Politecnico di Milano
AA 2017-2018

Computer Science and Engineering

Software Engineering 2 Project

TRAVLENDAR+

RASD

Requirements Analysis and Specifications Document

Version 1.0 - 29/10/2017

Pelinsu Çelebi-893636
Yusuf Yiğit Pilavcı-892973

Contents

1. INTRODUCTION 3
   A. Purpose: 3
      A.1 Goals 3
   B. Scope: 4
   C. Definitions, Acronyms, Abbreviations 4
      C.1 Definitions 4
      C.2 Acronyms 4
   D. Revision history 5
   F. Document Structure 5
2. OVERALL DESCRIPTION 5
   A. Product perspective: 5
   B. Product functions: 5
      B.1 Event Management 5
      B.2 Trip Management 6
   C. User characteristics: 6
   D. Assumptions, dependencies and constraints: 6
      D.1. Domain Assumptions and Dependencies 6
      D.2 Constraints 7
3. SPECIFIC REQUIREMENTS: 8
   A. External Interface Requirements 8
      A.1 User Interfaces 8
      A.2 Hardware Interfaces 11
      A.3 Software Interfaces 11
   B. Functional Requirements: 12
      B.1.1 [G_1]. Allow users to create new account 12
      B.1.2 [G_2]. Allow users to become logged in to existing account after entering his/her credentials. . 12
      B.1.3 [G_3]. Allow users to view his/her calendar. 12
      B.1.4 [G_4]. Allow users to manage events in calendar 12
      B.1.5 [G] Allow users to manage personal mobility preferences 13
      B.1.6[G_6]. Allow users to know mobility options which feasible travelling duration under user preferences, weather and traffic constraints. 13

B.2 Use Cases 14
B.2.1 Scenarios 14
B.2.2 Use Case Descriptions 16
B.2.3 Use Case Diagram 24
B.3 Sequence Diagrams 26
B.4 Statechart Diagram 31
B.5 Class Diagram 31
C. Performance Requirements 32
D. Design Constraints 32
D.1 Hardware limitations 32
E. Software System Attributes 32
E.1 Reliability 32
E.2 Availability 32
E.3 Security 32
E.4 Maintainability 32
E.5 Portability 32
4. FORMAL ANALYSIS USING ALLOY: 33
4.1 MODEL 33
4.2 GENERATED WORLD 37
5. EFFORT SPENT: 37
5.1 Hours of Work 37
6. REFERENCES: 38
7. APPENDIX: 38
7.1 Used Tools 38

1. INTRODUCTION

A. Purpose:

This document presents the Requirement Analysis and Specifications for Travelander+ application. This document aims to define functional and non-functional requirements, detailed analysis of the environment and constraints, realization of customer needs, and characteristic of users and their use case for the Travelander+ application. Typical audience of this document are the developers who aims to implement this system.

Travelander+ application provides an enhanced calendar with custom scheduling options to create appointments and offers travel planning assistance in between the arranged appointments for its possible users. The system enables its users to add/edit appointments, choose mobility preferences and to offer them the best options for travelling from one appointment to another by regarding environmental conditions (such as traffic, weather etc.), user specific situations and preferences and efficient usage of time and other available resources.

A.1 Goals

Users:

[G_1].Allow users to create new account
[G_2].Allow users to become logged in to existing account after entering his/her credentials.
[G_3].Allow users to view his/her calendar.
[G_4].Allow users to manage events in calendar

- [$G_{4.1}$].Allow users to add new event to the user if selected time slot is available.
- [$G_{4.2}$].Allow users to receive a warning if the selected time slot is occupied or not feasible by considering place and time constraints.
- [$G_{4.3}$].Allow users to edit his/her events.
- [$G_{4.4}$].Allow users to delete his/her events.
- [$G_{4.5}$].Allow users to add customized break time with a certain duration defined in a certain time interval
- [$G_{4.6}$].Allow users to pin particular event by regarding importance of the event.
- [$G_{4.7}$].Allow users to receive alerts for the pinned events.
- [$G_{4.8}$].Allow users to add periodic events in daily, weekly and monthly basis.

[G_5].Allow users to manage personal mobility preferences.

