Travlendar+

Requirement Analysis Specification Document

Calzavara Filippo, Filaferro Giovanni, Benedetto Maria Nespoli

Version 1.0.0

Contents

1 Introduction

1.1 Purpose

This document is the Requirements Analysis Specification Document (RASD) for Travlendar+, the system assigned to be developed for the mandatory project. The document describes the characteristics of the system, both functional and non-functional requirements for the software, design constrains and system interfaces. It is addressed to be read by the developers that will implement the final system.

1.2 Scope

The aim of this project is to create a smart calendar-based application that people can use to reduce delays at the appointments and supporting the users by identifying the best mobility option. People create appointments and they get notified in case they will not be on time for it. Moreover, the system needs to take into account weather forecast as well as strike days. Furthermore, the users should be able to adapt settings based on their preferences, such as walking-time constrains or a more eco-friendly way of transportation, and the possibility to define flexible appointments that given a window of time and a duration an arrangement of the schedule is proposed. Finally, the system should provide the user of a ticket or locate the closest car/bike sharing vehicle if it is the mean of transportation chosen.

1.2.1 Goals

- [G0] Allow people to use the app without a login function
- [G1] Allow people to view the daily schedule with coming up events at the top
- [G2] Allow people to view previous events
- [G3] Allow people to view the details of each event
  - [G3.1] Allow people to get travel information for a particular event
    - [G3.1.1] Allow people to get the ETA to the appointment
  - [G3.2] Allow people to explore a specific route
    - [G3.2.1] Allow people to purchase the transit ticket
    - [G3.2.2] Allow people to open the sharing services/ride provider's app and call the vehicle
  - [G3.3] Allow people to eventually view the purchased transit ticket
- [G4] Allow the users to create an event
  - [G4.1] Allow the user to define a name/description of the appointment

  - [G4.2] Allow the user to define the location
  - [G4.3] Allow users to set a flexible timing
    - [G4.3.1] Allow the user to select a time window in which the event will occur
    - [G4.3.2] Allow the user to select how long the event will last
  - [G4.4] Allow the user to set a starting time and ending time on non-flexible events
  - [G4.5] Allow the user to repeat the event regularly throughout the week
  - [G4.6] Allow the user to assign the event to a specific calendar
  - [G4.7] Allow the user to define the mean of transportations that he/she wants to use for the event
- [G5] Allow people to see daily events on a map
  - [G5.1] Allow the user to have information about the status of the journey to make in order to reach the appointment's location
  - [G5.2] Allow the user to see previous daily events on a map
- [G6] Allow the user to set preferences
  - [G6.1] Allow the user to set the eco-friendly mode
  - [G6.2] Allow the user to set the walkable distance limit
  - [G6.3] Allow the user to set the maximum biking range
  - [G6.4] Allow the users to set the owned bike and/or bike sharing providers
  - [G6.5] Allow the users to set a time range in which they can take public transits
  - [G6.6] Allow the users to set if they own a car and/or their favorite car sharing providers
  - [G6.7] Allow the users to set the preferred ride sharing service
- [G7] Allow the users to create calendars
  - [G7.1] Allow the users to color calendar
  - [G7.2] Allow the users to change calendar's color and name
- [G8] Notify the user that it's time to leave for the appointment

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Definition	Explanation
Appointment	A period of time in which something take place at a certain time
Travlendar+	The name of the platform to develop
Event	A synonym for appointment
System	A synonym of Travlendar+
Up Coming Event	A particular event that is expected to occur soon
Up Next Event	A synonym of Up Coming
User	A potential utilizer of this project
Ride	A service performed by a ride-sharing company and Cabs

1.3.2 Acronyms

Acronym	Explanation
GUI	Graphic User Interface
ETA	Estimated time of arrival
API	Application programming interface
RASD	Requirement Analysis and Specification Document
PNR	Passenger name record
QR	Quick Response Code
OS	Operative Systemk

1.3.3 Abbreviations

Abbreviation	Explanation
App	A synonym of Travlendar+
[Gn]	N-goal
[Dn]	N-domain assumption
[Rn]	N-functional requirement

1.4 Revision history

- 1.0.0 - Initial Version (16/10/2017)

1.5 Reference Documents

- Specifications document "Mandatory Project Assignments.pdf"
- ISO/IEC/IEEE 29148 - IEEE Standard on Requirements Engineering
- Software Abstractions, MIT Press

1.6 Document Structure

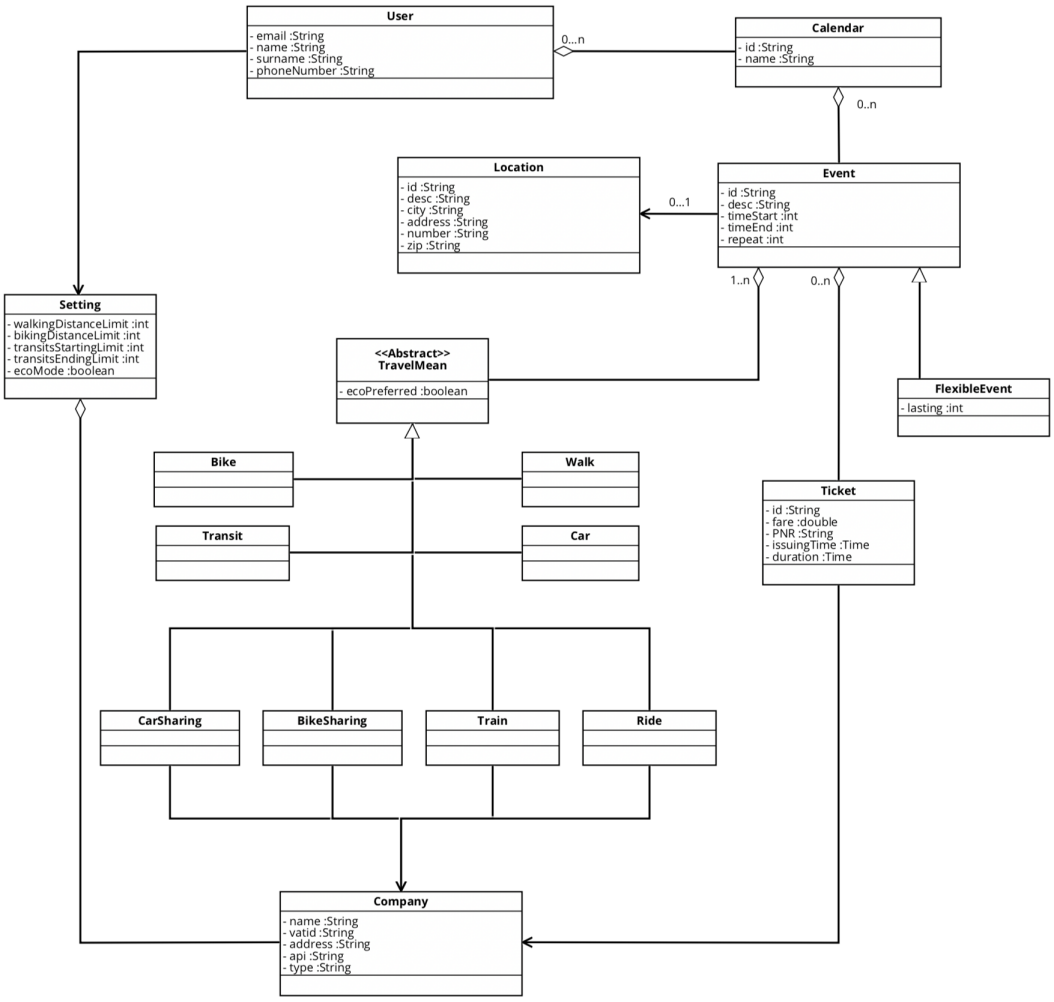
The paper includes six areas. The first one, is composed by the introductory information provided in order to give an orientative view of what this document is about. The second one, provides an overall description of the project including the features, the characteristics and the constraints that its development implies. The main body of this paper is contained in the third section, which assert the requirements and the design constraints. The paper is equipped with a fourth section that contains a formal analysis using Alloy. Finally, a fifth and a sixth section allows the reader to learn the effort spent on this project by each team member and the references made in the whole paper.

2 Overall Description

2.1 Product perspective

Travlendar+ is a standalone system that provides the functionality described in the Product Function Section 2.2. It would implement external services in order to offer weather forecast, maps services, sharing systems information, transits and strike information.

