar apps.

4 Formal Analysis Using Alloy

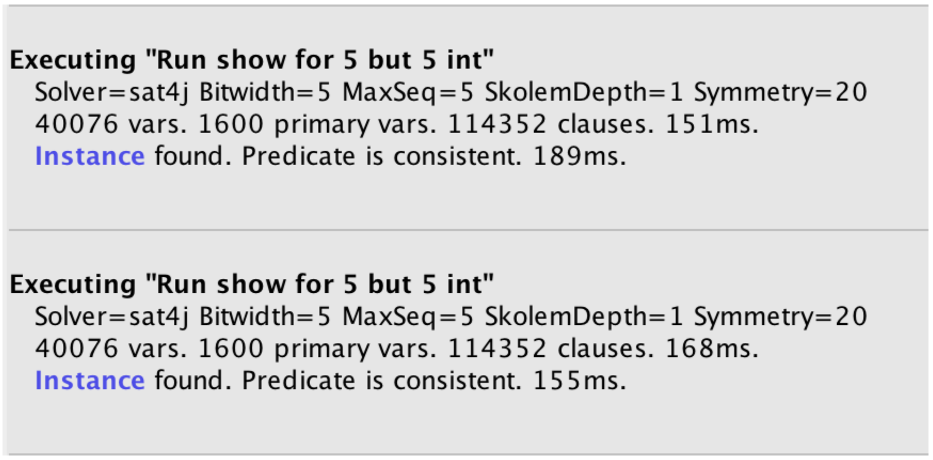
In this section is presented a possible Alloy formalization of the proposed system. This Alloy model illustrates the key concepts and entities that make up the system and the relationships between them, and is to be seen as an attempt of capturing the systems essential features. This model has the purpose of verifying if the properties defined for the system are being satisfied and if there are no constraints being violated. In this model it was considered a subset of user preferences and travel means. It was also considered that the destination of all the travel means used to get to an appointment is the same and that only the status of two fixed appointments have to be different (the fact that a flexible appointment didn't start but the Local Time is already in the possible interval that it can start was considered an implementation detail, too difficult if even possible to represent in Alloy, and so two Flexible Appointments can be in the "started" state). Although this model is a simplified version of the "real world", it is enough to show that the model stands in the scope of the project (including all of the details would only increase complexity).

```
open util/boolean
open util/integer

sig string {}

sig User {
username: one string,
email: one string,
realizes: some Appointment,
preferences: one Preferences
}

sig LocalTime{
time: one Int
}{time >= 0}

abstract sig Appointment{
description: lone string,
located: one Location,
has: some TravelMean,
time: one Time,
status: one AppointmentStatus
}

enum AppointmentStatus{
created,
started,
finished
}

sig FixedAppointment extends Appointment{}

sig FlexibleAppointment extends Appointment{
minDuration: one Int
}{minDuration > 2}

sig Time{
start: one Int,
end: one Int
}{start >= 0 end >= 0}


sig Location{
latitude: one Int,
longitude: one Int,
distance: one Int
} {distance > 0}

abstract sig TravelMean{
reaches: one Location,
active: one Bool
}

sig Car extends TravelMean{}

sig Bike extends TravelMean{}

sig PublicTransport extends TravelMean{}

sig Walking extends TravelMean{}

sig Preferences{
maxWalkingDistance: one Int,
publicTransportActive: one Bool
}{maxWalkingDistance > 0}
```

```
/*FACTS*/

fact usernameUnique{
//User username is unique
all disj u1, u2: User | u1.username != u2.username
}

fact emailUnique{
//User email is unique
all disj u1, u2: User | u1.email != u2.email
}

fact appointmentRealisedByOnly1User{
//An appointment can only be realised by one user
all disj u1, u2: User | u1.realizes & u2.realizes=none
}

fact appointmentDoesntExistWithoutUser{
//Appointment shall not exist when not associated with a User
all a1: Appointment | one u1: User | a1 in u1.realizes
}

fact timeDoesntExistWithoutAppointment{
//Time shall not exist when not associated with an Appointment
all t1: Time | one a1: Appointment | t1 in a1.time
}

fact preferencesDontExistWithoutUser{
//Preferences shall not exist when not associated with a User
all p1: Preferences | one u1: User | u1.preferences=p1
}

fact locationDoesntExistWithoutAppointment{
//A location shall not exist when not associated with an appointment
all l1: Location | one a1: Appointment | a1.located=l1
}

fact startTimeSmallerThanEndTime{
//An appointments start time has to be smaller than the end time