- [$G_{5.1}$].Allow users to enter personal mobility preferences.
- [$G_{5.2}$].Allow users to choose predefined mobility preferences such as preference lists enforcing minimizing carbon prints, not driving car, not using public transportation etc.
- [$G_{5.3}$].Allow users to activate or deactivate particular mobility options

[G_6].Allow users to know mobility options which minimize travelling duration under user preferences, weather and traffic constraints.

- [$G_{6.1}$]. Allow users to know the optimum mobility option travelling from current location.
- [$G_{6.2}$]. Allow users to know the optimum mobility option travelling from specific location.

B. Scope:

Travelander+ will be the mobile and web application that enables to manage appointments and find best mobility options for its users. Its users can be everyone who needs to plan his/her long or short term schedule. Since the system is able to take information from various sources such as maps, traffic analysis on Internet, weather forecasting, public transportation etc., Travelander+ is able to adapt the appointments and mobility options for maximizing efficiency on time and minimizing the latency and usage of other resources.

C. Definitions, Acronyms, Abbreviations

C.1 Definitions

- Event: Any appointment or customized break by the user
- Break: An event with a flexible duration assigned between the selected start and end time
- Periodic Event: An event that is repeated with a selected frequency. (e.g. break, gym)
- Activated/Deactivated: An activated mobility option is listed in the preference list with the given constraints and priority, whereas a deactivated mobility option is not listed in the preference list at all or only on the restricted times given by the user or based on the information received from APIs (e.g. walk and bike deactivated on rainy or snowy weather).
- Restricted Time Interval: A time interval set by the user for a certain mobility option to be temporarily deactivated.
- Distance Limit: A maximum distance set by the user to deactivate the certain mobility option for any larger distance.
- Preference List: A list constructed by the user with the Travlendar+'s given mobility options by giving them priorities to be selected or to activate-deactivate them based on the user's abilities and preferences.
- Default List: Ready to use preference lists offered by the Travlendar+.
    - Minimize Carbon Footprint
    - Minimize Expenditure
- Reachable/Unreachable: An attribute of the location of the event to be added which shows if the event is actually reachable by any means of mobility based on the already scheduled events, and if it is unreachable forbids the selection of that location.
- Public Transportation Information Provider API: General name for the public transportation APIs. Its purpose is to provide official information about the public transportation of the current city.

C.2 Acronyms

- RASD : Requirement Analysis and Specification Document
- PTIP: Public Transportation Information Provider
- API: Application Programming Interface
- JDBC : Java Database Connectivity

D. Revision history
Version 1.0

F. Document Structure

This document consists of 6 parts as table of content indicates:

1. Introduction: The problem definition is introduced and aims of both this document and desired system are explained. Goals of the system are listed with mapping abbreviations. Basically, this part provides the introductory information for giving full understanding of rest of the document
2. Overall Description:
3. Specific Requirements: As the title indicates, this part lists all the functional requirements of the system. Besides, various interfaces (user, software and hardware interfaces) of the application is explained in this section.
4. Formal Analysis Using Alloy: In this section the built alloy model is given with explanatory comments on what the model represents, what is tested via this model and proved and how it is related to the goals and requirements that were set in the beginning.
5. Effort Spent: The tables of the time spent on each topic by each contributor of the project are given in this section.
6. References: The reference documents that were benefited from while constructing this document are given in this section.

2. OVERALL DESCRIPTION

A. Product perspective:

Travlendar+ is an application developed to be used by the citizens and visitors of the adapted city (i.e. Milan) to enhance their ability to create, track and manage their schedules, and to save them the time to arrange their trips between these scheduled events by providing them the best mobility option suggestions based on their preferences and time-distance limitations. The application benefits its users greatly by allowing them to create these feasible schedules which they can easily follow since any undoable event assignment is prevented automatically by the time and reachability constraints and by offering them the information on all transportation means of the city on a single platform.

B. Product functions:

In this part, the main functions of the proposed system are introduced and explained. By providing these essential functions, the system aims to satisfy the main needs of the described problem within given boundaries of both environment and system itself.

B.1 Event Management

Event management is one of the main functions of this system. It basically enables to user add/edit/delete appointments and customized breaks and, visualize and plan his/her schedule. By accomplishing these purposes, the event manager promises that;

- The user is never late for any appointment
- No overlapping event might occur
- Travelling durations are always taken account for planning schedule.

With these given necessities of the system, this functionality of the system solves the event management part of the problem.

