POLITECNICO
MILANO 1863

RASD

Requirements Analysis and Specification Document

FEDERICO DI DIO - 893730
FLAVIO DI PALO - 898541
MATTEO BELLUSCI - 898380

INDEX

1. Introduction

1.A Purpose

The aim of RASD document is to analyze the most important features of the calendar-based application called "**Travlendar+**". This document is intended to illustrate system goals and functionalities, taking in account interests and needs of the users in a useful way for the relationship between the developer and the consumer and for the following implementations.

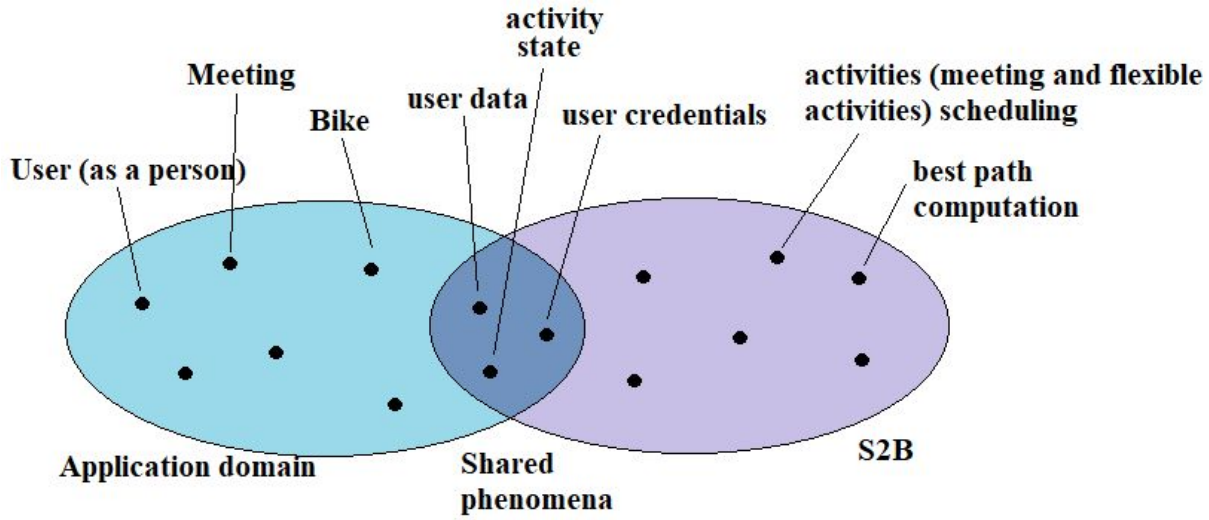
1.A.1 Goals

[G1] Allow a Visitor to become a User with a personal account
[G2] Allow a User to manage his Activities
[G3] The User has to be warned if a scheduled activity becomes unreachable in the accounted time.
[G4] Allow a user to choose which means of transport he prefers to reach the Activity
[G5] The application should suggest travel means depending on the nature of appointment.
[G6] Allow a User to reach an activity using the best mobility option and considering user Preferences
[G7] Allow a User to buy public transportation tickets/passes, and take advantage of the taxi service.
[G8] Allow a User to use bike-sharing and car-sharing services
[G9] Allow a User to schedule an activity in a flexible way according to his other meetings.
[G10] Allow the User to view a calendar with all his scheduled activities

1.B Scope

The application is finalized to support the user in his daily travels and Activities at various locations, identifying the best mobility solution. The software must grant the following functionalities in order to satisfy this purpose.

Jackson-Zave approach



The system wants to provide the user with personalized and flexible support which can help him in organizing his weekly commitments, from work to leisure. In doing so, it aims to suggest the best means of transport and allow the purchase of tickets or passes for public transport.

1.C Definitions, Acronyms, Abbreviations

1.C.1 Definitions

● **Activity:** it is a scheduled appointment it could be either a Meeting (or Fixed Activity) or a Flexible Activity

● **Flexible Activity:** a particular event characterized by a fixed duration that can be scheduled in a given time slice by the system.

● **Meeting (or Fixed Activity):** a scheduled appointment characterized by a precise hour, day and destination.

● **Route:** is the path from the user's current position to the position of his Scheduled Activity

● **Sub-Route:** is a portion of the Route that is performed with a single mean of transport

1.C.2 Acronyms

- DDoS attack: Distributed Denial of Service attack
- ETA: Estimated Time of Arrival
- MVC: Pattern Model View Controller Pattern
- JML: Java Modeling Language
- RASD: Requirement Analysis and Specification Document.
- API: Application Programming Interface
- S2B: System-to-be

1.C.3 Abbreviations

- **[Gn]**: identifier of n-th Goal.
- **[Dn]**: identifier of n-th Domain Assumption.
- **[Rn]:** identifier of n-th Functional Requirement.

1.D Revision history

- **Version 1.0,** 29th october 2017
- **Version 1.1,** 26th November 2017

1.E Reference Documents

- *Specification Document:* Mandatory Project Assignments.pdf
- *"Fundamentals of Software Engineering"*, Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli
- RASD Examples: "RASD+Sample+from+A.Y.+2016-2017.pdf","RASD+Sample+from+A.Y.+2015-2016"
- *"Mobile Application Design and Development"*, Erik Wilde, UC Berkeley school of Information, 2010
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.

1.F Document Structure

This document is essentially structured in four parts:

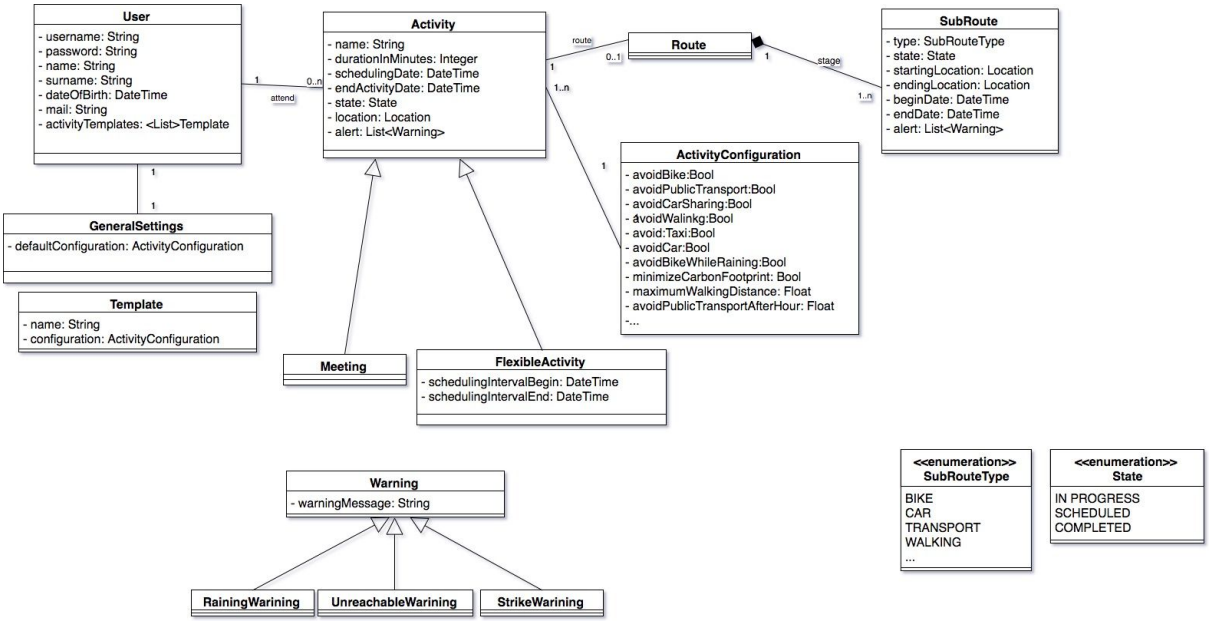
1. In this **first section**, it is provided an introduction to the general problem that is going to be develop in the software.
2. In the **second section** it is explained which are the overall functionalities of the product and its principal requirements, assumptions and constraints.
3. In the **third section**, the specific requirements (functional and nonfunctional) of our system and to describe the structure using Use Cases and Sequence Diagrams.
4. In the last **fourth section**, it is presented the Alloy contribution and team effort.

2. Overall Description

2.A Product perspective

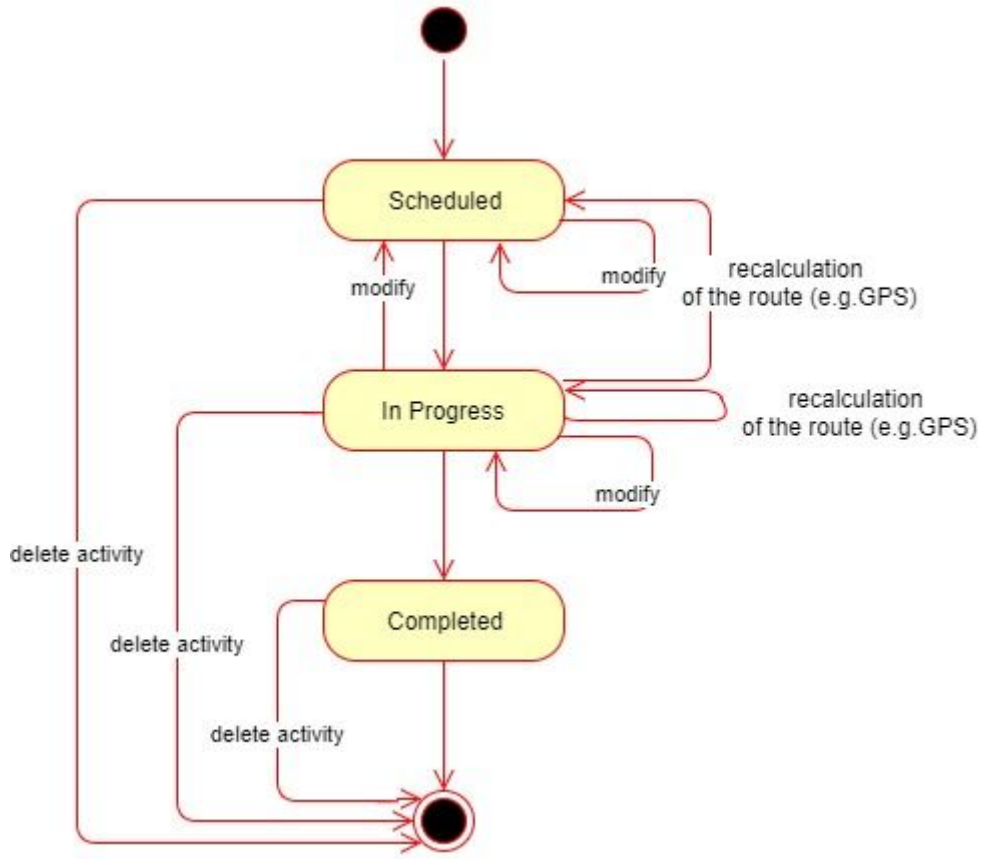
The system will be released as mobile Application available for iOS, Windows Phone and Android Operating Systems. Our product will be able to interact with many external services that will be further described later. Any payment is intended to be managed directly from the interested external service in order to guarantee the security of the transactions for the User.

2.A.1 Class Diagram



2.A.2 StateChart Diagram

The following diagram is the statechart of a generic activity (meeting or flexible activity).



The state “In Progress” means that the activity is ongoing or the user is travelling for reach that activity’s location.

2.B. Product functions

Here are described the main goals, functional requirements and assumptions included in Travlendar+.

[G1] Allow a Visitor to become a User with a personal account

[R 1.1] The User must be able to Sign Up to Travlendar+ by inserting their personal information and email address.

[R 1.2] During the Sign Up phase the System must verify that email address is not yet registered on the DBMS

[R 1.3] The Visitor must be able to Log In from Travlendar+

[R 1.4] The User must be able to Log Out from Travlendar+

[R 1.5] During Log In phase the system must verify that the entered username and password are already registered into the DBMS and correct to allow the user to access.

[R 1.6] The System must store User default preferences and general settings on his account

[R 1.7] The User must be able to change his default preferences and general settings in any time

[G2] Allow a User to manage his Activities

[R 2.1] The system must allow the User to create Activities only if the scheduled date and time are after the creation time

[R 2.2] The system must verify that the scheduled Activity is reachable on time

[R 2.3] The system must show a warning if a Scheduled Activity is not reachable on time

[R 2.4] The system must allow the User to delete Activity in any moment

[R 2.5] The system must allow the User to edit Activity preferences and settings in any moment

[D 2.1] The reachability of an Activity is computed considering all the possibilities and selecting the best route from the User's starting position

[D 2.2] The position of the User is correctly retrieved by GPS

[G3] The User has to be warned if a scheduled Activity becomes unreachable in the accounted time.