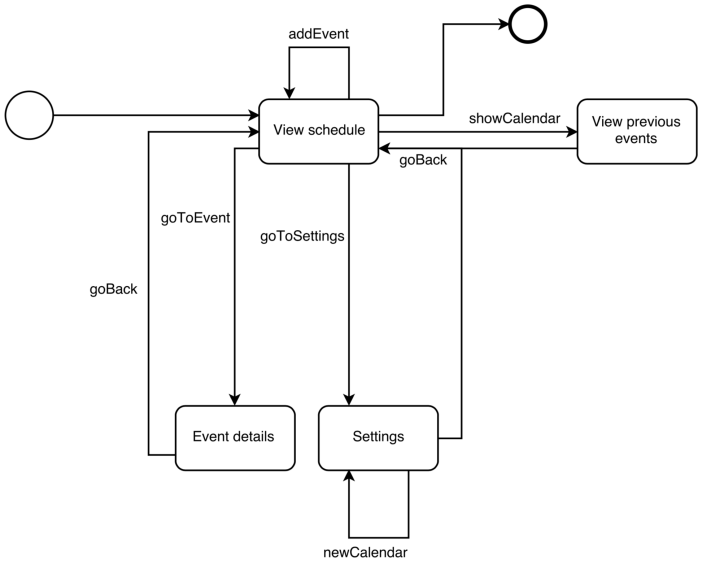
2.1.1 Class Diagram



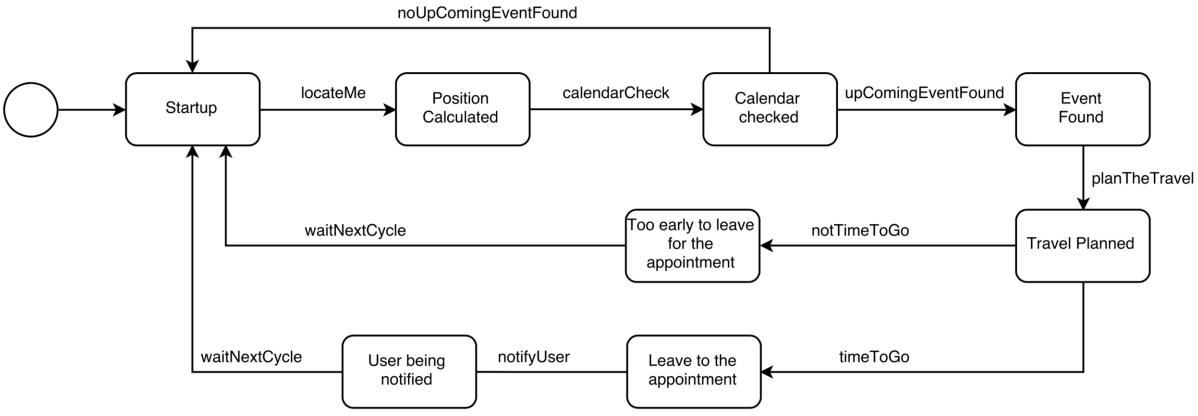
2.1.2 Statechart

The system is based on two different modes: The active function and the passive function. The first is in place when the person is actively using the application and its functionality. The latter is a background service that runs in order to keep track of the position, other information and it attract user's attention when a notification is risen.

Active mode:



Passive mode:



2.2 Product functions

Each goal described in Section 1.2.1 is function that will be implemented in the system. We report below the functionalities that require a further explanation and the boundaries between the system to be developed and thirty party system.

2.2.1 Allow people to purchase and view the transit ticket

The ticket buying system will use regular text messages to send a purchasing request at the local transit company which reply to our app with the PNR code of the ticket. This choice of designing was made to ensure compatibility among every transit provider and to allow users to pay it by charging the cost of the ticket on their cell phone credit/bill, letting the carrier provider handling the payment.

When the purchasing is completed, a QR code representing the ticket's PNR will be displayed on the user's screen.

2.2.2 Overlapped events

While creating an event, the system automatically rearrange the schedule in order to fit all the events in the calendar. In case an event overlaps another event of the same calendar, an exception is thrown. In case an event overlaps one event of another calendar nothing happen.

Moreover, an error is shown when the user tries to add an event that could not be reached in time.

2.3 User characteristics

This application is designed to be used by any person that would like to have a more organized schedule

2.4 Assumptions, dependencies and constraints

We assume that:

1. The client is using the app through a new generation smartphone
2. The mobile device has a GPS system
3. The mobile device has a regular 3G internet connection available
4. The software is designed to be utilized in Italy, precisely in the region of Lombardy
5. Public transits companies provide a sms-based purchasing system
6. The user has already logged in to his/her device account (Google or Apple) so that we can identify him/her by using their unique identifier.

7. Vehicle-sharing/ride-sharing companies' will provide APIs to locate the nearest vehicle available
8. Weather forecast are provided from an external service through API
9. Calculating the shortest path follows a time-based criteria instead of a cost-based one

2.4.1 Interfaces to other applications

At the first stages of the project, Travlendar+ will not release API interfaces to the public

2.4.2 Public transit pass

The system will not take into consideration whether the person owns a transit's pass, however it will always allows to buy one or more transit tickets if asked by the user

2.4.3 Domain assumption

- [D1] App data are stored remotely and retrieved by multiple devices
- [D2] Up coming events' section needs to contain appointments planned to happen 2h before
- [D3] Transportation information are not shown for the past events
- [D4] Indications on the ETA, calculated mean of transportation and time-to-leave are calculated on the current approximative position
- [D5] User's position is retrieved using GPS system
- [D6] Position is re-calculated every 2 hours, or when a significant change in GPS position is discovered
- [D7] The map can be browsed by the user
- [D8] The localization service is available and functionally working
- [D9] The map service is reliable
- [D10] The button to choose the calendar is, by default, the last calendar chosen, or the first calendar if the user has never added an event
- [D11] While creating an event, the system must take into account whether the eco mode has been turned on
- [D12] The map can be browsed by the user
- [D13] Every preference agrees with the actual record saved in the database

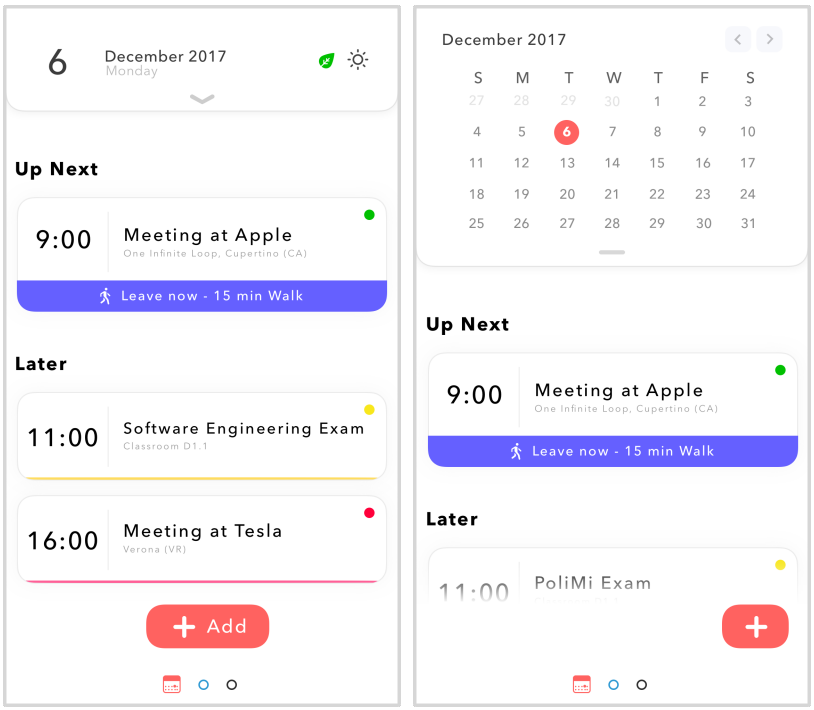
- [D14] Allowed services have been preinstalled
- [D15] The user has always at least one calendar
- [D16] Each calendar belongs to only one user
- [D17] Each event belongs to only one calendar
- [D18] The user won't insert overlapping events

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Home Screen

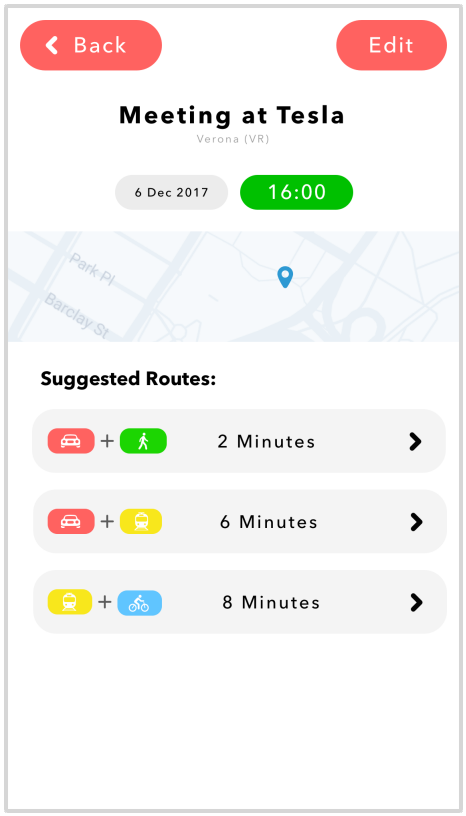


Here is shown the main page that will appear once opened the app. People can see directly their upcoming events of the day. By pulling down the top view (where 16 December 2017 is written) they can access the calendar view (Right Picture) and by tapping on a particular day the UI re-adapts to that day showing its regarding events. Onto this page users may see wether the ECO-Friendly mode is enabled (Green Leaf on top) and the current weather. Furthermore, on the mockup here proposed the reader may see a suggested route and an ETA for the up coming event. It is also shown a little dot at the upper right side of the cell that becomes:

- **Green** when the user is on time/early.
- **Yellow** when the user will be late if he/she doesn't hurry up.
- **Red** when they will centrally be late or there is a problem in that route.

In each cell there is a small bar on the bottom whom color depends on calendar's color where the event is saved to.

