Travlendar+

Barcella Lorenzo Bellini Alberto Mario Cavalli Luca

Travlendar+

Team members:

- Barcella Lorenzo
- Bellini Alberto Mario
- Cavalli Luca

INDEX

1. Introduction

1.1. Purpose

Nowadays most people live an intense and busy life. Many endeavors require scheduling meetings at various locations all over a city or a region. Moreover, when schedules get more and more intense, observing them while moving in the chaos of a city may become challenging.

The purpose of Travlendar+ is to provide a solution to this. We want to create a calendar-based application that automatically computes and accounts for travel times between appointments to make sure that the user is never late and to support the user in his/her travels by identifying the best mobility option at any time (taking into account events like current traffic, strikes, delays ...).

Furthermore we would like to highlight that Travlendar+ may even lead to a more eco friendly environment due to the exploitation of public transports and emission-free travel means such as bikes.

In order to have Travlendar+ reaching its full functionality the following goals have been defined:

1.1.1. Goals

[G1] to build a calendar-based application that schedules tasks and helps following the daily timetable

[G1.1] to allow every registered user to submit, edit or delete new tasks

[G1.2] when unattendable tasks are submitted (with time overlaps or in any way that makes the attendance of all tasks physically impossible for transportation limitations) a warning is created

[G1.3] to account automatically for travel times between appointments to make sure that the user is never late to attend his tasks

[G1.4] to give easy directions that guide the user to every appointment

[G2] to allow users to define their travel preferences

[G2.1] to support a multitude of travel means, including walking, biking (own or shared), public transportation (including taxis), driving (own or shared)

[G2.2] to allow users to provide reasonable constraints on different travel means

[G2.3] to allow users to specify preferences on proposed solutions (time-cost trade-offs, additional general constraints)

[G3] to suggest the best travel mean depending on the appointment, the day and the weather

[G3.1] to suggest the best combination of transportation means considering the preferences of the user

[G3.2] automatically adjust the proposed travel solutions and notify the change to the user every time an environmental event happens (such as weather conditions)

[G4] to allow users to register to T+, to access and manage their account

[G5] to support users in their travels

[G5.1] to allow users to declare a day/week/season pass and take it into account for cost computations

[G5.2] to allow users to buy transportation tickets and passes

[G5.3] to allow users to locate and reserve (if possible) the car or bike of a sharing system which is most suitable to their appointments

1.2. Scope

Travlendar+ is an application designed to support movements in and across cities. The initial geographical scope will be a single city, but we mean to address a much wider area.

Depending on size and location, each city can offer different transportation services. For instance, some will provide a car sharing while others may have a good underground network.

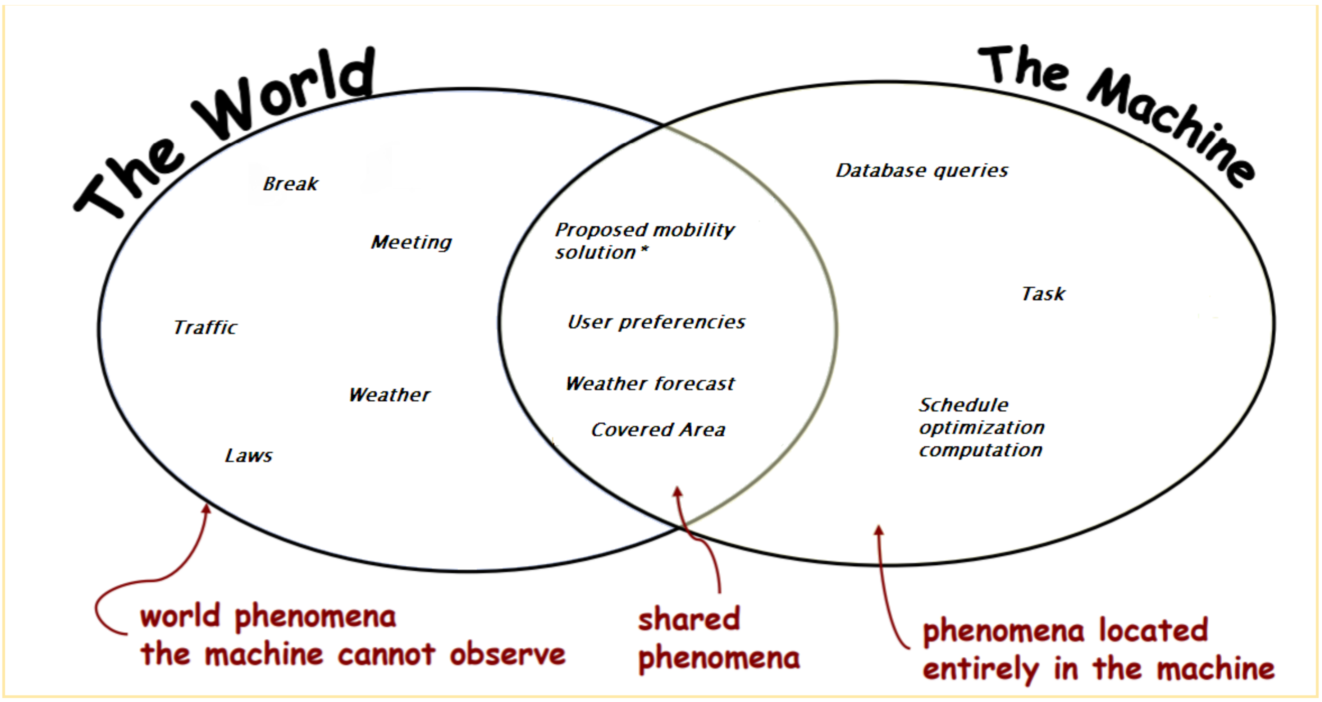
Also we assume that every city may have different rules about transportations, and that these rules may change with advanced notice.

Our users will not be able to fix appointments in uncovered areas, but they may fix appointments in partially covered, covered or fully covered areas.

Users may want to change their timetable at any moment, even during a task, or in a time-critical moment, and they may make their plan unfeasible.

To compute the most efficient combined path it is mandatory for the user to have a mobile phone with both a Global Positioning System and an internet connection and therefore we take it for granted to deliver our services.

All data from external sources (GPS, wheather, traffic… and all external services and sensors) is considered to be possibly inaccurate or corrupted and thus subject to sudden change in any moment.


The World
The Machine
Break
Meeting
Traffic
Weather
Laws
Proposed mobility solution *
User preferencies
Weather forecast
Covered Area
Database queries
Task
Schedule optimization computation
world phenomena the machine cannot observe
shared phenomena
phenomena located entirely in the machine

World, Machine and shared phenomena

1.2.1. Actors

- **User** : a person that exploits the application. Every User can be an Unregistered user or a Registered User.
  a. **Unidentified User** : an unidentified user can interact with the application but can't schedule any task nor customize his own travel preferences. He can log in or register.
  b. **Identified User** : an identified user can fully interact with the application by submitting new tasks and defining preferences. This user has been successfully authenticated through the login procedure.
- **Maintenance Team** : a group of people whose task is to keep the application up to date; this activity includes bug fixing, coverage increase and [customer service?].
  - **Developer** : operates directly on software developing and debugging.
  - **Operator** : keeps up to date the dataset for the different areas and follows their evolution.
- **External Services**: all services used to gather data about weather, supported transportation means, traffic and geographical locations.

1.3. *Definitions, Acronyms, Abbreviations*

1.3.1 Definitions

- **Task**: in the context of this document a task is considered to be the allocation of a precise amount of time in a given interval. It may optionally include the specification of a geographical place where the user should be at the start of the task. What is done in the allotted time is not important for our purpose: it may be a meeting, an appointment, a break, a step in a tour, or anything else.
- **Appointment**: a task having a specified geographical place of attendance.
- **Basic mean of transportation**: one of the following: walking, personal biking, personal driving.
- **Fully covered area**: a geographical area for which we have data about all available and supported means of transportation (if more than one service of the same type is available in this area, we must have data of at least the major one).
- **Covered area**: a geographical area for which we have data about at least one more mean of transportation other than basic ones.
- **Partially covered area**: a geographical area for which we have complete information about all and only the basic means of transportation.
- **Uncovered area**: a geographical area for which we lack information about at least one of the basic means of transportation.
- **Combined path**: a mixture of transportation means used to reach one location from another one.
- **Best Mobility Option**: either a combined or a simple path that satisfies a set of constraints and maximizes preference options.
- **Alternative Mobility Option**: either a combined or a simple path that satisfies a subset of user constraints and maximizes preference options. Proposed only when no Best Mobility Option can be found.
- **Time Coverage**: a time parameter that defaults to 24 hours. Time Coverage puts a limit to the depth of the mobility solutions provided by the application: tasks over the Time Coverage in the future will not be considered for travel calculations. TC is considered a general preference and thus it is user definable, but it cannot exceed 120 hours.
- **Maximum/Minimum Destination Waiting Time**: the maximum/minimum time a valid schedule is allowed/requested to consider waiting at the destination location when computing for a Best Mobility Option. These user-definable options default to 10/5 minutes.

1.3.2 Acronyms & Abbreviation

T+	Travlendar+
BMO	Best Mobility Option
AMO	Alternative Mobility Option
TC	Time Coverage
MDWT	Maximum Destination Waiting Time
mDWT	Minimum Destination Waiting Time

- **GPSG:** GPS gadget

1.4. *Revision history*

1.5. *Reference Documents*

- [1] Software Engineering 2 course slides.
- [2] Travlendar+ assignment.

1.6. *Document Structure*

2. Overall description

2.1. *Product perspective*

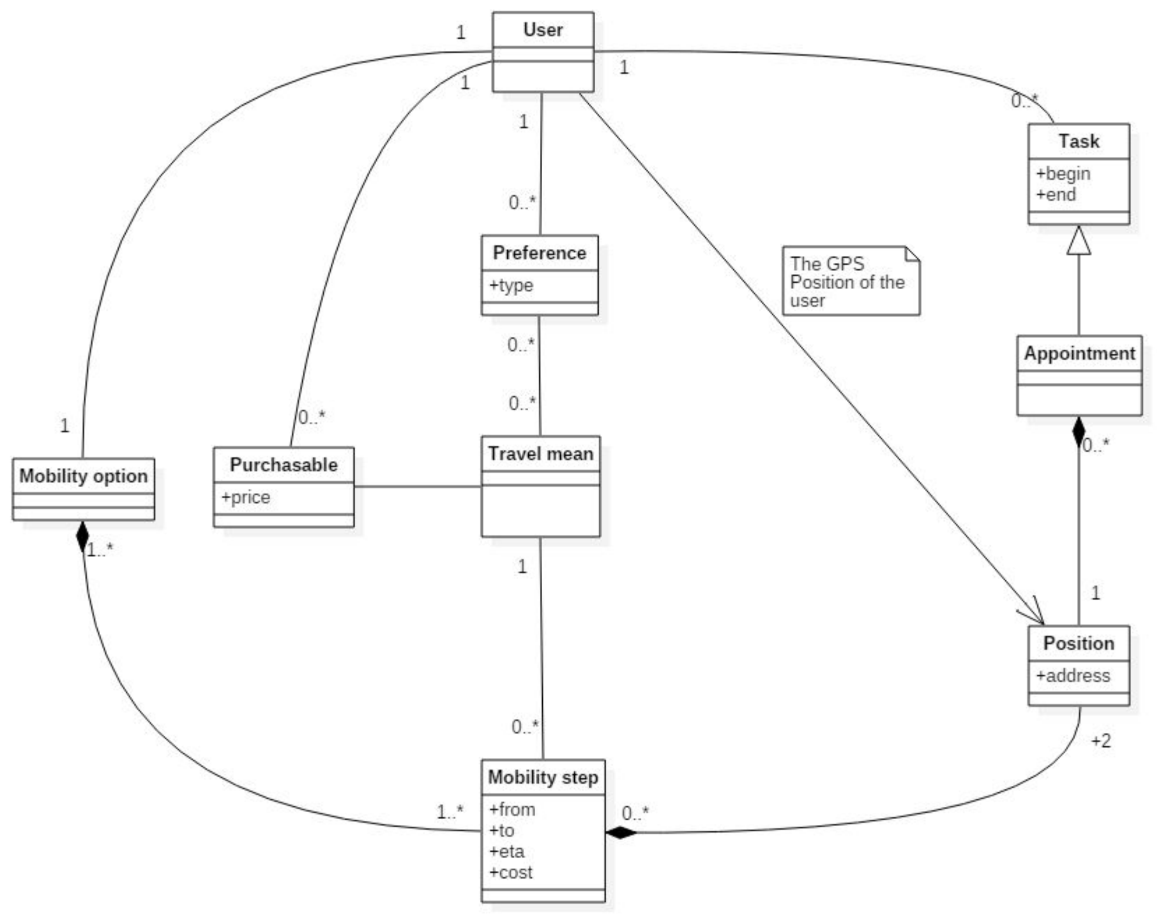
The product, Travlendar+, is designed to perform the tasks mentioned above.
To address its needs it will be made up of the following core components:

T+ Server :
The server is going to host a database containing all the required information about the users.
Servers will mainly provide two services:
- They will dispatch updates about all means of transportation needed by a specific client
- They will provide a synchronization service of tasks and computed BMOs among different devices

T+ Mobile App
This module is going to be the main interface throughout which the users are going to interact with our system.
Complex computational tasks to optimize the schedules of users will be held locally to increase the scalability of T+.
For portability reasons there will be a cross-platform application capable of running on iOS, Android and Windows Phone.
It will be available from the App Store of each device and will receive periodic updates.
Registration, task scheduling and core functions will be available with a very simple and intuitive interface to reduce the effort needed to use the app itself.