[R 3.1] The System must verify that the scheduled Activity is reachable on time considering both Public Transportation Strikes and Weather Conditions

[R 3.2] The System must correctly handle Public Transportation APIs

[R 3.3] The System must correctly handle Weather Information APIs

[R 3.4] The System must show a warning to the User If the scheduled Activity is not reachable on time.

[D 3.1] The reachability of a Activity is computed considering all the possibilities and selecting the best route from the User's starting position

[D 3.2] The position of the User is correctly retrieved by GPS

[D 3.3] The functionality of retrieve weather conditions and forecast is correctly provided by an external service with its APIs.

[D 3.4] The functionality of retrieve information about public transportation strikes is correctly provided by an external service with its APIs.

[G4] Allow a user to choose his Preferences to reach the Activity

[R 4.1] The User must be able to select which means of transport avoid for reaching that particular activity

[R 4.2] The User could select if the wants to avoid bike while raining

[R 4.3] The User could select if the wants to choose the route that minimizes the Carbon Footprint

[R 4.4] The User can select the maximum walking distance he wants to perform for each route

[R 4.5] The User can choose to avoid the Public Transport after a given hour

[R 4.6] The preferences has to be correctly stored into the System and available in any moment

[G5] The application should suggest travel means depending on the nature of appointment.

[R 5.1] The User must be able to create Activity Templates in which he can choose what travel mean he want to use for each different kind of Activity.

[R 5.2] The system must suggest to the User the travel mean basing on created templates

[G6] Allow a User to reach an activity using the best mobility option and considering user Preferences.

[R 6.1] The System must compute the best route by analyzing all possible paths considering traffic conditions, eventual accidents, weather conditions or Public transportation strikes.

[R 6.2] The System must take in account preferences expressed by the User to reach the Activity.

[R 6.3] The System must correctly handle Map and Traffic Service APIs.

[R 6.4] The System must correctly handle Weather Information APIs.

[D 6.1] The System computes the best route in any condition.

[D 6.2] The position of the User is correctly retrieved by GPS.

[D 6.3] The functionality of retrieve weather conditions and forecast is correctly provided by an external service with its APIs.

[D 6.4] The functionality of retrieve information about public transportation strikes is correctly provided by an external service with its APIs.

[G7] Allow a User to buy public transportation tickets/passes.

[R 7.1] The System must correctly handle the Public Transportation Company APIs.

[D 7.1] The possibility to buy ticket or passes is provided by an external service with its APIs.

[G8] Allow a User to use bike-sharing and car-sharing services.

[R 8.1] The System must correctly handle the Bike Sharing Service APIs.

[R 8.2] The System must show the position of the rentable bikes on the map.

[R 8.3] The System must correctly handle the Car Sharing Service APIs.

[R 8.4] The System must show the position of the rentable cars on the map.

[D 8.1] The position of bikes, cars and bike stations shown on the map really indicates the location.

[D 8.2] The bike-sharing functionality is correctly provided by an external service with its APIs.

[D 8.3] The car-sharing functionality is correctly provided by an external service with its APIs.

[G9] Allow a User to schedule an Activity in a flexible way according to his other meetings.

[R 9.1] The User must be able to specify a time interval in which will be done a Flexible Activity

[R 9.2] The system must compute the best time in interval specified by the User to place the Flexible Activity, according to the other meetings during the day

[G10] Allow the User to view a calendar with all his scheduled activities

[R 10.1] The System must show the User a Calendar from which he can visualize and manage his daily meetings.

[G11] Allow a User to take advantage of the taxi service.

[R 11.1] The System must correctly handle the Taxi Service APIs.

[D 11.1] The Taxi Service Functionality is provided by an external service with its APIs.

2.C User characteristics

The user must have a smartphone to run the application of Travlendar+ and must be able to install the application on the device.
The application must be able to access the GPS of the user's device.
If the User decide to insert the possibility of use a car in the route track then he must be in the age of majority and have the driving license.
The application cannot control if the User is in a driving condition: this is a problem only for the User, who declares to be in a position to drive.
If the User decide to insert the possibility of use a public transportation service in the route track then he must be in possession of a valid travel ticket or he must buy it through the Travlendar+ App.
The User agrees to these conditions also during the registration to the system.

2.D Assumptions, dependencies and constraints

Here we include further specifications in order to avoid any kind of ambiguity in the interpretation of the document.

2.D.1 Assumptions

- The internet connection works correctly.
- Credentials that a visitor has to provide to become a registered user are: name, surname, address, email, telephone number and username.
- Weather Forecast are provided and by an external service.
- The required functionalities provided are ambiguous about how the software suggest the travel mean “depending” on the appointment: we assume that the User can set his preferences for each kind of appointment using “Activity Templates”.
- When a destination address is inserted, the map with the route will be available in most 30 seconds.
- The Route computed by the software is the best Route according to ETA.
- The User must be identified uniquely by the username.
- User location is correctly retrieved by GPS.
- When the system shows a car or a bike in a certain position it means that it's actually there

2.D.2 Dependencies

The car/bike sharing, taxi service, weather service, bus ticket purchase functionalities are provided by external services with their APIs.

2.D.3 Constraints

The System must require these permissions to the User:

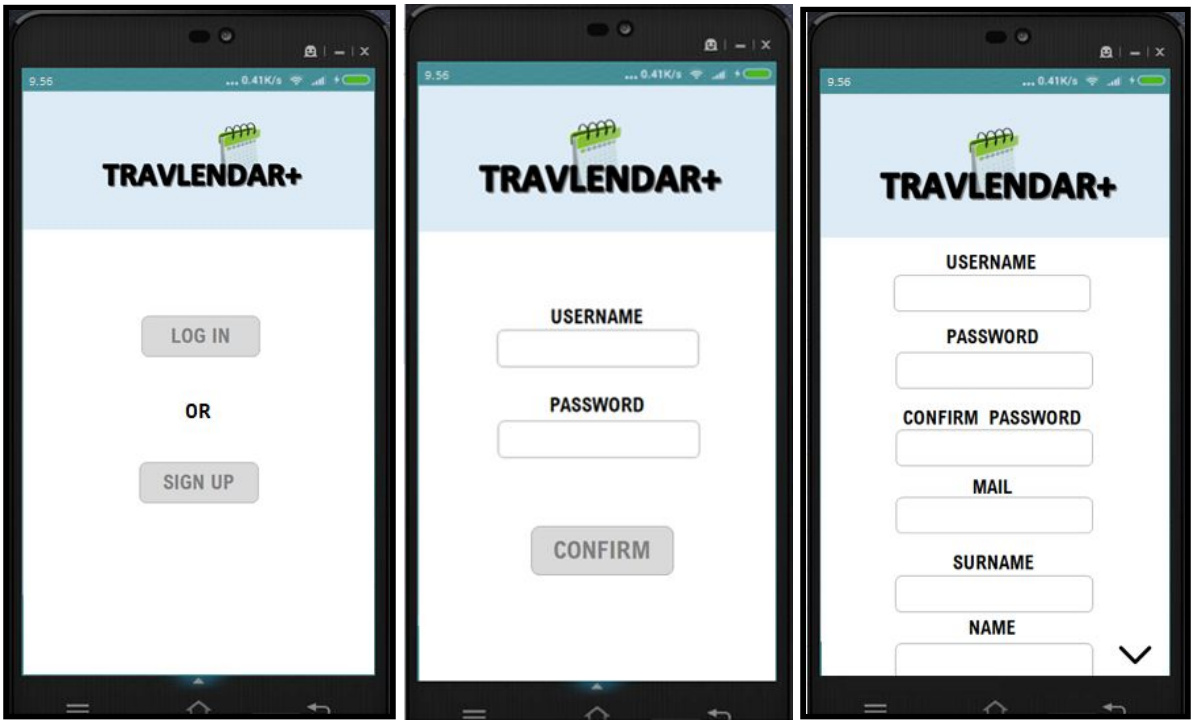
- Identity
- Contacts
- GPS Location
- SMS
- Wi-Fi Connection information.

3. Specific Requirements

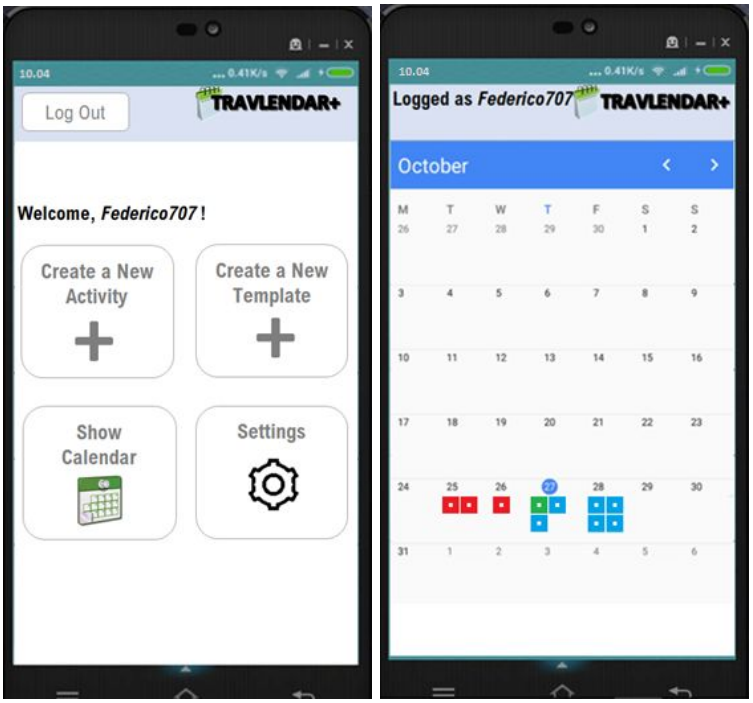
3.A External Interface Requirements

3.A.1 User Interfaces

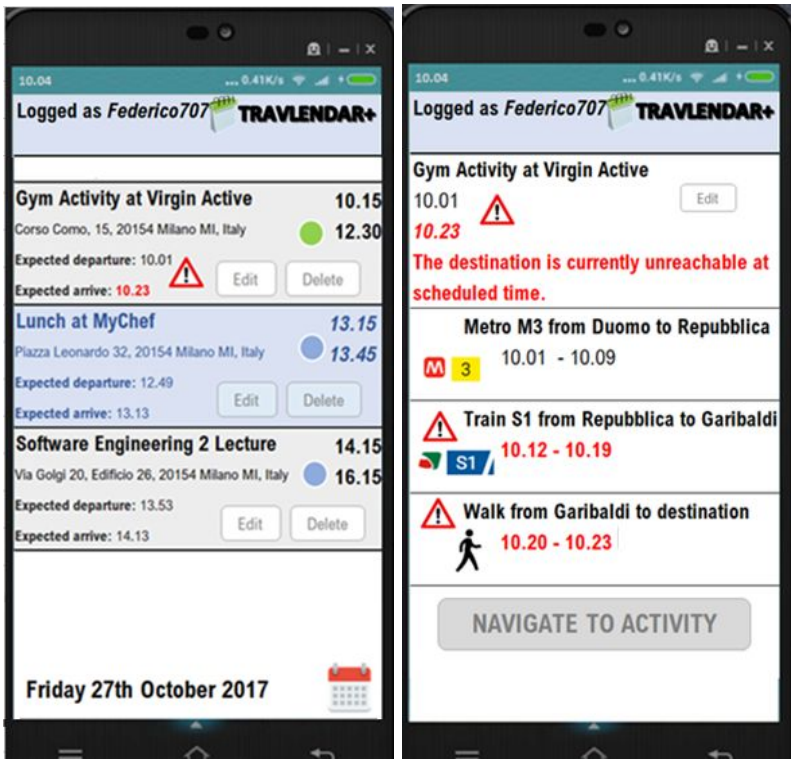
The following mockups represent a basic idea of what the mobile app will look like in the first release.



In these screens are represented the Login and Sign Up functionalities. The system will check the correctness of the entered data.