B.2 Trip Management

This functionality gathers the most recent information and does the necessary calculations for offering best mobility options to its user by considering current location, weather and traffic situation, recent events and also user preferences and conditions. This functionality successfully able to communicate with external APIs for gathering information related to current location, maps, public transportation, weather forecasting, traffic etc. and merge them with user preferences and added personal events.

C. User characteristics:

Only active actor for this system is the user; who aims to obtain optimized schedule planning, mobility options and calendar services. System requires from the user registration for his/her personal account at the first place. After registration, user is needed to input his/her appointments and personal preferences for the operation of main functions of the system.

D. Assumptions, dependencies and constraints:

D.1. Domain Assumptions and Dependencies

In the problem description that is proposed by the customer, some points related to the environment and user are not clearly identified. Since this lack of identification may lead to ambiguities, we needed to make assumptions on these points during development process. Our proposed system successfully operates on the environment that these assumptions hold:

1. During application run, operating device always receives Internet connection.
2. Information about weather forecasting and traffic conditions are published on related APIs.
3. Resources for the updated weather forecasting and traffic conditions always provide accurate information.
4. Information of public transportation can be accessed by Public Transportation Information Provider(PTIP) API for the current city.
5. Public transportation vehicles are assumed as punctual with their published programs.
6. Provided information on working hours and stops of public transportation means are always accurate.
7. Users who have car active in their preference list have driver license for the preferred vehicle.
8. Users are able to ride bike when biking is activated as mobility option.
9. Users are assumed to walk in average speed.
10. Users do not have any disability related to walking if walking is activated as mobility option.
11. During mobile application operation, GPS is on and at working status while current position is needed.
12. GPS API always provides accurate location information.

13. Each user account has only one calendar.
14. Users always know and enter correctly their credentials.
15. Usernames are unique and consists of only alphanumeric characters.
16. No user can be at different places at the same time.
17. Users accurately enter location addresses and date-time of the events to the system.
18. Break duration is always equal or smaller than given time interval for it.
19. Start time of events always are before the end time of events.
20. All mobility options have distinct priority number in all user preference lists.

D.2 Constraints

D.2.1 Regulatory Policies

Proposed system requires user input as appointment time and location which are details of personal schedule and only uses this information for event and trip management purposes. Also, it asks for the user's permission in case of the usage of current location is needed and it is only used for increasing the quality of trip planning. It is not served for commercial or any other purpose.

D.2.2 Interfaces to other applications

System needs to communicate for up-to-date information collection of weather, traffic, public transportation. These operations are conducted with related APIs such as google Maps, Open Weather Map etc. Also, it needs to communicate and manage a database system for storing and updating personal user information such as credentials, recent schedule and events, preferences for mobility options. For this purpose, MySQL will be deployed