T+ Desktop
This component is meant only for schedule-programming and preferences-setting.
It will perform queries on the server to perform updates that will later be dispatched to the Mobile App.
There will not be support for computing or viewing the BMO.

Class Diagram:



2.2 *Product functions*

[G1.1] to allow every registered user to submit new tasks:

[R1] T+ allows registered users to submit, modify and detail tasks whenever the application is not computing a BMO.
[R1.1] T+ will not accept tasks without a specified duration and a time range in which they must begin
[R1.2] T+ will allow the specification of a map location in which the task should be accomplished (appointment)

[G1.2] when unattendable tasks are submitted (with time overlaps or in any way that makes the attendance of all tasks physically impossible for transportation limitations) a warning is created

[R1.3] T+ will allow to specify whether the task is compatible or not with overlappings.
[R1.3.1] T+ will not accept overlapping tasks both with a map location specification. In case they are submitted, T+ will warn the user and point out the name of the conflicting task.
[R1.4] T+ will check for a BMO that satisfies all the preference options for the new task right after a successful submission. If no suitable BMO can be found, a warning is produced.

[G1.3] automatically accounts for travel times between appointments to make sure that the user is never late to attend his tasks

[R2] T+ automatically accounts for travel times between appointments to make sure that the user is never late to attend his tasks (excluding external and sudden complication)
[R2.1] If data provided by external sources is correct and complete, then the BMO proposed by the application guarantees the user to be able to arrive in time for the appointment if followed. (in case of unexpected external complications R2.1 is not applicable due to incompleteness of data)
[R2.2] If no mobility option can be found to arrive at least mDWT minutes before the appointment time to a destination inside the set TC, a warning is produced and a BMO is computed ignoring the mDWT.
[R2.3] If no mobility option can be found to arrive in time to a destination inside the set TC, a warning is produced and a BMO is computed ignoring the arrival time of that task.
[R2.4] If the TC is less or equal to the default, T+ will provide a BMO in no more than 10 seconds.

[G1.4] to give easy directions that guide the user to every appointment
[R7] T+ will offer the possibility to display the planned mobility option step by step on a map

[G2.1] to support a multitude of travel means, including walking, biking (own or shared), public transportation (including taxis), driving (own or shared)

[R3] T+ will support a multitude of different travel means in covered areas.
[R3.1] T+ must support all common transportation means for short and medium distance travels (walking, biking, bus, tram, taxi, driving, car/bike sharing, train).

[G2.2] to allow users to provide reasonable constraints on different travel means

[R4] T+ will allow to specify BMO constraints.
[R4.1] T+ will allow the user to specify common constraints either about a specific mean of transportation or about general limitations.

[G2.3] to allow users to specify preferences on proposed solutions (time-cost trade-offs, additional general constraints)

[R4.2] T+ will allow the user to specify a minimization preference tradeoff between cost and time efficiency.

[G3.1] to suggest the best combination of transportation means considering the preferences of the user

[R4.3] T+ will compute the BMO taking into account all specified limiting parameters such as maximum walking distances, carbon footprint and so on. Provided BMOs will satisfy all constraints.
[R4.4] T+ will compute the BMO with the target of finding the best option with regards to the minimization preference specified by the user.

[G3.2] Automatically adjust the proposed travel solutions and notify the change to the user every time an environmental event happens (such as weather conditions)

[R5] T+ keeps the proposed BMO up-to-date with the current environmental situation.
[R5.1] Every time environmental data changes, the BMO is reoptimized if the change influences either its optimality or its constraint satisfaction.

[G4] to allow users to register to T+, to access and manage their account

[R6] T+ will give the possibility to register and manage all user related data.
[R6.1] T+ will allow to change the anagraphics at any moment.

[G5.1] to allow users to declare a day/week/season pass and take it into account for cost computations

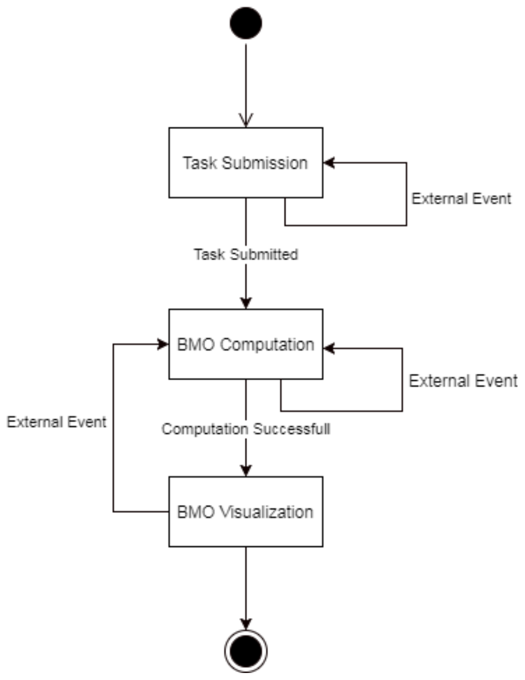
[R6.2] T+ will allow to submit and delete tickets and passes bought outside the application for specific supported means of transportation at any moment.

[G5.2] to allow users to buy transportation tickets and passes

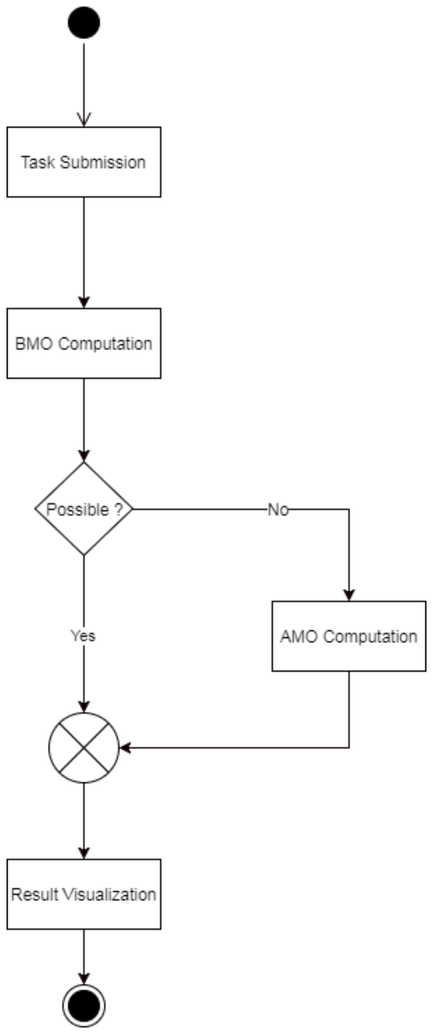
[R6.3] T+ will allow to buy new tickets for public transportation directly from the app for the most common services whenever the service is available.

[G5.3] to allow users to locate and reserve (if possible) the car or bike of a sharing system which is most suitable to their appointments

[R6.4] T+ will allow to reserve resources directly from the app for shared services whenever the service itself allows the operation.



Statechart diagram



Activity Diagram

2.2.1 **Scenario identification**

In this section we provide several different scenarios, each of which should clarify how the user interacts with the system in practical situations.

1. User registration and login

   John is a very enthusiast person but is always late for his meetings.
   He has just heard about Travlendar+ from a friend and decides to download the T+ Mobile App.
   He then opens the app and registers following the very simple procedure that requires him to enter his contact information, credentials to login later on (such as username and password) and payment method. Finally he is prompted to accept the EULA.
   After that he receives an email to confirm his address.
   John can now login into T+ (Either Mobile or Desktop) and schedule his meetings.

2. New task scheduling

   Alex has just been asked to make a speech at the University of the city where he lives in on next day. His schedule for that day is very busy, and therefore he decides to let T+ help him: he opens the mobile app, toggles the menu and with a single click he fills in the meeting details. After adding the new task T+ starts computing the BMO. Few moments later Alex receives a notification telling him that the BMO has been successfully computed and that a reminder has been set to inform him when to leave to be right on time.

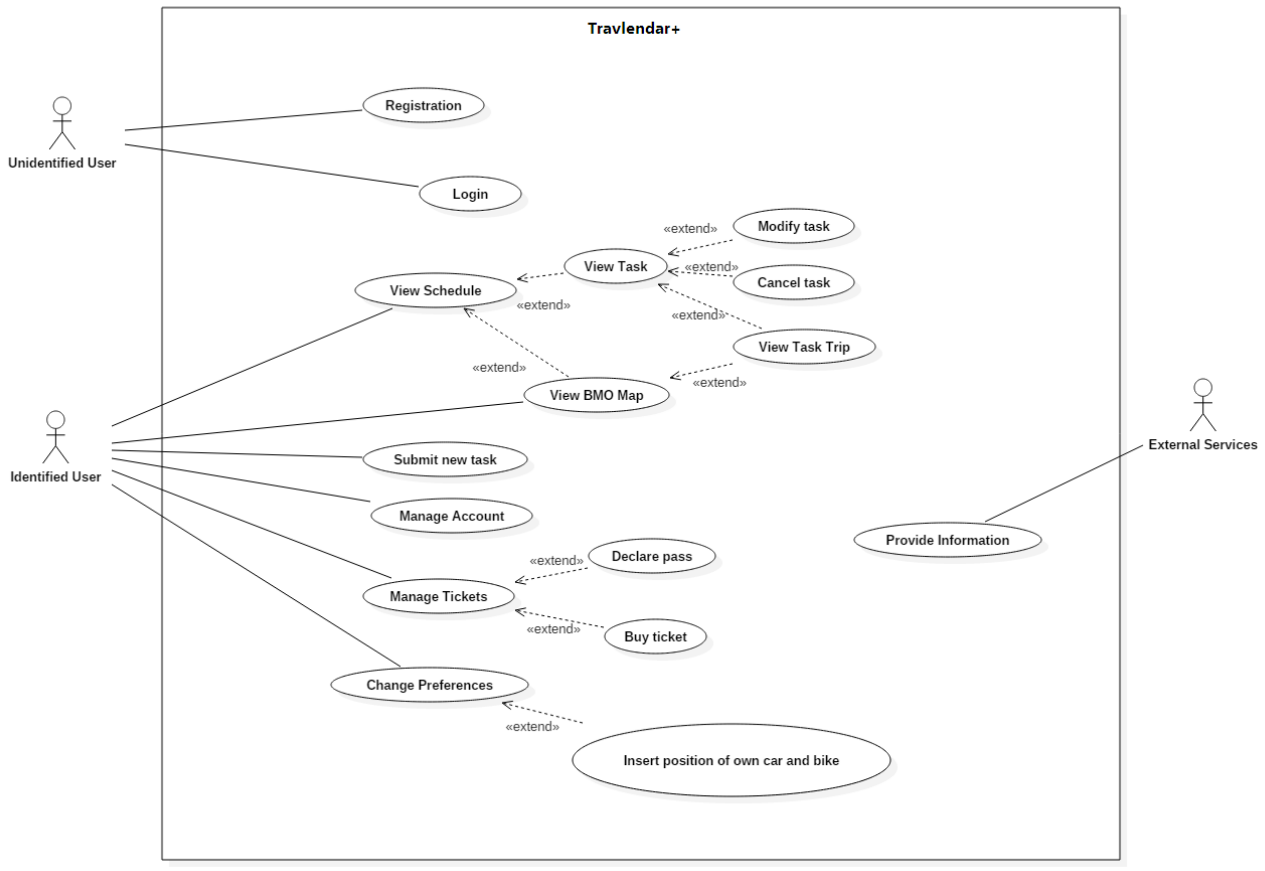
3. The city tour

   Bob is an enthusiast tourist with low budget. He used to spent a lot of time to plan his travels in order to find the best means of transportations in every city he has visited. For his next travel to visit a new city he has only to look for the destinations he would like to see and set times for each step. Then, being the city covered by T+, the app looks for the best travel options to visit all the places chosen by Bob in almost no time compared to the manual search he was used to, and with better results.

