Requirements Analysis and Specifications Document

Politecnico di Milano

- Antonino Caminiti (mat. 835724)
- Daniele Gonella (mat. 827091)
- Matteo Scerbo (mat. 899823)

1. Introduction

1.1. Purpose

This document represents the R.A.S.D. (Requirement Analysis and Specific Document). Our goal is to give a detailed and complete description of the system in terms of requirements (both functional and non-functional) and an analysis of the customer's needs to model the system, detail the constraints and the limit of the software, and show a rundown of the typical use cases that will occur after the release.
This document is directed to the developers who are going to implement the requirements and will also work as a contractual basis.

1.2. Scope

1.2.1. Description of the given Problem

We are going to design and implement a web application named "TravLander+".
The System will allow the users to

- Find the shortest available itinerary given the location of departure and destination, with possibility of modification in case of unforeseen circumstances
- Further customize said itinerary by stating their preferences of transport and desired pauses or breaks, customizing them by specifying how much the break should last and the given timeslot
- Buy and/or book tickets for public transports

The users will have to register (by inserting a username and a password) to be able to use the system. Every user has a set of travel preferences, which can be customized in general as well as for each itinerary, if necessary.
The main purpose will be to offer a quick, efficient, and reliable application to schedule the quickest routes complying to all the user's events within the limits of feasibility.

1.2.2. Actual System

The users will be logging in with their username (or, alternatively, their e-mail address or phone number) and password. The set of preferences will be the basis for all travels, and the users will only need to give as inputs the locations of departure and the destination, the time at which they should arrive to the destination, and the type of event.
After that, they will also be able to customize the preferences for the specific trip (such as some means of transport to avoid) and they will receive the shortest itinerary given these inputs.
The user will also have the further option to arrange the trip by buying/booking tickets for public transport if needed, once the itinerary has been set.

1.2.3. Goals of the System

- [G1] Allows a visitor to register an account
- [G2] Allows the user to set a new password in case he forgot the old one
- [G3] Allows the user to set preferences
- [G4] Allows the user to edit said set of preferences
- [G5] Allows the user to request for an optimized trip
- [G6] Allows the user to edit and arrange trips or completely cancel them
- [G7] Allows the user to set breaks by giving their timeframes and durations
- [G8] Allows to book/buy tickets for their trips
- [G9] Allows the user to visualize the scheduled trips yet to take anytime

1.3. Definition, Acronyms, Abbreviations

1.3.1. Definitions

- Event: Locations the user has to go to within a certain deadline for a given timeframe
- Trip: The description of route, including the transports, the user takes on to get from the starting location to the event
- Step: A single part of a trip, corresponds to one mean of transport
- Break: An optional pause to consider from all trips and events, it has to be within a chosen timeframe and last for at least a chosen amount of time

1.3.2. Acronyms

DB: DataBase
DBMS: DataBase Management System
RASD: Requirement Analysis and Specification Document

1.3.3. Abbreviations

[Gn] - nth goal
[Dn] - nth goal
[Rn] - nth functional requirements

1.3.4. Reference Documents

- Specification Document: "Mandatory Project Assignments"
- GPS Performances: "http://www.gps.gov/systems/gps/performance/accuracy/"
- Alloy Dynamic Model examples

1.3.5. Revision History

- 1.0 - First completed Draft (29/10/2016)
- 1.1 - Deleted Text Assumption, reordered the requirements, added [G9]

2. Overall Description

2.1. Product Perspective

We will develop the system from scratch, with the support of external systems for GPS tracking, mapping ,and paying (in case of rental transport or buying tickets), in order to guarantee a simplification of the overall implementation by decoupling mapping and payment management from our system, and to guarantee the security of both our client's transaction and the reliability of the provided information.

2.2. User Characteristics

2.2.1. Actors

- *Visitors*: A person using the application without being registered. They can only register (or log in)
- *Users/Registered Users*: A person who has registered and thus, once they log in, they are able to employ all of TravLander's functionalities

2.3. Assumptions, Dependencies And Constraints

2.3.3. Constraints

2.3.2.1. Regulatory Policies

Travlendar+ will be developed in HMTL5 and CSS3 according to W3C org. standards.
It complies with the EU law on cookie policies (D. Lgs. 69/2012 - 70/2012).

2.3.2.2. Hardware Limitation

Travlendar+ will support only web browsers newer than 2010. The responsive view will be suitable for devices with screen sizes starting from 4.2'.

2.3.2.3. Interfaces to other applications

Travlendar+ is strongly dependent on MySQL as the main DBMS.

2.3.2.4. Criticality of the application

The application doesn't have a critical role: any of the actions done by Travlendar+ can be accomplished from a phone.

2.3.2.5. Safety and security considerations

Since the payments are processed externally and the data is handled by an external DBMS, there are no specific security issues there.
The users' personal data is protected, as required by law (D.Lgs 196/2003).

2.3.3. Domain Properties

- [D1] - The username and e-mail must be unique for each registered user
- [D2] - The user can't fully employ the functionality of the application until he makes access from the email confirmation link
- [D3] - In case of forgotten password, the new password will be surely sent only to the one user who requested it
- [D4] - User and locations are found by the GPS
- [D5] - The user will be always available to be tracked
- [D6] - The information given by the third party servers (traffic, mapping GPS tracking etc.) are always true
- [D7] - The inputs given by the user are always right
- [D8] - The timeframes have always a limited amount of time
- [D9] - The breaks have to last for at least more than 1 minute
- [D10] - The events have to last for at least more than 1 minute
- [D11] - There can't be a negative numbers of trip per day
- [D12] - The trip will be automatically cancelled once the deadline is past
- [D13] -The system will always give the shortest trip for an event (within the set preferences)
- [D14] - Trips are organized daily (it's not possible to issue a more than 24 hours long trip)

3. SPECIFIC REQUIREMENTS

3.1 External Interfaces Requirements

3.1.1 User Interfaces

3.1.1.1 Account creation

Web

3.1.1.2. All Trips

Web

Mobile

3.1.1.3. Specific Trip

Web

Mobile

3.1.1.4. Master Page
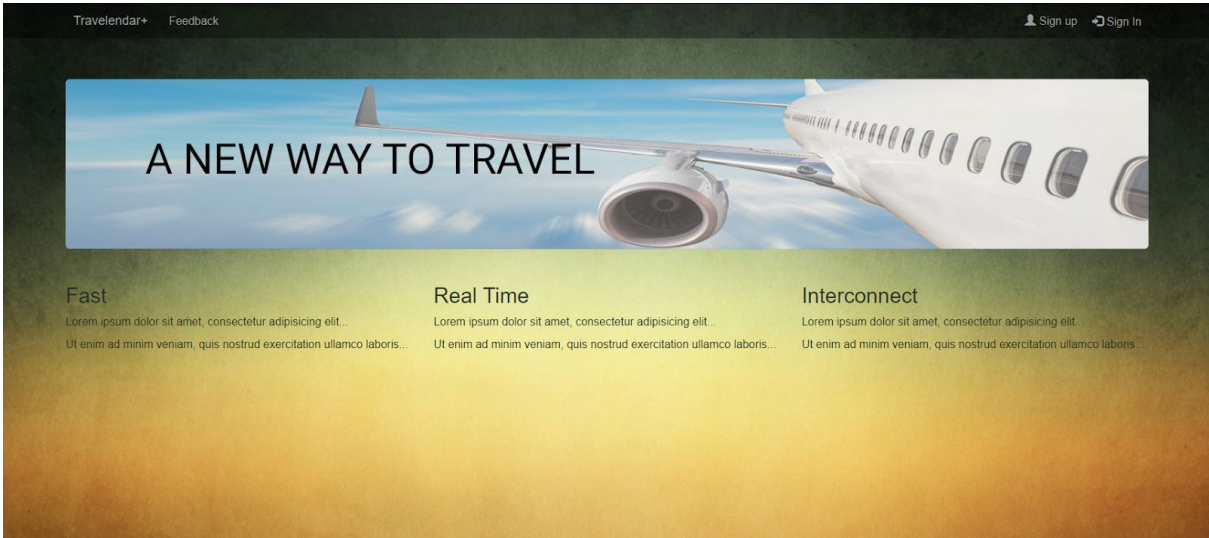
Web

Mobile

3.1.1.5. Password Reset Page

Web

Mobile

3.1.2. Hardware Interfaces

No hardware interface is provided.