3. SPECIFIC REQUIREMENTS:

A. External Interface Requirements

A.1 User Interfaces

A.1.1 Login Screen and Main Menu

Figure 1. Login Screen

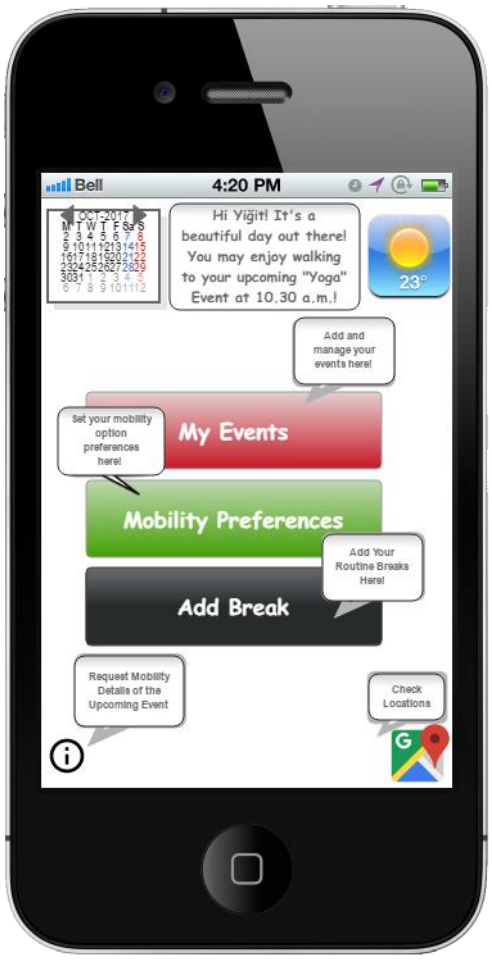


Figure 2. Main Menu Screen

A.1.2 My Events and Mobility Preferences Screen

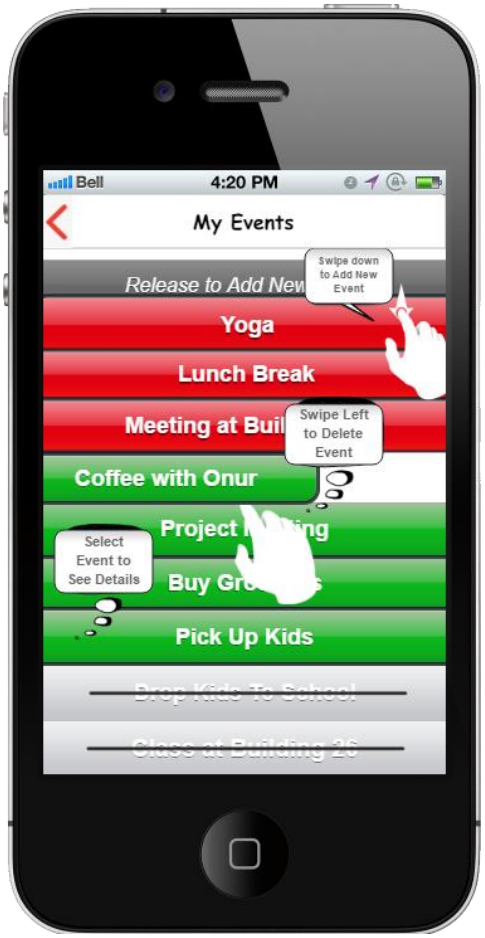


Figure 3. Event Details Screen

Figure 4. Mobility Preferences Screen

A.1.3 Calendar and Map View

Figure 5. Calendar View

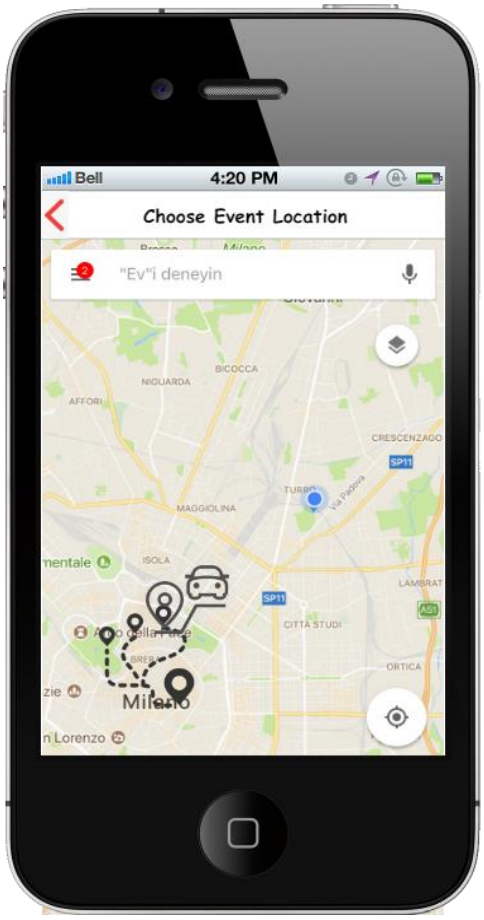


Figure 6. Maps View

A.1.4 Event Details and Customize Mobility Option Screen

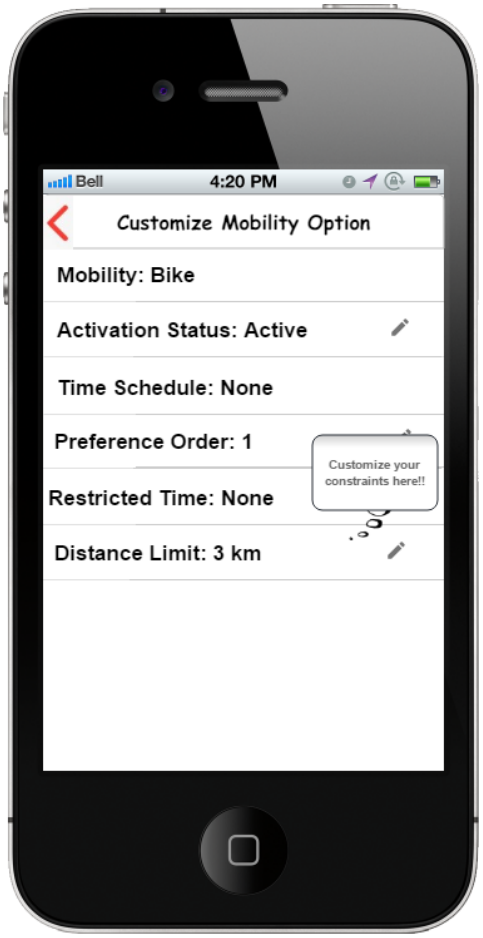


Figure 7. Customize Mobility Option Screen

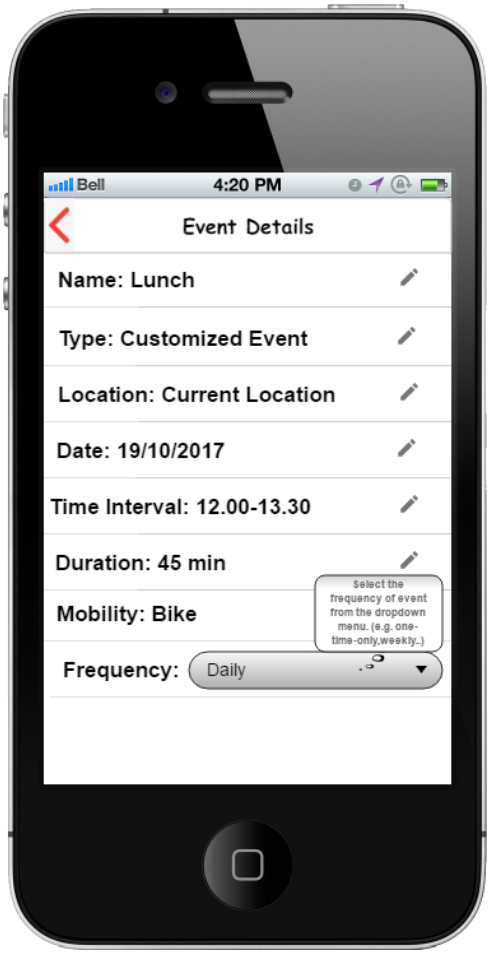


Figure 8. Event Details Screen

A.2 Hardware Interfaces

The hardware interface is needed to collect GPS data which is included in almost every mobile phone.

A.3 Software Interfaces

The software interfaces are mainly needed for collecting traffic, weather, public transportation data and this issue is mainly solved by related APIs. Another need for software interfaces arises in usage of databases for retrieval of user information. For this purpose, Java Database Connectivity (JDBC), which is another API that connects databases with applications, will be deployed with MySQL queries.

B. Functional Requirements:

B.1.1 [G_1]. Allow users to create new account

- [R_1]. The system must initiate new user registration with user input. During this process, the system must ask new credentials (username and password), mobile phone number and e-mail address to the user.
- [R_2]. The system must check whether the e-mail or mobile phone number already exist or not.
- [R_3]. The system must check whether the username is taken by another user or not.
- [R_4]. The system must check whether the username have written only with alphanumeric characters or not
- [D_{15}]. Usernames are unique and consists of only alphanumeric characters.

B.1.2 [G_2]. Allow users to become logged in to existing account after entering his/her credentials.

- [R_5]. The system must check whether the credentials are correct by querying in the database.
- [D_{14}]. Users always know and enters correctly their credentials.

B.1.3 [G_3]. Allow users to view his/her calendar.

- [R_6]. Each registered user must be able to log in to his/her personal account.
- [R_7]. The system must retrieve details the personal calendar of user from the database when it is required by the user.
- [D_{13}]. Each user account has only one calendar.