4. The urgent solution

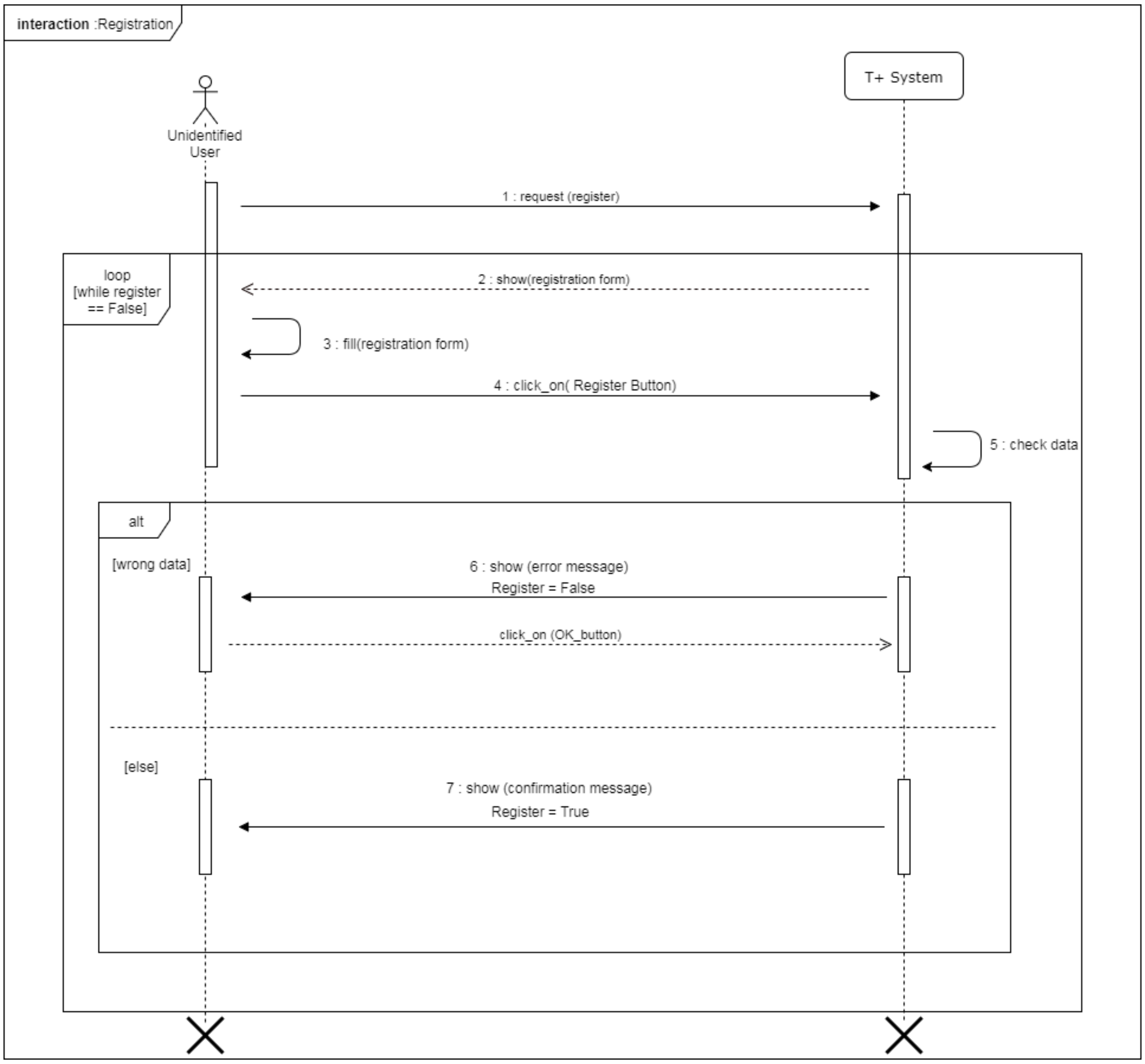
   David is a busy enterpreneur, he is used to attend many meetings in his avarage day and he uses T+ to be always sure to dinamically adjust his route and times to the city current traffic. Today his car had a breakdown while going to an important meeting. He was able to park it, then he had to disable his personal car option in T+ settings and wait some seconds to have an alternative solution. The app warned David that it couldn’t find any mobility option to arrive the minimum 5 minutes before the appointment time, and proposes a solution that is expected to arrive only 2 minutes before. Following T+ directions and purchasing all needed tickets through the app without having to stop anywhere to buy them, David could arrive just in time for the meeting.

2.2.2 Use Cases



- **Registration**

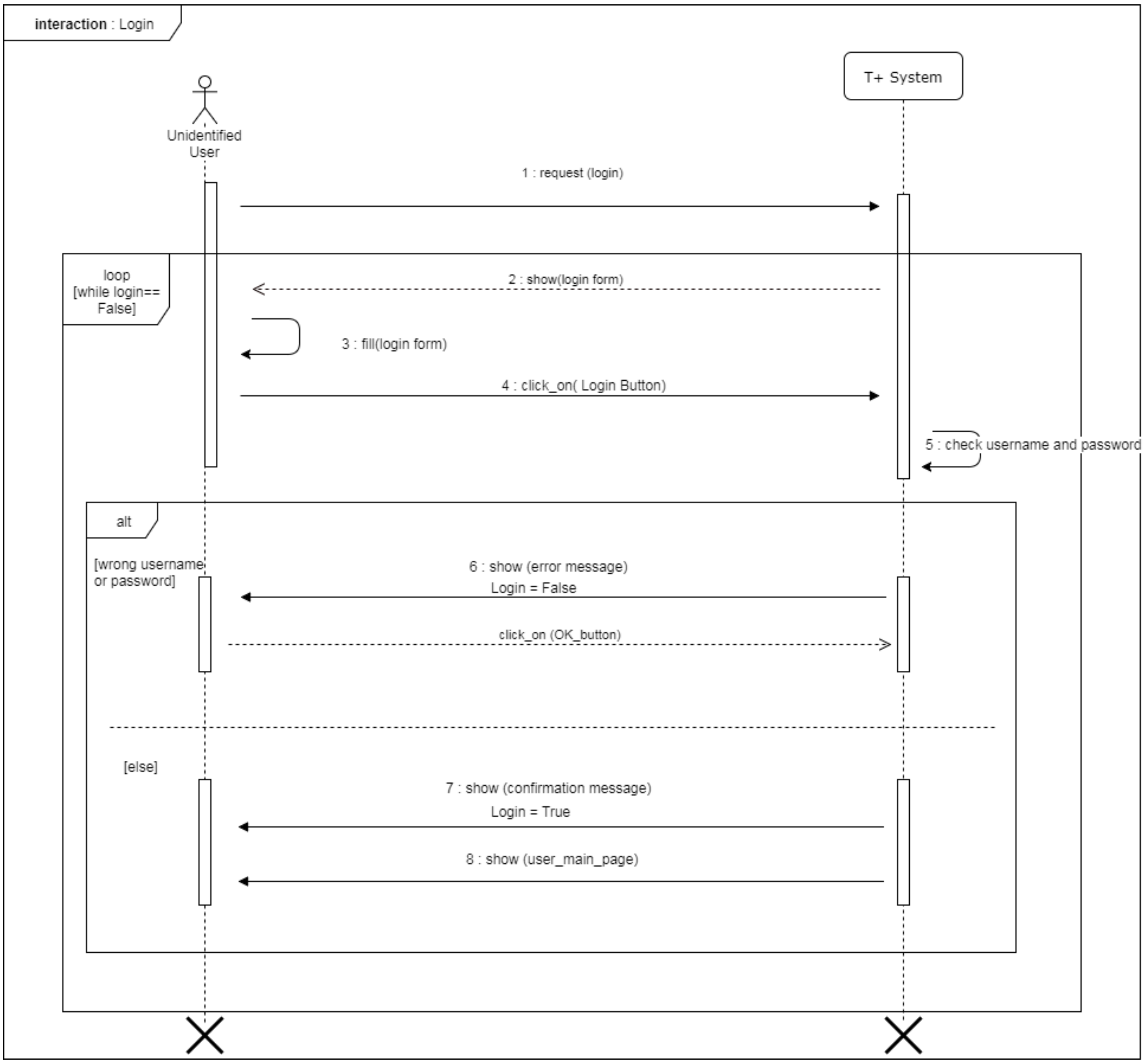
Name	Registration
Actors	Unidentified User
Related Goals	G5
Entry Condition	The user want to create an account on T+
Flow of Events	1. The user opens the registration form from her device 2. The user fill in the form with the required contact information (Name, Surname, Phone, Email, Age, Gender , Address, Username, Password) and the required payment information (Card number, CCV, Expiry) 3. The user clicks on submit 4. T+ checks the validity of the data inserted 5. T+ creates an account and notifies the user
Exit Condition	Data inserted are correct and T+ server saves on database the contact and payment information. The user receives notification of the successful registration.
Exceptions	• The personal information of the user are not valid. In this case the system stops the registration and sends a notification to the user. No data are stored. • If the registration is interrupded before its completion no data are stored. In all these two cases the user can restart the registration from the beginning.


interaction :Registration
Unidentified User
T+ System
1 : request (register)
loop [while register == False]
2 : show(registration form)
3 : fill(registration form)
4 : click_on(Register Button)
5 : check data
alt
[wrong data]
6 : show (error message) Register = False
click_on (OK_button)
[else]
7 : show (confirmation message) Register = True

Registration sequence diagram

- **Login**

Name	Login
Actors	Unidentified User
Related Goals	G5
Entry Condition	The registered user want to log in.
Flow of Events	1. The user select login in the welcome page 2. The user insert his username and password 3. The user clicks on the log in button 4. T+ redirects the user in his own main page
Exit Condition	The user insert correct username and password and is correctly redirected to his own main page.
Exceptions	If the credentials are not valid T+ doesn't redirect the user to his own main page but notifies him of the error and allows to input his username and password again.


interaction : Login
Unidentified User
T+ System
1 : request (login)
loop [while login== False]
2 : show(login form)
3 : fill(login form)
4 : click_on(Login Button)
5 : check username and password
alt
[wrong username or password]
6 : show (error message) Login = False
click_on (OK_button)
[else]
7 : show (confirmation message) Login = True
8 : show (user_main_page)

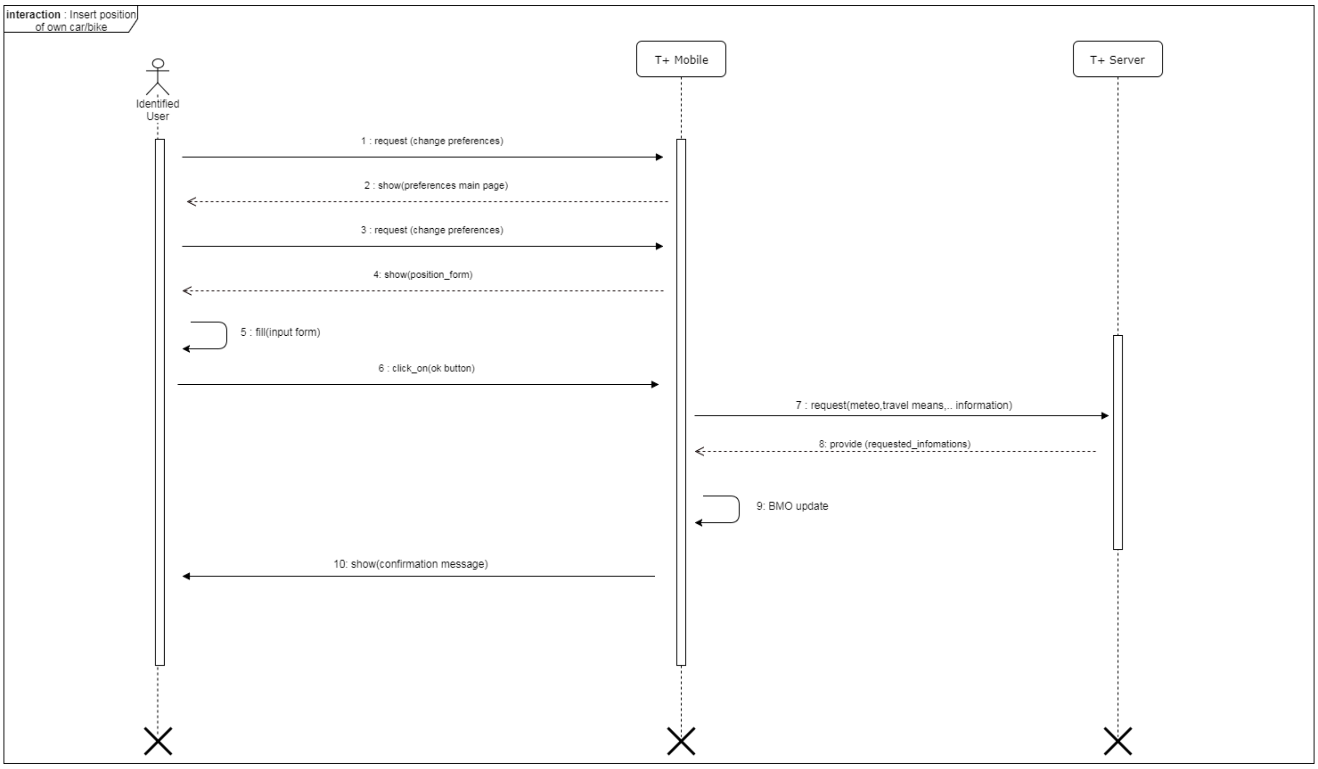
Login sequence diagram

- **Change Preferences**

Name	Settings
Actors	Identified User
Related Goals	G2, G3
Entry Condition	The user must be logged in. The user wants to modify his preferences.
Flow of Events	1. The user clicks on “Preferences” button from the menu of the application. 2. The user decides in which area to update his preferences and constraints (Mobility Option, Sharing Services, Taxi Services, Own transports, Public Transports). 3. The user modifies his preferences and costraints in all needed areas. 4. The user ends his modification exiting the settings menu. 5. T+ updates BMO if possible, otherwise it computes an AMO and notifies the user.
Exit Condition	The user updates all his preferences and T+ makes sure that a valid BMO can be found again
Exceptions	T+ is unable to find a BMO satisfying the new preferences, a warning is produced.