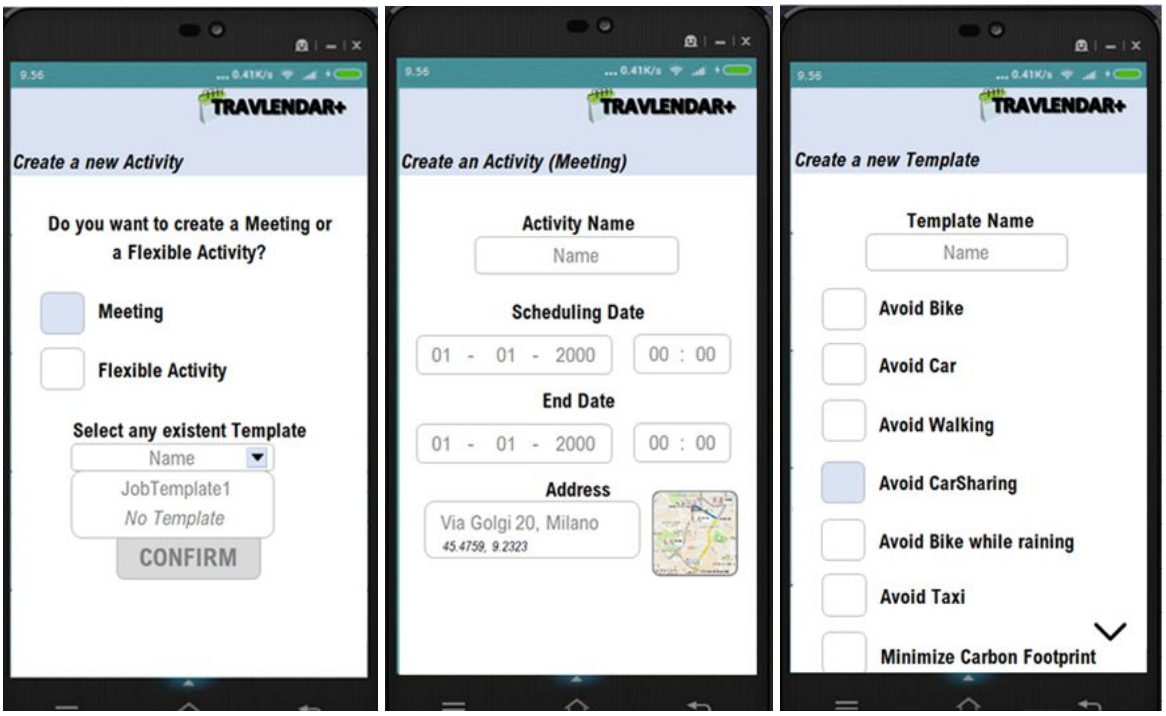


Here are some screens about the core of the application.
The first two are about the User's homepage, containing his Monthly calendar, in which is possible to click on a specific day to watch the Activities scheduled for it; if the activity is colored green, it is in a "In Progress" status, if it is blue is "Scheduled", if it is red is "Concluded".



These screen are related to the details about a specific day organization and a specific Activity. In the first screen daily Activities are described in their state, departure time, arrival time etc., then in the second are represented the subroutes relative to an Activity, each featuring a means of transport, departure time, arrival time and eventual warning.

This screen represents the proper Navigation to the destination in which is located the Activity (conducted through the car in the example); a map is shown with live indications to reach the stage. In addition, there is the expected arrival time and the remaining distance.



These screens represent the feature of Create an Activity, where it is possible to choose the type of Activity in the first screen, and then select the preferences in the specific case of a Meeting. There is also a screen containing the creation a specific template where the User can express his preferences for specific kind of activities.

3.A.2 Hardware Interfaces

The system does not have any hardware interfaces except for the Smartphone with it's GPS sensor

3.A.3 Software Interfaces

The system operates using software to store and save data, as well as databases and DBMSs. The following software are used:

- MySQL version 5.7.19.

The mobile application must support Android, iOS and Windows Phone.
The back-end stores its data in a RDBMS.
The back-end must offer APIs for user interfaces and external modules, like: taxi reservation, car/bike sharing, weather information and online payments.

3.A.4 Communication Interfaces

The system must interface with the **DBMSs** installed on the server, and the **APIs** through which he retrieves information from the external services (Weather Service, Public Transportation Service, Bike / Car Sharing Service, Maps and Traffic), in order to notify the User with warnings or recompute the best route.

3.B Functional Requirements

3.B.1 Scenarios

3.B.1.1 Sign up in the System and Log In

Francesco works as a trading agent for large company in Milan and he is in need of organizing a great deal of appointments in different areas of the city during the day. He finds the new service **Travlendar+** and decides to download it from the Play Store on his Android device and sign up to better organize his commitments. For the registration, the System provides a form with different fields: username, name,
surname, birth information, email address, password and optional personal information to complete the profile. Then Francesco fills in all the fields and pushes the registration button. The system approves the registration and sends to him a confirmation email. Now Francesco can better organize his activities during the day using Travlendar+. Some days after the registration, Francesco decided to log in the system with his email address and password, which will be stored into the system whenever he decides to relog to the application. If he would decide to close or uninstall the app, he could run a Log Out to disconnect from the application.

3.B.1.2 Creation of an Activity and Purchase of public transport tickets

It's half past four of a rainy afternoon and Francesco has a meeting with his boss at 6pm, then he decides to create an Activity. He inserts the scheduled time and the exact address but he have not any constraint about the mean of transport, and then he click on Create the Activity. The application calculates the best route, suggesting the metro as best mean, to be taken at 5.36 some stops away from the arrival station; after that he will have to walk for 6 minutes to reach the exact destination. The system preferred the metro to the car after detecting heavy traffic it that zone; then he decided to buy the ticket using the app, to not to waste time going to the metro ticket office.

3.B.1.3 Activity modification

Some minutes later, the boss calls Francesco by advising him of moving the appointment two hours later. Francesco then clicks on "Edit Activity" on the activity previously scheduled in his homepage, and enter the new scheduled hour for the appointment. The system recalculates the route, but now the app suggests to the user a car as means of transport since the road is tendentially much less busy for that hour.

3.B.1.4 Bike or Car Sharing service utilization

It’s half past ten in the following morning and Francesco has to reach a client almost 2 miles from his position at 11.15, then he decides again to create an Activity in Travlendar. It's fortunately a sunny day and Francesco would prefer to avoid taking the car to enjoy the good weather, so he selects the "Avoid Car" option and creates the Meeting. The app suggests a bike sharing service: the closest free bike is 300 meters away and he books it, then he starts to walk to reach the bike.

3.B.1.5 Creation of Activity Template

However, after that activity Francesco realizes that using a bike to get to job appointments, especially near the summer season, can be overly tiring. Noticing the Create to Template functionality then he decides to use it to create a specific Template suitable for work appointments, which excludes the bike from the means of transport. At the same time, going to sports appointments may require to arrive already warmed up, so bike could be the right means of transport: Francesco creates another template in which he excludes the option to take car or public transport.

3.B.1.6 Creation of Flexible Activity

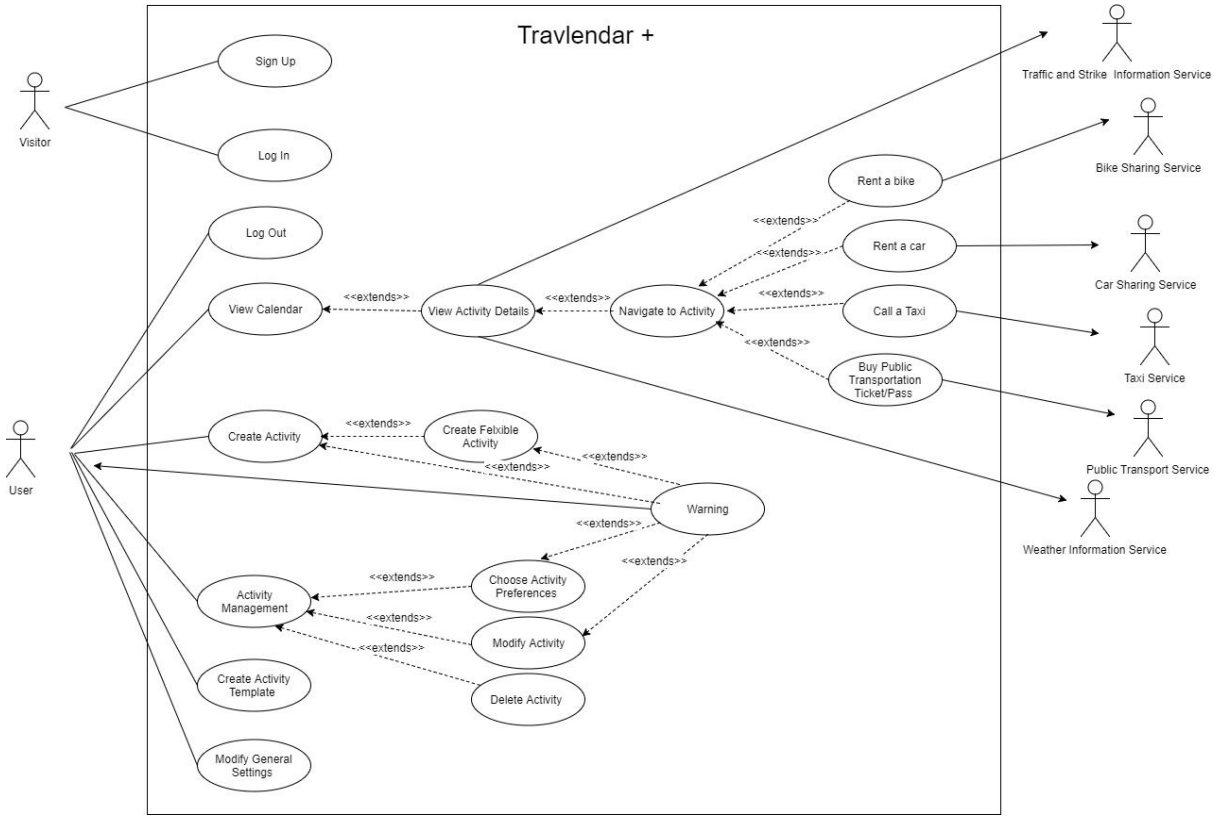
Lunch time is coming and Francesco needs to optimize the available time available due to many appointments scheduled from 2.30 pm onwards. He decided to use the Flexible Activity functionality, which offers him the possibility to specify a range in which positionate a task of a particular duration, choosing the best best time according to his next activities. Francesco therefore chooses a half-hour lunch to be entered between 12.30 - 14.30; the application chooses 13.15 to 13.45 to leave Francesco the time needed to reach the destination of first appointment.
To reach Activities scheduled for the first afternoon, the app does not consider the subway option according to information received from the system about an ATM strike which will last until 6 o'clock in the afternoon.

3.B.2 Use Case Model

We can deduce some use cases from the scenarios previously described. The main actors are:

- **Visitor**: a guest who is not yet authenticated in the system
- **User:** a Visitor who has already signed up in “Travlendar+” and have successfully Log In.

- **External Services**: the System interacts with these services through some APIs in order to guarantee some functionalities



Here we are going to analyze some Use Case in their particular features, such as their goals, requirements, assumptions, input/output conditions and event flow, in order to better describe the App functionalities. Some of use cases like “Call a Taxi” are omitted because of their similarity to the functionality about the Public Transportation TicketPass which is well described in one of the following tables.

3.B.2.1 Visitor Use Cases

Sign up to the system [UC1]

Goal	**[G1]** Allow a Visitor to become a User with a personal account.
Requirement	**[R 1.1]** The User must be able to Sign Up to Travlendar+ by inserting their personal information and email address.

	[R 1.2] During the Sign Up phase the System must verify that email address is not yet registered on the DBMS
Assumption	Visitor is not registered yet to the system.
Actors	Visitor, User
Input Condition	The Visitor must have the app downloaded on the smartphone.
Output Condition	The Visitor is a Registered User.
Event Flow	1) The Visitor opens the “Travlendar+” application. 2) The Visitor clicks on “Sign Up” button. 3) The Visitor fills all mandatory fields. 4) The Visitor clicks on “Confirm” button. 5) The System checks the correctness of the information inserted. 6) The Systems shows on the display the App homepage.
Exceptions	1) The Visitor is already a User. 2) The Visitor entered non valid information in mandatory field. 3) The Visitor chooses an email that has been associated with another User.

Log In to the System [UC2]

Goal	**[G1]** Allow a Visitor to become a User with a personal account
Requirement	**[R 1.3]** The Visitor must be able to Log In from Travlendar+.

	[R 1.4] During Log In phase the system must verify that the entered Username and password are already registered into the DBMS and correct to allow the user to access.
Assumption	Visitor is registered.
Actors	Visitor, User
Input Condition	The Visitor must have the app downloaded on the smartphone and have opened it.
Output Condition	The Systems shows on the display the App homepage.
Event Flow	1) The user opens the "Travlendar+" app. 2) The User clicks on “Log In” button. 3) The User enters his Username and Password in their respective fields. 4) The User clicks on “Confirm” button. 5) The System checks the correctness of the information inserted. 6) The Systems shows on the display the App homepage.
Exceptions	1) Username is not correct 2) Password is not correct **These exceptions are handled notifying the User and re-starting the Event Flow from the point 3.**

3.B.2.2. User Use Cases

Create an Activity [UC3]

Goal	**[G2]** Allow a User to manage his Activities.