3.1.3. Software Interfaces

Our application is suited for most available browsers.

3.1.4. Communication Interfaces

No communication interface is needed.

3.2. Functional Requirements

3.2.1. [G1] Allows a visitor to register

- [R1] - A visitor must be able to access the registration process. During said process they will be required to provide username, password, default preferences.
- [R2] To verify the email, the user is sent a confirmation link (?)

- [D1] - The username must be unique for each registered user
- [D2] - The user can't fully employ the functionality of the application until he makes access to the confirmation link

3.2.2. [G2] Allows the user to receive a new password

- [R3] A registered user must be able to receive a password by confirming the security questions and his e-mail/mobile phone

- [D3] - In case of forgotten password, the new password will be surely sent only to the one user who requested it

3.2.3. [G3] Allows the user to set a default set of preferences

- [R4] A registered user must be able to log-in by using their credential
- [R5] When the user is registered for the first time they are given the option to set further credential to state their preferred transport(s)
- [R6] The system has to take in consideration these preferences when setting up trips and send warning when a trip is not possible this way

3.2.4. [G4] Allows the user to edit said default set of preferences

- [R4] A registered user must be able to log-in by using their credential
- [R7] The user is able to edit the preferences anytime
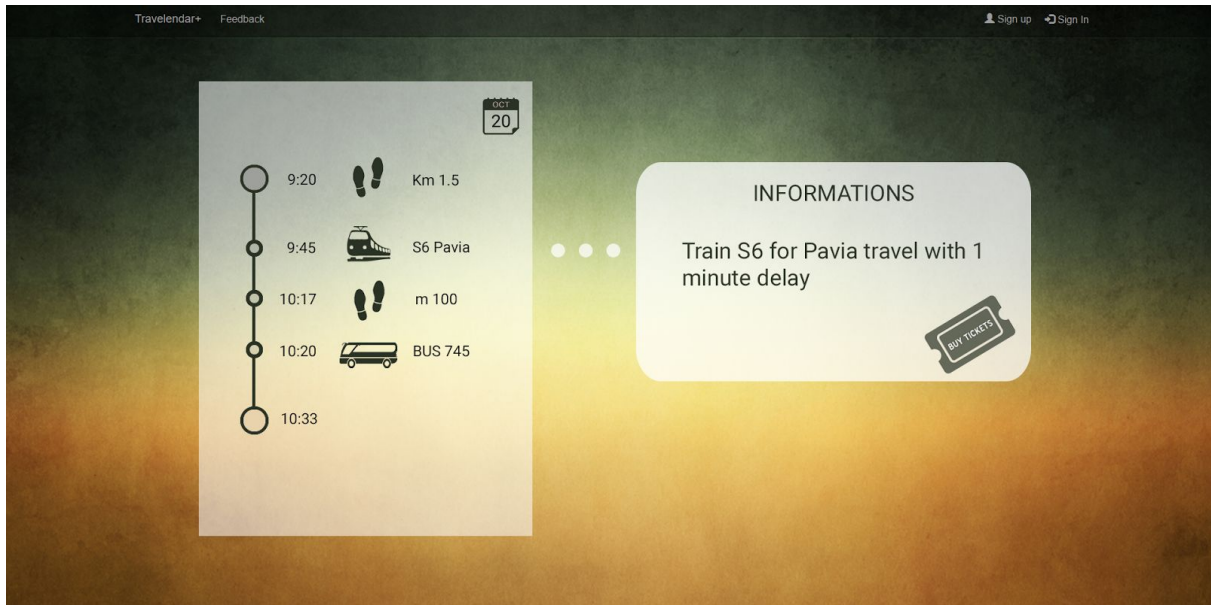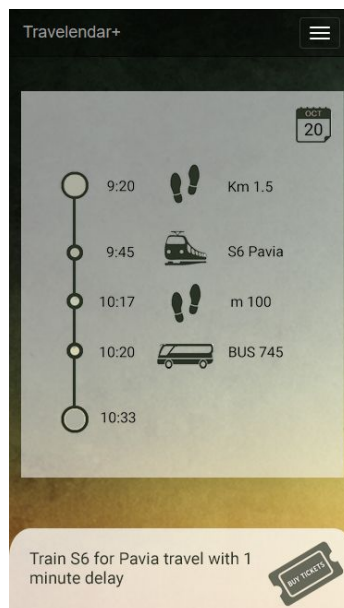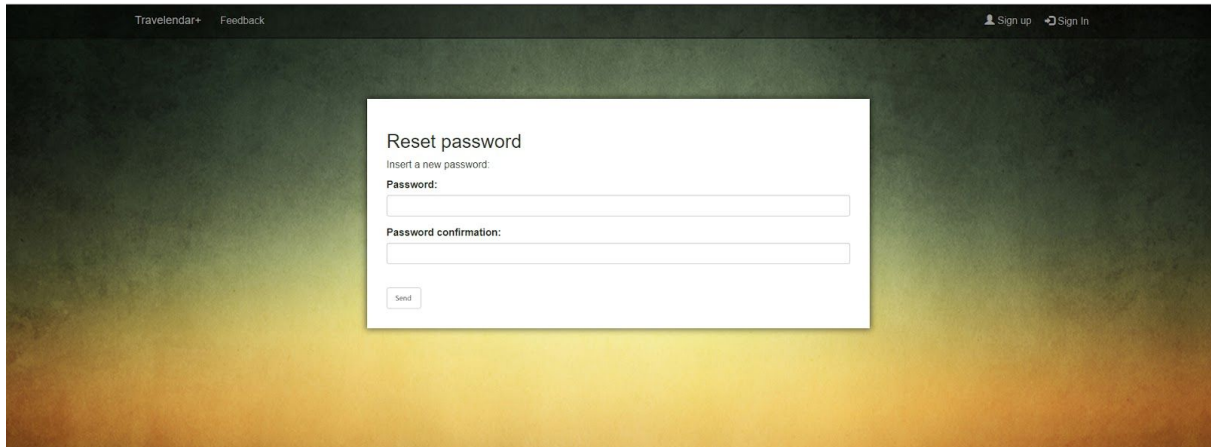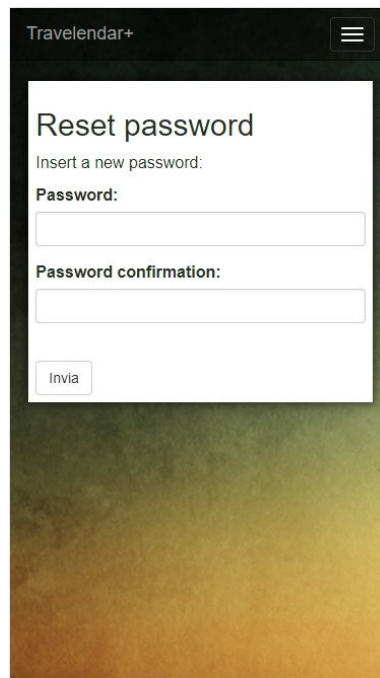- [R8] When he does so, all the trips that are using default preferences are changed as well (including the one he might be taking on at the moment)

3.2.5. [G5] Allows the user to request for an optimized trip

- [R4] A registered user must be able to log-in by using their credential
- [R9] - An user must be able to select the day from the calendar page, then add the time, the 'starting location' and the destination
- [R10] - The GPS is required to track said destination to establish the trip
- [R11] - The user must receive a scheduled trip
- [R12] - In case of unforeseen circumstances given by the mapping/Trafic/ GPS services the server uses, the trip is changed and rescheduled.
- [R13] - In case a trip is not feasible within the given deadline (or the event is located outside the region the user is in), a warning notification is issued

- [R14] - In case two trips overlap, the user is given the option to edit/cancel either one or either until they do not overlap anymore.
- [R15] - In case a trip overlaps with a break or an event, the break is moved down further down the possible timeframe. If the break is not possible, the user is given the option to edit/cancel either one or either until they do not overlap anymore.