- **Insert position of own car and bike**

Name	Own transport
Actors	Identified user
Related Goals	G2.1
Entry Condition	The user dosn't have GPSG and want to insert the position of his own car/bike.
Flow of Events	1. The user clicks on the preferencies button from the navigation interface of the application. 2. The user clicks on 'Own transport'. 3. The user inputs the position of his own car/bike. 4. T+ updates the current BMO.
Exit Condition	The user inputs the new position of his own car/bike.
Exceptions	The user entered an invalid address.


interaction : Insert position of own car/bike
Identified User
T+ Mobile
T+ Server
1 : request (change preferences)
2 : show(preferences main page)
3 : request (change preferences)
4: show(position_form)
5 : fill(input form)
6 : click_on(ok button)
7 : request(meteo,travel means,.. information)
8: provide (requested_infomations)
9: BMO update
10: show(confirmation message)

Insert position on own car/bike sequence diagram

- **Declare pass**

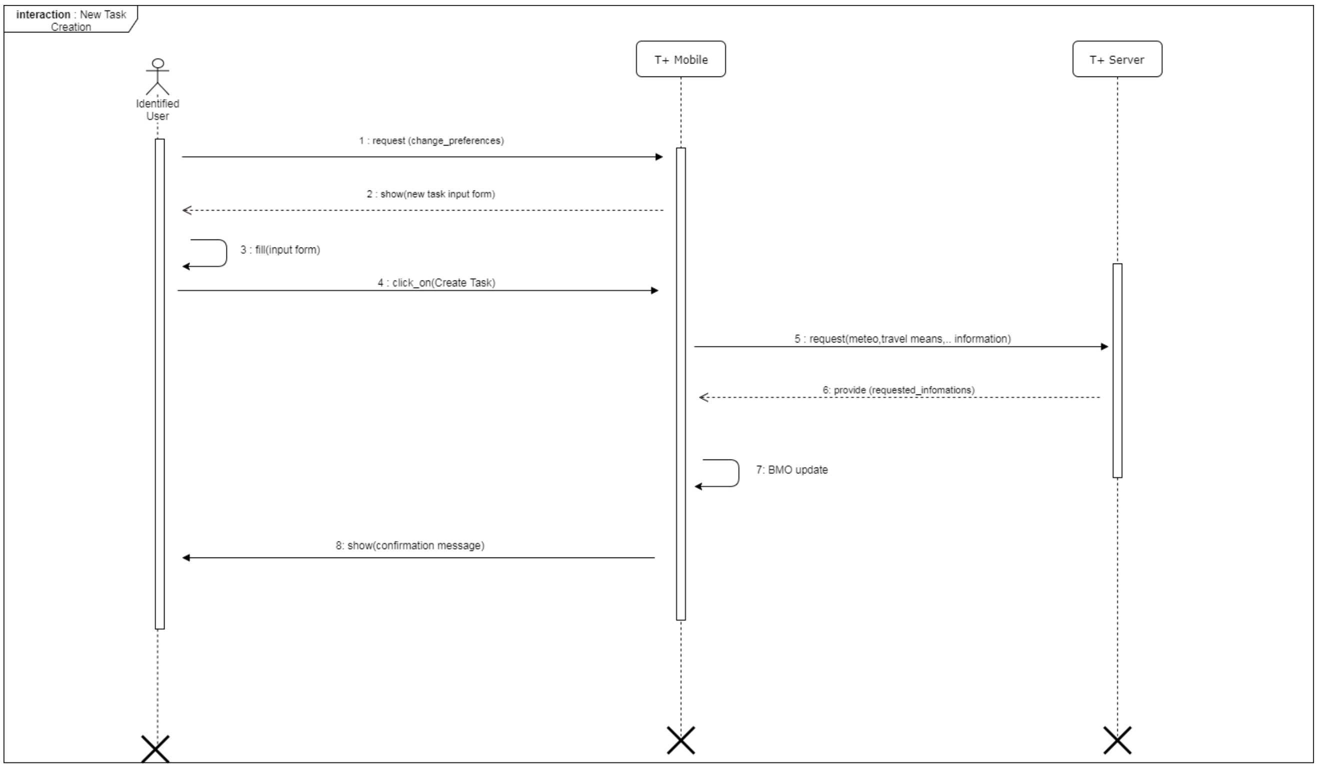
Name	Declare pass
Actors	Identified user
Related Goals	G5.1
Entry Condition	The user want to declare a week/monthly/season pass.
Flow of Events	1. The user clicks on the 'My passes' button from the navigation interface of the application. 2. The user clicks on 'Add pass'. 3. The user selects how he wants to add the pass he owns across the following options: a. Scan a QR Code on the pass. b. Scan a Bar Code on the pass. c. Scan the entire pass if it is card-sized. d. Enter the pass details manually. 4. The user clicks on 'Add pass' to finalize the procedure. 5. The pass is added to the owned passes of the user. 6. T+ updates the current BMO.
Exit Condition	The user inputs the features of the pass.
Exceptions	The user entered an invalid pass.

- **Buy pass**

Name	Buy pass
Actors	Identified user
Related Goals	G5.2
Entry Condition	The user wants to buy a pass.
Flow of Events	1. The user clicks on the 'My passes' button from the navigation interface of the application. 2. The user clicks on 'Buy pass'. 3. The user selects which type of pass he wants to buy from the various options: a. Public transportation passes. b. Airplane passes. c. Sharing services passes. 4. The then is prompted with additional options based on the previous selection (such as kind of public transport for which to buy the pass) 5. The user is prompted with the required details and pass options. 6. When ready the user clicks on 'Buy'. 7. The pass is added to the owned passes of the user. 8. T+ updates the current BMO.
Exit Condition	The user buy the pass
Exceptions	The user doesn't have a valid payment method

- **New task**

Name	New task
Actors	Identified User, External Sevices
Related Goals	G1
Entry Condition	The user must be logged in. The user wants to create a new task
Flow of Events	1. The user clicks on the "New task" button from the menu of the application. 2. The user inputs the required details for a new task (Title, Description, Location, Time span). 3. The user selects a color to assign the task to a specific category. 4. The user clicks on "Create task" to proceed. 5. External services provide all the necessary informations on the various means of transport. 6. T+ creates a new task 7. If the new task is in the TC range, T+ updates the current BMO.
Exit Condition	The data provided by the user is correct and the task is added to the user schedule. A success notification is showed.
Exceptions	• The information provided are badly formatted. In this case an error notification is raised and the user is asked to repeat the procedure. • The new task is incompatible with the other ones (such that a new BMO cannot be produced). In this case a warning is produced, and the new BMO ignores the new task starting time.


interaction : New Task Creation
Identified User
T+ Mobile
T+ Server
1 : request (change_preferences)
2 : show(new task input form)
3 : fill(input form)
4 : click_on(Create Task)
5 : request(meteo,travel means,.. information)
6: provide (requested_infomations)
7: BMO update
8: show(confirmation message)

New task sequence diagram

- **View schedule**

Name	My schedule
Actors	Identified user
Related Goals	G1
Entry Condition	The user want to see his schedule.
Flow of Events	1. The user clicks on the “my schedule” button from the menu of the application. 2. The user selects the month for which wants to see his schedule. 3. The user clicks on a particular day of the calendar to list the task for that day.
Exit Condition	The user is able to see all the tasks that were previously scheduled on that month/day.
Exceptions	• There are no scheduled tasks that month. • There are no scheduled tasks on the selected day.

- **View BMO Map**

Name	View map
Actors	Identified user
Related Goals	G1.4
Entry Condition	The user want to see the map of his journey.
Flow of Events	1. The user clicks on the ‘View BMO map ‘button from the navigation interface of the application.

- **Modify Task**

Name	Modify task
Actors	Identified user, External Services
Related Goals	G1
Entry Condition	The user wants to modify a task.
Flow of Events	1. The user clicks on the “my schedules” button from the menu of the application. 2. The user selects the month which contains the the task that he wants to modify. 3. The user clicks on the day where the task was scheduled. 4. The user clicks on the “Edit” button next to the task. 5. The user changes the desired fields throughout the dedicated UI. 6. External services provide all the necessary information on the various means of transport. 7. If the modified task is in the TC range, T+ updates the current BMO.
Exit Condition	The updated information about the task are correct. In this case a success notification is showed.
Exceptions	• The information provided are badly formatted. In this case an error notification is raised and the user is asked to repeat the procedure. • The new task is incompatible with the other ones (such that a new BMO cannot be produced). In this case a warning is produced, and the new BMO ignores the new task starting time.

- **Cancel Task**

Name	Delete task
Actors	Identified user, External Services
Related Goals	G1
Entry Condition	The user wants to modify a task.
Flow of Events	1. The user clicks on the “my schedule” button from the menu of the application. 2. The user selects the month which contains the the task that wants to modify. 3. The user clicks on the day where the task was scheduled. 4. The user clicks on the “Edit” button next to the task. 5. The user clicks on the “Delete task” button. 6. External services provide all the necessary informations on the various means of transport. 7. T+ updates the current BMO.
Exit Condition	T+ completely deletes the task and updates the database onto the server. A success notification is presented and the user is redirected to the main interface.
Exceptions	T+ is not able to delete the task. An error notification is presented.

- **Manage Account**

Name	My profile
Actors	Identified user
Related Goals	G4
Entry Condition	The user wants to edit his profile.
Flow of Events	1. The user clicks on the “My profile” button from the menu of the application. 2. The user changes his details from the dedicated UI.
Exit Condition	The user entered valid information (correct email address, valid phone number, unique username etc.). In this case a success notification is shown.
Exceptions	The user entered badly formatted information. An error notification is raised.

Goals and Requirements Coverage

Goals	Requirements	Use Case	Covered ?
G1	R1,R7	Cancel task, Modify task, View schedule, New task, View BMO Map, View Schedule	✓
G2	R2, R3, R4	Insert position of own car/bike, Change preferencies	✓
G3	R4	Change preferencies	✓
G4	R4, R6	Manage account	✓
G5	R6	Login, Registration	✓

2.3. *User characteristics*

Our Users are people living in a city that have a number of scheduled tasks to accomplish. These tasks may be meetings, lunch, breaks, reserving time to study, or anything else. Tasks may or may not require the User to be in a certain place at the given time, and may be either extraordinary or periodic. They are not required to be fixed in time, they might have a timespan in which they must begin. Tasks may change in every moment and Users want a possible solution in 10 - 15 seconds.

Users have different preferences about travel solutions, ranging from particular needs (constraints) to their best cost-time tradeoff.

Users taking advantage of our services need to find a good mobility option between their appointments that satisfies their personal needs and maximizes their preference options.

Users may have tickets/passes bought outside our application that might influence the cost they are actually charged for certain means of transportation, and they need to take it into account in their travel solutions.

Users do not need a travel solution going over the defined Time Coverage in the future.

Users can be mainly divided into two categories:
- Type A Users (Enterpreneur-like Users): these users highly rely on T+ to plan their daily movements across one or more cities. They have some routine periodical tasks and some exceptional tasks to schedule. They need T+ to be highly reliable, available and fast, to be able to submit and adapt the plan for a new appointment at the last minute. Offline calculation is highly appreciated, and an easy interface is essential as they do not want to spend too much time for setting up the tasks in the app. On the other side, efficiency of the proposed solution is not strictly essential as long as they are guaranteed not to be late.
- Type B Users (Tourist-like Users): these users occasionally use T+ as a tool to optimize movements among a great variety of travel options. They have some fixed tasks in a city they usually don't know, and they rely on T+ to explore their possibilities to find the best (usually the cheapest) way to reach the city and to move around. Good coverage and the optimality of the solution are of major importance to this type of user, while computational time and high availability of online services are not important. They appreciate a good depth on constraints and preferences specification to make T+ find the solution that really fits them most.

2.4. *Assumptions, dependencies and constraints*

We will make the following domain assumptions:

[D1] location and prices of transport services is well known
  [D1.1] the provided location of all available cars/bikes of sharing systems is correct (with an error tolerance of 10 meters)
  [D1.2] public transportations have well-defined timetables, stops and prices. All these features change only with advanced notice.

[D2] any strikes, delays or incidents are reported

[D3] the user's mobile phone is equipped with a GPS and his location is correct (with an error tolerance of 10 meters)

[D4] the user and his mobile phone always share the same position.

[D5] the user's mobile phone has always access to an Internet connection.

[D6] the user never misses sent notifications.

[D7] when a new notification arrives the user sees it in a short time.

[D8] if at least a personal car is submitted, the user either communicates the correct position of his personal car or has a connected GPS on it.

[D9] every city have different rules about transportation and these rules may change with advanced notice.

[D10] for every user the frequency of external event notifications that need the recomputation of the current BMO is less than 2 every minute on daily avarage.

2.5. *Future possible implementation*

- When the user has similar appointments or repeated trips in a given period the system could suggest a week/monthly/season pass for some means of trasportation.
- When the user schedules an appointment in a very busy period while other times are much lighter the application could suggest another possible date to fix it in case this is possible.
- The user could ask the app what is the best time to schedule a specific appointment.
- The user might be able to create a meeting with other users that use T+ : He would be able to enter the email address/cellphone number of his colleagues and send them an invite.

3. Specific requirements

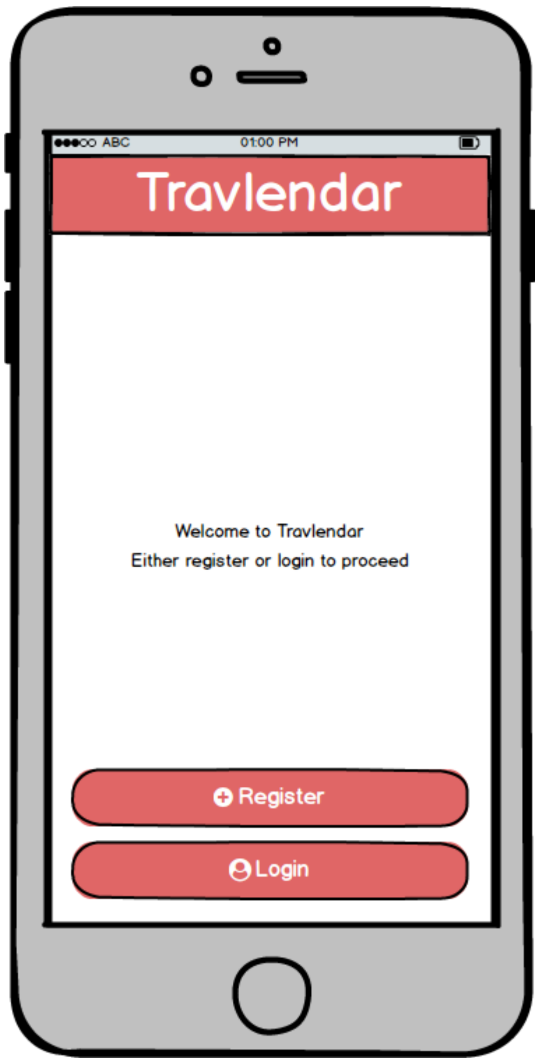
External Interface Requirements

User Interfaces

We want to provide the user a simple interface that allows him to schedule his meetings and reach them in a very transparent way without being worried of distances or travel times.