B.1.4 [G_4]. Allow users to manage events in calendar

- [R_6]. Each registered user must be able to log in to his/her personal account
- **[**R_8]. The system must be able to add event to calendar by registered user input and must check validity of calendar by controlling whether the selected time slot is available or not due to any reason.
- [R_9]. The system must be able to delete event from calendar by registered user input.
- [R_{10}]. The system must be able to edit an existing event from calendar by registered user input and must check validity of edited calendar by controlling whether the selected time slot is available or not due to any reason.
- [R_{11}]. The system must be able to pin events by user selection and must generate reminders or alarms for them.
- [R_{12}]. The system must be able to add events as periodic such as daily, weekly, monthly etc. based on user input.
- [R_{13}]. The system must be able to add breaks as special type of event that occurs in a time interval with smaller duration which are defined by the user.
- [D_{16}]. No user can be at different places at the same time.
- [D_{17}]. Users accurately enter location addresses and date-time of the events to the system.
- [D_{18}]. Break duration is always equal or smaller than given time interval for it.

- [D_5]. Public transportation vehicles are assumed as punctual with their published programs.

B.1.5 [G] Allow users to manage personal mobility preferences

- [R_6]. Each registered user must be able to log in to his/her personal account.
- [R_{14}]. The system must be able to get and save mobility preference list constructed by the user.
- [R_{15}]. The system must be able to propose to the user predefined mobility preference lists which aims to minimize carbon footprint or minimize travelling costs etc.
- [R_{16}]. The system must be able to deactivate or activate mobility options by user request.
- [R_{17}]. The system must be able to deactivate mobility options during their restricted time.
- [R_{18}]. The system must be able to get restricted time of mobility options from registered users or related APIs.
- [R_{19}]. The system must be able to get travel duration/distance limit about mobility options
- [R_{20}]. The system must be able to deactivate the mobility options if the travel duration is more than duration limit.
- [R_{21}]. The system must place only activated mobility options in registered user preference list.
- [R_{22}]. The system must decide on the mobility option of the user's preference list as chosen mobility option of related event based on the user given priorities and the mobility option's travel duration fitness between events.
- [D_7]. Users who have car active in their preference list have driver license for the preferred vehicle.
- [D_8]. Users are able to ride bike when biking is activated as mobility option.
- [D_{10}]. Users do not have any disability related to walking if walking is activated as mobility option.
- [D_{19}]. Start time of events always are before the end time of events.
- [D_{20}]. All mobility options have distinct order number in all user preference lists.

B.1.6[G_6]. Allow users to know mobility options which feasible travelling duration under user preferences, weather and traffic constraints.

- [R_6]. Each registered user must be able to log in to his/her personal account.
- [R_{14}]. The system must be able to get and save mobility preference list constructed by the user.
- [R_{23}]. The system must be able to get traffic and weather information from related APIs.
- [R_{24}]. The system must be able to get current location from GPS API when the current location is needed as starting point.
- [R_{25}]. The system must be able to deactivate unavailable mobility options due to weather, traffic and user preferences and constraints.
- [R_{26}]. The system must be able to delete deactivated mobility options from user preference list.
- [R_{27}]. The system must be able to suggest the mobility option which is the top of the user preference list.

- [R_{28}]. For car and bike mobility options, the system must be able to suggest possible shared bike or car applications.
- [D_{17}]. Users accurately enter location addresses and date-time of the events to the system.
- [D_5]. Public transportation vehicles are assumed as punctual with their published programs.
- [D_1]. During application run, operating device always receives Internet connection.
- [D_2]. Information about weather forecasting and traffic conditions are published on related APIs
- [D_3]. Resources for the updated weather forecasting and traffic conditions always provide accurate information.
- [D_4]. Information of public transportation can be accessed by Public Transportation Information Provider (PTIP) API for the current city.
- [D_5]. Public transportation vehicles are assumed as punctual with their published programs.
- [D_{11}]. During mobile application operation, GPS is on and at working status while current position is needed.
- [D_{12}]. GPS API always provides accurate location information.
- [D_9]. Users are assumed to walk in average speed.

B.2 Use Cases

B.2.1 Scenarios

B.2.1.1 Scenario 1

Deniz has recently moved to Milan for her new job with her family. She has many appointments and her usual daily routine with the kids and house which is very hard to manage since she doesn't know the roads; the public transportation stops and hours or how long would it take her to go from one appointment to another. Her neighbor Maria advises her to try the Travlendar+ app so Deniz picks up her phone and downloads the app. Then, she creates a new user account and starts to form her schedule on the calendar.