	[G3] The User has to be warned if a scheduled Activity becomes unreachable in the accounted time. **[G9]** Allow a User to schedule an Activity in a flexible way according to his other meetings.
Requirement	**[R 2.1]** The system must allow the User to create Activities only if the scheduled date and time are after the creation time. **[R 2.2]** The system must verify that the scheduled Activity is reachable on time and must show a warning if it is not. **[R 3.1]** The System must verify that the scheduled Activity is reachable on time considering also both Public Transportation Strikes and Weather Conditions. **[R 3.4]** The System must show a warning to the User If the scheduled Activity is not reachable on time.
Assumption	User is registered, has logged in and navigates on the homepage.
Actors	User
Input Condition	The User is in the Homepage.
Output Condition	The User has scheduled an Activity.
Event Flow	1) The User selects the “Create an Activity” option. 2) The User chooses the Activity type and eventually any created template. 3) The User enters Activity date and hour (and eventually Coordinates using a map to select the Activity place). 4) The User chooses his preferences about the Activity.

	5) The system divides the trip into Subroutes, suggesting the best means of transport for each one. 6) If the destination is not reachable at the scheduled time, the system notifies the user with a warning
Exceptions	1) The Activity is scheduled before the actual time **The system reports the error and the control flow returns to step 2**

Edit an Activity [UC4]

Goal	**[G2]** Allow a User to manage his Activities **[G3]** The User has to be warned if a scheduled Activity becomes unreachable in the accounted time. **[G6]** Allow a User to reach an activity using the best mobility option and considering user Preferences
Requirement	**[R 2.5]** The system must allow the User to edit Activity preferences and settings in any moment **[R 3.1]** The System must verify that the scheduled Activity is reachable on time considering also both Public Transportation Strikes and Weather Conditions **[R 3.4]** The System must show a warning to the User If the scheduled Activity is not reachable on time.
Assumption	User is registered, has logged in and has already created at least one Activity.
Actors	User
Input Condition	The User navigates in his Homepage.

Output Condition	All the modified information about the Activity are saved.
Event Flow	1) The User selects the “Edit” option on an Activity scheduled for a specific day. 2) The System shows all the preferences options about the Activity. 3) The User changes one or more settings or preferences about the Activity. 4) The system computes again the best route based on new preferences and settings. 5) If the destination is not reachable at the scheduled time according to the new settings or preferences, the system notifies the user with a warning. 6) The changes about the Activity are stored.
Exceptions	1) The User closes the “Edit View” 2) The User returns to Homepage

Delete an Activity [UC5]

Goal	**[G2]** Allow a User to manage his Activities
Requirement	**[R 2.4]** The system must allow the User to delete Activity in any moment
Assumption	User is registered, has logged in and has already created at least one Activity.
Actors	User
Input Condition	The User navigates in his Homepage.
Output Condition	The User has deleted an Activity.

Event Flow	1) The User selects the "Delete" option on an Activity in his Homepage. 2) The system asks the User a confirmation about his will to delete the selected Activity 3) The System deletes the Activity. 4) Eventual Flexible Activities are computed again according to new disposition of Activities
Exceptions	1) The User choose close the "Edit View" 2) The User returns to Homepage

Create a Flexible Activity [UC6]

Goal	**[G9]** Allow a User to schedule an Activity in a flexible way according to his other meetings.
Requirement	**[R 9.1]** The User must be able to specify a time interval in which will be done a Flexible Activity **[R 9.2]** The system must compute the best time in interval specified by the User to place the Flexible Activity, according to the other meetings during the day
Assumption	User is registered and has logged in.
Actors	User
Input Condition	The User navigates in his Homepage.
Output Condition	The User has created a new Flexible Activity.

Event Flow	1) The User selects the “Create an Activity” option. 2) The User chooses the “Flexible Activity” as a type of Activity. 3) The User enters Flexible Activity hour range and duration. 4) The User chooses his preferences about the Flexible Activity. 5) The system computes the best time into the range for the Flexible Activity, according to other Activities during the day. 6) The system divides the trip into Subroutes, suggesting the best means of transport for each one.
Exceptions	1) The User closes the “Create Activity” view. 2) The User returns to the Home Page.

Create an Activity Template [UC7]

Goal	**[G5]** The application should suggest travel means depending on the nature of appointment.
Requirement	**[R 5.1]** The User must be able to create reusable Activity Templates in which he can choose what travel mean he want to use for each different kind of Activity. **[R 5.2]** The system must suggest to the User the travel mean basing on created templates
Assumption	User is registered and has logged in.
Actors	User
Input Condition	The User navigates in his Homepage.

Output Condition	The User has created a new Template.
Event Flow	1) The User selects the “Create a Template” option in his Homepage. 2) The User selects Template general settings and preferences. 3) The new Template is stored in User account.
Exceptions	1) The User closes the “Create Template” view before completing all the steps. 2) The User returns to the Home Page.

Modify Account General Settings [UC8]

Goal	**[G1]** Allow a Visitor to become a User with a personal account
Requirement	**[R 1.6]** The System must store User default preferences and general settings on his account **[R 1.7]** The User must be able to change his default preferences and general settings in any time
Assumption	User is registered and has logged in.
Actors	User
Input Condition	The User navigates in his Account page.
Output Condition	The User has changed General Settings.
Event Flow	1) The User selects the “Edit General Setting”. 2) The User modifies some General Settings. 3) The changes are stored in User account.

Exceptions	1) The User closes the “Modify General Settings” view before completing all the steps. 2) The User returns to the Home Page.

View Calendar [UC9]

Goal	**[G10]** Allow the User to view a calendar with all his scheduled activities.
Requirement	**[R 10.1]** The System must show the User a Calendar from which he can visualize and manage his daily meetings.
Assumption	User is registered, has logged in and has already created at least one Activity.
Actors	User
Input Condition	The User in already in his Homepage.
Output Condition	The User has successfully viewed all his meetings.
Event Flow	1) The User select the option “View Calendar” in his home page.

View Activity Details [UC10]

Goal	**[G10]** Allow the User to view a calendar with all his scheduled activities. **[G2]** Allow a User to manage his Activities. **[G3]** The User has to be warned if a scheduled Activity becomes unreachable in the accounted time.
Requirement	**[R 10.1]** The System must show the User a Calendar from which he can visualize and manage his daily meetings. **[R 3.1]** The System must verify that the scheduled Activity is reachable on time

	considering also both Public Transportation Strikes and Weather Conditions. **[R 3.2]** The System must correctly handle Public Transportation APIs. **[R 3.3]** The System must correctly handle Weather Information APIs. **[R.3.4]** The System must show a warning to the User If the scheduled Activity is not reachable on time.
Assumption	User is registered, has logged in and has already created at least one Activity. **[D 3.1]** The reachability of a Activity is computed considering all the possibilities and selecting the best route from the User's starting position. **[D 3.2]** The position of the User is correctly retrieved by GPS. **[D 3.3]** The functionality of retrieve weather conditions and forecast is correctly provided by an external service with its APIs. **[D 3.4]** The functionality of retrieve information about public transportation strikes is correctly provided by an external service with its APIs.
Actors	User
Input Condition	The User is in the Calendar View.
Output Condition	The User has successfully received all details for his meeting.
Event Flow	1) The User selects an Activity in his calendar View. 2) The User views all details related to that meeting.
Exceptions	The User deletes the operation.

Navigate to Activity [UC11]

Goal	**[G6]** Allow a User to reach an activity using the best mobility option and considering user Preferences.
Requirement	**[R 6.1]** The System must compute the best route by analyzing all possible paths considering traffic conditions, eventual accidents, weather conditions or Public transportation strikes. **[R 6.2]** The System must take in account preferences expressed by the User to reach the Activity. **[R 6.3]** The System must correctly handle Map,Traffic information Service APIs. **[R 6.4]** The System must correctly handle Weather Information APIs.
Assumption	**[D 6.1]** The System computes the best route in any condition. **[D 6.2]** The position of the User is correctly retrieved by GPS. **[D 6.3]** The functionality of retrieve weather conditions and forecast is correctly provided by an external service with its APIs. **[D 6.4]** The functionality of retrieve information about public transportation strikes is correctly provided by an external service with its APIs.
Actors	User
Input Condition	The User is in “Activity Details” view.
Output Condition	The User has successfully reached the Activity.
Event Flow	1) The user select “Navigate to Activity” option.

	2) The User starts the navigation following the route computed by the System.
Exceptions	1) Warning is shown if there is a transportation strike and Public Transport is a mean to reach the activity. 2) Warning is shown if It will rain when the User is going to use bike. 3) The User deletes the operation.

Rent a Car / Bike [UC12]

Goal	**[G8]** Allow a User to use bike-sharing and car-sharing services
Requirement	**[R 8.1]** The System must correctly handle the Bike Sharing Service APIs. **[R 8.2]** The System must show the position of the rentable bikes on the map. **[R 8.3]** The System must correctly handle the Car Sharing Service APIs. **[R 8.4]** The System must show the position of the rentable cars on the map.
Assumption	**[D 8.1]** The position of bikes, cars and bike stations shown on the map really indicates the location. **[D 8.2]** The bike-sharing functionality is correctly provided by an external service with its APIs. **[D 8.3]** The car-sharing functionality is correctly provided by an external service with its APIs.
Actors	User, Bike/Car Sharing Service
Input Condition	The User is performing Navigation to his Activity.

Output Condition	The User has successfully rented a bike / car for the desired subroute.
Event Flow	1) The user clicks on an available Bike or a Car on the map. 2) The System using the APIs redirects the User to the page where he can pay his bike / car rent. 3) The User completes other procedures on the external Website.
Exceptions	1) The Internet connection is not available. 2) A Warning is shown and the user is invited to retry later to buy the ticket.

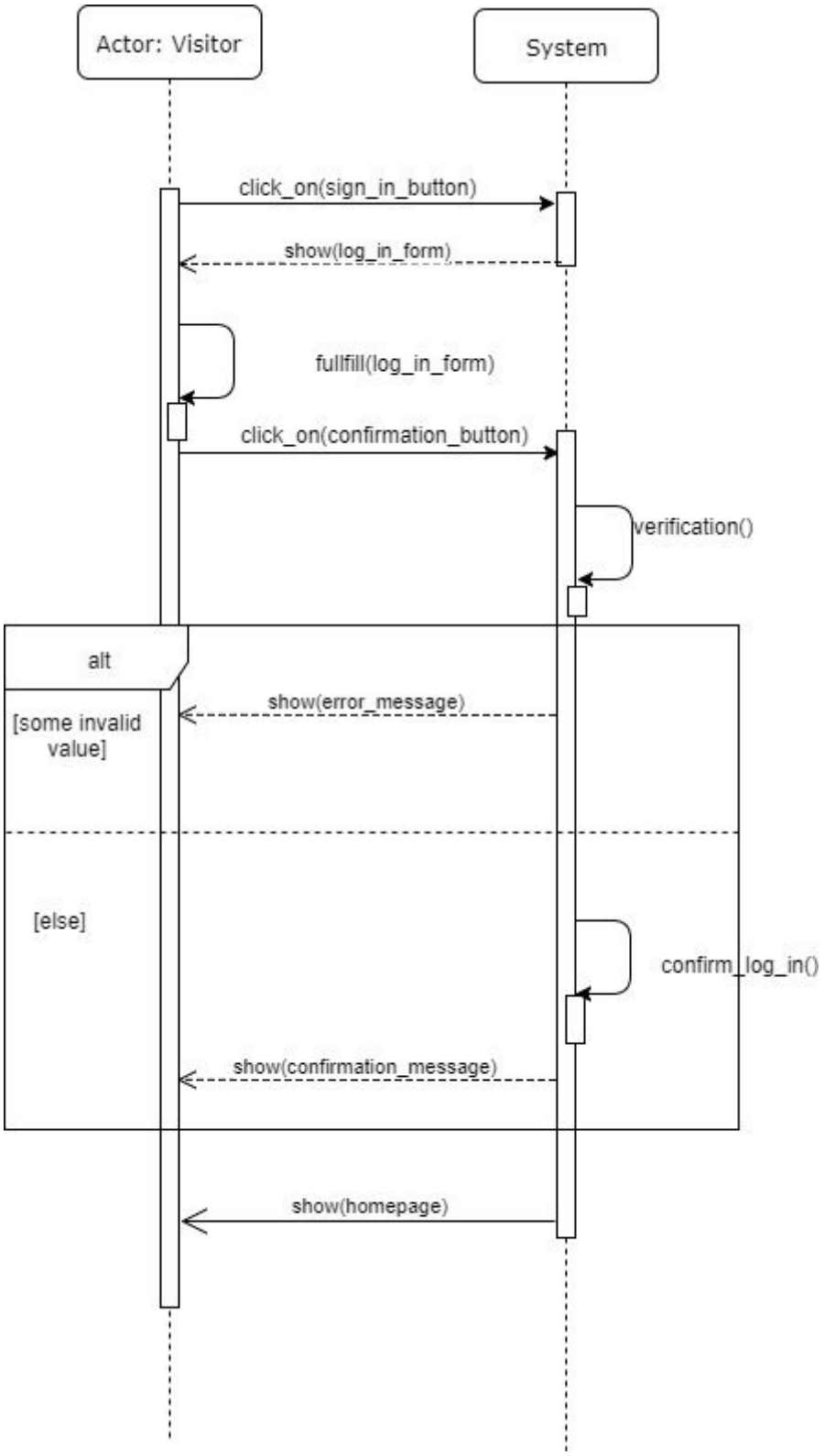
Buy Public Transportation Ticket [UC13]