- [D5] - The user will be always available to be tracked
- [D6] - The information given by the third party servers (traffic, mapping GPS tracking etc.) are always true
- [D7] - The inputs given by the user are always right
- [D10] - The program can't arrange trips for more than a regional radius
- [D11] - There can't be a negative numbers of trip per day
- [D12] - The trip will be automatically cancelled once the deadline is past
- [D13] -The system will always give the shortest trip for an event(within the set preferences)
- [D14] - Trips are organized daily (it's not possible to issue a more than 24 hours long trip)

3.2.6. [G6] Allows the user to edit, re-arrange trip or completely cancel a new trip

- [R4] A registered user must be able to log-in by using their credential
- [R12] - In case a trip is not feasible within the given deadline (or the event is located outside the region the user is in), a warning notification is issued
- [R13] - In case two trips overlap, the user is given the option to edit/cancel either one or either until they do not overlap anymore.
- [R16] - The user can access to the options of the trip to edit the specifics of the trip, or completely cancel it.
- [R17] - For the preferences of a trip, the system offers as default option the preferences the has set previously.
- [R18] - If the user doesn't wish to do so, he can customize said trips as he wishes by filling out the format.
- [R19] - When the settings are changed, the trip is rescheduled.

- [D5] - The user will be always available to be tracked
- [D6] - The information given by the third party servers (traffic, mapping GPS tracking etc.) are always true
- [D7] - The inputs given by the user are always right
- [D10] - The program can't arrange trips for more than a regional radius
- [D11] - There can't be a negative numbers of trip per day
- [D12] - The trip will be automatically cancelled once the deadline is past
- [D13] -The system will always give the shortest trip for an event (within the set preferences)6
- [D14] - Trips are organized daily (it's not possible to issue a more than 24 hours long trip)

3.2.7. [G7] Allows the user to set breaks by giving their timeframes and durations

- [R4] A registered user must be able to log-in by using their credential
- [R15] - In case a trip overlaps with a break or an event, the break is moved down further down the possible timeframe. If the break is not possible, the user is given the option to edit/cancel either one or either until they do not overlap anymore.
- [R20] The break can be issued either in the singular daily trip or in the Default preferences
- [R21] - If the break's minimum duration time lasts more than the whole timeframe, a warning sign is given
- [R22] - The break has to last at least as much as the minimum duration the user has issued.
- [R23] - The break will be done as soon as possible at the beginning of the timeframe, between the trips during the selected timeframe
- [R24] - The user can't take trips during breaks
- [R25] - When added the break, the system reschedules the trip(s)
- [R26] - In case a break is not feasible within the given timeframe due to the trip a warning notification is issued to the user, who is given the option to cancel the break or rearrange the trips.
- [R27] - Two breaks can't overlap

- [D7] - The inputs given by the user are always right
- [D8] - The timeframes have always a limited amount of time
- [D9] - The breaks have to last for at least more than 1 minut
- [D14] - Trips are organized daily (it's not possible to issue a more than 24 hours long trip)

3.2.8. [G8] Allows to book/buy tickets or set rentals of said itinerary to further arrange details

- [R4] A registered user must be able to log-in by using their credential
- [R28] - When a trip is scheduled, the user is given the option, in case of public transport, to go to a site where he can buy/book a ticket.

3.2.9. [G9] Allows the user to visualize the scheduled trips yet to take anytime

- [R4] A registered user must be able to log-in by using their credential
- [R29] - By selecting a date, the system shows the user the individual steps of the trips of that day

3.3. Performance Requirements

The system must be able to respond to a possibly great number of simultaneous requests and more generally to a great number of request throughout the day.
Basing the analysis on competitors' results an average of 2.000 (two thousands) subscribed users and 100 (one hundred) trips per day may have to be expected: our application will be capable of withstanding traffic up to twenty times higher.

3.4. Design Constraints

3.4.1. Standards compliance

The system will have to ask for users' permission in order to retrieve and use their positions without (at least in a first implementation) storing them. Telephone numbers and e-mail addresses won't be used for commercial uses.

3.4.2. Hardware limitations

- Mobile Browsers
    - iOS or Android smartphone
    - 2G/3G/4G connection
    - GPS
- Web Browsers
    - Modern browser able to retrieve user's location

3.5. Software System Attributes

3.5.1. Reliability

All the sensors used must provide positions' data with an error lower than 10 meters (GPS technology is theoretically able to provide locations with an error less or equal to 7.8 meters).
In case of connection problems to the server, the system will be restored in five minutes.
We guarantee at least 30 minutes of service between two consecutive incidents.

3.5.2. Availability

The system must guarantee a 24/7 service in case a trip is changed mid travel due to unforeseen circumstances. Small deviations from such requirement is acceptable

3.5.3. Maintainability

Maintenance events will require the system to be down for no more than 0.001% of the time.

3.5.4. Portability

Our system's compliance will be suitable for the most popular browsers (example: Explorer, Chrome, Opera, Safari etc.). It will also be supported by all devices (desktop, tablet, smartphone etc.)

4. UML Modeling

4.1. Use case description

4.1.1 Account Registration

Use CASE	Account registration
Description	Interested client register himself in order to access Travelendar+ services.
Actors	Unregistered client, DBMS, third party server (Email).
Pre conditions	Interested client isn't registered on Travelendar+ database.
Post conditions	Interested client becomes a member of Travelendar+'s users and is allowed to access to all features, like schedule an event and more others.
Normal step	Interested client who wants to use Travelendar+ application clicks the "Sign up" button and goes to the Register page. Here he insert his data, like username, password, e-mail, and more. Clicking "Submit" button the system verifies the inserted information and, if everything is ok, store it in the Database. After that, an e-mail confirmation will be sent within 5 minutes. From that moment the new user can access Travelendar+ features.
Alternative step	Interested client insert incorrect informations or the Travelendar+ system is unavailable or required information aren't inserted. In all these cases an error message will be displayed containing more information about the error occurred.

4.1.2 Preference Setting

Use CASE	Preference setting
Description	User modify preference settings.
Actors	User, DBMS

Pre conditions	User is registered on DB. User is logged in, with an active session.
Post conditions	Preference settings will be edited according user's request.
Normal step	On the "preferences" page, the user edits previously inserted settings. The data in the submitted form is used to update the Database.
Alternative step	If the system is temporarily unavailable an error message will be shown with more details. No preferences have been changed.

4.1.3. Editing an Event

Use CASE	Editing an event
Description	To be able to change the specifics of a trip to better suit your new preferences and/or circumstances
Actors	External services, User, DBMS
Pre conditions	User is registered on DB. User is logged in, with an active session
Post conditions	(if the event exists) The event's details have been set to new user's preference

Normal step	The user is on "Specific Trip" page, by pressing the edit button a pop up asks for confirmation on the action, and in case of consent a request is sent to the web server, specifying the ID of the event.
	The application queries the DBMS to check that an event with the requested ID exists, if a following and/or preceding event exists, and if a trip is associated with these events.
	A pop up appears with the filled datas of the trip to edit
	● If there is no feasible trip (and if there are eventual breaks - See Other Use Cases) between the rescheduled event to the following/preceding one (if either exists), the event is NOT edited and a pop up with a message error appears, with the option to either cancel the action or delete the rescheduled event.
	● Otherwise, If a new trip is feasible and a previous trip was associated with the event to be edited, a popup asks the user whether to cancel the action or continue anyway and reschedule the following trip.
	    ○ If the user decides to cancel, nothing happens
	    ○ If the user decides to continue anyway, the event is edited, and old trips are erased and the rescheduled

	ones are registered
Alternative step	If anything goes wrong during the procedure, the user is shown a failure popup.

4.1.4. Buying a Ticket

Use CASE	Buying a ticket
Description	The user is given the option sent on a third party page where to buy the tickets
Actors	External services, User, DBMS
Pre conditions	User is registered User is logged in, with an active session There is a trip already registered in the Database
Post conditions	The User is sent on another page, where the Ticket is bought/booked.
Normal step	Either once the user has made a trip automatically, or in the 'Selected Trip' page, a pop up question appear: given the public transport(s) suggested by the trip, the user are asked whether they wish to buy the ticket ● If the user chooses 'Yes', the user is sent to the respective third party page, where he can buy the ticket.
Alternative step	If anything goes wrong during the procedure, the user is shown a failure popup.