Event Detail

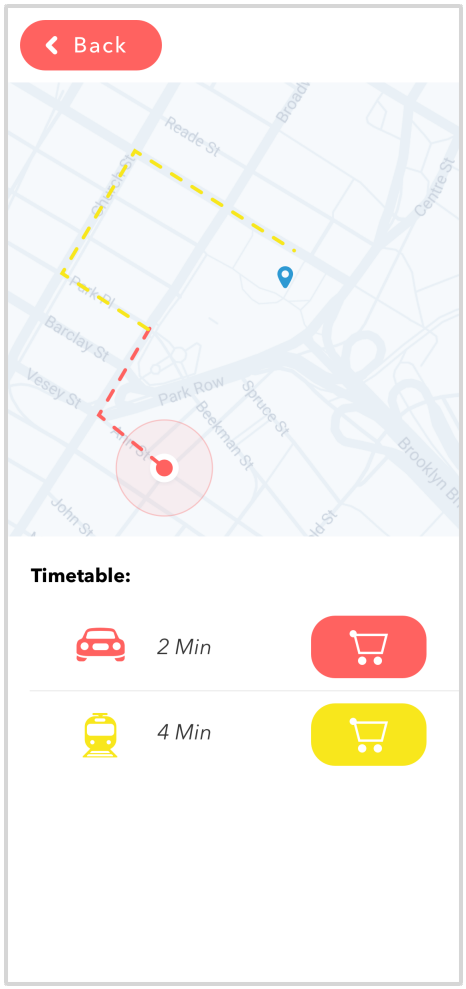


This is the place where, once tapped on an event in the Home Screen, people can have a fast look at its more interesting details:

- Basic details of the event such as day, time, position and name.
- There will be a map view where the location of the event will be shown.
- Suggested routes (compound ones too) to take based on the user preferences

From this view the user is allowed also to edit the event by tapping the "Edit" button.

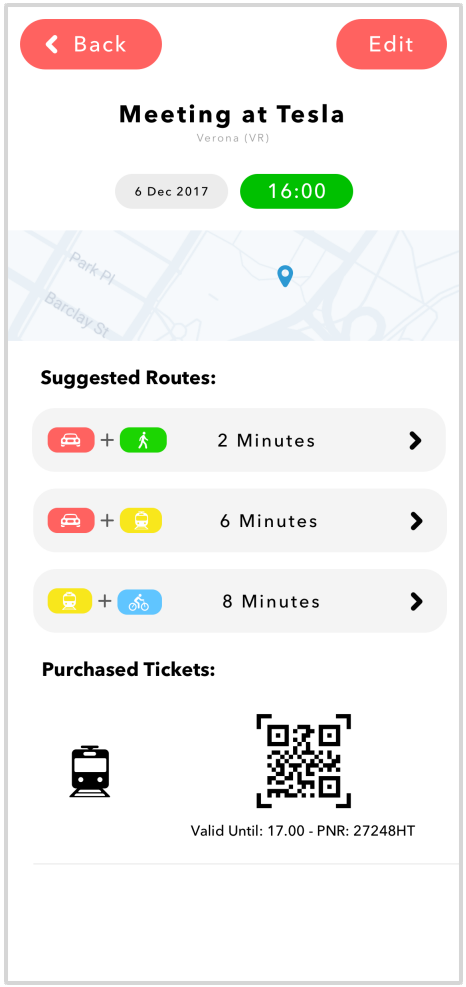
Event Detail / Route Detail



Just by tapping to one suggested route from "Event Detail" view, the app will present to the user this view in which they can look at the routes to reach the event.

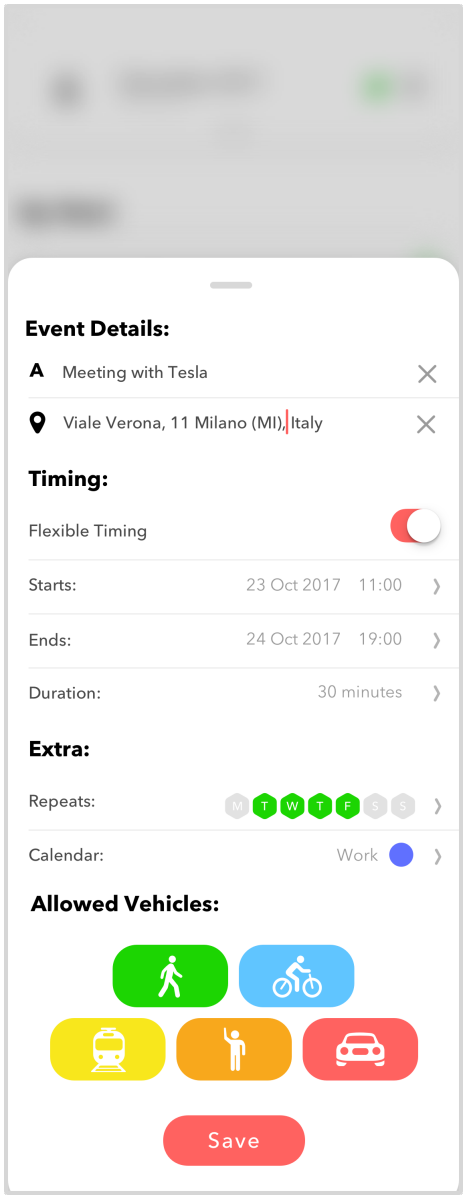
In particular all the parts composing the route are shown on map and just by tapping the button on the right of each cell the app will prompt the user to buy a ticket.

Event Detail / Ticket Bought



When a user buys a ticket it appear in this view which is the main "Event Detail". A QR code containing the PNR code of the ticket is shown.

Add Event

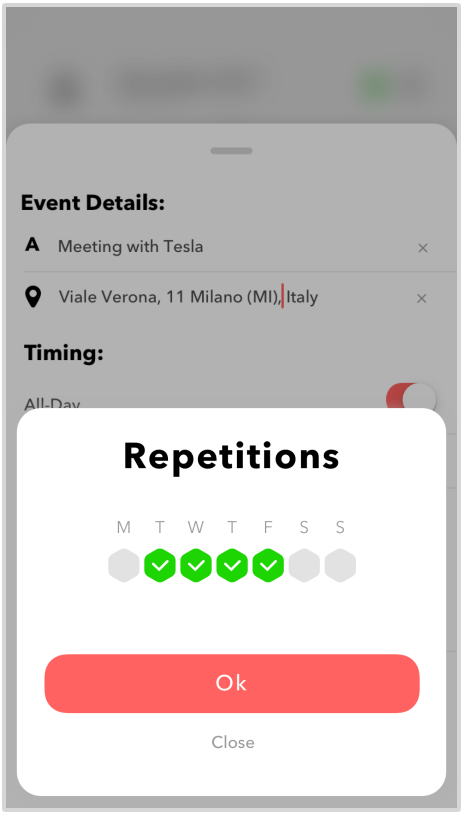


On the main view, once the user taps the add button this pop-up is shown. From this area, the person is able to choose:

- Name of the event

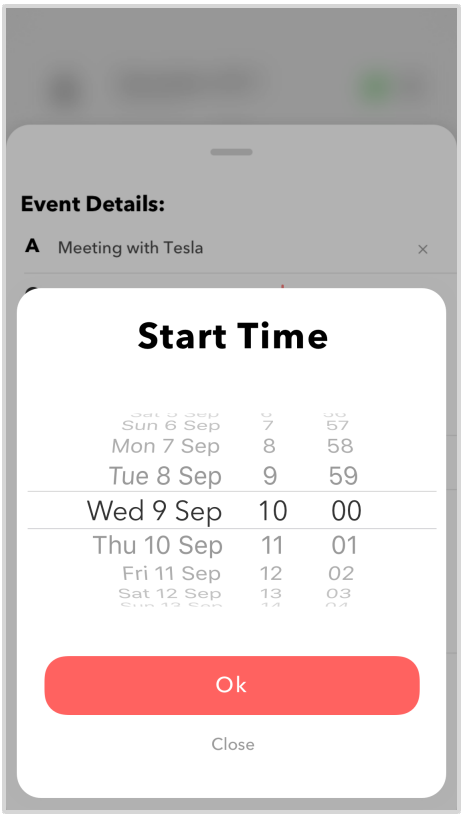
- Location of the event
- Start time and end time
- By toggling the "Flexible Timing" switch a new cell called "Duration" would appear so that they can just say indicate the how long for this event will last, given a time window. This feature is useful to set breaks during the day.
- Repetition days and a calendar where the event will be saved
- Vehicles and travel means to compute the best route based on them

Add Event / Repetitions



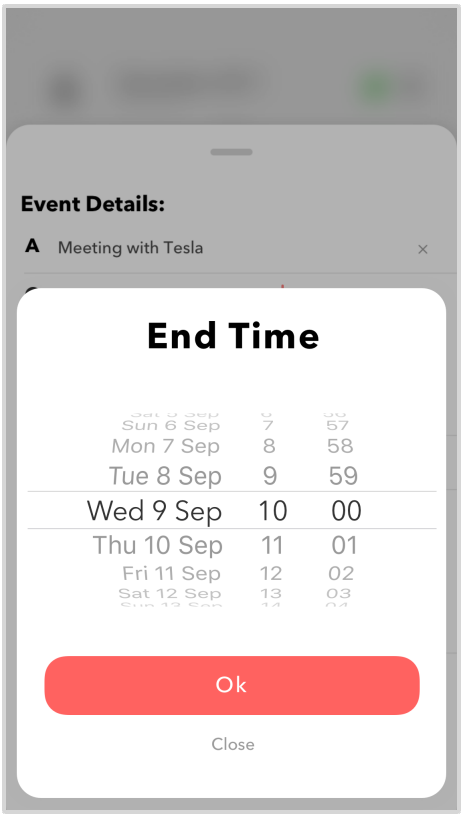
Here is shown the pop-up that will appear once the user taps on the "Repeats" cell from the "Add Event" screen.