B.2.1.2 Scenario 2

Deniz starts to add events on her calendar using the Travlendar+ app. She tries to add an appointment with an old friend for coffee by entering the time and the location of the café. Travlendar+ gives a warning stating that the appointment is unreachable/undoable between the events "Drop Kids to School" and "Resident Permit Appointment" and does not add it to the calendar and offers to select a new location. Deniz calls her friend and suggests him that they meet somewhere close to the police station where she has the appointment for the resident permit. Her friend accepts, and she adds the appointment by choosing the new reachable location.

B.2.1.3 Scenario 3

Deniz decides to add a 45-minute walk to her schedule in the mornings between 6 a.m. and 8 a.m. before dropping her kids to school so she chooses to add break on the Travlendar+ selects the start and end times as 6 a.m. and 8 a.m. and sets the break duration to 45 minutes.

Then she selects the periodicity of the break as daily so the Travlendar+ duplicates this break on every day on the calendar which is available and selects a 45 minute in the selected time interval for Deniz to have her morning walk.

B.2.1.4 Scenario 4

Deniz likes to walk and bike to her appointment as much as possible because she likes the fresh air and she is an environmentalist, but she realizes that Travlendar+ sometimes suggests her a public transport even though she thinks the appointment can be considered close enough to bike. She asks Maria about how she could fix this, and Maria tells her that in fact she could prioritize or even customize her mobility option preferences. Then shows her the manage mobility options feature on the app. She also advises her that she could just choose the default minimize carbon footprint list instead of sorting the list herself since the default list perfectly fits her preferences. Maria also tells her that she could remove any mobility option from her preference list by deactivating the option, so since Deniz doesn't have a driver's license she deactivates the Car and removes it from the list.

B.2.1.5 Scenario 5

After a couple of weeks Deniz knows the city of Milan a little better, and she realizes that there is traffic on the main streets at the start and end of business days, so she decides that she doesn't want to use taxi during these rush hours since it is very costly and time consuming. She also decides that walking more than 2 km to an event, even though it does not cause any inconvenience about timing is too tiring and senseless, so she goes to the manage mobility option menu on the Travlendar+ selects Taxi and sets the rush hours as "Restricted Time Interval" and then selects Walk and sets the "Distance Limit" a 2 km to make sure the application does not suggest these options if they do not satisfy the constraints.

B.2.1.6 Scenario 6

After a month, Deniz realizes that it started to be still dark between 6 a.m. and 7 a.m. in the morning and that she does not want to go for a morning walk when it is still dark outside, so she decides to change the time interval on her schedule. She selects "My Events" on the Travlendar+, then chooses the "Morning Walk" event on the list and edits the times as 7 a.m. to 7.30 a.m., Travlendar+ gives a warning stating that the time interval is less than the desired break duration. Deniz realizes she chose 7.30 instead of 8.30 a.m. and fixes the time and saves her changes.

B.2.1.7 Scenario 7

After a couple of months, Deniz learns how to drive and gets her driver's license since she thinks that it would be easier to drive the kids to their events. Although she does not own a car, since she knows that there are many car sharing options in Milan, she decides to have the Car option in here preference list. Thus, she goes to "Manage Mobility Options" on Travlendar+ reactivates the Car on the mobility options list, so the Travlendar+ puts the Car at the end of her preference list. She decides that she would rather drive an electric car than taking a Taxi, so she moves the Car above the Taxi option on her preference list to give it priority.

B.2.2 Use Case Descriptions

Use Case 1	Log in
Actors:	User
Goals:	G_1,G_2
Entry Condition:	The user is signed up to the system.
Flow of Events:	1. The user opens the Travlendar+ application on his/her device. 2. The system displays the main screen. 3. The user enters his/her e-mail address or user name and the predefined password. 4. The user selects log in. 5. The system displays tools menu with the profile information.
Exit Condition:	The use case terminates when the user is successfully logged in.
Exceptions:	• The user information entered by the user i.e. the email address/user name or the password is wrong hence an error message is displayed.

