POLITECNICO
MILANO 1863

Travlendar+ project Jerkenhag, Kverne, Rebner

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Jerkenhag, Kverne, Rebner
Version:	1.0
Date:	October 27, 2017
Download page:	https://github.com/Rebnersaurus/JerkenhagKverneRebner
Copyright:	Copyright © 2017, Jerkenhag, Kverne, Rebner – All rights reserved

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The goal of the project is to create a dynamic calendar-based application that will assist the user in managing her meetings as well as the travel means in between them in a given city or region. The application should allow for a customizable way of getting the directions that the user needs based on her preferences when it comes to mode of transportation as well as the ecologic impact. Consideration should also be taken to the day, for example if there's a strike or the weather can have an impact the app will notify the user and route accordingly.

In short we want Travlendar+ to be:

- Knowledgeable
- Reliable
- Versatile
- Environmental Friendly

1.1.1 Goals

Clients:

G1 Allow clients to access the calendar.

G2 Allow clients to customize their preferences regarding mean of transportation.

G3 Allow clients to customize/rate/evaluate/assess their prefered way of travel between economics, environment and weather sensitivity.

G4 Allow clients to add and delete events in their calendar.

G5 Allow clients to add a flexible lunch of a minimum of 30 minutes that the application will take into consideration.

G6 Clients should receive a warning if meetings collide.

G7 Allow clients to access an accurate description of an optimal path to take to an event according to their own settings.

G8 Allow clients to purchase tickets to transports within the application.

G9 Clients should receive a warning if a destination is unreachable at the set time.

G10 Clients should be able to create a profile.

1.2 Scope

Our goal is for the Travlendar+ to be in contact with multiple points of interest around the world, gathering information to give the user the best option of transporting and scheduling considering the given constraints.

The world that Travlendar+ will interact with draws inspiration of the city of Milan since this is where the development team is working. The different transportation means included in the app are MoTs that you can find around Milan and big cities in general. Travlendar+ will work best in these types of areas where multiple choices are offered for arriving at a destination (e.g. if you had an environment where

the only way to get somewhere was by car Travlendar+'s planning suggestions would be of little use). Since Travlendar+ is generating its results based on the goings on of the real world, the schedules and availabilities of the transportations will greatly affect the output. Further, if there was a MoT available in a city not supported by Travlendar+ it would not be used to compute the best route to a destination.

Since Travlendar+ is first and foremost intended as a calendar-based application the focus of the program is to deliver a continuously updated best way to travel between the users meetings for each day. In the current state the application will require of the user to manually enter the addresses and will not keep live track of the users geoposition, but it is an idea for future development.

Constraints:

Travlendar+ will require an Internet connection in order to update, fetch and display results. Travlendar+ will only be able to use open shared API's and will not show traveling option for companies that does not share their API's. Travlendar+ will only be able to reflect the best way to get to a destination based on the means that are available.

1.3 Definitions, acronyms, abbreviations

Acronyms:

- MoT - Method of Transportation
- FR - Functional requirements
- NFR - Non Functional requirements

Definitions:

- Event - A time block on a day in the calendar with a geolocation.
- Journey - The complete route to take to arrive at an event.
- Path - One piece of the journey, i.e. in a journey you might first have to take the bus to the train station and then the train to arrive at your meeting. In this case the journey would consist of 2 paths.
- Preferences - Concerns the importance the user places on the following points: Economy, Environment, Weather Sensitivity
- Relevance - When Travlendar+ generates possible journeys for an event they will be ordered by relevance with the most relevant option on top.
- Economy - How important price is in the user for the relevance of a journey.
- Environment - How important carbon footprint of different MoT is for the user for the relevance of a journey.
- Weather Sensitivity - How willing/unwilling the user is to travel by weather sensitive MoT if there is a bad weather forecast.

1.4 Revision history

First edition.

1.5 Reference documents

1. Assignment document.
2. Software Engineering [1]
3. Fundamentals of software engineering [2]

1.6 Document structure

This document will include an introduction about the scope of the project, then it will present the product perspective in further detail, including relevant models and diagrams. In part three the requirements will be presented. Finally the document will conclude with a detailed alloy model, complete with the code and the output. In the appendix you will find information about test cases for the requirements and further development of the product as well as some additional diagrams.

2 Overall Description

2.1 Product perspective

Travlendar+ will be a web application accessible through mobile devices. The central part of Travlendar+ will be the calendar where all the users meetings are stored ordered by date. Here it is easy to get an overview over the schedule for the day as well as the travel options that are available. The schedule for each day consists of events added by the user which can be edited and removed at will. For each event and journey will be displayed, giving the user suggestions on how to reach the destination from their starting point or the previous meeting.

It will allow different preferences for different users and each user is offered to customize their settings. These settings will be stored in a database within the system in order to filter the traveling options for the user. These settings will be pulled when the application creates a journey for an event. Below is a representation of the domain through a class diagram.