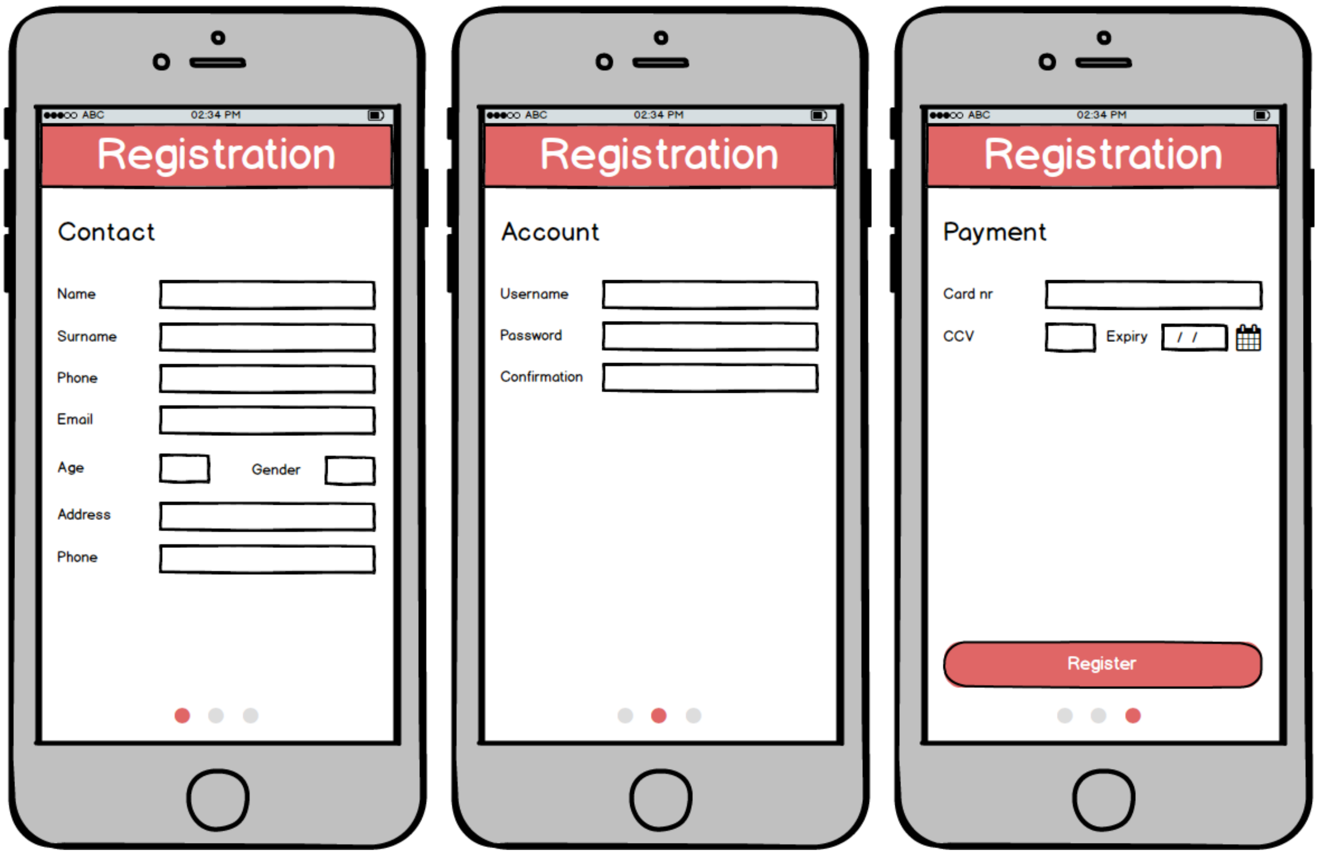
Mobile UIs

- **Welcome Interface:** This will be the interface the user will be presented with as soon as he launches the application.



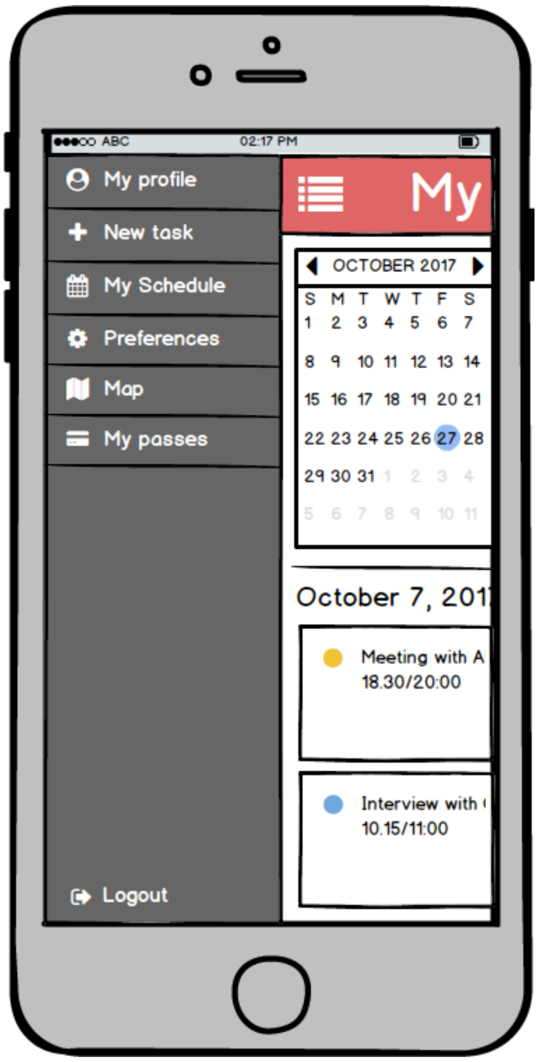
Welcome interface

- **Registration interface:** here the user will be able to fill in all the requested information to proceed in using the app.



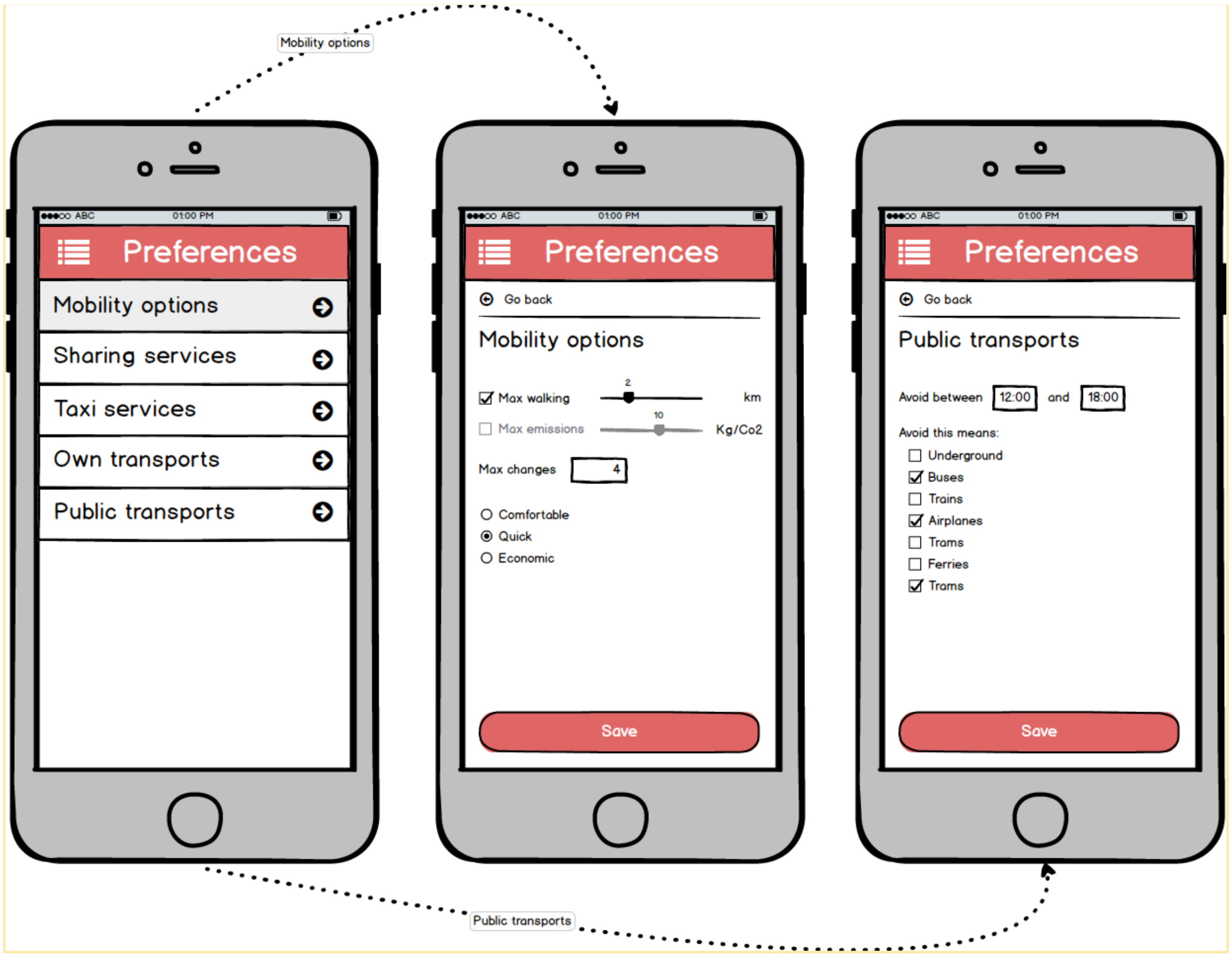
Registration procedure

- **Navigation interface:** to navigate within the app a simple UI will allow the user to select which action should be performed.



Navigation interface

- **Preferences interface:** this section allows the user to specify his own preferences. It consists of a main list of selectable entities and each of them, upon selection, will permit to toggle all the details relative to it. It is worth saying that there are two main differences across the preferences:
  - Mobility options: This section accounts for transportation mean independent options such as maximum desired carbon footprint or walking distances.
  - Other preferences: These entries focus on a single transportation mean / service.



Mobility options

☑ Max walking 2 km

☐ Max emissions 10 Kg/Co2

Max changes 4

○ Comfortable
● Quick
○ Economic

Public transports

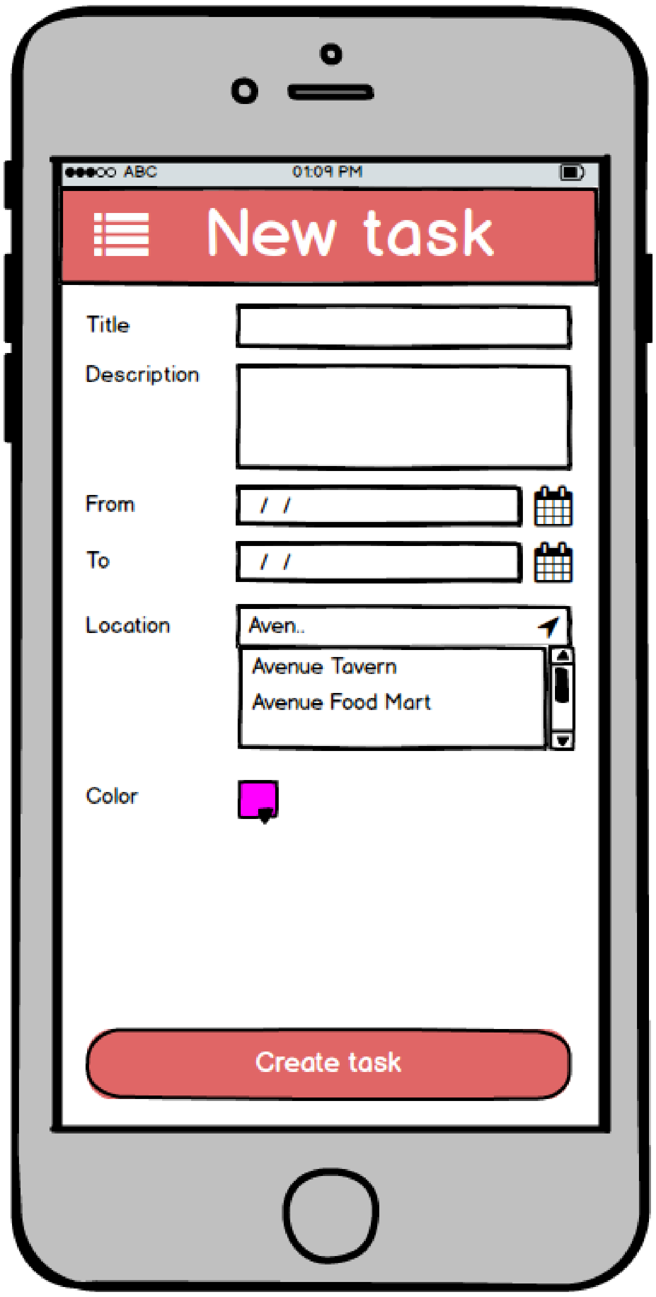
Avoid between 12:00 and 18:00

Avoid this means:

☐ Underground
☑ Buses
☐ Trains
☑ Airplanes
☐ Trams
☐ Ferries
☑ Trams

Preference UIs flow

- **Task interface:** there are three main UI for the tasks. The first one provides an input form to gather details about a new task, the second one displays a calendar with the possibility to see day by day all the scheduled tasks and finally a third one that renders onto a map the BMO, allowing the user to buy tickets for specific travel mean.


●●●○○ ABC 01:09 PM

New task

Title:

Description:

From: / /

To: / /

Location: Aven..