all t1: Time | t1.start < t1.end
```

```
}

fact minDurationHasToBeIncludedInInterval{
/*The minimum duration of a flexible appointment has to "fit"
in the time interval of the appointment*/
all fa: FlexibleAppointment | sub[fa.time.end, fa.time.start] >
fa.minDuration
}

fact FixedAppointmentsDontOverlap{
//Two different Fixed Appointments can't overlap
all disj f1, f2: FixedAppointment | (f1.time.end < f2.time.start ||
f2.time.end < f1.time.start)
}

fact ifTravelMeanIsUsedActiveIsTrue{
//If a travel mean is used it has to be active
all a1: Appointment, tv1: TravelMean | tv1 in a1.has implies
tv1.active.isTrue
}

fact travelMeanReachesAppointmentsLocation{
/*The travel mean chosen for an appointment has to reach
the appointments location*/
all a1: Appointment, tv1: TravelMean | tv1 in a1.has implies
tv1.reaches = a1.located
}

fact differentLocationsCannotHaveSameLongitudeAndLatitude{
//Two locations can't be identical
no disj l1, l2: Location | (l1.latitude = l2.latitude &&
l1.longitude = l2. longitude)
}

fact flexibleAppointmentDoesntOverlapFlexibleAppointment{
/*Two different Flexible Appointments can't overlap if
there's not enough space for the minDuration of both*/
all disj fa1, fa2: FlexibleAppointment |
(((fa1.time.start <= fa2.time.start) && (fa1.time.end <= fa2.time.end))
implies ((sub[fa2.time.end, fa1.time.start] >=
add[fa1.minDuration, fa2.minDuration]))) ||
(((fa1.time.end <= fa2.time.start) &&
(fa1.time.end <= fa2.time.end)) implies
((sub[fa2.time.end, fa1.time.start] >=
add[fa1.minDuration, fa2.minDuration])))
}

fact flexibleAppointmentDoesntOverlapFixedAppointment{
/*A Flexible Appointments and a Fixed Appointment can’t overlap if
there’s not enough space for the minDuration of the Flexible one*/
all disj f: FixedAppointment, fa: FlexibleAppointment |
sub[fa.time.end, f.time.end] > fa.minDuration
}

fact travelMeansNeedToMeetPreferences{
//Simplified to two preferences, shows that the model stands nonetheless
all u1: User, pt1: PublicTransport | (pt1 in u1.realizes.has) implies
u1.preferences.publicTransportActive.isTrue
all u1: User, w1: Walking | ((w1 in u1.realizes.has)) implies
(u1.preferences.maxWalkingDistance >= w1.reaches.distance)
}

fact appointmentStatusCoherence{
//The Appointment status has to be coherent with the local time
all a: Appointment | a.status = created <=>
(one lt: LocalTime | lt.time < a.time.start)
all a: Appointment | a.status = started <=>
(one lt: LocalTime | (a.time.start <= lt.time) &&
(lt.time <= a.time.end))
all a: Appointment | a.status = finished <=>
(one lt: LocalTime | a.time.end <= lt.time)
}

fact minDurationCoherence{
//Not a requirement, but needed to avoid overflow in Alloy
all fa: FlexibleAppointment | fa.minDuration < 8
}

fact SingletonClasses{ // Singletons
#LocalTime=1
```

```
}



pred show{
#User=1
#FlexibleAppointment=2
#FixedAppointment=2
#TravelMean=5
}
run show for 5 but 5 int
```



```
Executing "Run show for 5 but 5 int"
   Solver=sat4j Bitwidth=5 MaxSeq=5 SkolemDepth=1 Symmetry=20
   40076 vars. 1600 primary vars. 114352 clauses. 151ms.
   Instance found. Predicate is consistent. 189ms.

Executing "Run show for 5 but 5 int"
   Solver=sat4j Bitwidth=5 MaxSeq=5 SkolemDepth=1 Symmetry=20
   40076 vars. 1600 primary vars. 114352 clauses. 168ms.
   Instance found. Predicate is consistent. 155ms.
```