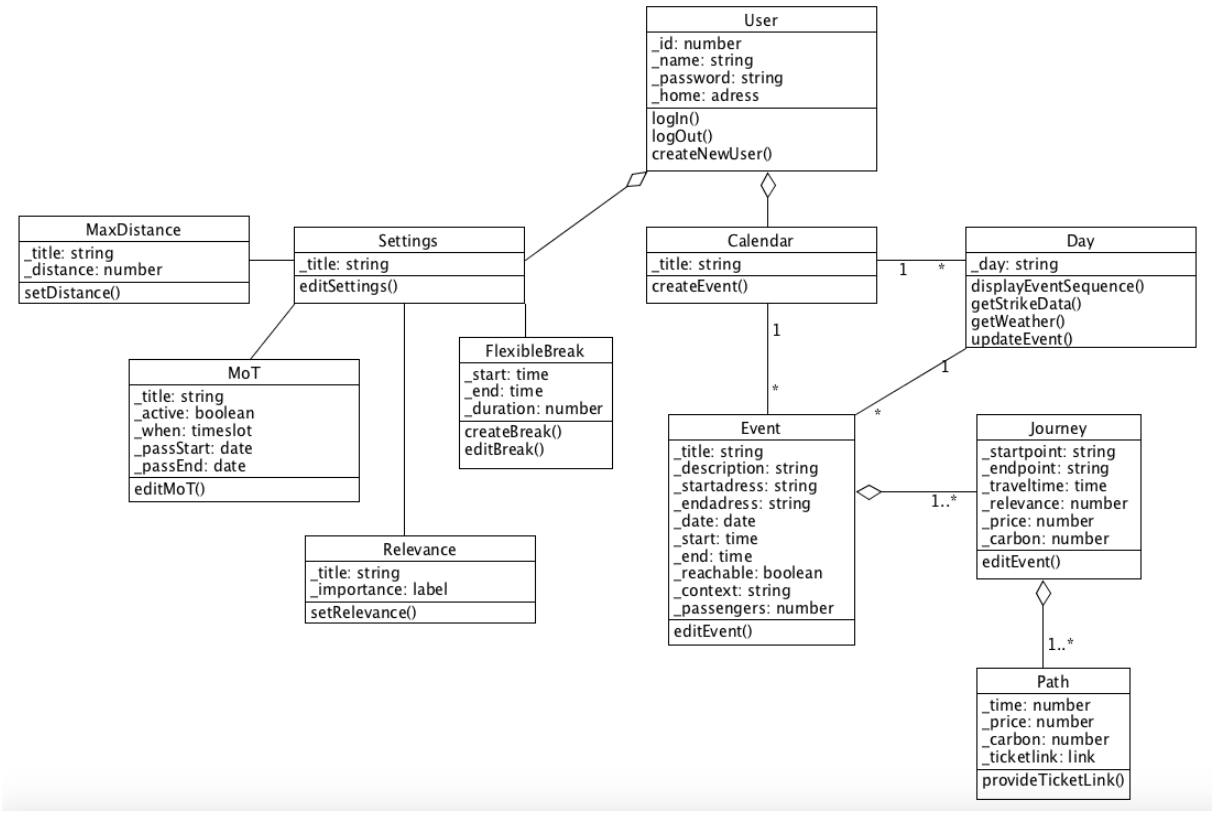

Figure 1: Class diagram

The system will communicate with external services in order to get information about the map and the different travel means that are available from the world that the user is located in. This will be from traveling companies that use an open shared API that are identified in the area, the MoTs will be identified and compared to the users settings. The above state chart depicts how the application uses the event information to derive the possible journeys to get there, and also how the users preferences plays into this in order to customize the results according to the clients needs.

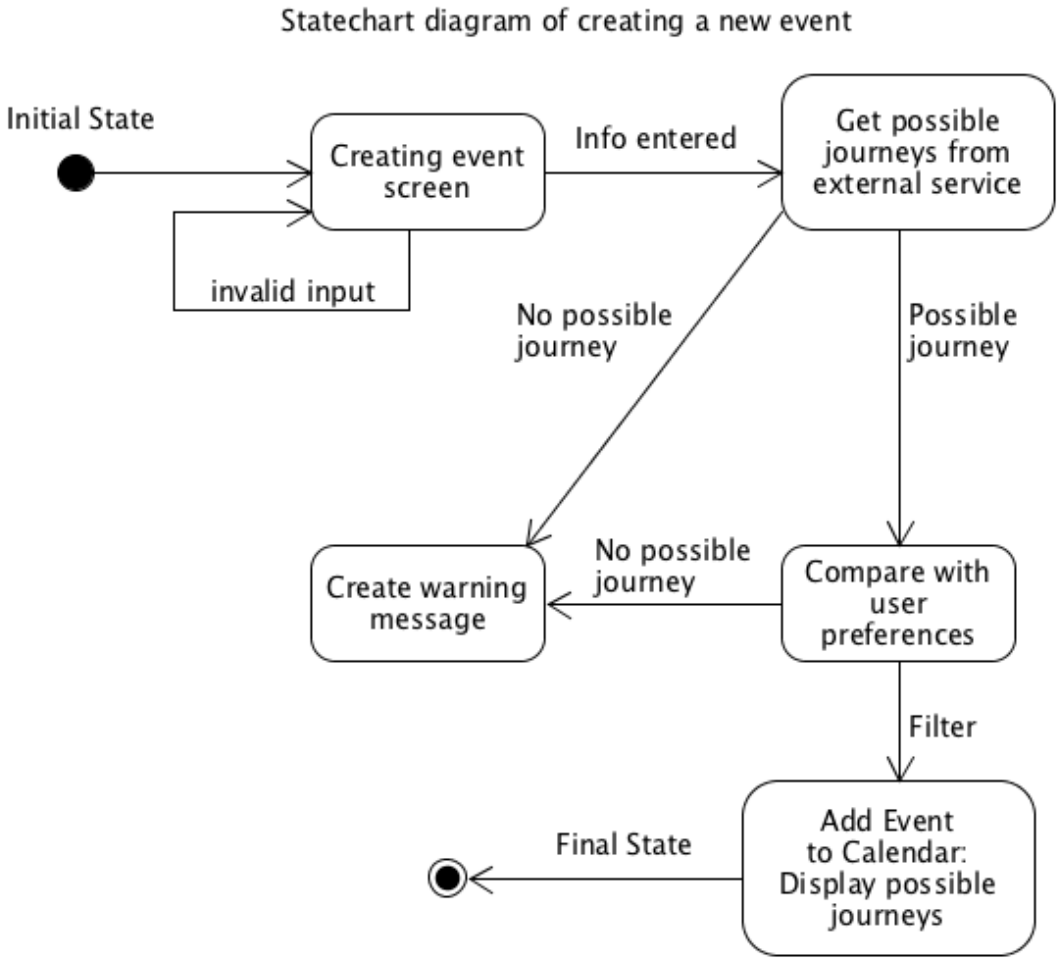

Figure 2: State chart

2.2 Product functions

For Travlendar+ to work as intended it has to deliver on some critical points.

The user should by using application have a reliable calendar where they can easily get an overview over their upcoming schedule. They should be able to add events to the calendar which the application automatically will suggest the potential journeys to by relevance. What an optimal journey is will be different for each user as the application allows the user to tailor their experience by selecting which MoTs they want to use and if they have a seasonal pass to one of these. It should also take into consideration their preferences regarding economic situation, environmental friendliness and weather by calculating the relevance of a path depending on these predispositions. When a journey is created the app should allow the user to buy tickets if required for the journey to be taken. The app should also make the user able to create breaks of different durations that will be included in the daily schedule.

In summary the core functionalities are envisioned as:

- Event creation and event management with the most relevant journey suggested.
- Intuitive journey presentation with the functionality to buy tickets if required.

- Allow for customization of how they want their journeys to be planned through user settings.
- The functionality of a flexible lunch and breaks that the calendar will take into account.

2.3 User characteristics

The application is aimed towards one type of user that will interact with the system which will be individuals as opposed to companies or organizations since the calendar is intended to be for personal use.

Assumptions about the user:

- The user is relying on the application throughout the day and will use it for several meetings.
- The user is responsible for understanding that if he takes his personal vehicle somewhere it will be left there unless he chooses to use it for the next meeting. (e.g Google maps.)
- The user is responsible to plan the meeting accordingly so that they have time to prepare if needed in before their event. The app will consider the travel route successful if it determines that you will arrive before the meeting.
- The user is responsible to keep track of his current location.

2.4 Assumptions, dependencies and constraints

Assumptions about the assignment text:

- We assume the calendar is utilized to create events for a single user, and calculate their journey to and between appointments.
- We assume that the lunch break can be considered as a regular break. We have chosen to solve this with having a break that can be modified into lunch or coffee, by selecting name, duration, and in which time period the break is intended.
- The text states that the application should support a multitude of travel means beyond certain specific ones. For the choices of public transportation we have drawn inspiration from the collective transports of the city of Milan.
- We have assumed from the text that the opportunity of buying transportation tickets will be satisfied by being connected to the ticket purchase system of the transportation in question.

Assumptions about the region:

- There is only one provider of each MoT in the region.
- We assume that external services with an open API will allow us to pull from their data.
- We assume that there is a Weather forecast which data we can pull from.
- We assume that there is a Strike forecast which data we can pull from.

Assumption about the transportation:

- There is only one pass for each vehicle and it makes it free for a duration.