Goal	**[G7]** Allow a User to buy public transportation tickets/passes
Requirement	**[R 7.1]** The System must correctly handle the Public Transportation Company APIs
Assumption	**[D 7.1]** The possibility to buy ticket or passes is provided by an external service with its APIs.
Actors	User, Public Transportation Service
Input Condition	The User is performing Navigation to his Activity
Output Condition	The User has successfully bought his public transportation Ticket for the desired subroute.
Event Flow	1) When a Public Mean of transport is used to reach an Activity the user

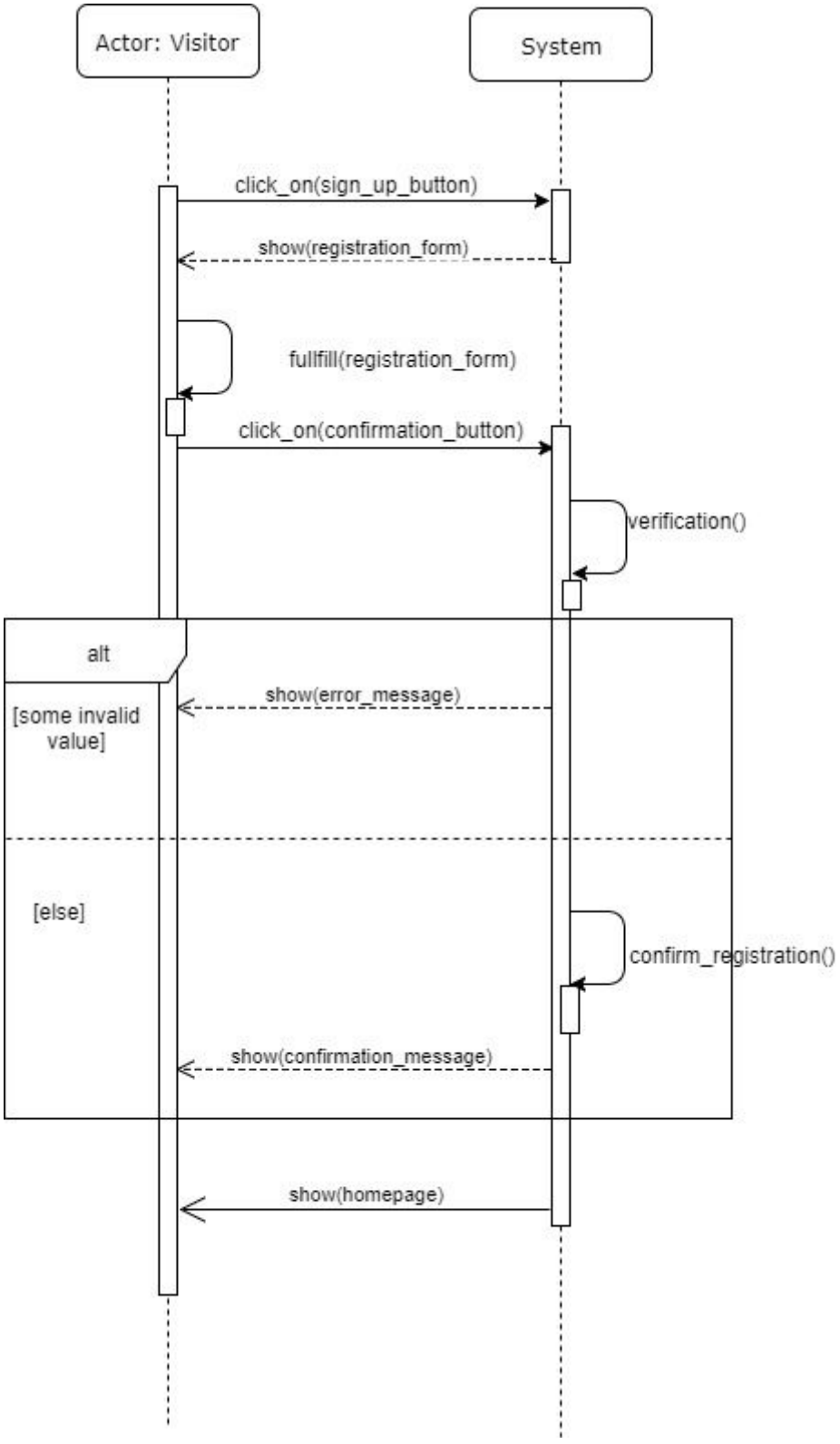
	can select the button “Buy Public Transport Tickets”. 2) The System using the APIs redirects the User to the page where he can buy the Ticket for the interested Trip. 3) The User completes other procedures on the external Website.
Exceptions	1) The Internet connection is not available. 2) A Warning is shown and the user is invited to retry later to buy the ticket.

3.B.3. Sequence Diagrams

3.B.3.1. Log In Sequence Diagram

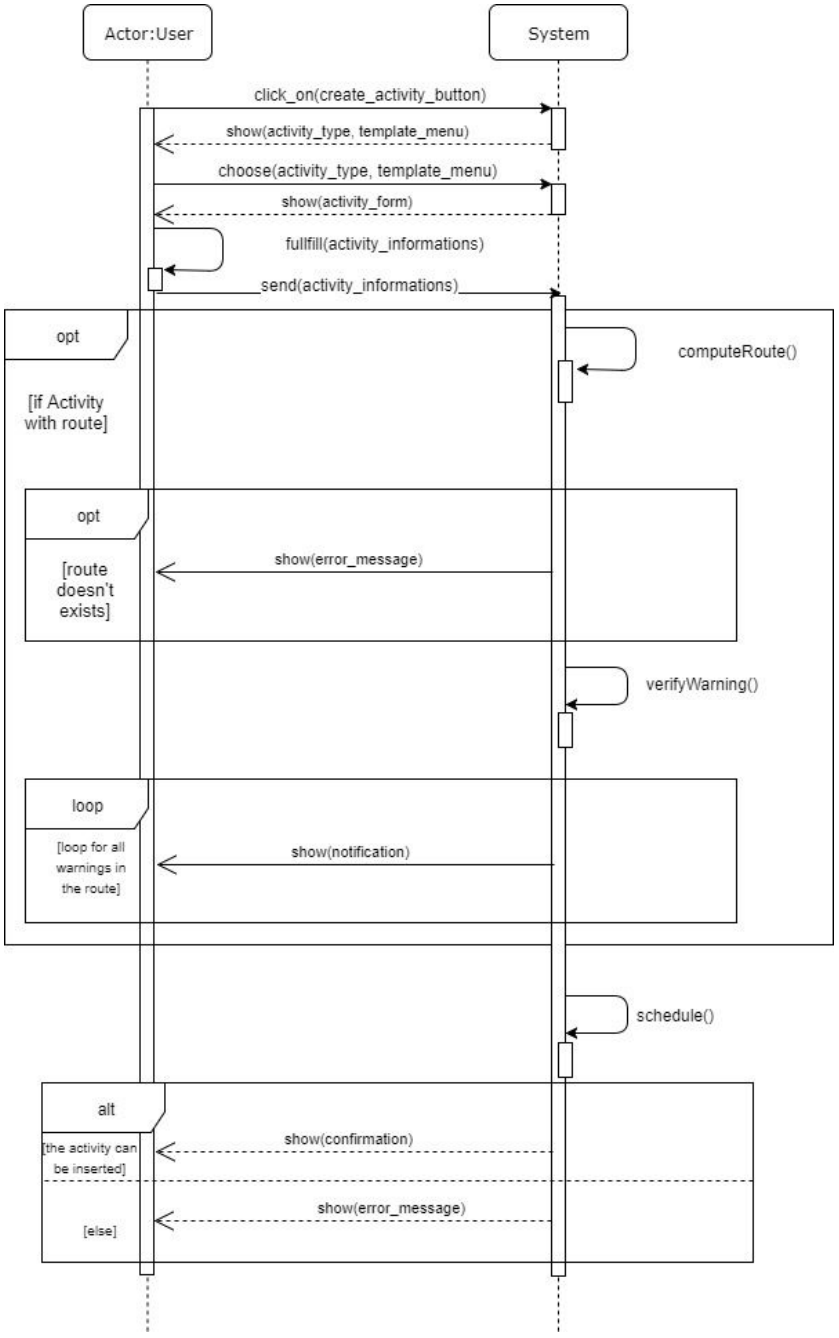


3.B.3.2. Registration Sequence Diagram



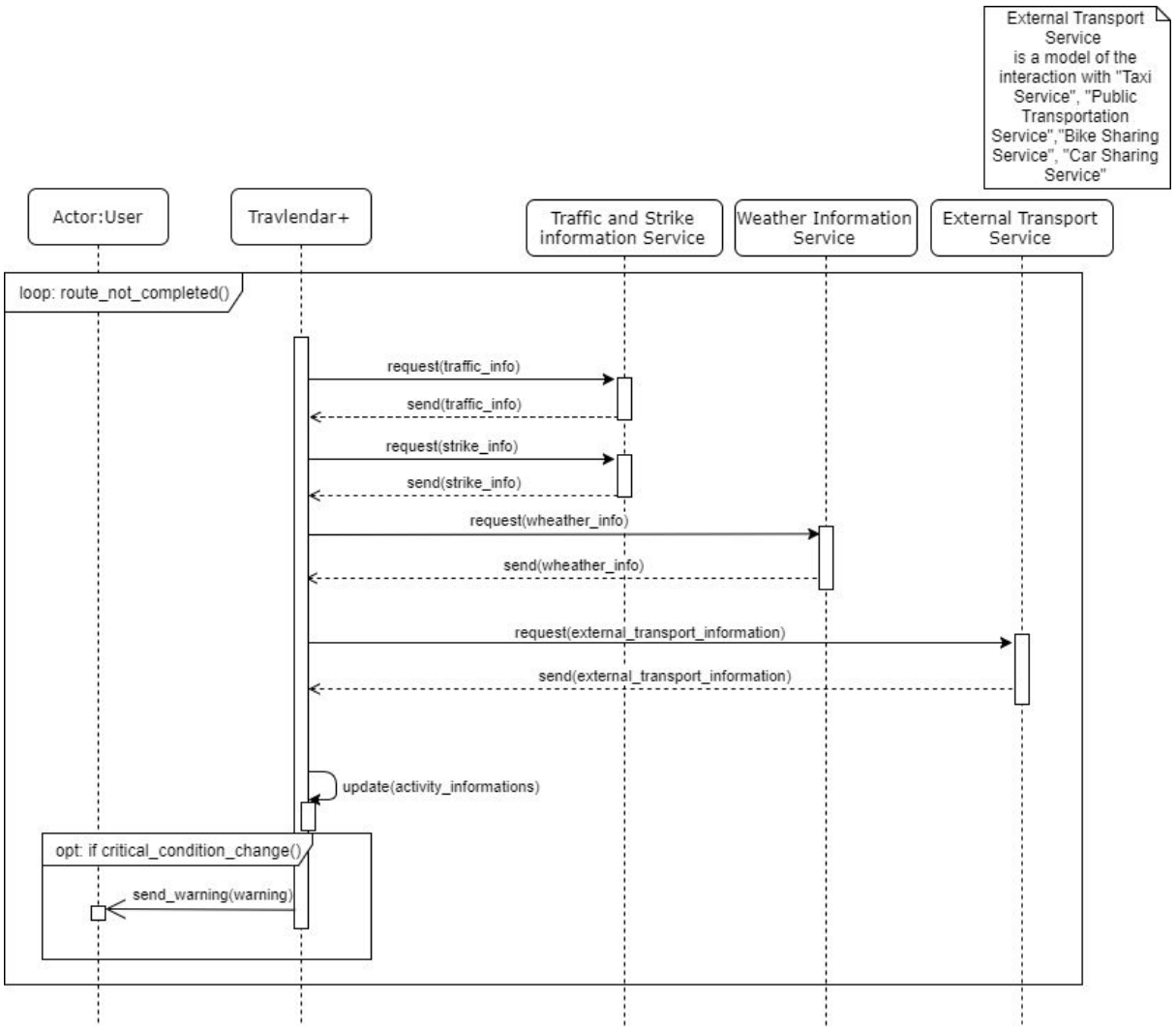
3.B.3.3. Activity Creation Sequence Diagram

This Sequence Diagram describes the functionality of creating a new Activity.