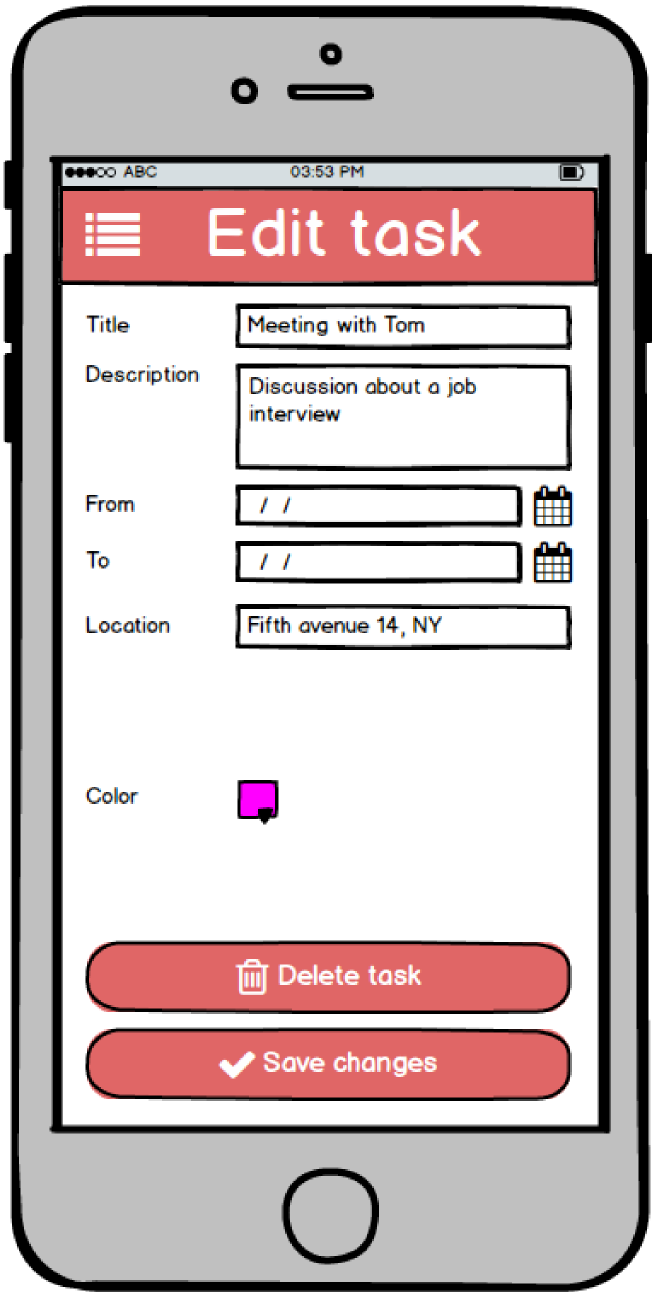
Avenue Tavern

Avenue Food Mart

Color:

Create task

New task UI


●●●○○ ABC 03:53 PM

Edit task

Title: Meeting with Tom

Description: Discussion about a job interview

From: / /

To: / /

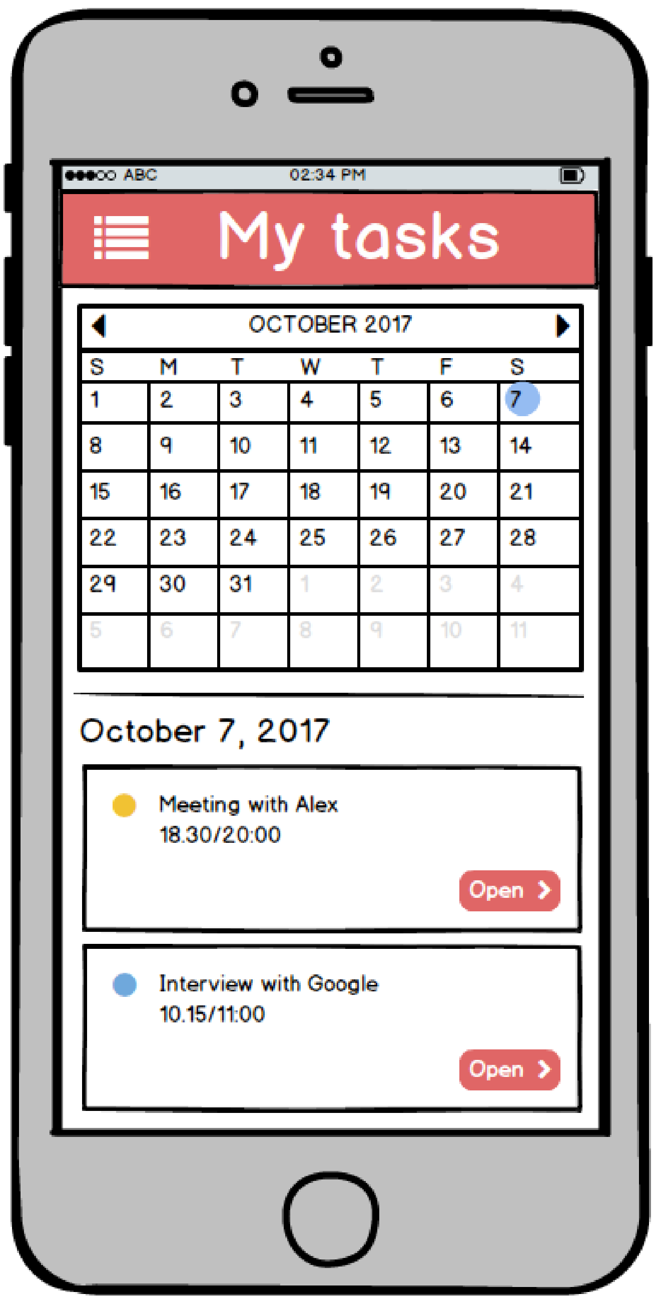
Location: Fifth avenue 14, NY

Color:

Delete task

Save changes

Edit task UI


●●●○○ ABC 02:34 PM

My tasks

OCTOBER 2017

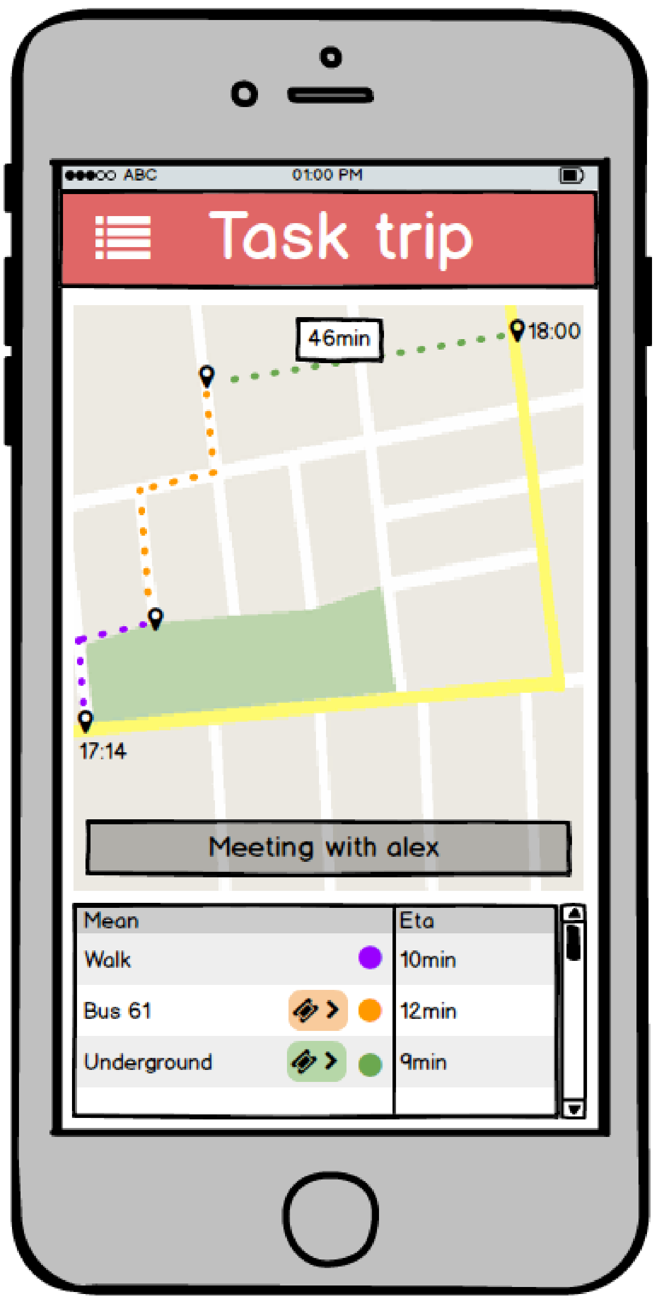
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

October 7, 2017

● Meeting with Alex
18.30/20:00
Open >

● Interview with Google
10.15/11:00
Open >

My schedule UI


●●●○○ ABC 01:00 PM

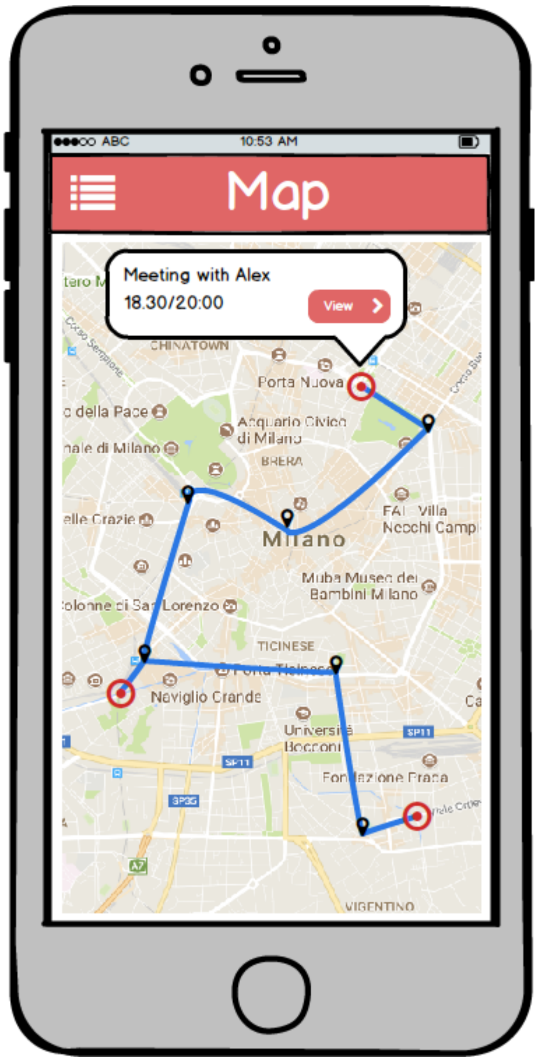
Task trip

Meeting with alex

Mean		Eta
Walk	●	10min
Bus 61	●	12min
Underground	●	9min

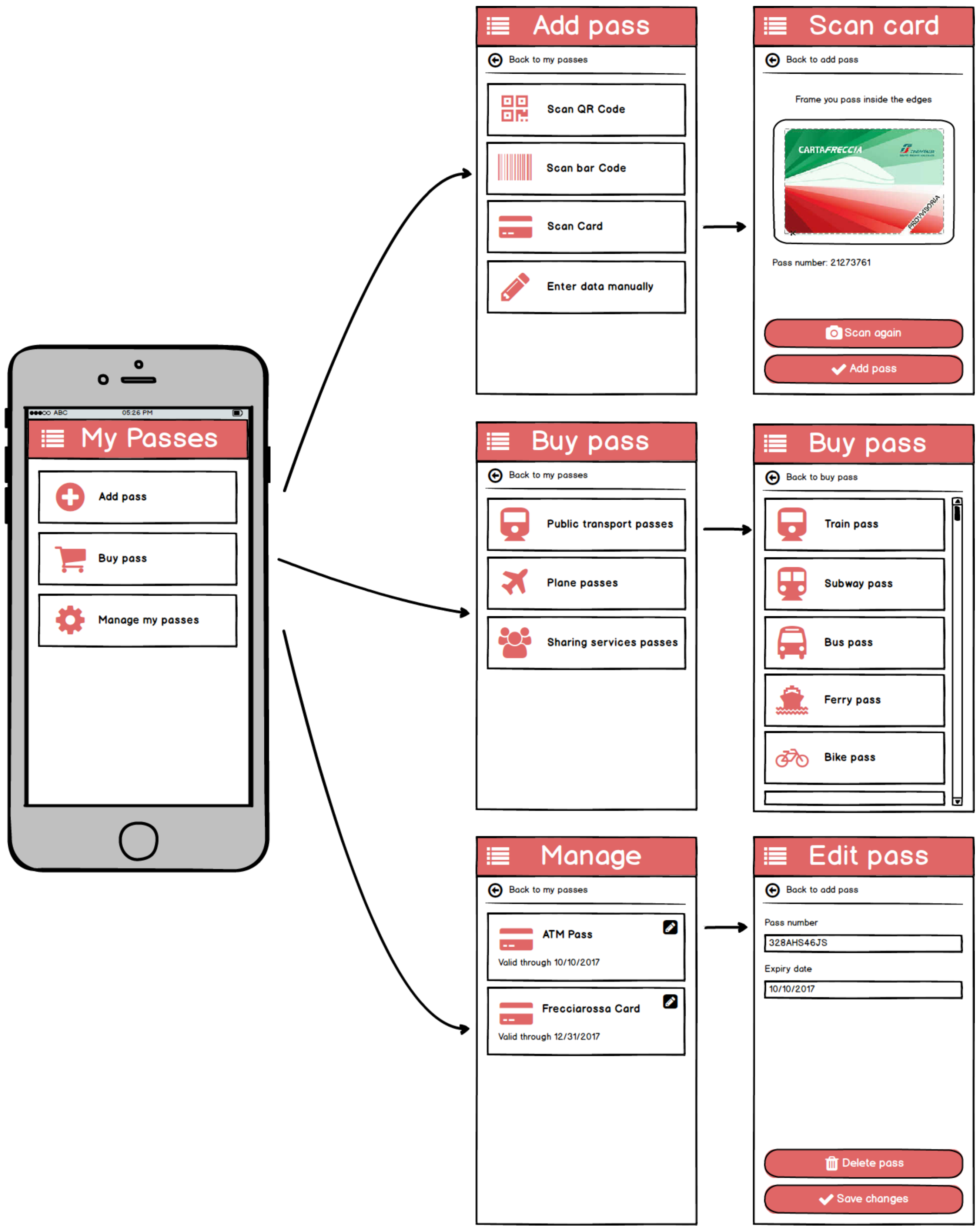
Task trip UI

- **BMO Map interface:** This interface provides a global overview of the BMO computed for the tasks in the time coverage. The red markers indicate a task: upon click a popup is shown with some brief details about the task; a button allows to see the **task trip** with more accuracy. The black markers indicate the points in which the user will have to switch transportation mean accordingly to the BMO. The blue path is a graphical representation of the course that the user is going to follow.