2.5 Identifying stakeholders

Name	Impact How much impact does the project have on the stakeholder? (low - medium - high)	Influence How much influence do they have on the project? (low - medium - high)	What is important to the stakeholder?	How can the stakeholder contribute to the project?	Strategy for engaging the stakeholder
Our team	High	High	To create an application that provides the desired features	To communicate well with the other stakeholders to secure optimal decisions about the application	The team is driven by creating a good delivery and also by the course's' final grade.
Project supervisor: Di Nitto	Medium	High	To have their student groups perform optimal solutions and learn as much as possible.	To communicate well the purpose of the project and be clear about the constraints and expectations.	The project supervisors is driven by helping the students, and will therefore be engaged the entire project.
Students, Workers & Senior citizens	High	Low	Using Travlendar + as an efficient time - and travelling manager in their daily life.	Have no or little possibility to influence the project.	The user is interested in an optimal application that takes care of the daily schedule.

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

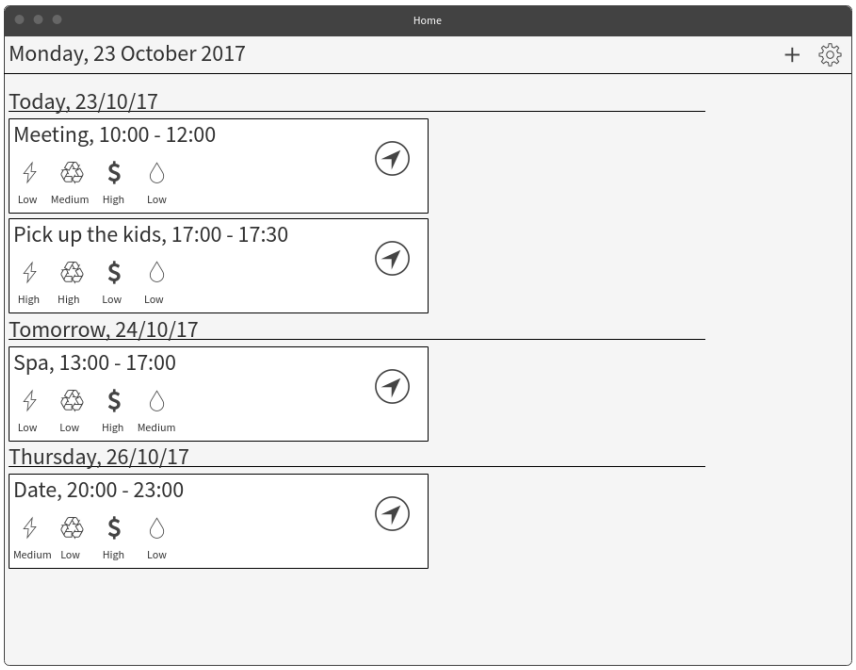

Figure 3: Home-screen

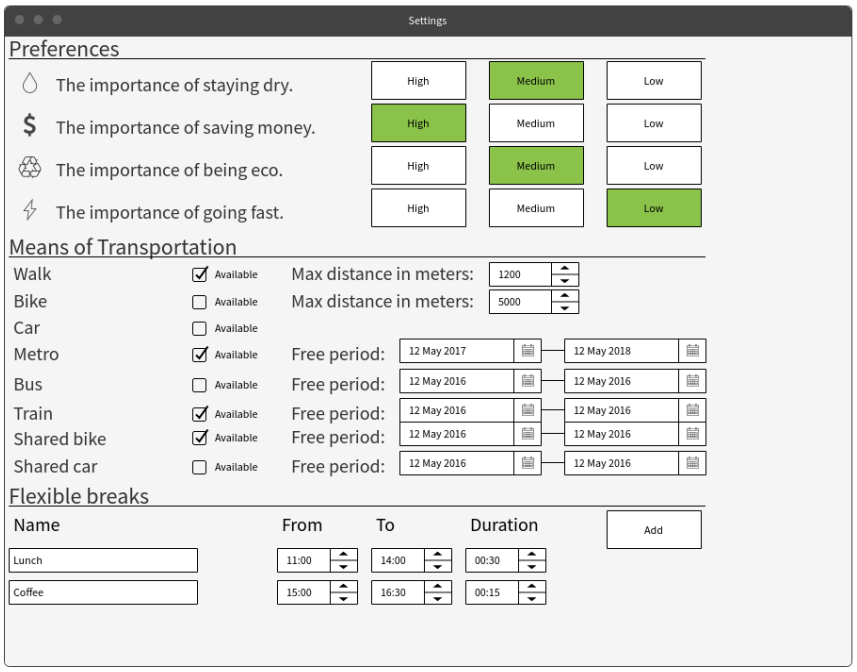

Figure 4: Settings-screen

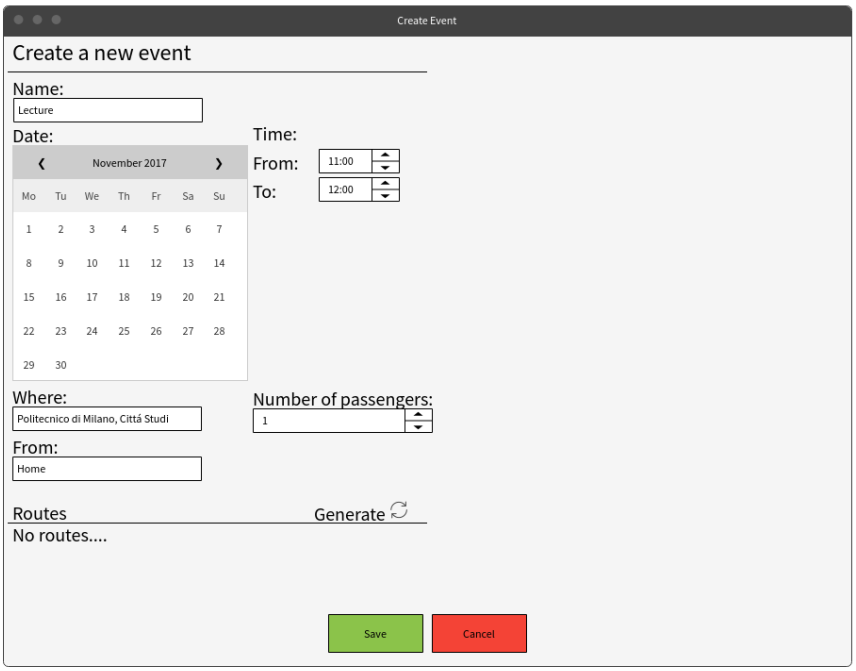

Figure 5: Event-screen

3.1.2 Hardware interfaces

Travlendar+ is intended as a web application that is compatible with iOS- and Android- powered devices. It should also be possible to use it on your computer on a desktop. Settings update, new user, add/delete events are handled by the Travlendar+ server. All interactions always run through the central server first. Travlendar+ is being developed to support these platforms:

Mobile Devices:

- iOS: from version 11.0.3
- Android from version 7.0

Standard browsers:

- Chrome from version 61.0.3163.100
- Mozilla from version 49.0.2
- Safari from version 11.0

All sensitive data will have SSL encryption.

The application should support interactions between different open API's and also able to import Travlendar+ calendar into their standard calendar.

3.1.3 Software interfaces

When showing an event Travlendar+ communicates with the external services in order to get a visual representation of a map and the different journeys to get there. The journeys that are displayed will be in accommodation to the settings and preferences that is stored on the user in the system database. In both of these cases Travlendar+ will only use reading operations to get the data and not modify it.