Add Event / Start Date Time



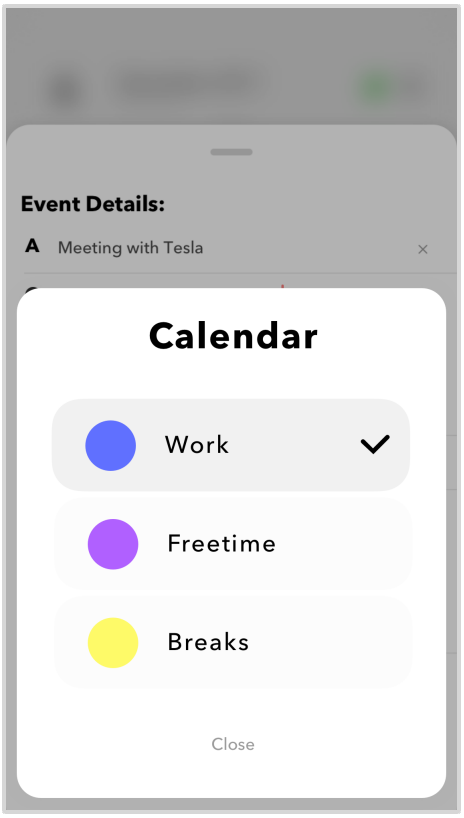
This is the pop-up that will appear once the user taps on the "Starts" cell from the "Add Event" screen.

Add Event / End Date Time



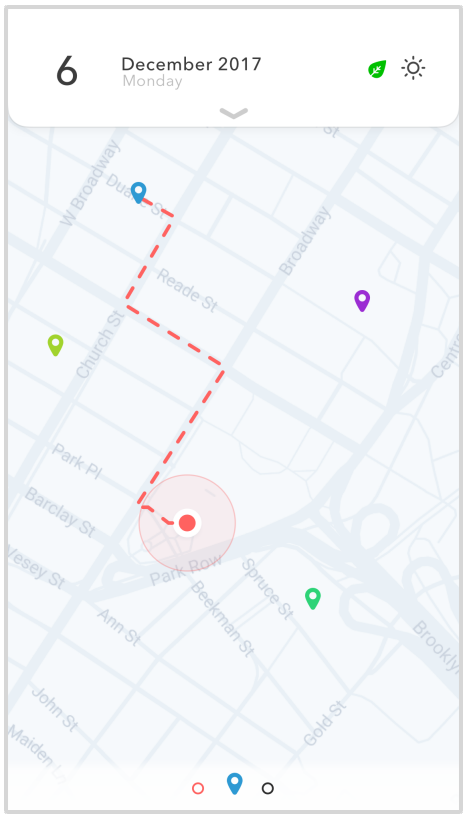
This is the pop-up that will appear once the user taps on the "Ends" cell from the "Add Event" screen.

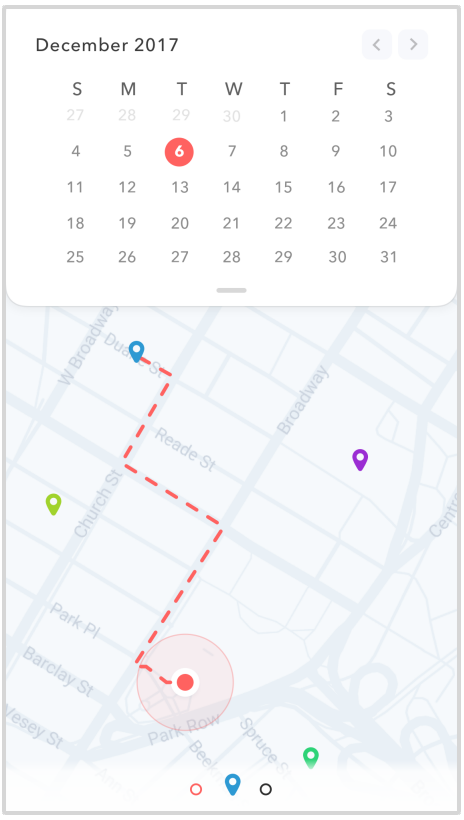
Add Event / Calendar Pick



This is the pop-up that will appear once the user taps on the "Calendar" cell from the "Add Event" screen.

Map


6
December 2017
Monday
Broadway
Reade St
Church St
Park Pl
Barclay St
Park Row
Spruce St
Beekman St
Ann St
John St
Gold St

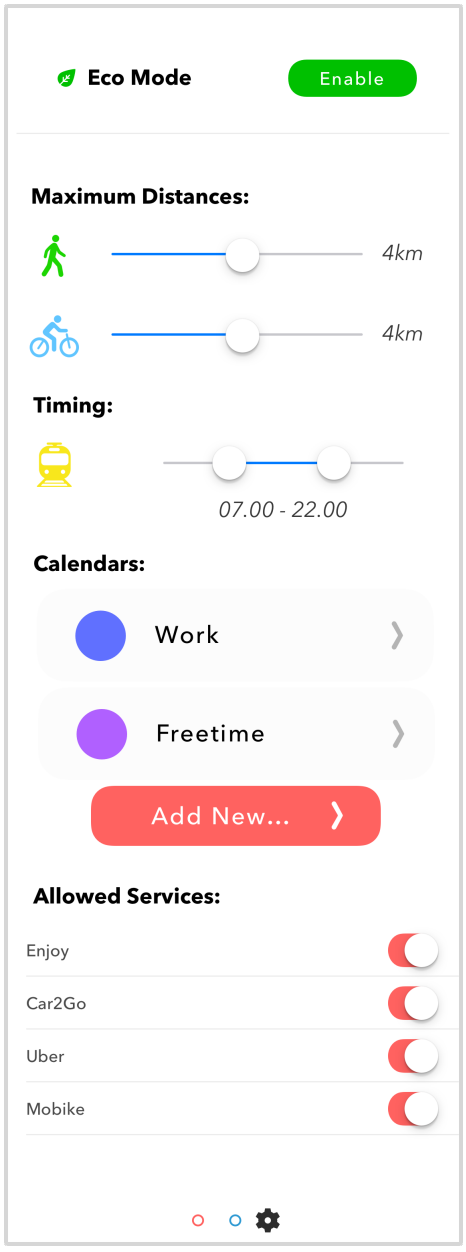


Swiping right from the main view or by tapping on the bottom icon the app will present this view.

Here the user can have a look at their daily events on a map. The screen shows also the suggested route from the actual position to the first event location.

By pulling down the calendar, the user can change day and have a look at those specific events.

Settings

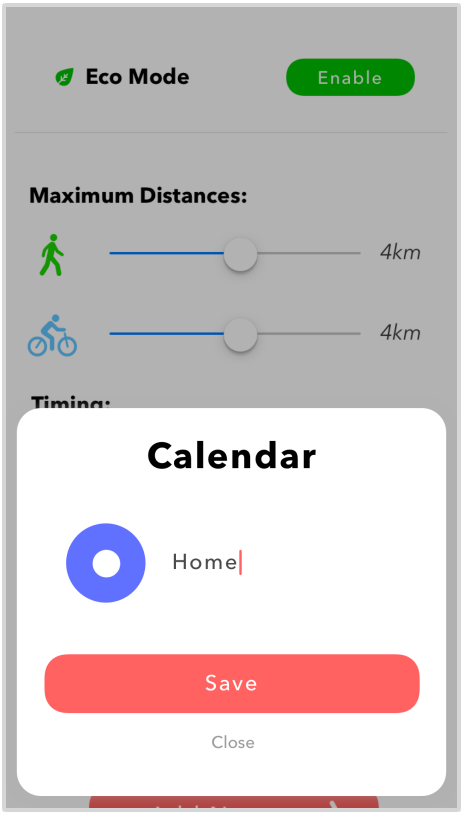


Swiping right from the map view or by tapping on the bottom icon the app will present this view.

Here we can find five sections:

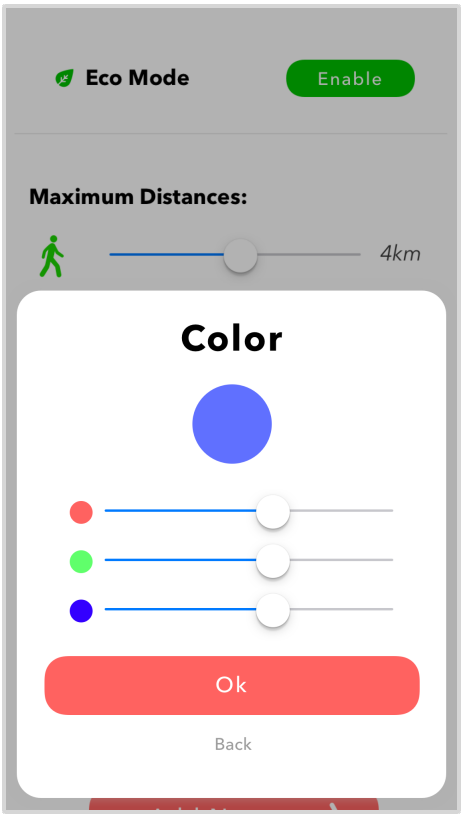
1. **Eco Mode:** By enabling it the system will differently weight routes that require fossil fuel based vehicles
2. **Maximum Distances:** User is allowed to set a maximum amount of distance by walk or bike so that the system will re arrange its calculations based on this new values.
3. **Timing:** Here the user can set the time window where transits will be available
4. **Calendars:** In this section the user can edit his existing calendars or create a new one by tapping "Add New..." button
5. **Allowed Services:** Here the user can just activate or disable vehicle-sharing services (such as Enjoy or Mobike) so that the app will not consider them on paths calculations.

Settings / Add Calendar



This is the view that will be shown when the user wants to add a new calendar.