4.1.5 Adding a break

Use CASE	Adding a Break
Description	Add a new break in a either specific trip or the user preferences
Actors	External services, User, DBMS
Pre conditions	User is registered User is logged in, with an active session
Post conditions	A new break is added

Normal step	Either once 'Selected Trip' page or the 'Preference Setting' pages, by choosing the option to add another break, a pop up with a form to edit appear which gives ● The start of the break time window ● The end of the break time window (if the end of the break is before the start, an error pop up appears) ● The least amount of minutes the break should happen (if the number is not a positive integer, an error message pop ups) In case it is done in preference setting, once the break is added, all the trips registered in the database are checked (or in case of 'Selected Trip' only said trip). If there is one where such a break is not feasible, a pop up error message appear, with the option to either edit or cancel the break
Alternative step	If anything goes wrong during the procedure, the user is shown a failure popup.

4.1.6. Visualization of a specific trip

Use CASE	Visualization of a specific trip
Description	A user viewing the calendar wants to see the details of one trip in particular.
Actors	User, DBMS, Third part actors (transportation, weather, traffic)
Pre conditions	The user is already registered on the DB. The user is logged in, with an active session.
Post conditions	The user received the correct data.

Normal step	When the user navigates to the "Specific trip" page, a request is sent to the web server, specifying the ID of the trip. The application queries the DBMS to check that a trip with the requested ID exists among the trips scheduled for the user that requested it, and to obtain all steps of the trip.
	For each step, the application communicates with third-part actors to check if the transportation is still available:
	● for trains, it is asserted that the train wasn't canceled or delayed;
	● for cars/buses, traffic servers are inquired for roadblocks;
	● for (motor)bikes or walking, the weather forecast is checked.
	While this happens, the user is directed to the "Specific trip" page: at first void of content (except for a loading icon), it is filled asynchronously.
	If nothing is wrong, the page displays each step in detail.
Alternative step	Instead of the trip data, the page displays an error message if:
	● No trip with the specified ID was scheduled for the user.
	● The connection dropped at any time during the processing.
	● Any step has to be rescheduled.
	○ If a ticket was bought, a link for a refund is displayed.
	○ If a ticket needs to be bought, a link to do it is displayed.
	○ If no other path was found, the user is notified.

4.1.7. Password Reset

Use CASE	Password reset
Description	A user forgot the chosen password and needs a new one.
Actors	User, DBMS.External server (emails)
Pre conditions	None.
Post conditions	(if the account already existed) The account password is set to the new chosen password.

Normal step	The user is on any page visible without login. By clicking on the link below the login textboxes, a request is sent to the web server that directs the user to a new page that prompts the user to write the email address / username linked to the account to be retrieved. As the address is written, asynchronously, the application queries the DBMS to check if the said address is registered. If it is, the page loads the security question associated to the user's account. If the user's answer is correct, two textboxes will appear for the new chosen password (and confirmation). If the user, within a time limit, submits the form, the new chosen password will be registered over the old one.
Alternative step	If the requested email address is not in the system, an error will pop up. If the user does not register a new password within the time limit, an email will be sent to warn the user that someone tried changing the password and all evidence of the act will be canceled. If the user fails the security question five times, the same email is sent and the account is locked for 24 hours.

4.1.8. Cancelling an Event

Use CASE	Canceling an event
Description	A user needs to cancel an event, as well as the previous and following trips, and may want to create a new trip for the following event.
Actors	User, DBMS
Pre conditions	The user is registered on the DB. The user is logged in, with an active session.
Post conditions	(if the event existed) The event is absent from the DB.

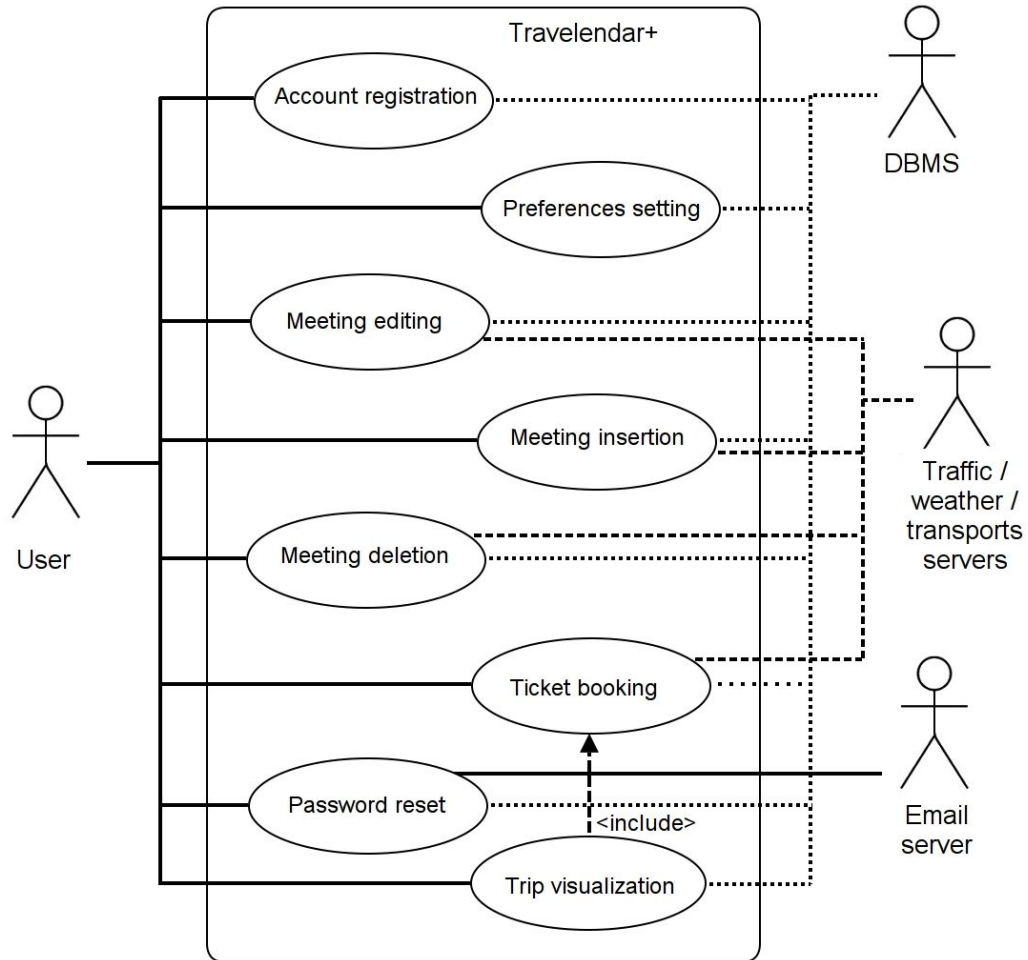| Normal step	The user is either on the "All trips" page or on the "Specific trip" page. Either way, when clicking on "cancel event" a popup asks for confirmation on the action, and in case of consent a request is sent to the web server, specifying the ID of the event.
	The application queries the DBMS to check that an event with the requested ID exists, if a following event exists, and if a trip is associated with either event.
	If no trip is associated with the following event, the event is erased from the DB and a confirmation popup is asynchronously loaded for the user.
	If a trip is associated with the event to be canceled, it is erased as well.
	If a trip is associated with the following event, a popup asks the user whether to cancel the action, continue anyway, or reschedule the following trip (see another use case).
Alternative step	If anything goes wrong during the procedure, the user is shown a failure popup.