3.1.4 Communication interfaces

The application support a push protocol to give information about(when the application is in use):
Notifications about warnings:

- Due to strike
- Due to weather

Notifications regarding event creation:

- Colliding with event or break
- Unreachable destination

The application will use a pull protocol (when the application is in use):

- To consistently update the user's journey
- Extract and sync the newest update

3.2 Functional requirements

In this section the use case diagram depicts the different kinds of interactions the user can have with Travlendar. A breakdown of the different functional requirements is included.

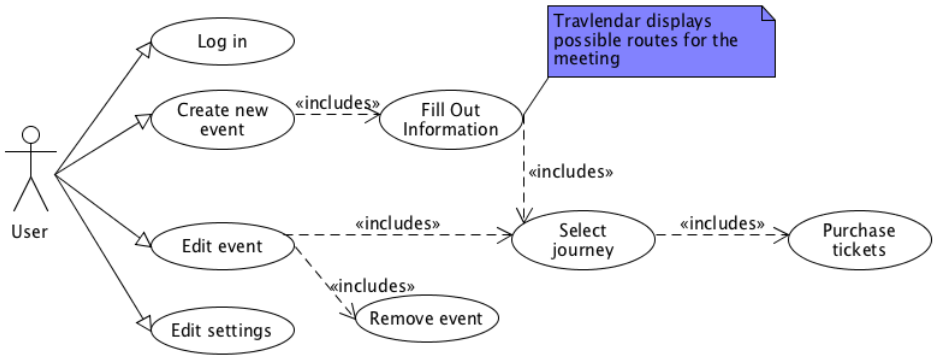

Figure 6: Use Case Diagram

Story ID	Goal ID	Functional requirement	NFR ID	Testing ID	Priority (low - medium - high)
FR1	G1	**Calendar requirements** FR1.1 User can access the calendar through a login page. FR1.2 The calendar should be able to give warnings. FR1.3 The calendar should show the entire day, week, month and year. FR1.4 The user should be able to click on different appointment to acquire further details about their journey.	NFR1, NFR2 & NFR3	T1, T11, T12, T13	High

FR2	G2 & G3	**Customize transportation preferences** FR2.1 Support of walking, biking (own, or shared), bus, train, tram, taxis, driving (own or shared) FR2.2 A user can deselect transportation options FR2.3 A user have the possibility to give constraints on total time of travel FR2.4 A user can specify maximum walking distance FR2.5 A user can specify timeslots for using public transportation. FR2.6 A user can select the possibility to travel alone or with friends/family. FR2.7 A user can limit the number of changes in transport. FR2.8 A user should specify if walking or biking in rain is acceptable. FR2.9 A user should have the possibility to select an environmentally friendly travel, as specified in FR3.1. FR2.10 The application should give the user the possibility to prioritize their transport favorite, as specified in FR3 FR2.11 A user can in default settings create their own starting point for calculation of journey	NFR1	T2, T11	High
FR3	G3	**Prioritize their transportation favorites** FR3.1 The application will, based on carbon footprint, provide the most environmental friendly traveling option. FR3.2 The application should get info from the travelling companies to calculate the ticket price. FR3.3 The application should get information from a weather service to have knowledge about optimal travel according to the weather. FR3.4 If the user does not want to prioritize the standard setting will be to find the quickest travel.	NFR1	T3, T11	Medium

FR4	G4	**Creating and deleting new appointments in the calendar** FR4.1 The user can create new appointments. FR4.1.1 If meetings collide see FR6. FR4.1.2 If the traveling time between meetings is too short, a warning message should occur. FR4.1.3 If meetings are created at locations that are unreachable see FR9. FR4.1.4 The user can specify, when an event is created, the address to travel from and too. FR4.1.5 The user can specify the event time and duration FR4.1.6 The user can click 'calculate path' and the application will suggest a journey, see FR7 FR4.2 If a meeting require more traveling time than expected, the app should notify the user. FR4.3 The user should be able to delete appointments.	NFR1 & NFR3	T4, T13	High
FR5	G5	**Creating breaks** FR5.1 The user should be able to specify that lunch must be possible every day between 11:30 - 14:30. FR5.1.1 The application will always make room for a break in the timeslot. FR5.1.2 The user should be able to specify the length of the lunch. FR5.2 The user should be able to add additional coffee breaks. FR5.2.1 The length of the break has to be similar during the day, but can be specified by the user. FR5.2.2 The application will provide space for breaks in the calendar. FR5.2.3 The number of breaks can be specified by the user.	NFR1	T5, T11	Medium
FR6	G6	**Colliding events** FR6.1 If two meetings collide, a specific warning message will occur. FR6.2 If a meeting and a break collide, a specific warning message will occur. FR6.3 The application will give the user the possibility to change the appointment, finding a suitable space in the calendar.	NFR1	T6, T11	High

FR7	G7	**An accurate path description** FR7.1 The application will based on the client's preferences use available API's to calculate shortest travel. FR7.2 The application should give the client several options regarding different transportation	NFR1	T7, T11	High
FR8	G8	**Purchase tickets for public transportation** FR8.1 The application should allow clients to buy public transportation tickets when a bus, train or tram will be taken. FR8.2 The client will have the possibility to add information about having a day/week/season pass. FR8.3 The application should also provide information about the nearest car or bake of a vehicle sharing system. FR8.4 The application should use the client's payment information to order tickets from the different public transportation companies.	NFR3 & NFR1	T8, T11, T13	Medium
FR9	G9	**Unreachable destination** FR9.1 If a destination is unreachable a warning message will occur. FR9.1.1 If the destination is unreachable because of strike, the application should provide alternatives. FR9.1.2 If the destination is out of reach by public transport a warning message should occur. FR9.2 If the destination is not found, a warning message should occur.	NFR1	T9, T11	High
FR10	F10	**Creating profile** FR10.1 The client should have the possibility to create a new profile to access the system. FR10.2 The client should be able to log in with their facebook or google account. FR10.2.1 The client can login without facebook, but then needs to provide certain information. FR10.3 The client should be asked to leave payment information, so the application can buy tickets. FR10.3.1 The client should be able to proceed using the app without having to leave payment information.	NFR3, NFR1 & NFR2	T10, T11, T12, T13	High

For further detailed information about different interactions between the user and the involved systems please refer to the sequence diagrams found in the appendix.

3.3 Performance requirements

Support of access points: The application is intended to be general and can be used in all large cities. However this projects' main focus is Milan.

For the access and storage of the service we will use One.com as hosting service.

According to their contract we will have the power of:

- Unlimited bandwidth
- 25 GB of storage (counting database and files)
- 512 MB RAM
- 1 CPU

We make the following estimations about our users and content:

- All files, excluding the database, will take up maximum 100 MB of storage
- If needed, every past event can be removed
- One event takes up an average of 2.5 KB of data in the database
- An average user have about 10 upcoming events

By these estimations we can calculate an average of how many concurrent users we can support and also how many we can have in total.

We give our system a maximum of 699 050 total users given from the following calculation, we give the database 20GB (20971520 KB) to work with. Every user uses 30KB (events, user info and settings) also including indexing.