Settings / Add Calendar / Color



This is the view where the user is allowed to edit the color of an existing or new calendar just by adjusting the RGB Sliders.

3.1.2 Hardware Interfaces

The system will require, in order to work properly, one or more dedicated servers in which the server management software will run.

3.1.3 Software Interfaces

The system will need:

- A server database where it will store user data like calendars, events and preferences.
- A client database where it will store user data on device so that the app will work offline too.
- The App that will be installed on the users's smartphones.

3.1.4 Communication Interfaces

In order to establish a communication between the app and the server, there will be the need to build an API that lets the app to store and/or retrieve data from the database (located server-side). The connection will be created though the HTTPS protocol and the data will be exchanged with a JSON encoding.

3.2 Functional Requirements

3.2.1 [G0] Allow people to use the app without a login function

- [R1] When the app is lunched for the first time, the system recognizes the new users and create a record of their data to be locally updated on the phone
- [D1] App data are stored remotely and retrieved by multiple devices

3.2.2 [G1] Allow people to view the daily schedule with coming up events at the top

- [R2] The system must be able to provide the list of daily events scheduled by the user
- [R3] The events must be divided between "Up coming" and "Later on"
- [R4] Information of event's description, calendar saved and time must always be shown, while the location of the event is optional
- [R5] Button add event must always be visible
- [D2] Up coming events' section needs to contain appointments planned to happen 2h before

3.2.3 [G2] Allow people to view previous events

- [R6] A calendar screen must be shown
- [R7] Information of event's description, calendar saved and time must always be shown, while the location of the event is optional
- [R8] Button add event must always be visible
- [D3] Transportation information are not shown for the past events

3.2.4 [G3] Allow people to view the detail of each event

- [R9] ETA and trip information must be provided
- [R10] Tickets services must be shown
- [R11] Different suggested routes can be explored, if available
- [R12] The system must provide a way to buy tickets or book a sharing service vehicle
- [R13] The system must show the purchased ticket
- [R14] The paths color in the map changes accordingly to the means of transportation

- [D4] Indications on the ETA, calculated mean of transportation and time-to-leave are calculated on the current approximative position
- [D5] User's position is retrieved using GPS system
- [D6] Position is re-calculated every 2 hours, or when a significant change in GPS position is discovered
- [D7] The map can be browsed by the user
- [D8] The localization service is available and functionally working
- [D9] The map service is reliable

3.2.5 [G4] Allow the users to create an event

- [R15] Information of event's description, calendar saved and time must always be provided, while the location of the event is optional
- [R16] The system must ask for the description and the location of the appointment
- [R17] The system must ask whether the event should be time-flexible. In this case it must require a time window and how long the appointment will last
- [R18] Button for repeating an event weekly must be shown
- [R19] Button for choosing the calendar in which the event will be saved must be shown
- [R20] A multiple choice list of available means of transportation must be included
- [D10] The button to choose the calendar is, by default, the last calendar chosen, or the first calendar if the user has never added an event
- [D11] While creating an event, the system must take into account whether the eco mode has been turned on
- [D17] Each event belongs to only one calendar
- [D18] The user won't insert overlapping events

3.2.6 [G5] Allow people to see daily events on a map

- [R21] All the events must be shown on a interactive map
- [R22] The map must show a path to the next event
- [D12] The map can be browsed by the user
- [D5] User's position is retrieved using GPS system
- [D9] The map service is reliable

3.2.7 [G6] Allow the user to set preferences

- [R23] The system must provide of a way to change the maximum distance that could be traveled by bike or walk
- [R24] The system must provide a time window in which public transits can be taken
- [R25] All the available calendars must be shown
- [R26] The system must allow to toggle the available external sharing services
- [D13] Every preference agrees with the actual record saved in the database
- [D14] Allowed services have been preinstalled

3.2.8 [G7] Allow the users to create calendars

- [R27] The app should allow the user to add a new calendar
- [R28] The system allows to change the name of a calendar and eventually its color
- [D15] The user has always at least one calendar
- [D16] Each calendar belongs to only one user

3.2.9 [G8] Notify the user that it's time to leave for the appointment

- [R29] The app must send a notification when it's time to leave for the appointment.
- [R30] The system must take into account the means of transportation that the user has selected
- [D5] User's position is retrieved using GPS system

3.3 Scenarios

3.3.1 Scenario 1

Mike was invited to John's Party in Los Angeles and he really wants to go there. Unfortunately he has a lot of memory problems... That's the reason why John told him to use Travlendar+.

Mike picked up his smartphone and installed it, added the event details on the main calendar and saved!

Later he received a notification that it was time to leave and so he could arrive on time at his friend's birthday.

3.3.2 Scenario 2

Steve is always too busy and he participates to a huge variety of events so he needs to create new calendars in which he can store them. He also cares a lot on tidiness he wants his calendars have great and appealing colors in order to focus on his events in a better way.

3.3.3 Scenario 3

Benedetto is a convinced environmentalist. He wants to make the world a better place by reducing carbon emissions. For this reason he downloaded and added all his events in Travlendar+. He then enabled "Eco Mode" and now he can really help the environment.

3.3.4 Scenario 4

Filippo is a man who doesn't like to walk and neither use a bike therefore he wants to minimize his effort to reach an event.

He now installed Travlendar+ on his smartphone and set "Maximum Distance" to the minimum value for bike and walk, so he now doesn't need to struggle to reach an appointment.

3.3.5 Scenario 5

Giovanni likes to travel a lot, for this reason he started using this app.

He likes a lot to reach locations using public transport or rides, but he is under age to be allowed to use car sharing providers so he just disabled suggestions for car sharing services right on the app and now he is happy more than ever.

3.4 Definition of use cases

3.4.1 CreateCalendar

Allow the user to create a new calendar with a name and a color chosen by the user

Name	CreateCalendar
Actors	User
Goals	[G7]
Input conditions	None
Flow of Events	1. The users swipes to reach the settings tab 2. Under the Calendar section, the user choses "Add new" 3. A pop-up is shown with a name and color input 4. The user chooses the calendar name and its color 5. The system records the new calendar in the database
Output conditions	The calendar is saved in the database
Exceptions	None

3.4.2 AddStandardEvent

Allow the user to insert an event in the calendar

Name	AddStandardEvent
Actors	User
Goals	[G4]
Input conditions	None
Flow of Events	1. The user selects "Add event" from the "schedule" view 2. A pop-up is shown with a form to fill out 3. The user chooses the location and a description 4. The user selects the starting and ending date-time 5. Eventual repetitions during the week and type of calendar are chosen 6. Preferred means of transportation are selected 7. The system checks if the appointment can be reached in time with the user's preferred vehicles 8. The system rearrange flexible events in order to fit the schedule 9. Event's details are shown
Output conditions	• The event is added to the calendar • Flexible events are rearranged to fit the schedule
Exceptions	Users are notified immediately with a warning if their event will not be reached on time

3.4.3 AddFlexibleEvent

Allow the user to insert a flexible event in the calendar

Name	AddFlexibleEvent
Actors	User
Goals	[G4], [G4.3]
Input conditions	None
Flow of Events	1. The user selects "Add event" from the "schedule" view 2. A pop-up is shown with a form to fill out 3. The user choose the location and a description 4. The user toggle the switch for a Flexible Event 5. The user selects the starting and ending date-time window 6. The user choose the duration of the event 7. Eventual repetitions during the week and type of calendar are chosen 8. Preferred means of transportation are selected 9. The system checks if the appointment can be reached in time with the user's preferred vehicles 10. The system rearrange this and other flexible events in order to fit the schedule 11. Event's details are shown
Output conditions	• The event is added to the calendar • Flexible events are rearranged to fit the schedule
Exceptions	Users are notified immediately with a warning if their event will not be reached on time

3.4.4 EditPreferences

Allow the users to edit their settings: preferred vehicles (cars, bikes, public transportation, trains, sharing services), depending on the weather, if help lowering carbon emission or not, and specify a preferred lunch time

Name	EditPreferences
Actors	User
Goals	[G6]
Input conditions	None
Flow of Events	1. The user selects the "Settings" tab 2. A window with all the available settings is displayed 3. The user chooses preferences 4. The settings are automatically saved
Output conditions	The preferences are saved into user's profile
Exceptions	None

3.4.5 BuyTickets

Allow the user to buy public transportation tickets directly from the application.