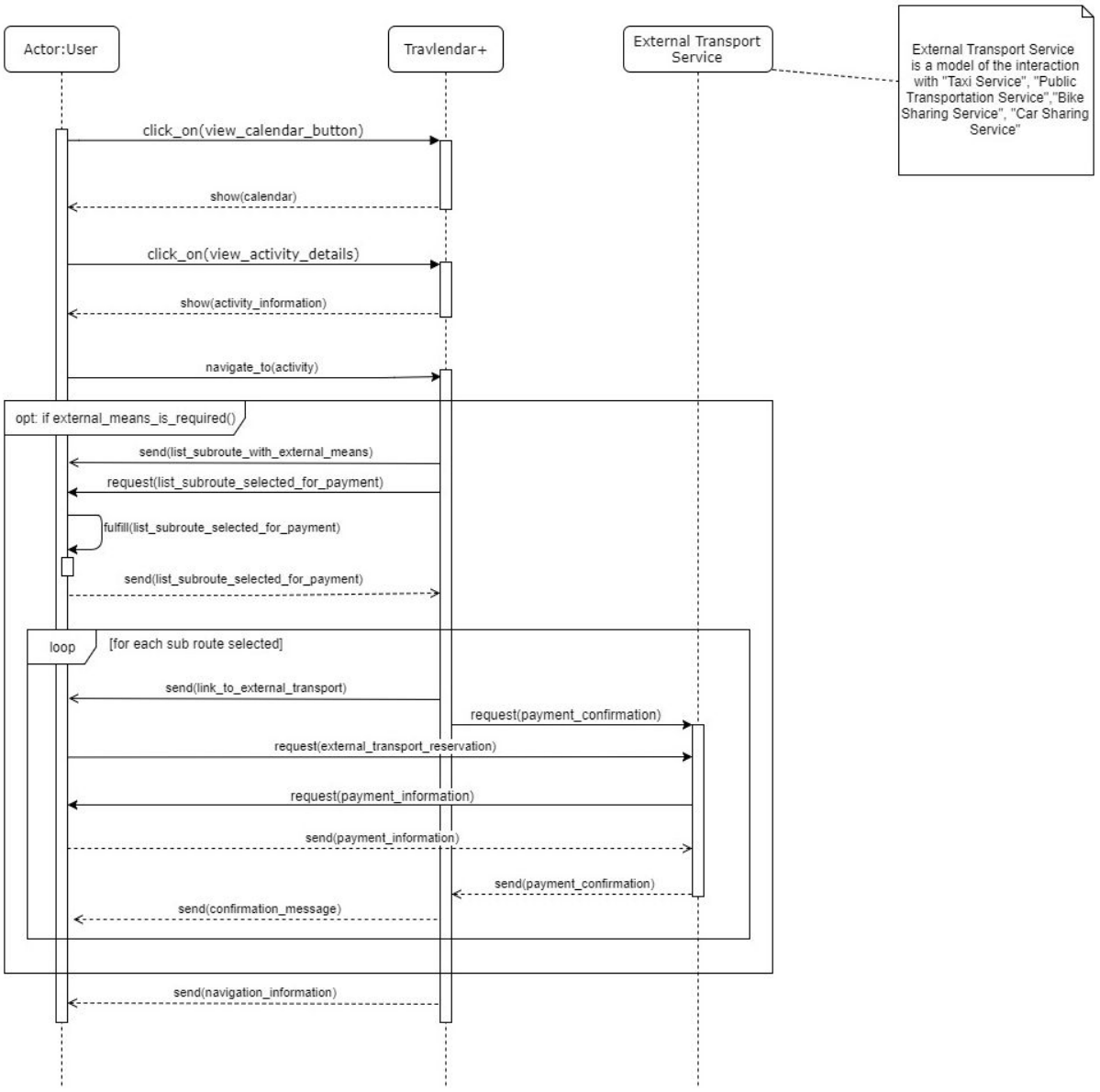


***The method computeBestRoute() recalls all the activities of the methods described in the one of the following Sequence Diagram, "Compute Navigation Sequence Diagram", aim to calculate the best path by collecting external information.**

3.B.3.4 Navigation Computation Sequence Diagram

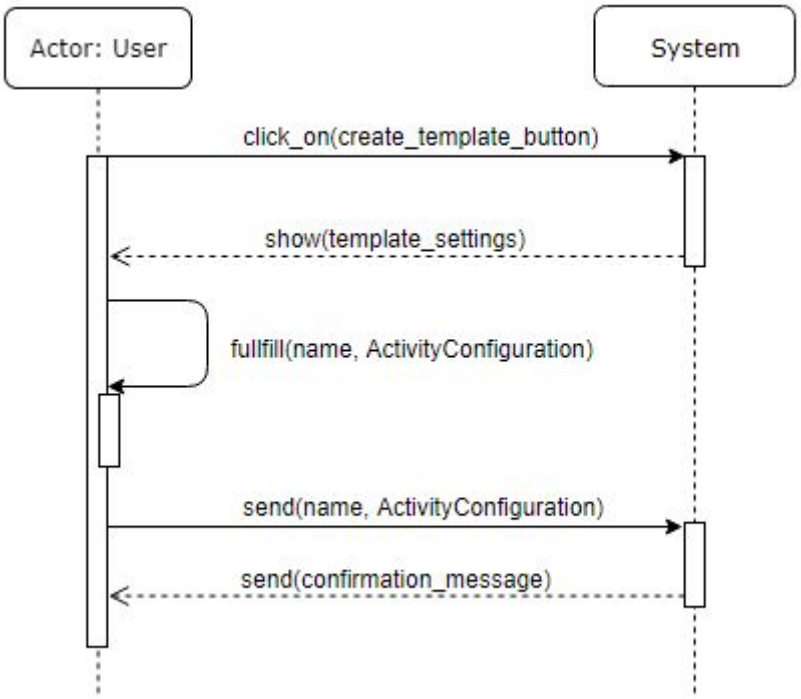


3.B.3.5 Navigation to Activity Sequence Diagram



Note: "list_subroute_selected_for_payment" may contain also a car / bike to rent.

3.B.3.6. Template Creation Sequence Diagram



3.C Performance Requirements

The system has to have low response time and to held a great number of simultaneous requests in an acceptable time, which means always less than 25 seconds for each one. When a destination address is inserted, the map with the route will be available in most 15 seconds.

3.D Design constraints

3.D.1 Standards compliance

The user must ensure that the personal use of the application is in compliance with current regulations and laws of his own country.
The user must give consent to the application to access the GPS before using the application. The application will store and process personal data only for the purpose of properly running the service.

3.D.2 Hardware limitations

The mobile application works only in Android, iOS and Windows Phone devices. The client app must be able to access the GPS data of the user's phone. Connection for mobile: 3G or

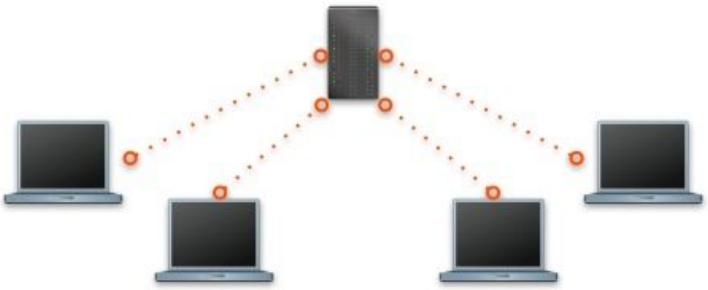
4G connection. To install and use the app the user must have 120MB of free space in his device. Display resolution must be 640x960 or more.
Internet connection: The application must be able to work if the user dispose of a 2Mb/s connection.

3.D.3 Any other constraint

The application may prohibit scheduling activities in some specific destinations, for example for military reasons. The system is not designed for life-critical applications.

3.E Software System Attributes

3.E.1 Reliability



The system is designed for the only centralized environment. The reliability of the system is strictly related to the reliability of the server it runs on. The system must ensure the truthfulness and accurateness of the data shown to the Users.

3.E.2 Availability

The system must guarantee an availability of 99.4%. This means approximately 2 days of downtime per year. Normally the system should be accessible 24 hours per day. Within 2 years we want to bring this percentage to 99.6%. Application updates are frequent but only a very small part could stop ordinary operations.

3.E.3 Security

Because Activity scheduling is a strictly personal matter, for privacy issues it must be ensured that User passwords are properly stored in the system. In particular, passwords and also personal data are stored, encrypted and not in text format, in a proper database.
The users' passwords are stored in the database using a proper hashing mechanism.
Every 60 days the application reminds users that there is the possibility to change their password in order to gain more security of their data.

All the communications between clients and server must be protected by strong encryption using the SSL protocol. The server does not allow visitors and users to establish unsafe connections.
The system has a software that detects bots (non-human users) in order to prevent potential external malicious actions. The server is thus able to protect itself against possible DDoS attacks.

3.E.4 Maintainability

The entire application code, written mainly in Java, will be documented to facilitate the work of current developers and to well inform future developers about how it has been developed and how application works. A standard for writing code will be used: JavaDoc and, if necessary, JML specifications. Additionally, the MVC Pattern will be used for consistency reasons and for facilitate code maintenance.

3.E.5 Portability

The mobile application must be supported by the last 3 major versions of Android, iOS and Windows Phone. There is no web application for this software. The system must also guarantee an high level of scalability.

4. Formal analysis using Alloy

The alloy model is mainly focuses on activity scheduling. The constraints related to the warnings (unreachable, rain...) and time sliding were explained. For brevity issues some parameters and classes have been omitted. One purpose is to show the constraints related to the subroutes of a path and their logistics scheduling. One of the results is the complete simulation of a calendar; After the presentation of the code, a practical explanatory example has been illustrated (section 4.5 Generated world).