We set the threshold for feedback to 1 second. This means that every action which need a connection to the database or website needs to be reported back to the user within one second, such as when clicking on an event, the new page should open within said threshold. We do however want to differentiate between the feedback of an action and the finished state. We set the finished state to be reached within 5 seconds, this includes loading of routes for the event and calculation of the weather. In conclusion this means that when opening an event it should take less than one (1) second to open the new page and no more than an additional four (4) seconds to load the content of said page.

Since our web host does not specify the power of the CPU other than we have access to only one it is hard to calculate the amount of concurrent users. We do however get a set number on the RAM, which is 512 MB.

The RAM needs to hold the tables from the database and the variables from the scripts running when calculating events. We first reserve up to 50MB of RAM, leaving 462MB RAM, and 3.0% of processing power to the background work of our system, which is not affected by the amount of current users.

By our estimation we can reserve up to 462 MB, 473 088 KB, of RAM for the users. We estimate our average user to take 50KB RAM and 0.06% of processing power. This gives us an maximum of 1 616 concurrent users due to the lack of processing power. If our bottleneck would be RAM we could support up to 9 461 concurrent users.

User queries based on served users - see section software system attributes for more info

3.4 Design constraints

3.4.1 Standards compliance

The application will only use the standardised web functions. The only deviation may be the use of JavaScript, which some users may have blocked by default but when they activate it for the website it should run as expected.

3.4.2 Hardware limitations

The system will be available as a web application which should run on the most common browsers of today, Chrome, Mozilla and Safari. The functions of which the system uses are very lightweight (how lightweight?) and thus any hardware that can run a web browser with the latest update should be able to run the application.

3.4.3 External API's limitations

The system will be dependent on external open API's to produce an optimal journey for the user. If the travel company does not have an external open API, we cannot use that traveling option in our journey planner, and therefore will not be able to give the best traveling solution. However, the system will deliver the optimal solution given this constraint.

3.5 Non Functional Requirements: Software system attributes

3.5.1 NFR1 Reliability

NFR1.1 Access time: The time to access the calendar, from login to complete view of schedule, should be under 2 seconds. The system should be able to produce fast outputs, due to small data packages that needs to be processes.

NFR1.2 Response time: The time for the system to respond should be immediate, as there is no large data files that needs processing.

- NFR1.2.1: Ticket purchase: Regarding response time for other sites when our application sends the user beyond our application we cannot ensure a similar response time.

NFR1.3 Delivery of correct data: The system should continuously check for errors with a proper testing system, to produce correct data outputs at all times. If error occurs there should be a mechanism for error message and report the failure.

3.5.2 NFR2 Availability

NFR2.1 Uptime: The system is planned to be located on a web host called One.com which does have a uptime guarantee for its customers, however they do have a registered uptime of about 99.99

3.5.3 NFR3 Security

NFR3.1 Site security issues

- NFR3.1.1 SQL injection: The system should always use commands to remove the threat of SQL injection since all of the data will be stored in SQL-database.
- NFR3.1.2 Cross- site scripting XSS - The system should use commands to avoid XSS to ensure to attackers hijack their session, deface the website or redirects the user to other malicious websites.

NFR3.2 Personal security issues

- NFR3.2.1 Having secure storage of profiles: No unnecessary data of the user should be stored in the case of an attack which results in stolen information.
- NFR3.2.2 Secure passwords: The user's password should also be stored with an MD5-encryption, in the unfortunate case of intrusion the passwords of the users are still held secret.

- NFR3.2.3 Having secure and personal profiles: Since some users may have private events planned which they do not want anyone to know the system should also take extra measures whereas to secure that the event is displayed for the right user.

3.5.4 NFR4 Maintainability

NFR4.1 Small modules: The system should contain of small modules to easily add or remove parts, to ensure scalability in the future. In addition will a smaller module more easily be updated. With the small modules will a clear documentation make it easier for an administrator to locate the necessary parts of which to adjust to achieve the desired update.

3.5.5 NFR5 Portability

NFR5.1 Web based application: By making it a web based application it would be very easy to port it onto other systems, such as implementing it directly into phones or computers.

For example when porting to a Android application the changes of the current files could be kept to a minimum, if changed at all, while the mobile application only would need to send HTML-requests to function.

4 Formal Analysis Using Alloy

4.1 Alloy Model discussion

As part of this Requirement Analysis and Specification Document we created an alloy model to visualize the world that is generated by Travlendar. The purpose of the model was to get an overview over the structure of the application and the different layers that the user will interact with. The model is intended more as an illustration of what the application will accomplish and produce when up and running as opposed to how it will actually work on the back-end. Therefore the focus has been on the components inherent to the internal software and external interactions has been omitted in this model. Found below is the code for the model, the execution results along with an example world obtained from running it.

As we conceptualized the structure of the application the alloy model assisted us in ensuring that the logic of the relationships between our elements was intact by seeing that the world our model generated was feasible. The usage of alloy helped us to visualize the sheer numerous potential combinations of components a calendar could include, as well as how interrelated these components are vertically. For example, if a day is to populate within a calendar it has to include an event, and for an event to be successfully created it is required to be reachable by having a generated journey (composed of one or multiple paths). Therefore, for any calendar that is not empty there must be at least one instance of every type of component. Beyond this, at any level of this hierarchy of related components, multiple "children" can belong to the same "parent".

4.2 Model

```
pred show() {
        #Database = 1
        #User = 3
        #Event = 6
        #Journey = 10
        #Path = 12
}

sig Database {
        users: seq User
}

fact noDuplicateUsers {
        all d: Database | not d.users.hasDups
}


sig User {
        calendar: one Calendar,
        settings: one Settings
}

fact userMustHaveDB {
        Database.users.elems = User
}

sig Settings {
```



```
}

fact settingsMustHaveUser {
        User.settings = Settings
}

sig Calendar {
        days: seq Day
}

fact calendarMustHaveUser {
        User.calendar = Calendar
}

fact allUsersHasSeperateCalendar {
        all u1,u2: User | u1 != u2 => u1.calendar != u2.calendar
}

fact allUsersHasSeperateSettings {
        all u1,u2:  User | u1 != u2 => u1.settings != u2.settings
}

sig Day {
        events: seq Event
}{
        #events > 0
}

fact dayMustHaveCalendar {
        Calendar.days.elems = Day
}

fact everyDayBelongsToOneCalendar{
        all c1,c2: Calendar, e1: c1.days.elems, e2: c2.days.elems |
           c1 != c2 => e1 != e2
}

fact daysAreUniqueForACalendar {
        all c: Calendar | not c.days.hasDups
}

sig Event {
        journeys: seq Journey
}{
        #journeys > 0
}

fact eventMustHaveDay {
        Day.events.elems = Event
}
```



```
fact everyEventBelongsToOneDay{
        all d1,d2: Day, e1: d1.events.elems, e2: d2.events.elems | d1
            != d2 => e1 != e2
}

fact eventsAreUniqueForADay {
        all d: Day | not d.events.hasDups
}

sig Journey {
        paths: seq Path
}
{
        #paths > 0
}

fact journeyMustHaveEvent {
        Event.journeys.elems = Journey
}

fact everyJourneyBelongsToAEvent{
        all e1,e2: Event, j1: e1.journeys.elems, j2: e2.journeys.
           elems | e1 != e2 => j1 != j2
}

fact journeysAreUniqueForAEvent {
        all e: Event | not e.journeys.hasDups
}

sig Path {
}

fact pathMustHaveJourney {
        Journey.paths.elems = Path
}

fact everyPathBelongsToAJourney{
        all j1,j2: Journey, p1: j1.paths.elems, p2: j2.paths.elems |
           j1 != j2 => p1 != p2
}

fact pathsAreUnqiueForAJourney {
        all j: Journey | not j.paths.hasDups
}

run show for 12
```