Figure 10: Execution Details

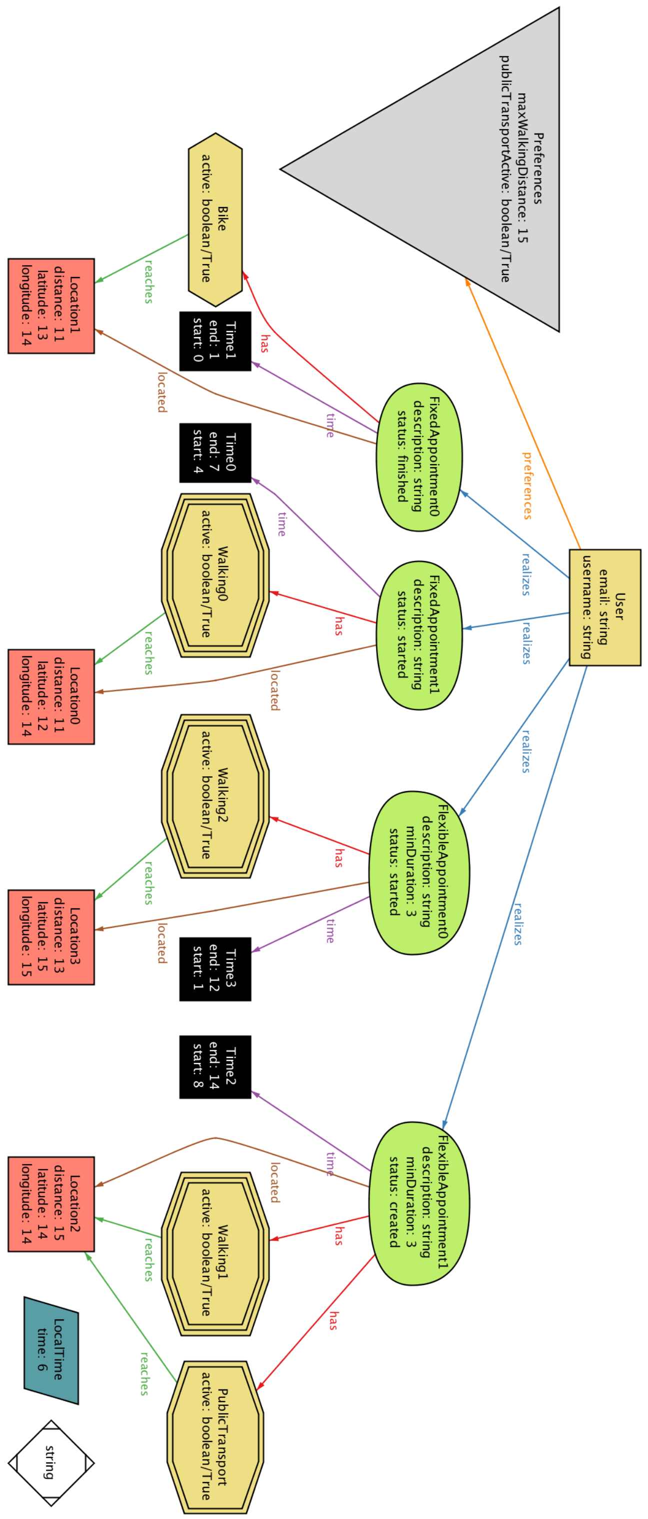


Figure 11: Generated World 1

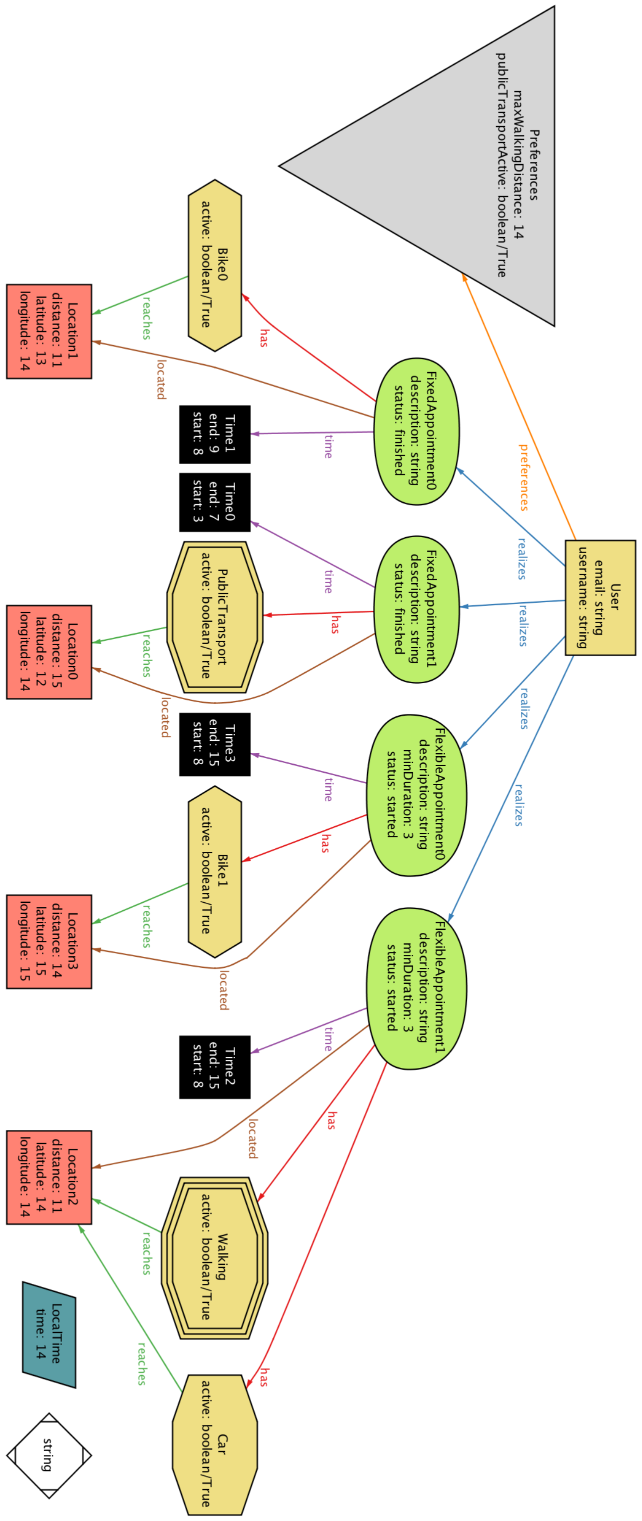


Figure 12: Generated World 2

5 Effort Spent

5.1 Francisco Cristóvão

DATE	TASK	HOURS
03/10/2017	Requirements Analysis	1
08/10/2017	Domain Model and Class Diagram	1,5
09/10/2017	Purpose and Document Structure	1
11/10/2017	Scope and Purpose	1
12/10/2017	User, Software and Hardware Interfaces, Product Functions and User Characteristics	4,5
13/10/2017	Software System Attributes	3
16/10/2017	Goals and Requirements, Use case Description	2
17/10/2017	Use Case Description and User Interface	4
18/10/2017	Goals, Requirements and Domain Assumptions	2
19/10/2017	Use Case Description, Performance Constraints	2
20/10/2017	Goals and requirements, Use-Cases, UML Diagrams	3
21/10/2017	BPMN Diagrams, Goals and Requirements, User Interface	2
24/10/2017	Alloy	2
25/10/2017	Alloy	3
26/10/2017	Alloy and General Document Aspects	3
27/10/2017	Alloy	6
28/10/2017	Alloy and Final details	4
TOTAL	45	