Name	BuyTickets
Actors	User
Goals	[G3.2.1]
Input conditions	• The event has already been created • The user is seeing a suggested route • The suggested route has a public transit option
Flow of Events	1. The user taps on the "Purchase ticket" button 2. The text message app is displayed in foreground with a predefined text and the local transit company number as receiver 3. The user sends the text message 4. The user receives a reply from the local transit company with a PNR code 5. The text message sent from the local transit company is read by the application
Output conditions	The ticket is recorded in the database
Exceptions	The purchased is aborted if the user doesn't have enough credit on the phone

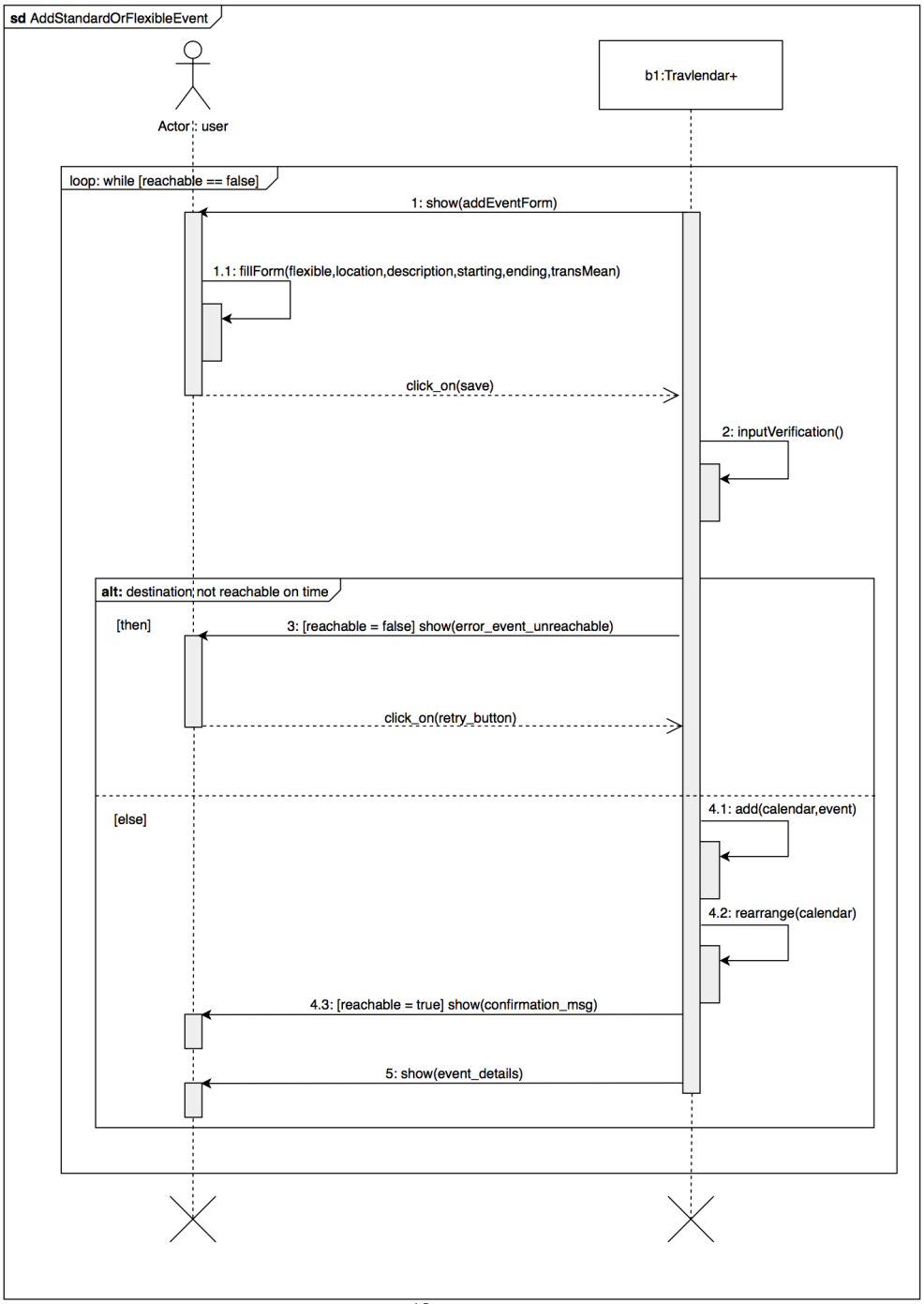
3.4.6 NotifyUser

When an upcoming event is detected, the system sends a notification to the user

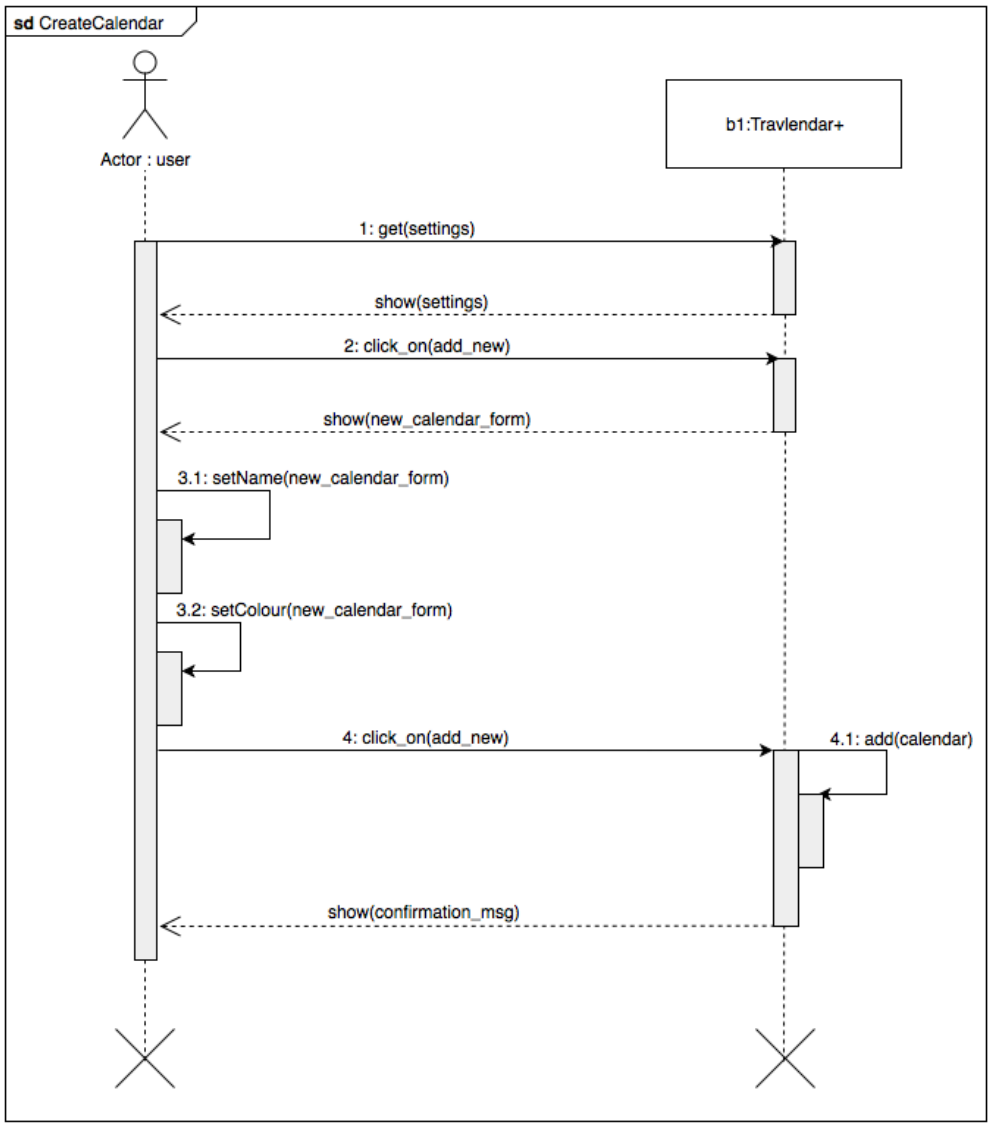
Name	NotifyUser
Actors	User, System
Goals	[G8]
Input Conditions	The event was created by the user
Flow of Events	1. The system checks the current user position 2. The system checks wether there is an upcoming event 3. When an event is found the system plans the travel 4. If it's time to go, the system sends the notification to the user
Output conditions	The user is now aware to leave to the appointment
Exceptions	• The system waits for the next cycle if there is no upcoming event • The system waits for the next cycle if it's too early to leave to the appointment