4.3 Model Results

```
Executing "Run show for 12"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  141127 vars. 5424 primary vars. 261303 clauses. 1357ms.
  Instance found. Predicate is consistent. 2924ms.
```

Figure 7: Alloy results

4.4 Alloy World

Figure 8: Alloy model

5 Effort Spent

5.1 Effort Spent

5.1.1 Axel Rebner

- 4/10/17: Kick-off and planning - 1,5 hours
- 8/10/17: Overview, layout and mindmap - 3 hours
- 10/10/17: Latex structure, text writing and sketch UML - 3 hours
- 15/10/17: Use Case, Sequence Diagram - 3,5 hours
- 15/10/17: Class Diagram, Statechart - 1 hours
- 18/10/17: Planning and revision of document - 3 hours
- 22/10/17: Working with the Alloy model - 5 hour
- 24/10/17: Introduction, Product Perspective and Alloy - 1,5 hours
- 25/10/17: Reviewing Document - 4 hours
- 25/10/17: Assumptions - 2 hours
- 26/10/17: Final review - 1,5 hours

5.1.2 Caroline Kverne

- 4/10/17: Kick-off and planning - 1,5 hours
- 10/10/17: Requirements start and overall structure - 1,0
- 11/10/17: Functional requirements - 4,5 hours
- 16/10/17: NFR and rest of requirement section - 3,0 hours
- 18/10/17: Planning and revision of document - 3 hours
- 23/10/17: Test Case and Appendix - 4 hours
- 25/10/17: Reviewing Document - 4 hours
- 25/10/17: Assumptions - 2 hours

5.1.3 Joakim Jerkenhag

- 4/10/17: Kick-off and planning - 1,5 hours
- 8/10/17: Overview, layout and mindmap - 3 hours
- 10/10/17: Latex structure, text writing and sketch UML - 3 hours
- 15/10/17: Use Case, Sequence Diagram - 3,5 hours
- 18/10/17: Planning and revision of document - 3 hours
- 18/10/17: Reformating requirements to LaTeX - 1 hour

- 22/10/17: Working with the Alloy model - 5 hour
- 23/10/17: Designing mockups - 1,5 hours
- 24/10/17: Finishing mockups - 1,5 hours
- 25/10/17: Reviewing Document - 4 hours
- 26/10/17: Final review - 1,5 hours