4.1 Signatures

```
sig Stringa{}

sig User{
        name: one Stringa,
        surname: one Stringa,
        username: one Stringa,
        password: one Stringa,
        attend: set Activity
}
```

```
abstract sig Activity{
	state: one ActivityState,
	route: lone Route,
	schedulingDate: one Int,
	endActivityDate: one Int,
	alert: set Warning
}{
	schedulingDate >= 0 and
	endActivityDate > schedulingDate
}

sig Meeting extends Activity{}

sig FlexibleActivity extends Activity{
	schedulingIntervalBegin: one Int,
	schedulingIntervalEnd: one Int
}{
	schedulingIntervalBegin >= 0 and
	schedulingIntervalEnd > schedulingIntervalBegin
}

sig Route{
	stage: some SubRoute
}

sig SubRoute{
	state: one ActivityState,
	beginDate: one Int,
	endDate: one Int,
	alert: lone Warning
}{
	beginDate >= 0 and
	endDate > beginDate
}

abstract sig ActivityState{}
sig IN_PROGRESS extends ActivityState{}
sig SCHEDULED extends ActivityState{}
sig COMPLETED extends ActivityState{}
```

```
abstract sig Warning{}
sig UNREACHABLE extends Warning{}
sig RAIN extends Warning{}
sig STRIKE extends Warning{}

//Used for time clock
sig System{
	currentTime: one Int
}{
	currentTime>=0
}

//There is only one system with its current Time
fact onlyOneSystem{
	#System = 1
}
```

4.2 Facts

```
//There is only one system with its current Time
fact onlyOneSystem{
	#System = 1
}

//There is not two users with the same username
fact noTwoUsersWithSameUsername{
	no disj u1, u2: User | u1.username = u2.username
}

//An Activity is completed if and only if it's already finished
fact completedActivity{
	all a: Activity | a.state in COMPLETED
		iff
	(all sys: System | sys.currentTime >= a.endActivityDate)
}

//An Activity is scheduled if and only if has not yet begun and neither the route has begun
fact scheduledActivity{
	all a: Activity, sys: System | a.state in SCHEDULED
```

```
                iff
        (sys.currentTime =< a.schedulingDate and
         (all s: SubRoute | s in a.route.stage implies s.state in SCHEDULED))
}

//Each activity has its (unique) creator
fact allActivityhasACreator{
        all a: Activity | one u : User | a in u.attend
}

//There are no routes without the corresponding activity
fact allRouteshasAMotherActivity{
        all r: Route | one a : Activity | r in a.route
}

//There are no subroutes without the corresponding route
fact allSubRouteshasAMotherRoute{
        all s: SubRoute | one r : Route | s in r.stage
}

//All subroutes scheduled prior to the current date are completed
fact completedSubRoute{
        all s: SubRoute | s.state in COMPLETED
                iff
        (all sys: System | s.endDate <= sys.currentTime)
}

//All subroutes scheduled after to the current date are scheduled
fact scheduledSubRoute{
        all s: SubRoute | s.state in SCHEDULED
                iff
        (all sys: System | s.beginDate >= sys.currentTime)
}

//all subroutes in progress have sys.currentTime between begin-time and end-time
fact inProgressSubRoute{
        all s: SubRoute | s.state in IN_PROGRESS
                iff
                        (all sys: System | s.beginDate < sys.currentTime
                                and sys.currentTime < s.endDate)
```

```
}

//there can be no two activities at the same time
//e.g. Luca has two meeting, one from 2:00 to 5:00 and  one from 3:00 to 6:00. This is
//impossible because at 4:00 Luca should attend two different meetings
fact noTwistedActivity{
	no disj a1, a2: Activity | one u: User |
		a1 in u.attend and a2 in u.attend and
			(some t: Int |
				t >= a1.schedulingDate and
				t >= a2.schedulingDate and
				t < a1.endActivityDate and
				t < a2.endActivityDate
			)
}

//The example also applies to subroutes:
//If I'm traveling for meeting 2 then I can not attend the meeting 1 at the same time
fact noTwistedActivityOneWithARoute{
	no disj a1, a2: Activity | one u: User |
		#a1.route = 1 and #a2.route = 0 and
		a1 in u.attend and a2 in u.attend and
			(some t: Int |
				t >= a2.schedulingDate and
				t < a1.endActivityDate and
				t < a2.endActivityDate and
				(some s1: SubRoute | s1 in a1.route.stage and
					t >= s1.beginDate)
			)
}

//The example also applies to two subroutes:
//If I'm traveling for meeting 1 then I can not traveling for meeting 2 at the same time
fact noTwistedActivityTwoWithARoute{
	no disj a1, a2: Activity | one u: User |
		#a1.route = 1 and #a2.route = 1 and
		a1 in u.attend and a2 in u.attend and
			(some t: Int |
				t < a1.endActivityDate and
				t < a2.endActivityDate and
```

```
                        (some s1: SubRoute | s1 in a1.route.stage and
                                t >= s1.beginDate) and
                        (some s2: SubRoute | s2 in a2.route.stage and
                                t >= s2.beginDate)
                )
}

//there can be no two subroutes related of the same route scheduled at the same time
fact noTwistedSubRoutes{
        no disj s1, s2: SubRoute | one r: Route |
                s1 in r.stage and s2 in r.stage and
                        (some t: Int | t >= s1.beginDate and
                                t >= s2.beginDate and
                                t < s1.endDate and
                                t < s2.endDate)
}

//there can no be a subroute scheduled after the end of activity
fact noLateOrUselessSubRoutes{
        no s: SubRoute | one a: Activity |
                s in a.route.stage and
                s.endDate >= a.endActivityDate
}

//all flexible activities must comply with their construction constraint
fact allFlexibleActivityAreScheduledInTheGivenInterval{
        all f: FlexibleActivity |
                f.schedulingIntervalEnd >= f.endActivityDate and
                f.schedulingIntervalBegin =< f.schedulingDate and
                (all s: SubRoute | s in f.route.stage implies
                        f.schedulingIntervalBegin =< s.beginDate)
}

//WARNING CONSTRAINS

//a warning always belongs to one activity
fact oneActivityForEachWarning{
        all w: Warning | one a:Activity| w in a.alert
}
```

```
//a warning always belongs to one or zero subroute
fact oneSubRouteForEachWarning{
	all w: Warning | lone s:SubRoute | w in s.alert
}

//if an activity contains a warning and a route then there is a subroute that caused that
//warning. This subroute belongs to the activity's route
fact thereIsAlwaysACauseOfTheWarning{
	all a: Activity | #a.route = 1 implies a.alert = a.route.stage.alert
}

//an activity without a route can have unreachable warning,
//e.g. Luca, in Italy, wants to arrive at New York in 5 minutes
fact activityWithoutRouteCanHaveOnlyUnreachableWarning{
	all a: Activity | #a.route = 0 implies
		(all w: Warning | w in a.alert implies w in UNREACHABLE)
}

//a subroute is called unreachable if it makes the user late for the appointment.
fact definitionOfUnreachableSubRoute{
	all s: SubRoute |
		(some u: UNREACHABLE | u in s.alert)
			iff
		(s.endDate > s.~stage.~route.schedulingDate)
}

//the system can not scheduled unreachable flexible activity when it can fine scheduled the
//same flexible activity reachable.
fact noUselessUnreachableWarningForFlexibleActivities{
	all f: FlexibleActivity |
		(some u: UNREACHABLE | u in f.alert)
			implies
		((some a: Activity | f.endActivityDate = a.schedulingDate
			and f.~attend = a.~attend)
				or
		(some s: SubRoute | f.endActivityDate = s.beginDate
			and f.~attend = s.~stage.~route.~attend))
}
```

4.3 Assertions

```
//if an activity with a route is completed then all its subroutes, that belongs to the
//activity's route, are completed
assert completedSubRoutesOnCompletedActivity{
        all a: Activity | a.state in COMPLETED and #a.route = 1
                implies
        (all s: SubRoute | s in a.route.stage implies a.state in COMPLETED)
}
//check completedSubRoutesOnCompletedActivity
//OK
```

```
Executing "Check completedSubRoutesOnCompletedActivity"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  11455 vars. 480 primary vars. 29992 clauses. 213ms.
  No counterexample found. Assertion may be valid. 4ms.
```

```
//if an activity with a route is scheduled then all its subroutes, that belongs to the activity's
//route, are scheduled
assert scheduledSubRoutesOnScheduledActivity{
        all a: Activity | a.state in SCHEDULED and #a.route = 1
                implies
        (all s: SubRoute | s in a.route.stage implies a.state in SCHEDULED)
}
//check scheduledSubRoutesOnScheduledActivity
//OK
```

```
Executing "Check scheduledSubRoutesOnScheduledActivity"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  11455 vars. 480 primary vars. 29992 clauses. 281ms.
  No counterexample found. Assertion may be valid. 3ms.
```

```
//if an activity with a route is in progress then there at most one subroute in progress
assert subRoutesConstrainsOnActivityInProgress{
        all a: Activity  | a.state in IN_PROGRESS and #a.route = 1
                implies
        (thereIsOneSubRouteInProgress[a] or thereIsNoSubRouteInProgress[a])
}
```

```
//check subRoutesConstrainsOnActivityInProgress
//OK
```

```
Executing "Check subRoutesConstrainsOnActivityInProgress"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  11439 vars. 477 primary vars. 29941 clauses. 162ms.
  No counterexample found. Assertion may be valid. 60ms.
```

```
//A user can have at most one activity in progress
assert onlyOneActivityInProgressForEachUser{
        all u: User | lone a:Activity |
                a in u.attend and a.state in IN_PROGRESS
}
//check onlyOneActivityInProgressForEachUser
//OK
```

```
Executing "Check onlyOneActivityInProgressForEachUser"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  11414 vars. 477 primary vars. 29870 clauses. 117ms.
  No counterexample found. Assertion may be valid. 339ms.
```

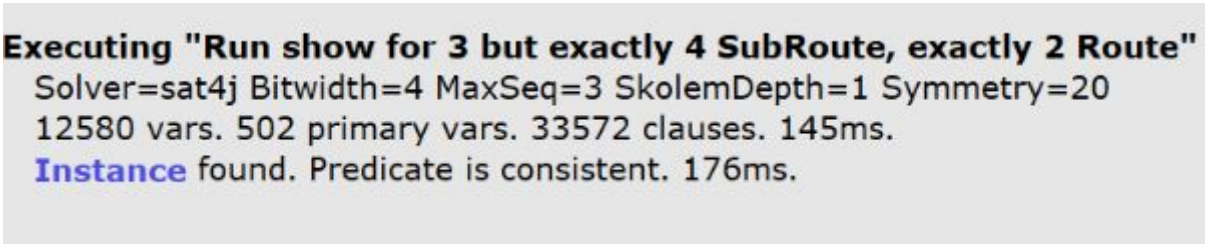
```
//If an activity has no warning then all his subroutes has no warning
assert noWarningActivityImpliesNoWarningSubRoutes{
        all a: Activity | a.alert = none implies
                (all s: SubRoute | s in a.route.stage implies s.alert = none)
}
//check noWarningActivityImpliesNoWarningSubRoutes
//OK
```

```
Executing "Check noWarningActivityImpliesNoWarningSubRoutes"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  11457 vars. 480 primary vars. 29941 clauses. 148ms.
  No counterexample found. Assertion may be valid. 26ms.
```

4.4 Predicates

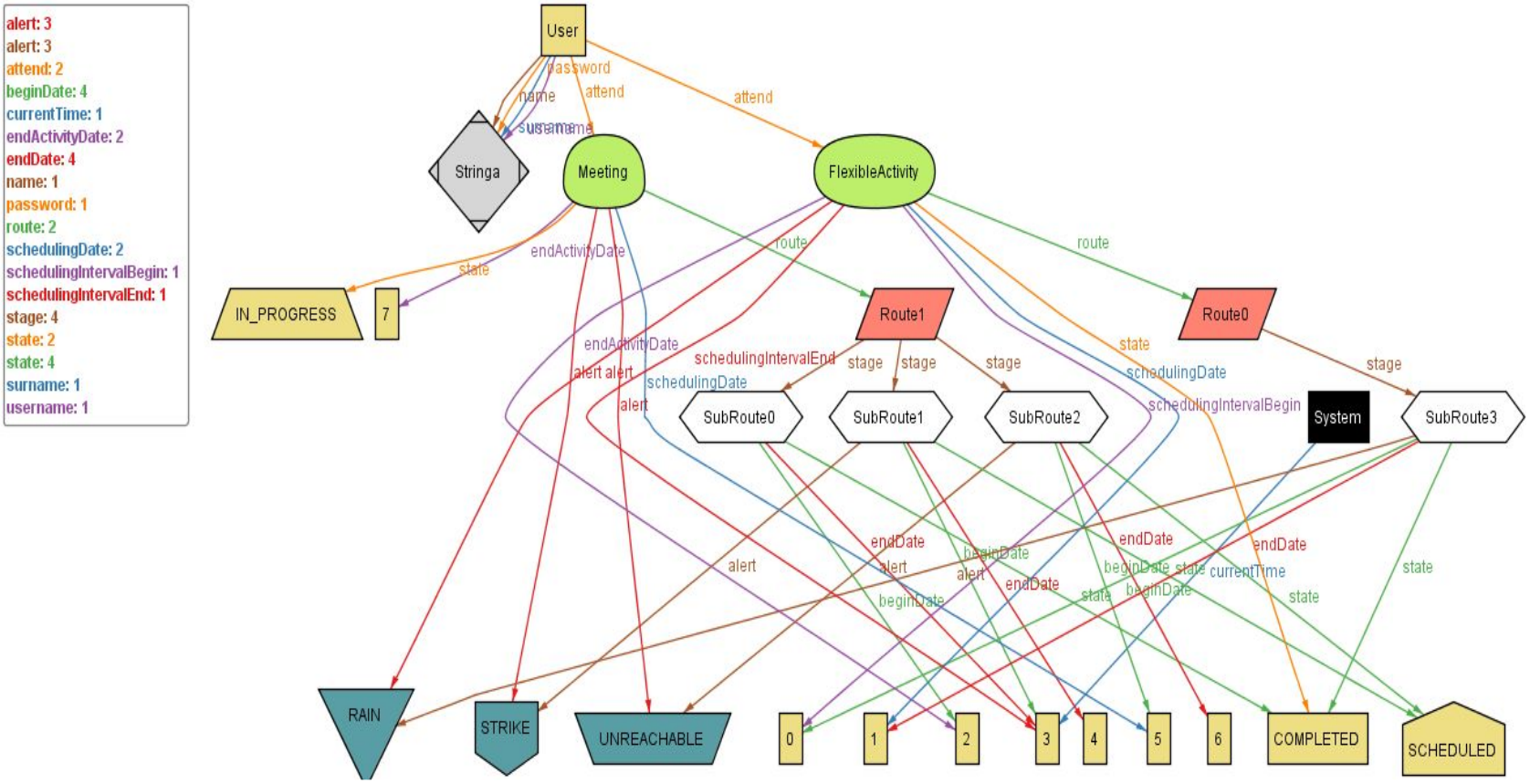
```
pred addSubRoute[r,r':Route, s: SubRoute]{
        r'.stage = r.stage + s
}
```