3.5 Sequence Diagrams

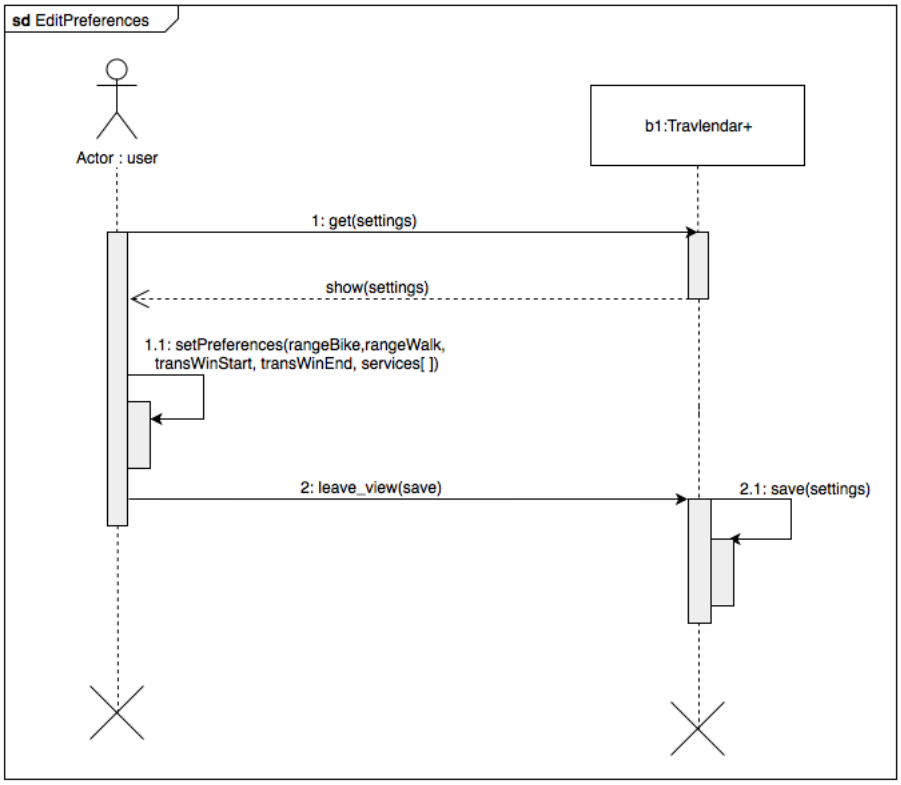
3.5.1 AddStandardOrFlexibleEvent



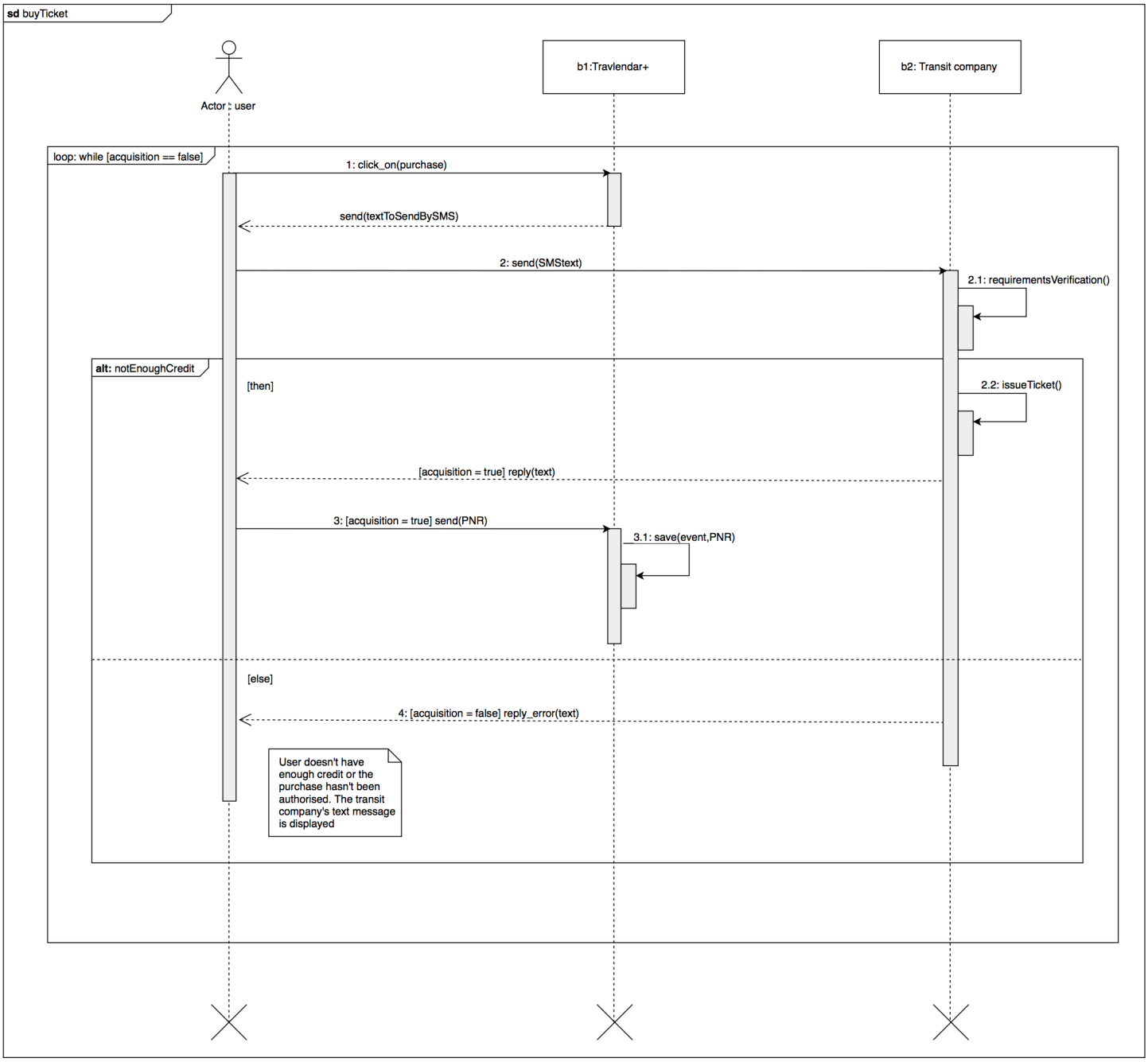
3.5.2 CreateCalendar



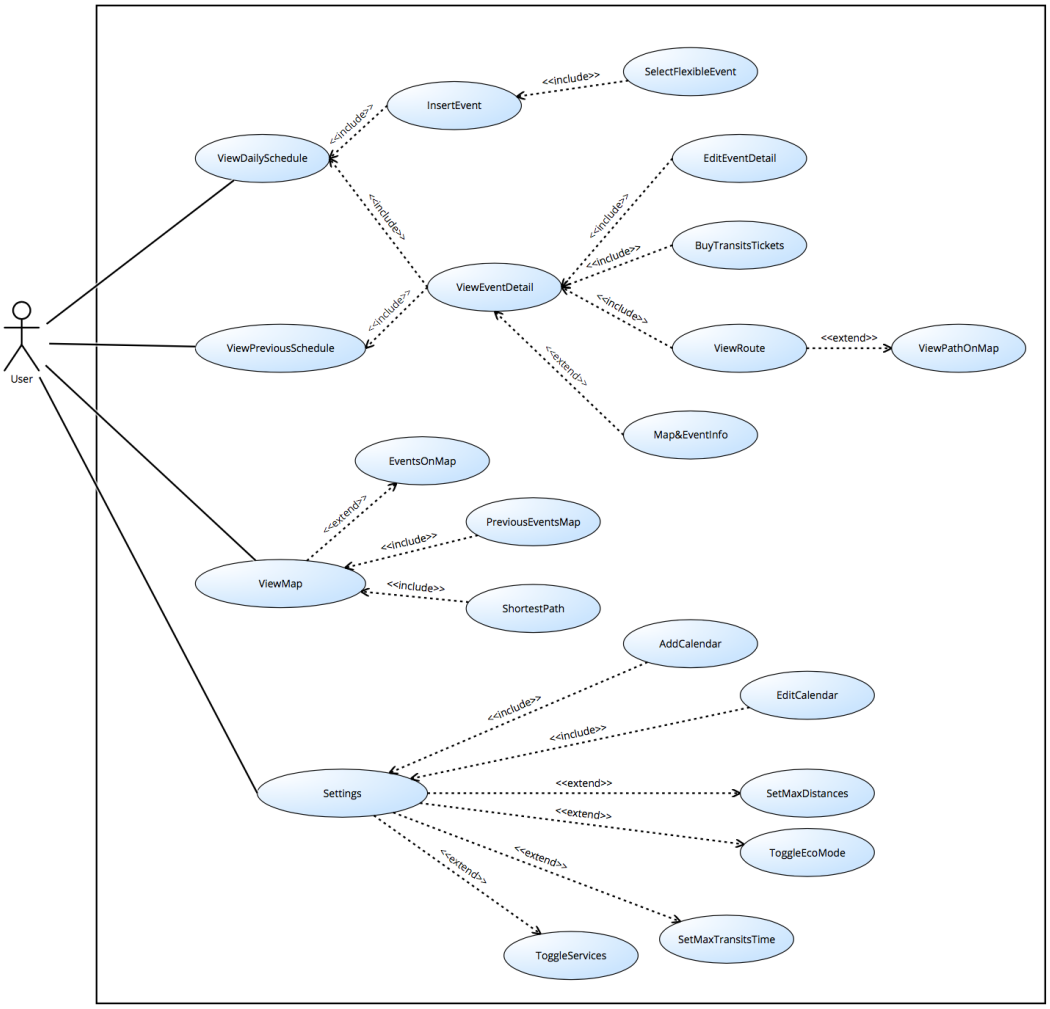
3.5.3 Edit Preferences



3.5.4 BuyTickets



3.6 Use Case Diagram



3.7 Performance Requirements

The system must be able to fulfill a big number of requests in the shortest time possible. Approximately a request should not last more than a second.

The server database should be fast enough to store and search data in order of milliseconds even if million of records are stored.

The server should be located near the maximum concentration of final users, in this case in Lombardy so that network latency will be minimized.

The application should not present any kind of lag in any situation and should never crash.

3.8 Design Constraints

3.8.1 Standards compliance

The app must comply to the standard "iOS Human Interface Guidelines" and "App Store Review Guidelines" by Apple regarding the iOS App and "Google Play Guidelines" for Android.

Regarding the interconnections between the app and the server the system will use some APIs which will be REST-Compliant and they should output data using XML or JSON.

3.8.2 Hardware limitations

- Mobile App
  - iOS, Android or Windows Phone Smartphone
- Connectivity
  - 2G, 3G or LTE
  - GPS

3.8.3 Any other constraint

The system will ask the user permissions to get her/his current position.

The user won't require login because it will use the iCloud unique identifier in order to synchronize events and calendars to the server.

3.9 Software System Attributes

3.9.1 Reliability

The system must guarantee a 24 hours per day and 7 days per week service.

The server should never be offline.

Very small deviations will be accepted only if the system will go in maintenance mode.

3.9.2 Availability

The app will be available at worldwide level but can be used only in Lombardy.

3.9.3 Security

The app will use the Cloud-Service unique identifier (iCloud, Google) to identify the user also when he/she uninstalls the app and reinstalls it.

This identifier is kept secret and it won't be stored on the server for promotional or ambiguous purposes. Its intent is to identify the user while using the system.

The app will communicate with the server via SSL certificates enabling HTTPS secure requests.

3.9.4 Maintainability

The whole system will be developed using a modular architecture that represents an important aspect to make it more maintainable and stable.