6 Appendix

6.1 Sequence Diagrams

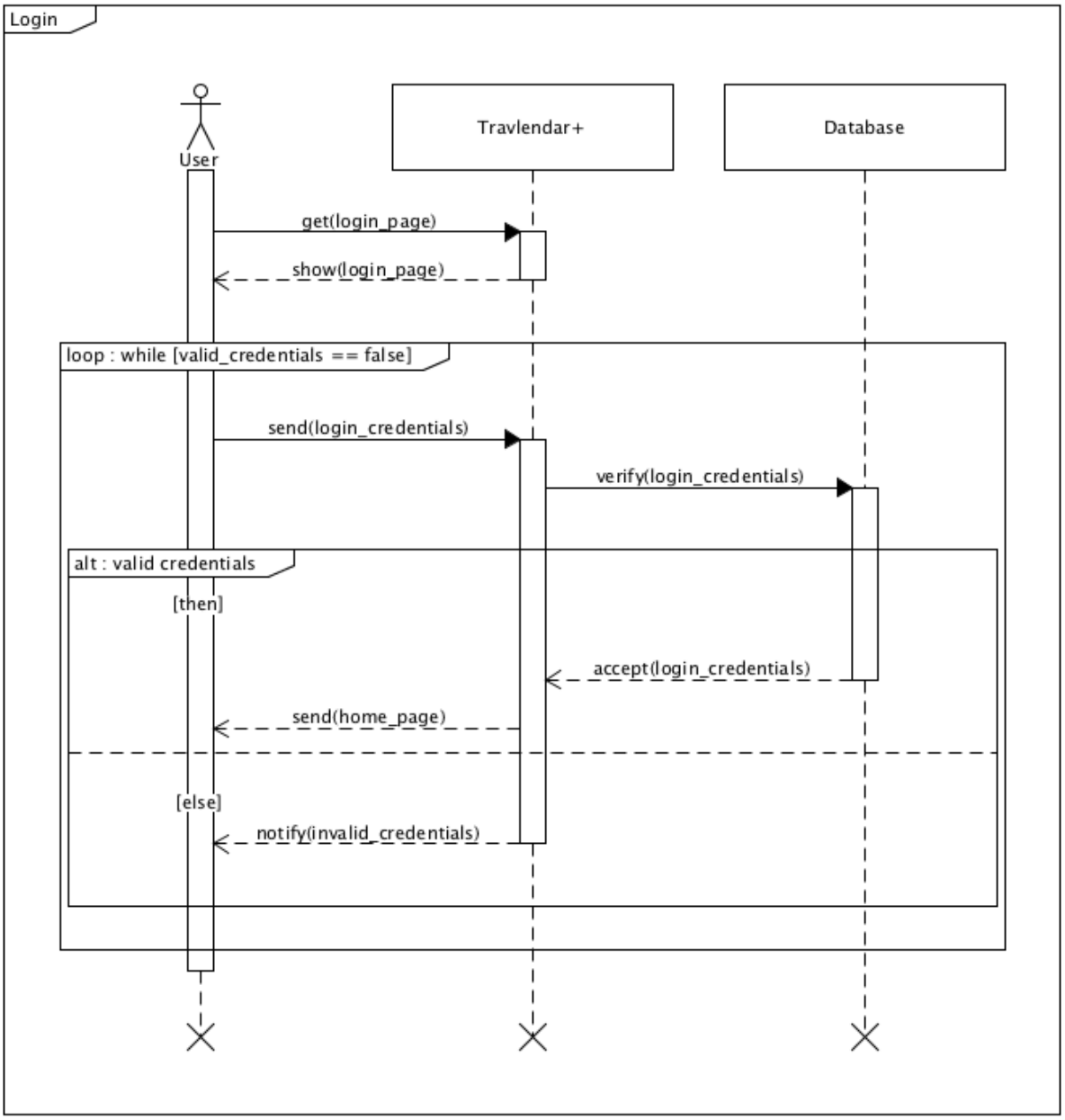

Figure 9: Sequence Diagram - User Log in

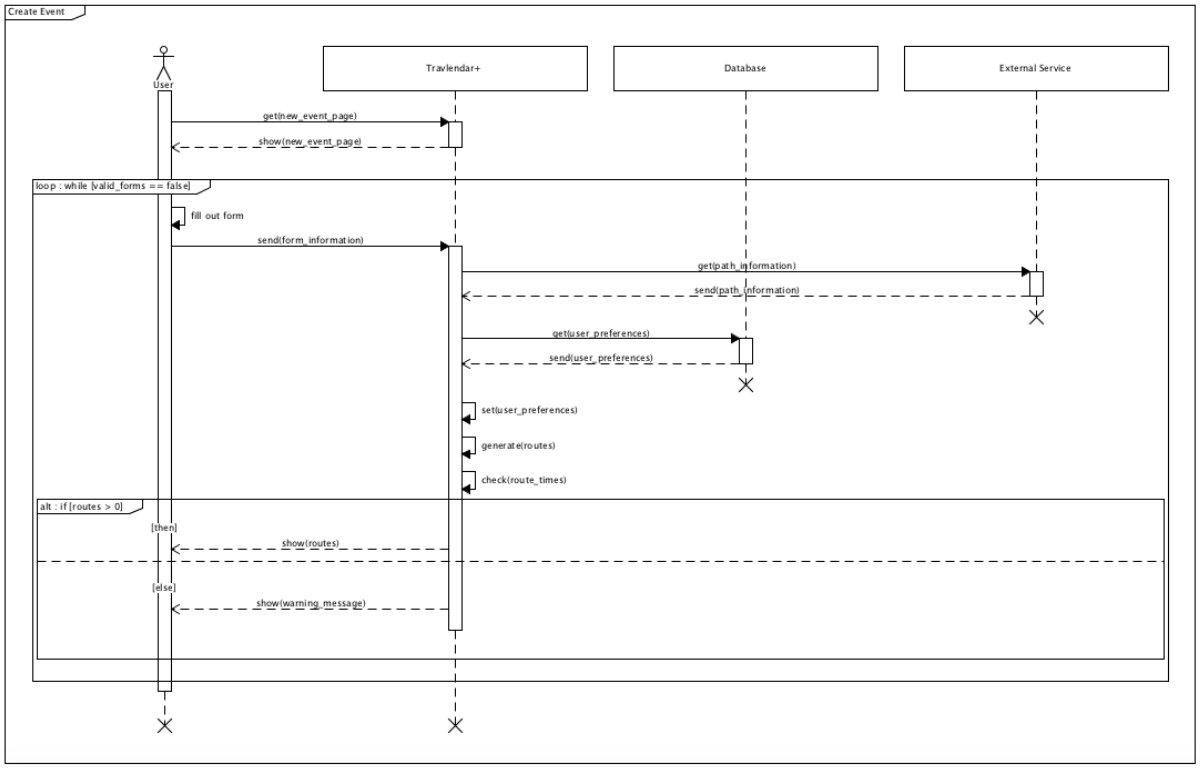

Figure 10: Sequence Diagram - Event Creation

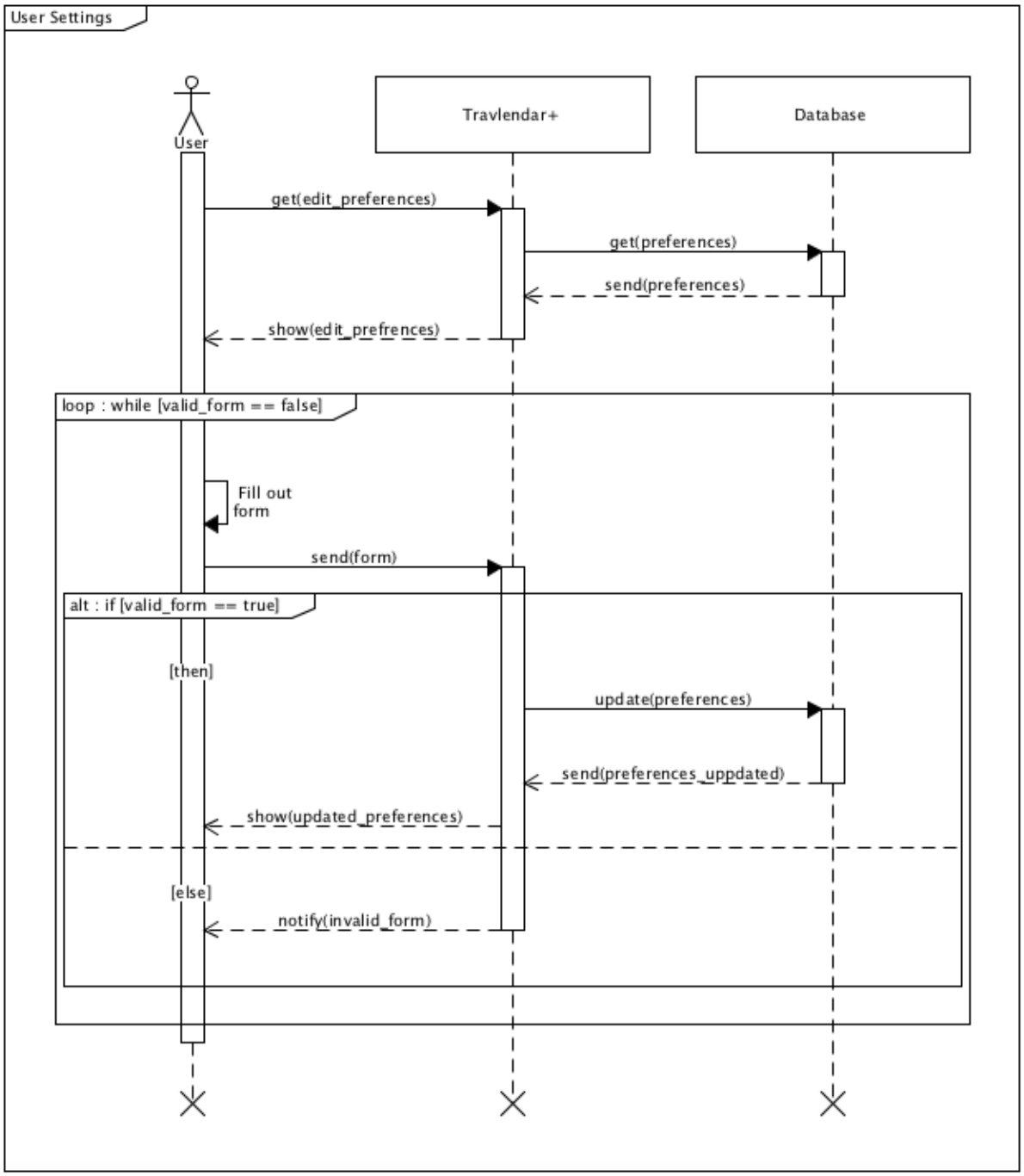

Figure 11: Sequence Diagram - User Settings

6.2 Test Case Description

As our team has decided to do the implementation we decided to add a part about the testing of requirements to our RASD. This is to ensure that our requirements are testable and verifiable. This test case document will describe the evaluation of criterias for completeness of the coherent requirement and present some test cases for the specific requirement. It is not complete, as not every test case for every requirement is listed, however it gives an overall understanding of how we will proceed during our implementation. All assumptions are described previously in the RASD document introduction.

TestID with description

- T1 Calendar

  Completeness criteria: The system has a functional login page, that redirects the user to the calendar. The main page of the calendar shows the upcoming events. The user should be able to click on different appointments to acquire further details about their planned events.

  T1.1 LOGIN PAGE

  Description: A registered user should be able to successfully login at travlendar+.com

  Precondition: The user must already be registered with an email address and password.

  Assumption: A supported browser is being used.

  Input:

  1. Navigate to travlendar+.com
  2. In the email field type registered email for user
  3. In the password field type registered password for user
  4. Click 'sign in'

  Output: The main calendar page should load, showing the upcoming events for the specific user.