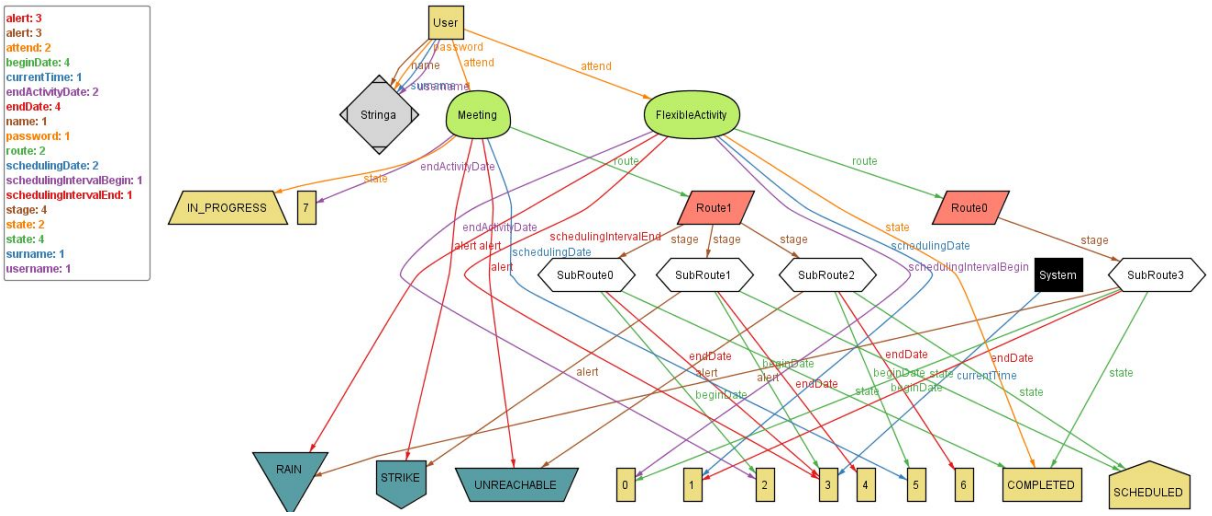
```
pred thereIsOneSubRouteInProgress[a: Activity]{
	one s: SubRoute | s in a.route.stage and s.state in IN_PROGRESS
}

pred thereIsNoSubRouteInProgress[a: Activity]{
	no s: SubRoute | s in a.route.stage and s.state in IN_PROGRESS
}

pred twousers{
	#User = 2
	#Activity = 3
	#{x: Warning | x in UNREACHABLE} > 0
}

pred activitywithoutroute{
	#User = 1
	#{x: Activity | #x.route = 0} >= 1
	#Activity > 2
}

pred rainday{
	#User < 3
	#RAIN = 3
	#{x: Warning | x not in RAIN} = 0
	#{x: Meeting | #x.alert = 1} = 1
}

pred showOnlyMeetingInProgressWithoutSubRouteInProgress{
	#{x: Meeting | thereIsNoSubRouteInProgress[x]} = #Activity
}

pred show{
	#User = 1
	#{x: FlexibleActivity | not x.alert = none} = 1
	#{x: Warning | x not in UNREACHABLE} = 1
	#{x: Activity | x.state in IN_PROGRESS} > 0
}
run show for 3 but exactly 4 SubRoute, exactly 2 Route
```



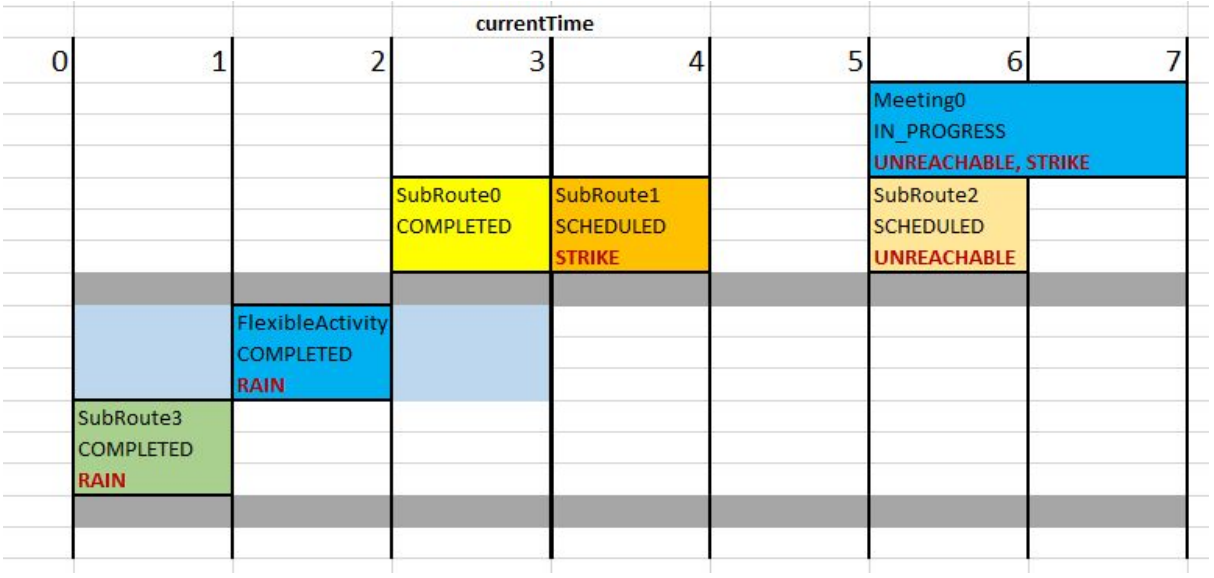
Alloy Analyzer 4.2 example of console output

4.5 Generated world


alert: 3
alert: 3
attend: 2
beginDate: 4
currentTime: 1
endActivityDate: 2
endDate: 4
name: 1
password: 1
route: 2
schedulingDate: 2
schedulingIntervalBegin: 1
schedulingIntervalEnd: 1
stage: 4
state: 2
state: 4
surname: 1
username: 1
User
password
name
attend
attend
Stringa
Meeting
FlexibleActivity
route
route
state
endActivityDate
IN_PROGRESS
7
Route1
Route0
endActivityDate
schedulingIntervalEnd
stage
stage
stage
state
schedulingDate
stage
alert alert
schedulingDate
alert
schedulingIntervalBegin
System
SubRoute0
SubRoute1
SubRoute2
SubRoute3
endDate
endDate
endDate
alert
beginDate
alert
beginDate
state
currentTime
alert
endDate
state
beginDate
state
state
RAIN
STRIKE
UNREACHABLE
0
1
2
3
4
5
6
COMPLETED
SCHEDULED



Explanation:

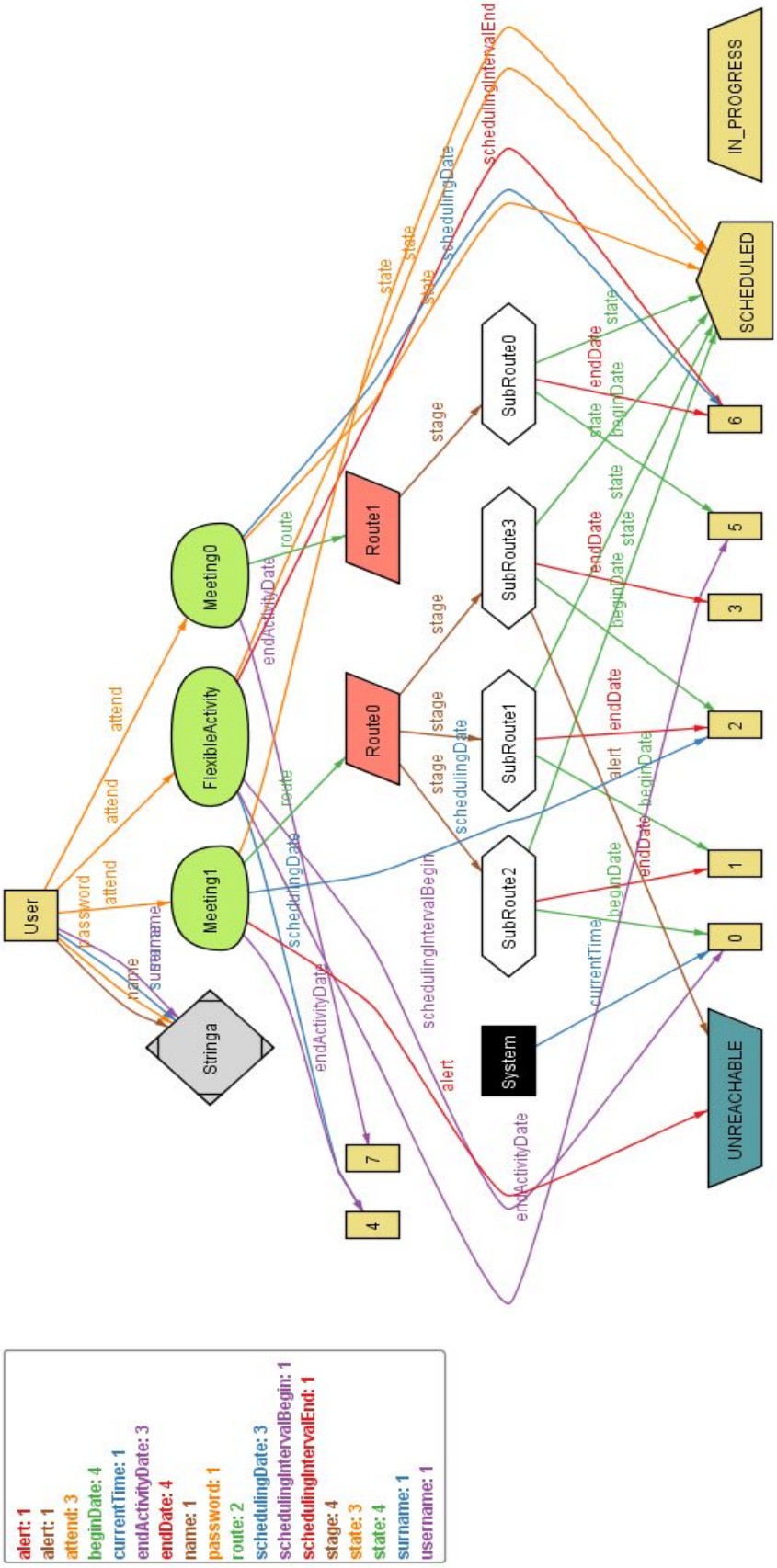


CurrentTime is 3. From 0 to 1 the user completed the subRoute3, for example by using bike. The application pointed out that it rained. The duration of the flexible activity was 1 and could be entered at any time between 0 and 3. In order not to hinder the route to reach meeting 0, the application has scheduled the activity from 1 to 2. In this way we let the user use the time from 2 to 3 to take the train (subRoute0). The application reported also the possibility of a strike for the bus line 90 (subRoute1) from 3 to 4. The next airplane lands at 5, so the user cannot arrive punctually at meeting 0: a unreachable warning is then signaled. The meeting at currentTime = 3 is in progress because the user is travelling for reach it.

Another possible show:

```
pred activitywithoutroute{
        #User = 1
        #{x: Activity | #x.route = 0} >= 1
        #Activity > 2
}
```

run activitywithoutroute for 3 but exactly 4 SubRoute, exactly 2 Route



5. Effort Spent

Hours of work per day:

5/10/2017 (Together): 2hrs
7/10/2017 (Together): 2hrs
10/10/2017 (Together): 4hrs
12/10/2017 (Together): 2hrs
17/10/2017 (Together): 4hrs
21/10/2017 (Together): 8hrs
23/10/2017 (Together): 5hrs
26/10/2017 (Together): 4hrs
28/10/2017 (Together): 4hrs

Hours do not include individual work, about approximately 9-10 hours of work per member of the group.

6. References