4.1.9. Insert Event

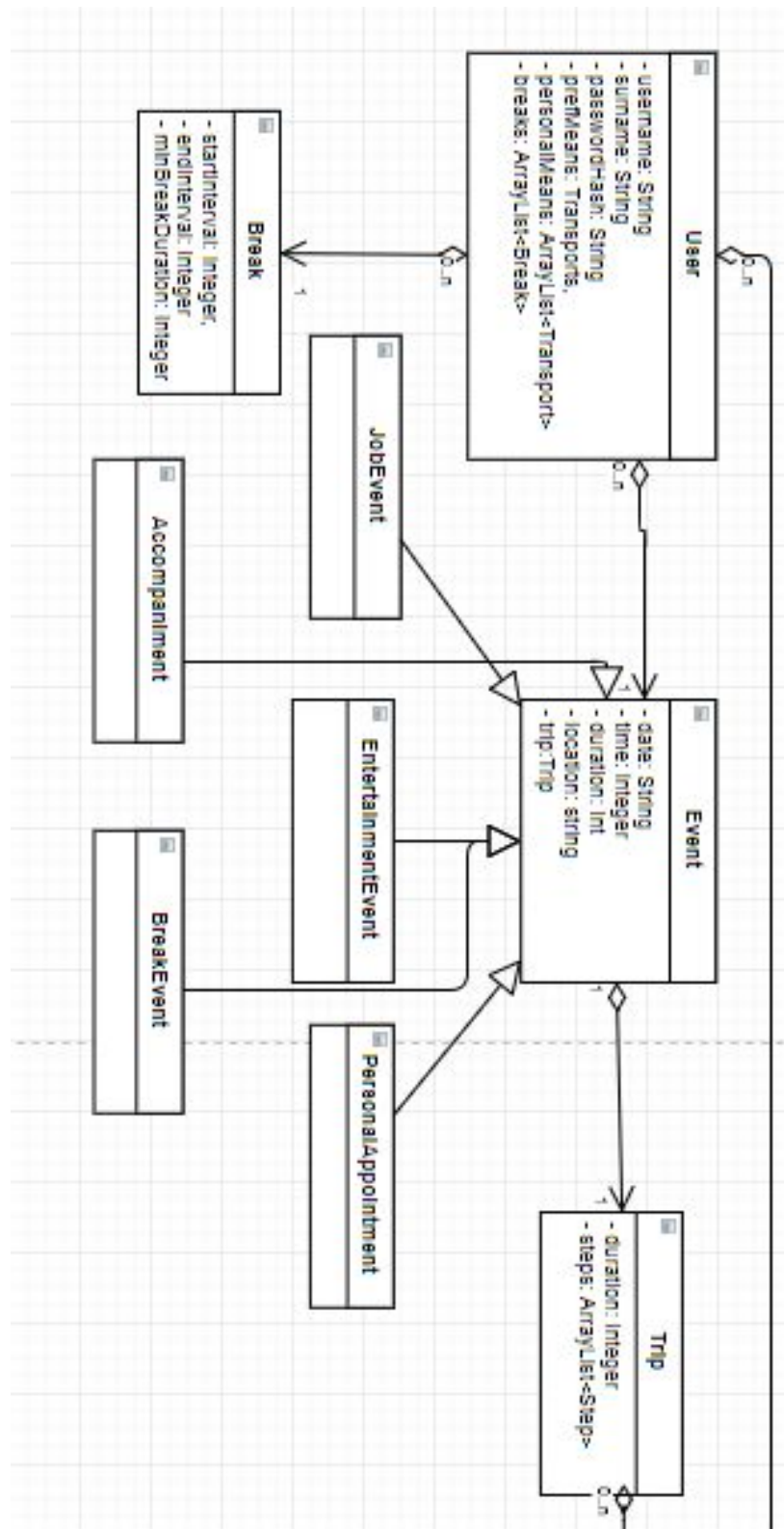| Use CASE	Insert Event
Description	User insert an event in order to schedule a trip.
Actors	User, DBMS, Third part actors
Pre conditions	The user is registered on the DB. The user is logged in, with an active session.
Post conditions	A new event is registered for the user who requested it.
Normal step	User selects the date on "All Trips" page by clicking or tapping on it on the calendar. A schedule appears on the right side, showing all events registered at the moment. The time of the new event is selected by clicking or tapping one of the empty time slots in the schedule. A new modal popup prompts the user to write the remaining information about the event (like location and type). Travlendar+ finds the best series of steps, taking into account not only the shortest travel time, but also (if needed) traffic and weather information obtained from third parties, and choosing the most fitting mean of transport for the event's type. A popup confirmation will appear for the user.

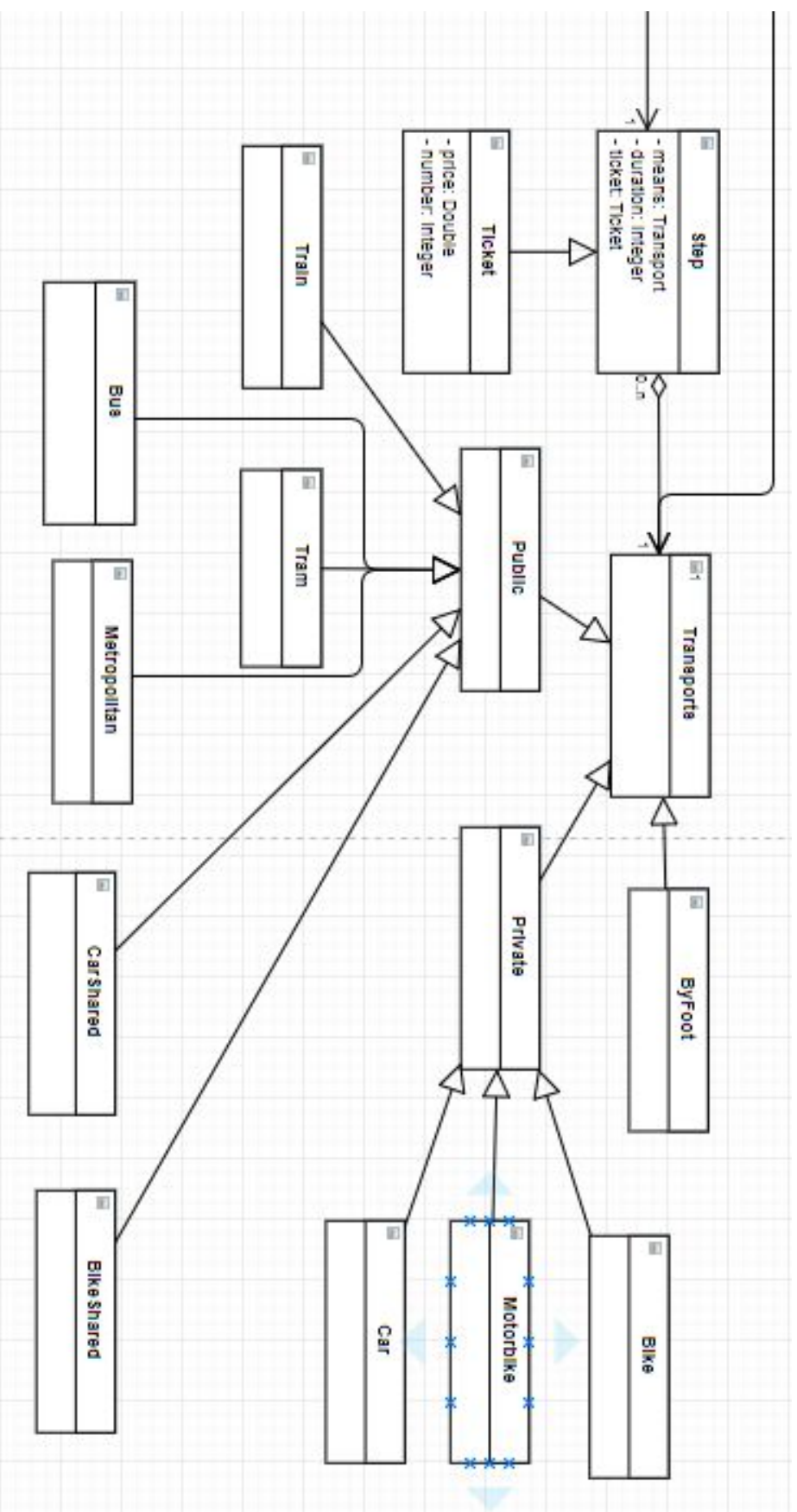| Alternative step	If system is temporarily unavailable or trip cannot be scheduled an error message will be shown with error details and no trip has been scheduled.