5.2 Samsom Tsegay Beyene

DATE	TASK	HOURS
03/10/2017	Requirements Analysis	2
05/10/2017	First Draft of Scope and Purpose	2
08/10/2017	Purpose	2
10/10/2017	Document Structure	3
12/10/2017	Product Perspective and User Characteristics	3
13/10/2017	Communication Interface	1
14/10/2017	Use Case Diagrams	3
16/10/2017	Goals and Requirements, Use case Description	2
19/10/2017	BPMN Diagrams	2
21/10/2017	Alloy Signatures	2
27/10/2017	RASD Final Review	2
TOTAL	24	

6 Used Tools

- ShareLateX
- StarUML 2.8.0
- Sketch 47.1
- Alloy Analyzer 4.2

7 References

References

[1] https://belitsoft.com/php-development-services/software-requirements-specification-document-example-international-standard

[2] https://en.wikipedia.org/wiki/Portability_testing

[3] https://en.wikipedia.org/wiki/Maintainability

[4] https://en.wikipedia.org/wiki/Software_reliability_testing

[5] Nielsen, Jakob, Budiu, Raluca. *Mobile Usability, 1st Edition*

[6] http://www.enforcive.com/regulatory-compliance-software

[7] https://data.oecd.org/pop/working-age-population.htm