3.9.5 Portability

Portability will be highly guaranteed by distribution in OS such as Android and iOS that represents 98% of the smartphone market.

4 Formal analysis using Alloy

4.1 Signatures

```
open util/time

sig Location {
  lat: one Int,
  lng: one Int
}

sig Settings {
  transitStart: one Time,
  transitEnd: one Time,
  maxWalkingDistance: Int
}

sig DB {
  users: set User
}

sig User {
  userId: one Int,
  settings: one Settings,
  calendars: some Calendar
}

sig Calendar {
  events: set Event
}

sig Event {
  eventId: one Int,
  start: one Time,
  end: one Time,
  location: lone Location
}
```

4.2 Facts

```
// event ID unique
fact {
  all e1, e2: Event | e1.eventId=e2.eventId implies e1=e2
}

// Each event belong to one and only one calendar
fact {
  all e: Event | e in Calendar.events
  all c1, c2: Calendar, e: Event | e in c1.events and e in c2.events implies c1=c2
}

// Each user belong to one DB
fact {
  all u: User | u in DB.users
}

// Each location belong to one event
fact {
  all l: Location | l in Event.location
}

// ---
// Each calendar belong to one and only one user
fact {
  all c: Calendar | c in User.calendars
  all u1, u2: User, c: Calendar | c in u1.calendars and c in u2.calendars implies u1=u2
}

// Each setting belong to one and only one user
fact {
  all s: Settings | s in User.settings
  all u1, u2: User, s: Settings | s in u1.settings and s in u2.settings implies u1=u2
}

// The beginning of an event must precede its end
fact {
  all e: Event | gt[e.end, e.start]
}

// The beginning of the transitStart must precede transitEnd
fact {
  all s: Settings | gt[s.transitEnd, s.transitStart]
}

// If two events belong to the same calendar, they cannot overlap
fact {
  all e1, e2: Event, c: Calendar | e1!=e2 and e1 in c.events and e2 in c.events
    implies
  gt[e1.start, e2.end] or gt[e2.start, e1.end]
}
```

4.3 Dynamic Model

```
// Show
pred show {
  #DB=1
  #users>1 and #users<6
  #User.calendars>0 and #User.calendars<4
  #User.calendars.events>3 and #User.calendars.events<8
}

// Determine if transit are available, based on user settings
pred isTransitAvailable[u: User] {
  all e: Event | e in u.calendars.events
    implies
  gt[e.start, u.settings.transitStart] and lt[e.end, u.settings.transitEnd]

  #DB.users<3
  #u.calendars>0 and #u.calendars.events>0
  #u.calendars<3 and #u.calendars.events<7
}

// Planar approximation for simplicity
fun distanceSquare[l1, l2: Location] : Int {
  ((l1.lat - l2.lat).mul[l1.lat - l2.lat] + (l1.lng - l2.lng).mul[l1.lng - l2.lng])
}

// Can walk to the event 'e', accordingly to user settings?
pred canWalkToEvent[currentLocation: Location, e: Event, u: User] {
  #e.location=1
  u.settings.maxWalkingDistance.mul[u.settings.maxWalkingDistance] <= distanceSquare[currentLocation
}

pred addEvent [e: Event, c, c1: Calendar] {
  c1.events = c.events + e
}

pred delEvent [e: Event, c, c1: Calendar] {
  c1.events = c.events - e
}

pred testAdd [e: Event, c, c1: Calendar] {
  addEvent[e, c, c1]
  e in c1.events
}

pred testDel [e: Event, c, c1: Calendar] {
  delEvent[e, c, c1]
  e not in c.events
}
```

4.4 Results

```
run isTransitAvailable for 5 but 8 Int, exactly 1 DB
run canWalkToEvent for 5 but 8 Int, exactly 1 DB

run testAdd for 8 but 8 Int, exactly 1 DB
run testDel for 8 but 8 Int, exactly 1 DB

run show for 8 but 8 Int, exactly 1 DB
```

4.4.1 Proof of Consistency

Executing "Run isTransitAvailable for 5 but 8 int, exactly 1 DB"
Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
42706 vars. 6635 primary vars. 81733 clauses. 143ms.
Instance found. Predicate is consistent. 83ms.

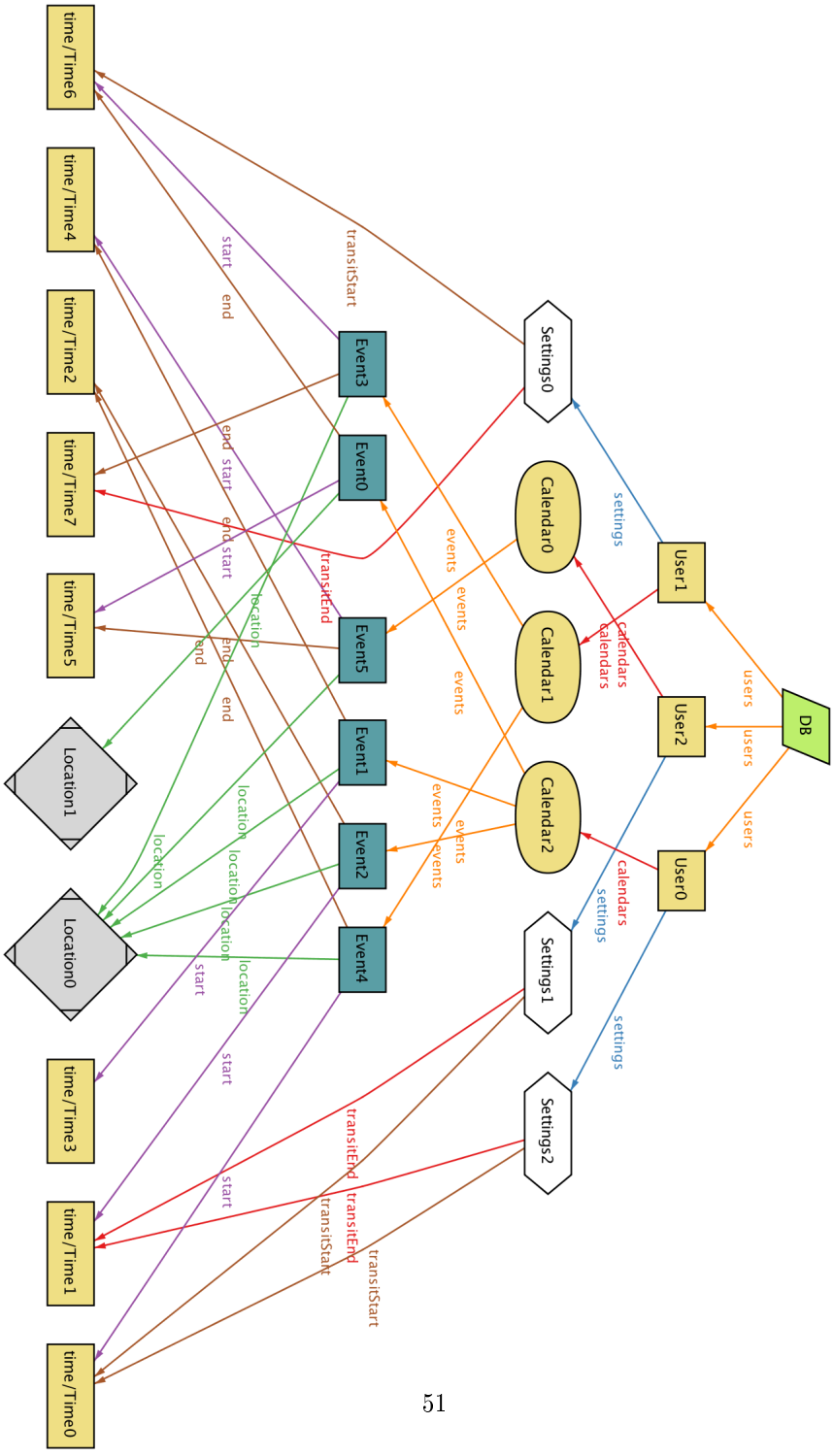
Executing "Run canWalkToEvent for 5 but 8 int, exactly 1 DB"
Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
67746 vars. 6645 primary vars. 169200 clauses. 186ms.
Instance found. Predicate is consistent. 314ms.

Executing "Run testAdd for 8 but 8 int, exactly 1 DB"
Solver=sat4j Bitwidth=8 MaxSeq=8 SkolemDepth=1 Symmetry=20
80599 vars. 10824 primary vars. 152646 clauses. 214ms.
Instance found. Predicate is consistent. 164ms.

Executing "Run testDel for 8 but 8 int, exactly 1 DB"
Solver=sat4j Bitwidth=8 MaxSeq=8 SkolemDepth=1 Symmetry=20
80599 vars. 10824 primary vars. 152647 clauses. 163ms.
Instance found. Predicate is consistent. 102ms.

Executing "Run show for 8 but 8 int, exactly 1 DB"
Solver=sat4j Bitwidth=8 MaxSeq=8 SkolemDepth=1 Symmetry=20
80452 vars. 10800 primary vars. 152620 clauses. 206ms.
Instance found. Predicate is consistent. 164ms.

4.4.2 Generated world



5 Effort spent

- Filippo Calzavara: 30.00h
- Giovanni Filaferro: 30.00h
- Benedetto Maria Nespoli: 30.00h

6 References

- iOS Human Interface Guidelines (https://developer.apple.com/ios/human-interface-guidelines/)
- App Store Review Guidelines (https://developer.apple.com/app-store/review/guidelines/)
- Google Play Guidelines (https://play.google.com/about/developer-content-policy/#!?modal_active=none)