4.2. Use case Diagram

4.3. Class diagram

User
- username: String
- surname: String
- password-hash: String
- prefMeans: Transports,
- personalMeans: ArrayList<Transport>
- breaks: ArrayList<Break>

Break
- startInterval: Integer,
- endInterval: Integer
- minBreakDuration: Integer

JobEvent

Accompaniment

EntertainmentEvent

BreakEvent

PersonalAppointment

Event
- date: String
- time: Integer
- duration: Int
- location: string
- trip:Trip

Trip
- duration: Integer
- steps: ArrayList<Step>

UML Class Diagram

Step
- means: Transport
- duration: Integer
- ticket: Ticket

Ticket
- price: Double
- number: Integer

Train

Bus

Metropolitan

Tram

Public

Transporte

CarShared

BikeShared

Private

ByFoot

Car

Motorbike

Bike

4.4. Sequence Diagrams

4.4.1. Event Insertions
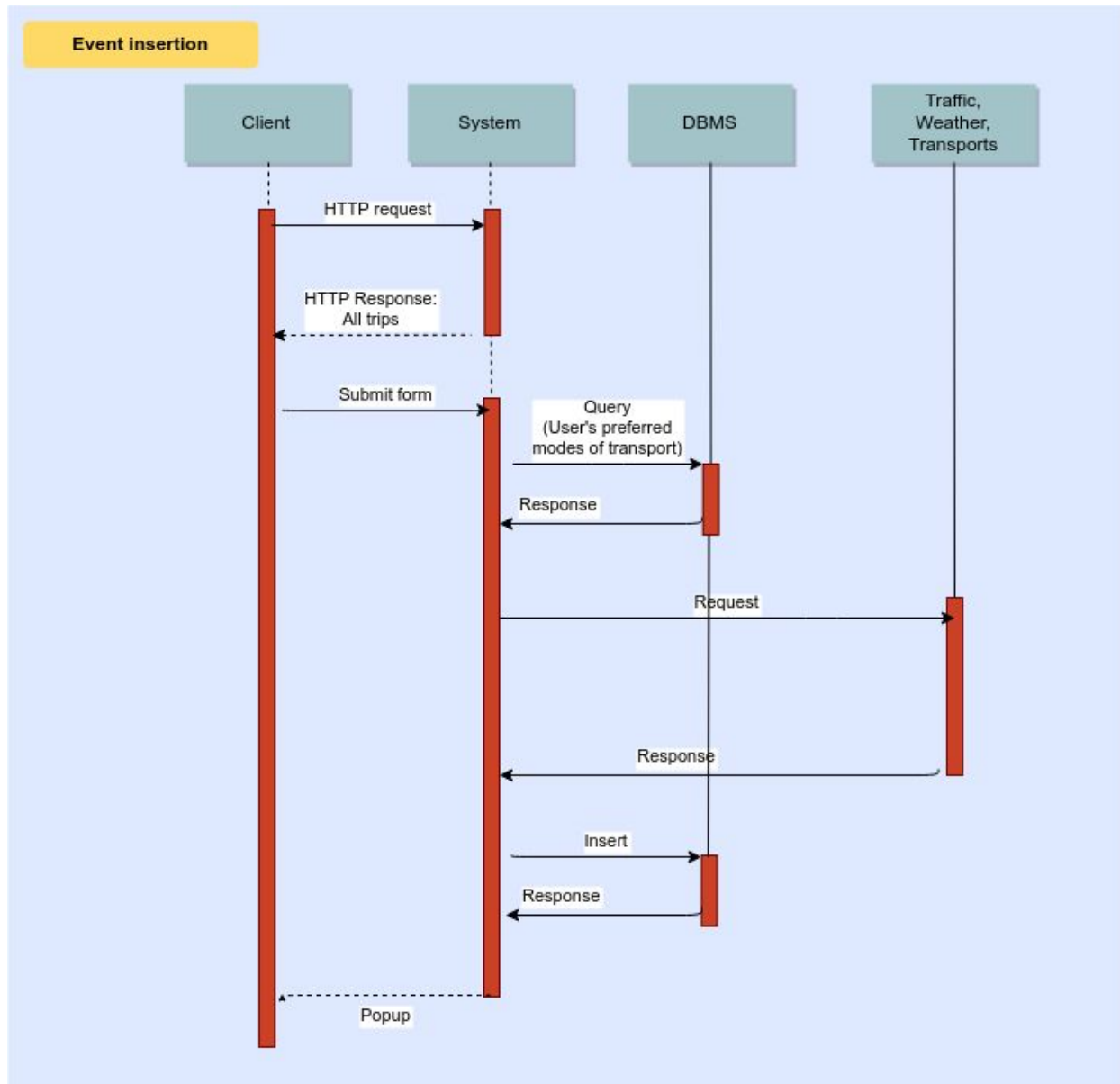
4.4.2. Event Insertion

4.4.3. Account Registration

Account registration

| Client | System | DBMS | Email server |

- HTTP request
- HTTP Response: Registration page
- Submit form
- Insert
- Response
- Request
- Response
- Popup
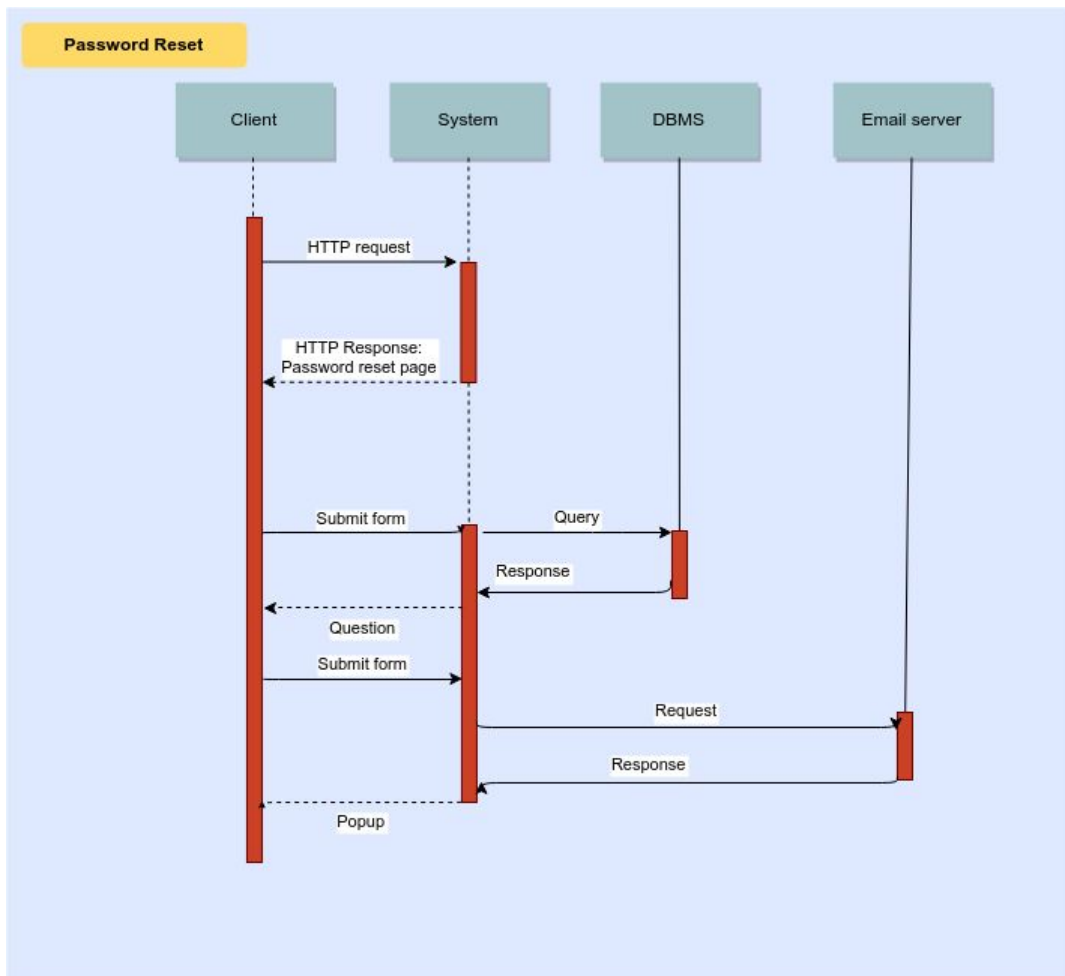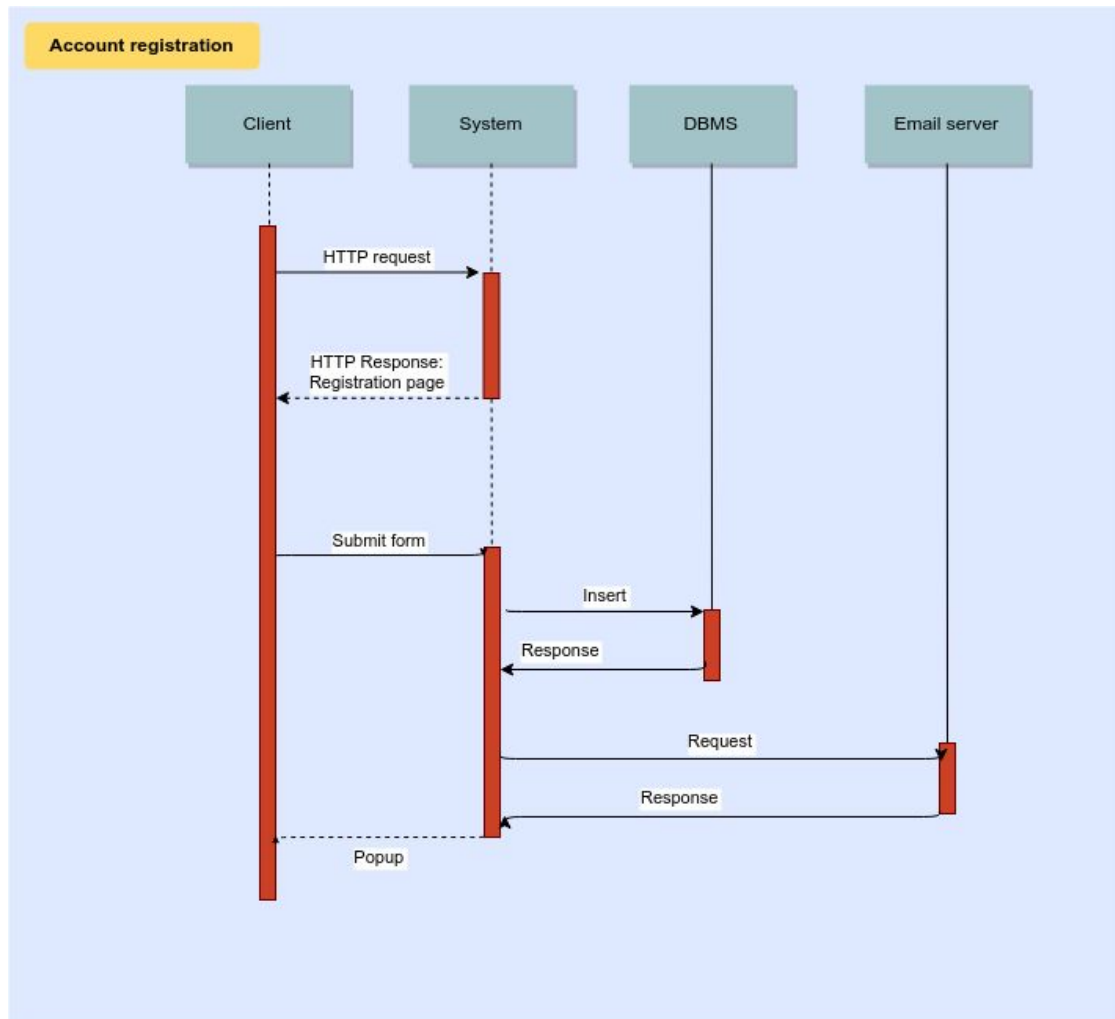
4.5 Activity Diagram
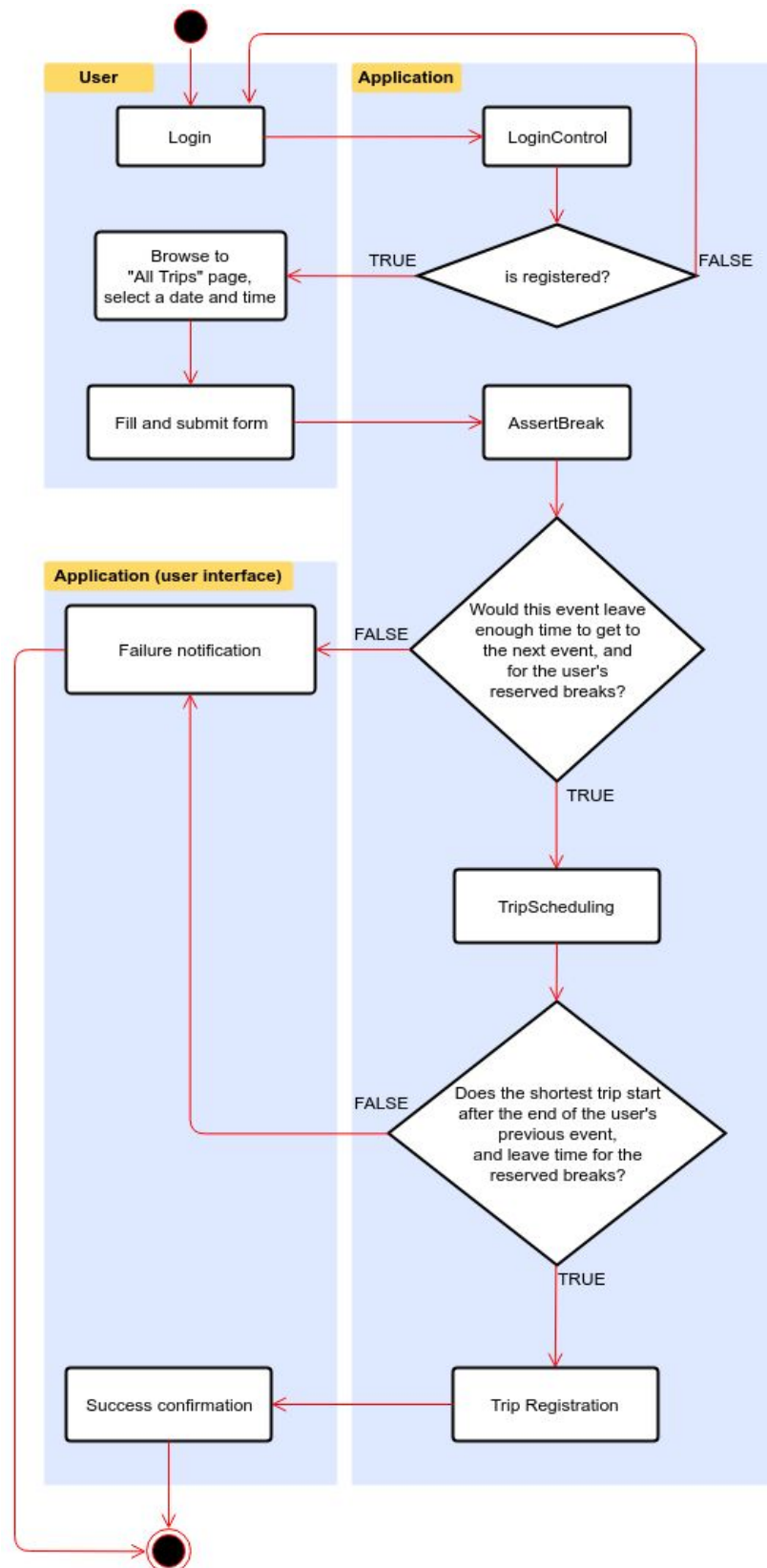
5. Allow Modeling

```
open util/boolean

sig User{
/*
username: String,
email: String,
password_hash: String,
*/
settings: one Preferences,
events_of_user: set Event}
{}

sig Preferences{
user_of_pref: one User,
pref_means: set Transport,
has_car: one Bool,
has_bike: one Bool,
breaks: set Break}
{}

sig Break{
start_interval: Int,                     //should be string, but needed to enforce logic
end_interval: Int,                       //should be string, but needed to enforce logic
min_duration: Int}
{start_interval > 0 end_interval > 0 min_duration > 0
end_interval > start_interval
min_duration < (end_interval - start_interval)}

sig Event{
/*
date: String,
location: String,
*/
user_of_event: one User,
time: Int,
duration_of_event: Int,
type: one EventType,
trip_of_event: lone Trip}                //absent if and only if break
{time > 0 duration_of_event > 0}

abstract sig EventType{}