BMO Map UI

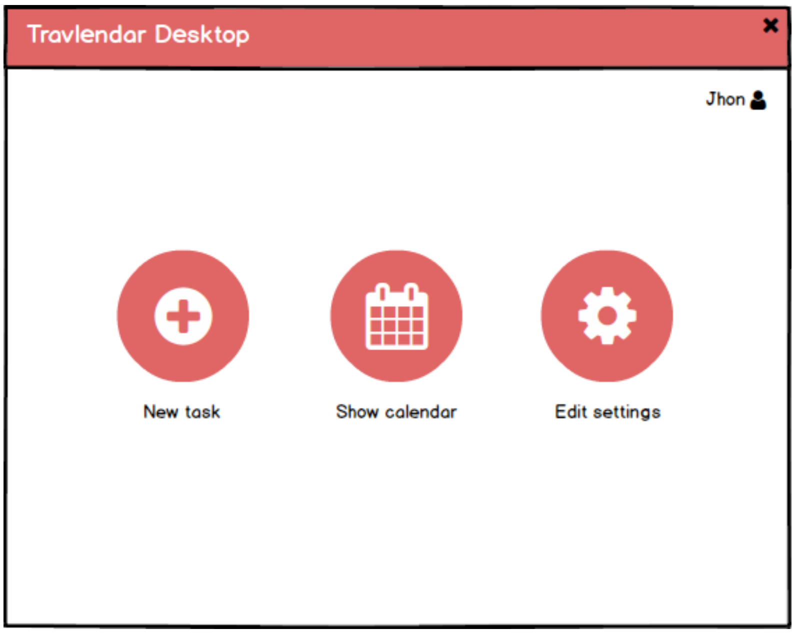
- **My passes interface:** Here the user can manage all his passes. There are three main sections which allow to manage and add a pass either via a purchase procedure or camera scan. For the buying procedure the user is redirected to the ticked provider website which is loaded within the application. T+ Mobile accounts for automatically inputting all the data already provided by the user in the preferences pane such as profile details as well as payment information.



My passes UIs flow

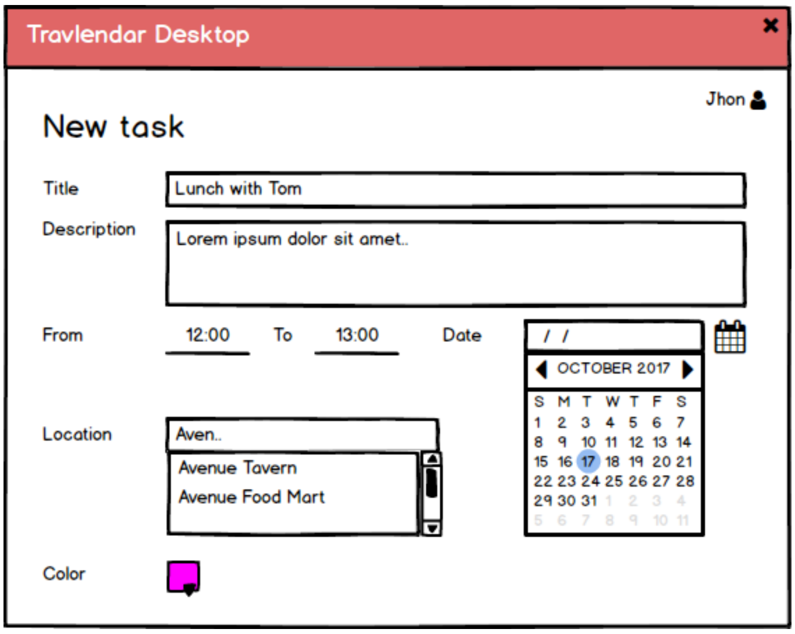
Desktop UIs

- **Main interface**

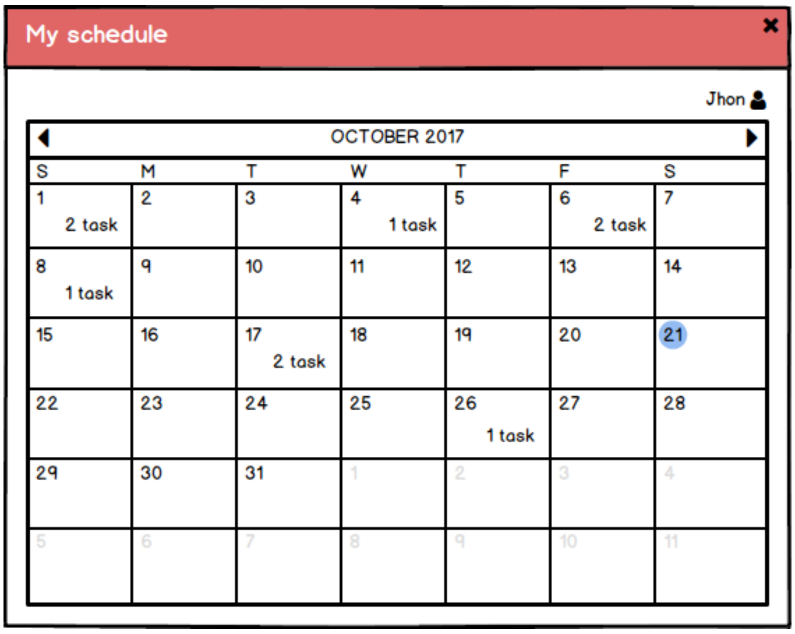


Main desktop interface

- **New task interface**



- **My schedule interface**



My schedule

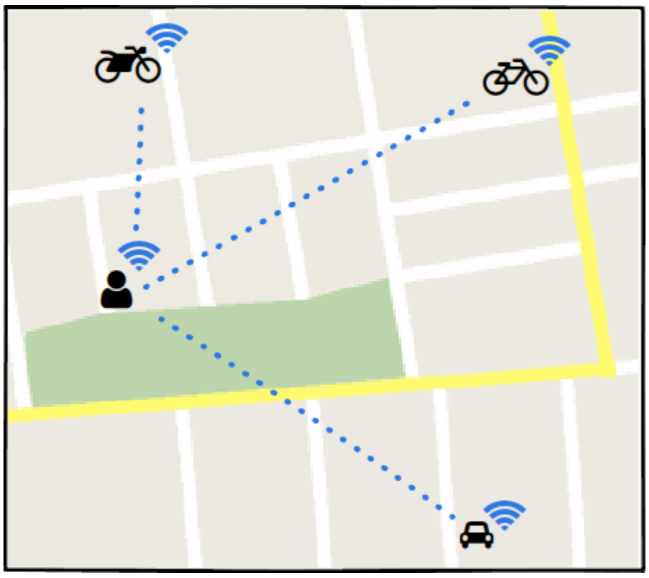
Jhon

OCTOBER 2017

S	M	T	W	T	F	S
1 2 task	2	3	4 1 task	5	6 2 task	7
8 1 task	9	10	11	12	13	14
15	16	17 2 task	18	19	20	21
22	23	24	25	26 1 task	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

Hardware Interfaces

- **GPSG (GPS Gadget) :** A small and embedded device that could be left in the car of the user to provide its remote location to T+. The same device could even be attached to his bike or other personal travel means. This will allow T+ to compute a more accurate BMO among the personal travel means.
  The GPSG system targets users that frequently use personal means and that have the need to know exactly the position of their car, bike, motorbike or whatever else they attach it to without having to update it manually every time it is moved.
  Every GPSG item will be provided with a unique ID (printed onto the item itself), used to pair its location with any personal linkable mean into the application, and with a password (not printed onto the item) to protect this information.
  The server will reject pairing the same item with different entities, to recycle a GPSG item the owner should unpair it before.



Software Interfaces

T+ exploits the following APIs :

- **Geolocation API** : in order to know the location of the user we will use some APIs, for instance Google Location Services.

- **Public Transportation Services APIs** : in order to know timetables, prices and stops of public transportation we will use some APIs which are made available by the suppliers of these services. For instance with ViaggiaTreno API we can gather all the information about FS trains.

- **Car/Bike Sharing Services APIs:** in order to know the position of the cars/bikes closest to the user we will use some APIs. For instance car2go API offers access to up-to-date car2go information such as vehicles available, parking spots and gas stations.

- **Maps APIs** : in order to access to all geographical data such as maps, locations, distance computation we will use Google Maps API.

- **Weather APIs** : in order to access to all meteo forecast data we will use some weather APIs (for instance OpenWeatherMap API).

Communication Interfaces

- **Internet :** In order to interact in the right way, the server, as well as the mobile and desktop applications, will make use of the TCP/IP stack. This will grant no data loss and good performance for sending messages across the various applications over the network. Furthermore, in order to send receipts via email, protocols such as POP3 and IMAP will be used.

Functional Requirements

Performance Requirements

Our users may need to change their schedule at the last minute. For this reason the BMO computation for local movements should take no more than 10 seconds (see R5.4).

Software System Attributes

Reliability
The reliability of the core service is granted by the fat-client architecture: the possibility to submit new tasks and compute BMOs inside the home city is fully autonomous on the client application, and thus this main service is granted even in the case of server failure. Secondary services such as device synchronization and transportation data updates will count on a redundant architecture to grant at least a 99.9% reliability during the uptime. We will cache latest data to ensure to provide support to specific means of transportation even when their related external services are down.

Availability
We assume that the load of our server based services (device sinchronization and transportation data updates) will vary on a daily and weekly basis: it will be much lower during night and weekends. For this reason we reserve a server downtime of two hours weekly on Sundays from 2:30 to 4:30 a.m. for server maintenance.

Security
T+ has to manage delicate information such as payment card data and account information. The latter will be encrypted on our servers and will never be sent back to grant privacy: it will only be used for device synchronization. Payment card information will be kept local on the device where it is submitted (and thus it will not be supported by device synchronization) to make sure that even our server system never receives this information.

Maintainability
A map-based application such as T+ is critically depentent on maintainability, with particular attention to extensibility. T+ wants to expand its coverage constantly, making extensibility of the software system a crucial aspect in order to easily connect to more and more transportation systems in the time. Furthermore, along with ordinary maintenance, we will often need to update the data of any transportation system that changed its rules.

Portability
T+ targets all main portable operating systems: the software will be available in iOS, Android and Windows Phone platforms.

4. Formal analysis using Alloy

Alloy modelization has been used to highlight possible design problems and the importance of certain domain assumptions towards the goal realization. The model represents the evolution of a single mobile application over time, which is subject to settings preference changes, notifications from T+ servers and new tasks submission. It describes the state of the application and its ability to interact with the user in each considered time instant, as well as the travel solution that has to be found.
Some important considerations can be derived from this model:

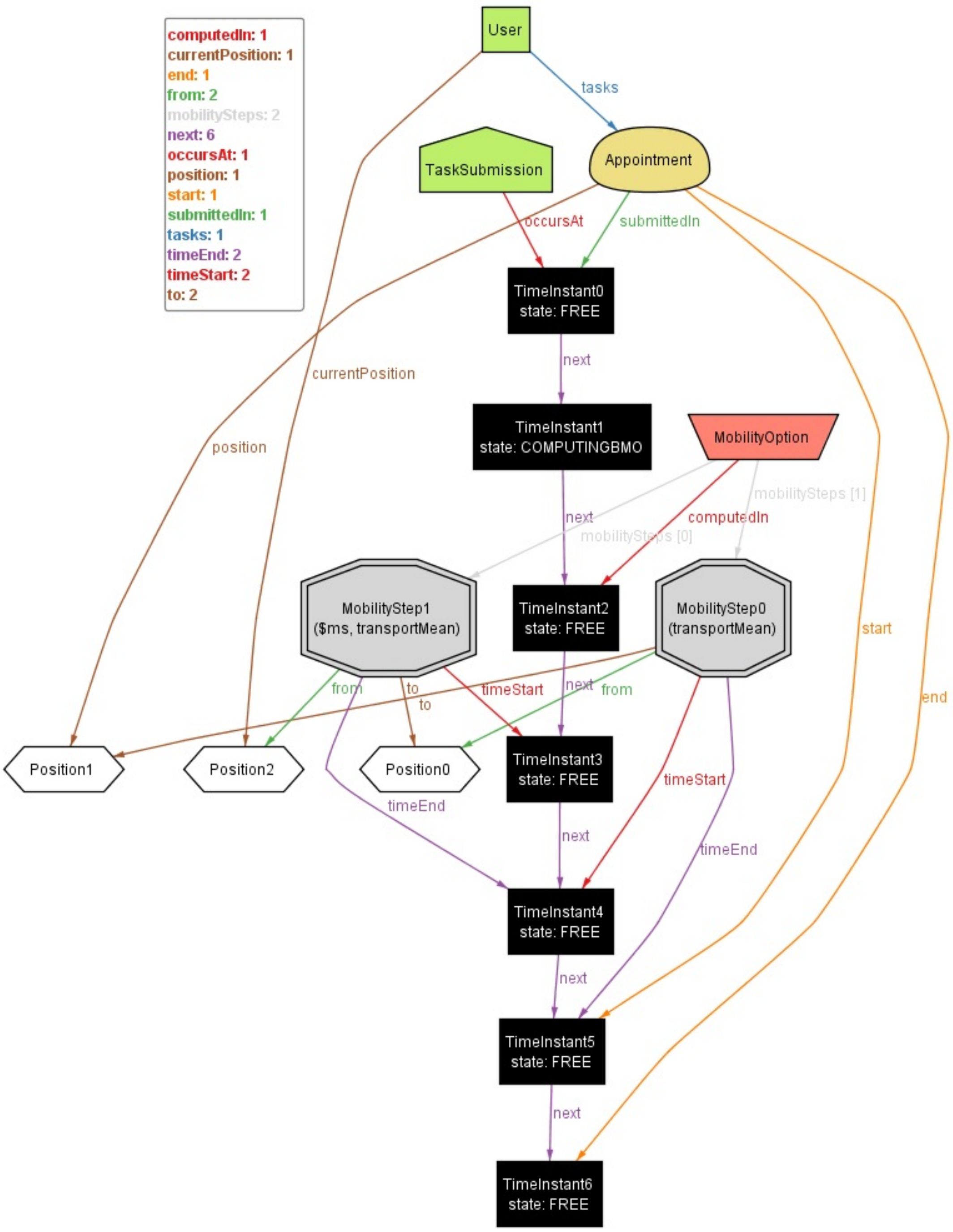
1. Domain assumption D10 about the frequency of relevant external notifications is vital to grant goals G1, G2 and G4: cutting down the fact corresponding to D10 Alloy can find counterexamples to the assertion representing G1, G2 and G4 in which the application is overwhelmed by server notification and is forced to constantly update the proposed solution even before being able to display it. This problem highlights the importance of caching frequently evolving data (such as traffic information) and forwarding them very selectively to the end-user applications.
2. Even with D10 enabled, running the model with a sufficient time range (TimeInstant cardinality more than 12-15) shows instances in which the mobility solution is frequently changed to adapt to events (either user-made or server notifications). An unstable solution may result in the user perceiving the application as unreliable and confusing, even if it is for the sake of saving a little more time. To overcome this issue T+ mobile application must be conservative on the proposed solution and change it only if it becomes unfeasible.

```
// Time progression and ordering
sig TimeInstant {
        before: set TimeInstant,
        state: one APPSTATE,
        previous: lone TimeInstant,
        next: lone TimeInstant
}

// Complete ordering definition
fact BeforeOrdering {
        all disj t1, t2: TimeInstant | t2 in before[t1] iff ! t1 in before[t2]
        all disj t1, t2, t3: TimeInstant | t2 in before[t1] && t3 in before[t2] implies t3 in before[t1]
        all t: TimeInstant | ! t in before[t]
}

// Handy previous definition
fact previousDefinition {
        all disj t1, t2: TimeInstant | t2 in previous[t1] iff t2.before - t1.before = t1
        all t: TimeInstant | ! t in previous[t]
        next = ~previous
}

// Any event that can occur to the application
abstract sig Event{
        occursAt: one TimeInstant
}
// User-caused events
abstract sig UserInteraction extends Event {}
{
        occursAt.state = FREE
}
sig TaskSubmission extends UserInteraction{}
{
        some t: Task | t.submittedIn = occursAt
}
sig SettingsChange extends UserInteraction{}

// External server notifications
sig ExternalServiceEvent extends Event {}


fact ReduceModelComplexity {
        // Forbid useless submissions: multiple tasks can be generated by a single task submission entity
        all disj ts1, ts2: TaskSubmission | ts2.occursAt != ts1.occursAt
        // Forbid useless settings change: multiple settings can be changed by a single settings change entity
        all disj sc1, sc2: SettingsChange | sc2.occursAt != sc1.occursAt
        // Forbid useless service events in the model: one single service entity is equivalent to multiple events at the same time
        all disj e1, e2 : ExternalServiceEvent | e2.occursAt != e1.occursAt
        // Forbid useless positions
        no pos: Position | NotSignificantPosition[pos]
        // Forbid unconcluded models [do not use when checking UserCanAlwaysInteract assertion, useful to generate cleaner models]
    all t : TimeInstant | t.next = none implies ( t.state = FREE and !eventHappened[t])
}

// States in which the app could be
abstract sig APPSTATE{}
one sig COMPUTINGBMO extends APPSTATE{}
one sig FREE extends APPSTATE{}

// Make computational times coherent to be intervals with a maximum length
fact ComputingRule {
        all t: TimeInstant |                (eventHappened[t.previous] implies t.state = COMPUTINGBMO) &&
                                                                                ( t.state = COMPUTINGBMO implies
                                                                                (no t.previous or eventHappened[t.previous.previous] or eventHappened[t.previous]))
}
```

```

// Handy shortcut to check if any event has happened in a given time istant
pred eventHappened(t: TimeInstant){
        t != none and some e: Event | e->t in occursAt
}

// Events cannot happen too frequently (from domain assumption D10)
fact SparseExternalEvents {
        #(ExternalServiceEvent->(1+2+3)) < #TimeInstant
}

// Necessary condition for goals G1, G2, G4
assert UserCanAlwaysInteract{
        some t: TimeInstant | t.state = FREE
}

/*position definition*/
sig Position{
}

pred NotSignificantPosition (p: Position){
        (no ms: MobilityStep | ms.from = p or ms.to = p)
        and (no apt: Appointment | apt.position = p)
        and (User.currentPosition != p)
}

/*The user of the app and its data*/
one sig User {
        tasks : set Task,
        currentPosition : one Position
}

/*appointment definition*/
sig Appointment extends Task{
        position: one Position,
}

/*task definition*/
sig Task{
        start: one TimeInstant,
                end: one TimeInstant,
                submittedIn: one TimeInstant
}{
        start->end in before
        submittedIn->start in before
        this in User.tasks
        some ts: TaskSubmission | ts.occursAt = submittedIn
}

fact noContemporaryTasks {
        all disj t1, t2 : Task | (t1.end -> t2.start in before or t2.end -> t1.start in before)
}

/*mobility step definition*/
sig MobilityStep{
        from : one Position,
        to : one Position,
        timeStart : one TimeInstant,
        timeEnd : one TimeInstant,
}
{
        /* no useless mobility steps */
        some mo : MobilityOption | this in mo.mobilitySteps.elems
        /*start and end cannot be the same place */
        to != from
        /*end must happen after starting*/
        timeStart->timeEnd in before
}

pred consecutiveSteps(ms1, ms2: MobilityStep){
        ms2.from = ms1.to
        and         (ms1.timeEnd = ms2.timeStart
                        or (some apt : Appointment |( apt.position = ms1.to and apt.start = ms1.timeEnd and apt.end = ms2.timeStart))
                        or (some t: Task | ( t not in Appointment and t.start = ms1.timeEnd and t.end = ms2.timeStart))
                        )
}

pred coversAppointment(ms: MobilityStep, apt: Appointment){
        ms.to = apt.position and ms.timeEnd = apt.start
}

/*mobility option definition*/
sig MobilityOption{
        /*seq is an ordered set*/
        mobilitySteps : seq MobilityStep,
                computedIn: one TimeInstant
}
{
/* there is at least one mobility step in a mobility option*/
        #mobilitySteps >= 1
/* all steps in a mobility option are consecutive */
                all i: Int | i in mobilitySteps.butlast.inds implies consecutiveSteps[mobilitySteps[i], mobilitySteps[add[i, 1]]]
/* all steps in a mobility option are scheduled after computation */
                computedIn -> mobilitySteps.elems.timeStart in before
/* no double elements */
                #mobilitySteps = #mobilitySteps.elems
/*starts where user currently is located */
                mobilitySteps[0].from = User.currentPosition
/* a mobility option covers all appointments submitted before its computation */
                all apt : Appointment | apt.submittedIn-> computedIn in before implies one ms : MobilityStep | ms in mobilitySteps.elems and coversAppointment[ms, apt]
/* the mobility option ends in a valid appointment */
                some apt: Appointment | apt.submittedIn -> computedIn in before and coversAppointment[mobilitySteps.last, apt]
/* no task can be done during a mobility step */
                all ms: mobilitySteps.elems | no t: Task | t.submittedIn->computedIn in before and uncompatibleTask[t, ms]
}

// A task is not compatible with a mobility step
pred uncompatibleTask(t: Task, ms: MobilityStep){
        (t.start->ms.timeEnd in before and ms.timeStart->t.start in (before+iden))
        or (t.end->ms.timeEnd in (before+iden) and ms.timeStart->t.end in before)
        or (t.start->ms.timeStart in before and ms.timeEnd -> t.end in before)
}

// Mobility options are computed only as a result of a ComputingBMO phase
fact MobilityOptionComputationTimes {
        all disj mo1, mo2: MobilityOption | mo1.computedIn != mo2.computedIn
        all t: TimeInstant | t.previous = none or ((t.state = FREE and t.previous.state = COMPUTINGBMO) iff one mo: MobilityOption | mo.computedIn = t)
}
```

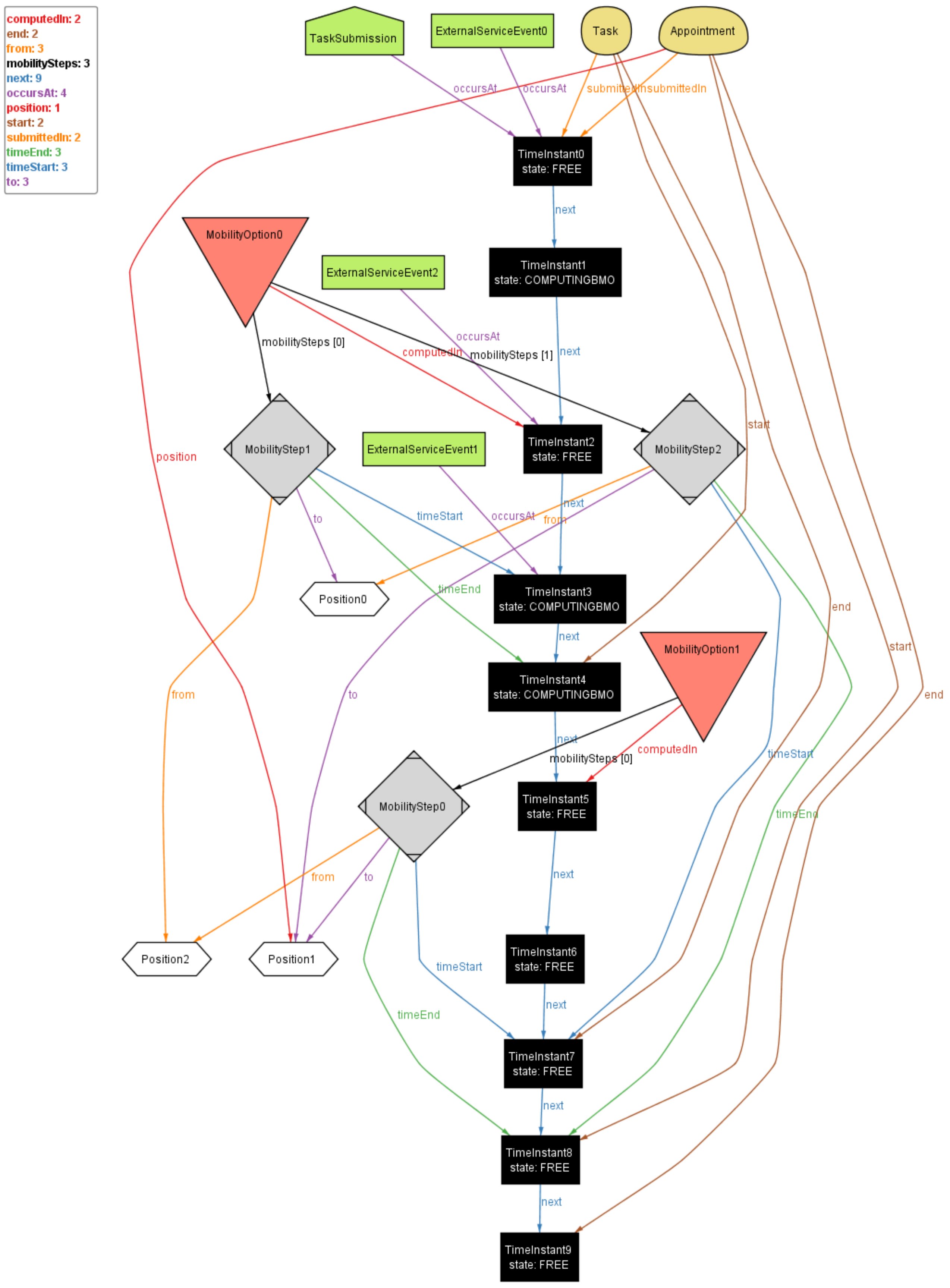
World instance model for:

```
run {} for 5 but exactly 7 TimeInstant, exactly 1 Task
```



World instance model for:

```
run {} for 5 but exactly 10 TimeInstant, exactly 2 Task
```



UserCanAlwaysInteract check:

```
check UserCanAlwaysInteract for 5 but exactly 10 TimeInstant
```

```
Executing "Check UserCanAlwaysInteract for 5 but exactly 10 TimeInstant"
   Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
   48466 vars. 1090 primary vars. 101918 clauses. 718ms.
   No counterexample found. Assertion may be valid. 12230ms.
```

5. Effort Spent

Name	Time spent [hours]
Barcella Lorenzo	21.5
Bellini Alberto	18.75
Cavalli Luca	23.5