  T1.2 SHOW EVENT

  Description: The user should be able to click on an upcoming event to see detailed information about traveling journey

  Precondition: The user must be logged in.

  Assumption: A supported browser is being used. Input:

  1. Click on any upcoming event

  Output: The calendar should show the event with the journey planned (description of event, travel time, transportation description, possible warning message), and also present an option to change transportation to/from event.

- T2 Customize transportation preferences

  Completeness criteria: The system has a functional system settings page. The system settings should be able to support all the transportation methods listed. Also the user should be able to specify their preferences regarding way of travel, and also add breaks. In addition it should be possible to be redirected to other transportation companies sites' to purchase tickets.

  T2.1 SUPPORTED TRANSPORTATION

  Description: A user should see all transports available (all are standard activated) and then independently choose which one to deactivate.

  Precondition: The user must be logged in.

  Assumption: A supported browser is being used.

  Input:

1. Navigate to settings found in the corner of the calendar main page
2. Click on transportation
3. Select from a list which transportation methods that suits you
4. Click ‘save’

Output: The settings will be saved, and the calendar will only use the transportation methods activated to search for an optimal journey.

- T3 Prioritize transportation favorites

  Completeness criteria: The system will calculate carbon footprint, calculate ticket price and get information from a weather service to give the user a correct estimate on their preferred transportation type.

  T3.1 PRIORITIZATION OF TRANSPORT MEANS

  Description: A user can choose between three prioritized way of travel. So that the application will present the most optimal journey.

  Precondition: The user must be logged in.

  Assumption: A supported browser is being used.

  Input:

  1. Navigate to settings found in the corner of the calendar main page
  2. Click on preferences
  3. Select the degree of importance of environment, price and weather.
  4. Click ‘save’

  Output: The settings will be saved, and the calendar will only present the optimal journey based on the degree of importance. If all of the preferences is maximized, the journey with shortest travel time will be presented.

- T4 Creating and deleting new appointments in the calendar

  Completeness criteria: The system should have an add and delete function of events in the calendar. Also when creating the event, if there is a collision, unreachable destination, collision in break or similar: a warning message should be present.

  T4.1 CREATE NEW EVENT

  Description: A user should be able to create new events and get a warning message if any problem occurs.

  Precondition: The user must be logged in.

  Assumption: A supported browser is being used.

  Input:

  1. At the main calendar page, click ‘+’ that is add event
  2. Type in ‘description’
  3. Type in event ‘start’ and ‘finish’ time
  4. Click calculate journey
  5. Select journey from the list presented
  6. Click ‘save’

  Output: The event will be created, and shown in the main calendar.

- T5 Creating breaks

  Completeness criteria: The system has a break system that independently make sure your day always includes a break with your customizable length.

T5.1 CREATE LUNCH BREAK

Description: A user should be able to specify that in a customizable time interval, the user will have a break of customizable time.

Precondition: The user must be logged in.

Assumption: A supported browser is being used.

Input:

1. Navigate to the settings menu
2. Click on ‘preferences’
3. Activate ‘Breaks’
4. Select time interval for your break
5. Select break duration
6. Click ‘save’

Output: The break setting will be saved, and an the calendar will always have an opening in the specified time interval with a duration of your break.

- T6 Colliding events

  Completeness criteria: The system has a warning system that shows a little warning triangle next to the event if there is a collision in events or collision in break and event.

  T6.1 COLLIDING EVENT Description: A user shoul d be able to create a new event and get a warning triangle if the event collides with another event.

  Precondition: The user must be logged in, and already have an event created.

  Assumption: A supported browser is being used.

  Input:

  1. Click ‘+’ add event
  2. Type in ‘description’
  3. Type in ‘start’ and ‘end’ time
  4. Click on the warning triangle
  5. Get a warning ‘colliding event, please change timeslot’
  6. Change time slot
  7. Calculate journey
  8. Click ‘save’

  Output: The event will be created with no collision in the calendar. If the user decides not to change the timeslot, a little warning triangle will appear next to the event.

- T7 An accurate path description

  Completeness criteria: The system has a shortest path system that always show the shortest journey regarding travel time.

  T7.1 SHORTEST TRAVEL TIME

  Description: A user that calculates journey should get the most optimal travel based on their preferences and time.

  Precondition: The user must be logged in and have created an event.

  Assumption: A supported browser is being used.

  Input:

  1. Click on an upcoming event

2. See the list of journeys presented with increasing time
3. Either click ‘save’ or change journey.

Output: The system should present a list with increasing time, however it is always possible to change journey later.

- T8 Purchase tickets for public transportation

  Completeness criteria: The system will redirect you to other transportation companies sites’ to purchase tickets.

  T8.1 PURCHASE TICKETS

  Description: A user should be able to purchase ticket for the selected journey.

  Precondition: The user must be logged in and created an event that requires a valid ticket.

  Assumption: A supported browser is being used.

  Input:

  1. Click on the specified event
  2. Choose ‘buy ticket’ (The system will redirect the user to another site and there the user will fulfil the payment. The system will then redirect the user to the specified event)
  3. Click ‘save’

  Output: The event will now appear in the calendar as usual and you will be able to do the journey with a valid ticket.

- T9 Unreachable destination

  Completeness criteria: The system has a warning system that shows a little warning triangle next to the event if there is an unreachable destination.

  T9.1 UNREACHABLE DESTINATION

  Description: A user should be able to create a new event and get a warning message if the event destination is unreachable.

  Precondition: The user must be logged in.

  Assumption: A supported browser is being used.

  Input:

  1. Click ‘+’ add event
  2. Type in ‘description’
  3. Type in ‘start’ and ‘end’ time
  4. Calculate journey
  5. Click on the warning triangle
  6. Get a warning ‘unreachable destination please change place’
  7. Change destination place or change event
  8. Click ‘save’

  Output: The event will be created with a valid destination.

- T10 Creating profile

  Completeness criteria: The system should have a ‘create profile’ possibility to be able to login.

T10.1 CREATE NEW PROFILE

Description: A user should be able to create a new profile to access the system.

Precondition: The user must have a valid email account.

Assumption: A supported browser is being used.

Input:

1. Navigate to travlendar+.com
2. In the sign-in box click ‘new user’
3. In the email field type email
4. In the password field type password
5. Click ‘register’

Output: The system will add the new user to the database and the user will now be able to sign in and use Travlendar+.

- T11 Reliability

  Completeness criteria:

  1. The system should access calendar, from login to calendar, in <2 seconds. Response time should be immediate: when doing a command the system should respond in <0.1 second.
  2. The system should deliver correct data 95

- T12 Availability

  Completeness criteria:

  1. The system should have an uptime of 99.99

- T13 Security

  Completeness criteria:

  1. The system should have no SQL injection.
  2. The system should have no XSS.
  3. The system will not store unnecessary data about the user
  4. Passwords will have MD5- encryption

6.3 Travlendar+ further improvements

Adding a geoposition function that keeps track of the users location for added versatility when getting to a meeting.

Branching out the options for travelling, allowing compatibility with more environments. A taxi ordering functionality could be envisioned.

References

[1] Software Engineering: Principles and Practice, Hans van Vliet (c) Wiley, 2007.

[2] Fundamentals of software engineering (2. ed.), Prof. Dr.-Ing. Axel Hunger, April 2009.