```

```
sig JobMeeting extends EventType{}

sig EntartainmentMeeting extends EventType{}

sig PersonalAppointment extends EventType{}

sig Accompaniment extends EventType{}

sig BreakEvent extends EventType{}

sig Trip{
event_of_trip: one Event,
duration_of_trip: Int,
steps_of_trip: set Step}
{duration_of_trip > 0 #steps_of_trip > 0}

sig Step{
trip_of_step: one Trip,
mean: one Transport,
duration_of_step: Int,
ticket: lone Ticket}
{duration_of_step > 0
#ticket = 1 => (mean in Public)}

sig Ticket{
//somedata: String,
step_of_ticket: one Step
}

abstract sig Transport{}

abstract sig Public extends Transport{}

abstract sig Private extends Transport{
//location: String
}

sig ByFoot extends Transport{}

sig Train extends Public{}

sig Tram extends Public{}

sig Metro extends Public{}
```

```
sig Bus extends Public{}

sig CarShared extends Public{}

sig BikeShared extends Public{}

sig Car extends Private{}

sig Bike extends Private{}

/*
fact EmailAreUnique {
all u1,u2: User | (u1 != u2) => u1.email !=u2.email }

fact NamesAreUnique {
all u1,u2: User | (u1 != u2) => u1.username !=u2.username }
*/

fact StepSum{
all t: Trip | t.duration_of_trip = (sum s: t.steps_of_trip| s.duration_of_step) }

fact PreferredMeans {
all s: Step | s.mean in s.trip_of_step.event_of_trip.user_of_event.settings.pref_means }

fact HasCar {
Car in Preferences.pref_means => Preferences.has_car.isTrue }

fact HasBike {
Bike in Preferences.pref_means => Preferences.has_bike.isTrue }

fact NoTripForBreak {
all e: Event |#e.trip_of_event = 0 <=> e.type = BreakEvent }

fact ReciprocalSettings{
user_of_pref = ~settings}

fact ReciprocalEvent{
user_of_event = ~events_of_user}

fact ReciprocalTrip{
trip_of_event = ~event_of_trip}

fact ReciprocalStep{
trip_of_step = ~steps_of_trip}

fact ReciprocalTicket{
```
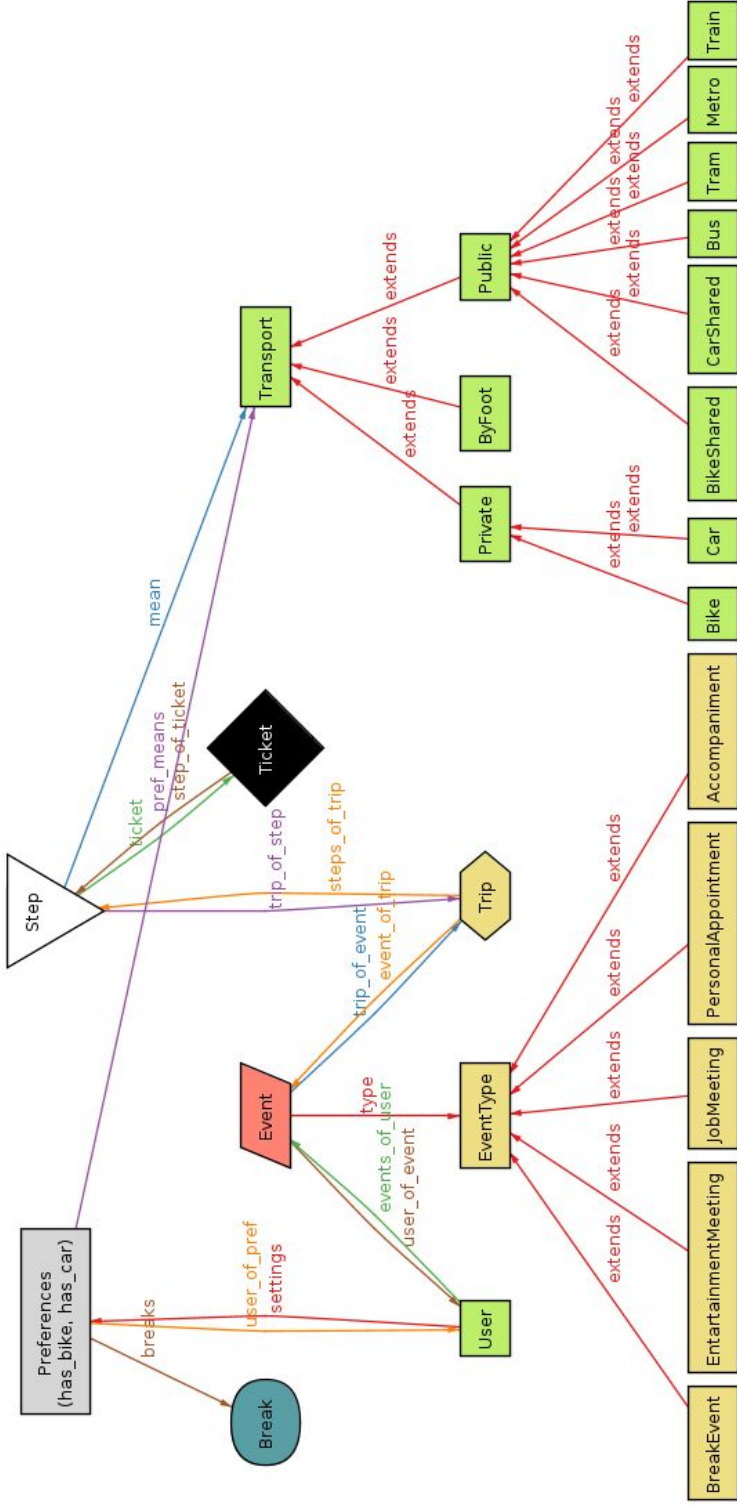
```
ticket = ~step_of_ticket}


assert relations{
#User = #Preferences
#Event >= #Trip
#Step >= #Trip
}


pred general() {
#User > 3
#Break > 0
#Event > 9
#Trip > 6
#Transport > 6
#ByFoot = 1
#Ticket > 3
}


check relations
run general for 8
```

6.Appendix

6.1. Effort spent

Antonino Caminiti
- 4/10/2017 (3 hours) + 1 hour to set down all of our inputs and begin a first draft of the RASD
- 11/10/2017 (2 hours)
- 18/10/2017 (3 hours) + 2 hours (3 use cases)
- 25/10/2017 1hours+3 hours (Table of contents + Overview)
- 28/10/2017: 5 hours
- 29/10/2017: 3 hours (Corrections and others)

Daniele Gonella
- 4/10/2017 (3 hours)
- 11/10/2017 (3 hours) + 5 hours (provide Mockup)
- 18/10/2017 (3 hours) + 1 hours (2 use cases)
- 25/10/2017 1hours+3 hours (Class diagram)
- 28/10/2017: 5 hours
- 29/10/2017: 3 hours

Matteo Scerbo
- 4/10/2017 (3 hours)
- 11/10/2017 (3 hours) + 2 hours (3 use cases)
- 18/10/2017 (3 hours)
- 25/10/2017 1hours+3 hours (Class diagram)
- 28/10/2017: 5 hours (Alloy Modeling)
- 29/10/2017: 3 hours

6.2 Tools employed

- Google Docs
- www.draw.io - Flowchart and Online Diagram software
- Alloy Analyzer 4.2 (2012/09/25 EDIT)
- Photoshop