Table 1. Login Use Case

Use Case 2	Add Event
Actors:	User
Goals:	G_3,G_4
Entry Condition:	The user is logged in.
Flow of Events:	1. The user selects the My Events function from the Main Menu. 2. The user slides the screen down. 3. The system creates a new line on the top of the events list. 4. The user enters a name for the event. 5. The system directs the user to the calendar automatically. 6. The user selects the date that he/she wishes to add the event from the calendar view. 7. The user selects the event's start and end time on the selected date. 8. The system directs the user to the map. 9. The user selects the event location by entering the specific location name or the address. 10. The user saves the event details. 11. Travlendar+ displays a confirmation message notifying the user that the event is added to the calendar.
Exit Condition:	• The use case terminates when the selected date and time is assigned to the new event. • The selected date and time interval is now unavailable to

	be allocated by any other new event to be added in the future • The travel duration intervals associated with the added event are also now unavailable to be allocated by any other event to be added in the future.
Exception 1:	• If the time interval on the date selected by the user is already assigned to another event or overlaps with the travel duration of between other events, the system displays a warning message indicating that the desired date/time is unavailable. • The system displays another message after the warning asking the user if he/she would like to change the date/ time • If the user selects "Yes" the flow starts again from the second step. • If the user selects "No", the use case terminates without any date/time allocation.
Exception 2:	• If the time interval on the date selected by the user is free but the location is unreachable for the selected time according to the system calculations based on the current schedule, the system displays a warning message indicating the event is unreachable. • The system displays another message after the warning asking the user if he/she would like to change the event location. • If the user selects "Yes" the flow starts again from the third step. • If the user selects "No", the use case terminates without any date/time allocation.

Table 2. Add Event Use Case

Use Case 3	Create Customized Event
Actors:	User
Goals:	G_3,G_4
Entry Condition:	The user is logged in.
Normal Flow of Events:	1. The user selects the My Events function from the Main Menu. 2. The user slides the screen down. 3. The system creates a new line on the top of the events list. 4. The user enters a name for the event. 5. The system directs the user to the calendar automatically. 6. The user selects the date that he/she wishes to add the event from the calendar view. 7. The user selects the event's start and end time on the selected date. 8. The system directs the user to the map.

	9. The user selects the event location by entering the specific location name or the address. 10. The user slides the map view to the left to see event details. 11. The system displays a list screen including the event details which are Name, Date/Time, Location, Duration and Frequency. 12. The user edits the frequency which is set to null as default based on the event's nature (e.g. daily, weekly, monthly or customized (e.g. Weekly: Monday-Wednesday)). 13. The user may edit the duration which is set to the whole-time interval between the start and end time of the event as default based on the event's nature (e.g. 45-minute lunch break between the selected start and end time). 14. The user saves the event details. 15. Travlendar+ displays a confirmation message notifying the user that the event is added to the calendar with the defined frequency.
Exit Condition:	• The use case terminates when, starting with the selected date, the selected time is duplicated according to the frequency, assigned to the event and added to the other dates on the calendar. • The selected dates and time intervals appearing on the calendar based on the frequency are now unavailable to be allocated by any other new event to be added in the future
Alternative Flow of Events:	1. The user selects the My Events function from the Main Menu. 2. The user selects the existing event he/she would like to repeat. 3. The flow continues as in the normal flow starting from the fifth step.
Exceptions:	• The system displays a warning message indicating the event cannot be repeated since one or more date/time that needs to be allocated based on the frequency are unavailable. • The system displays another message after the warning asking the user if he/she would like to change the date/ time or frequency. • If the user selects "Yes" the flow starts again from the sixth step. • If the user selects "No", the use case terminates without any date/time allocation in the normal flow or only allocating the existing single event date/time in the alternative flow.

Table 3. Add Customized Event Use Case

Use Case 4	Edit Event Date/Time
Actors:	User
Goals:	G_3,G_4
Entry Condition:	• The user is logged in. • The event list is not empty.
Flow of Events:	1. The user selects the My Events function from the Main Menu. 2. The user selects the event he/she would like to edit. 3. The system directs user to the date selected on the calendar. 4. The user edits the date and/or time. 5. The system displays a message asking the user "Would you like to save the changes?". 6. The user selects Yes. 7. Travlendar+ displays a confirmation message notifying the user that the changes are saved.
Exit Condition:	• The use case terminates when the new date and time is assigned to the event. • The new date and/or time interval is now unavailable to be allocated by any other new event to be added in the future. • The previous date and/or time assigned to the event is now available to be allocated by any other new event to be added in the future.
Exceptions 1:	• The user selects No after the system displays the "Would you like to save the changes?" message. • The use case terminates without changing the date and/or time of the event. • The event's current date and time interval stays unavailable to be allocated by any other new event to be added in the future.
Exceptions 2:	• If the time interval on the new date and/or time selected by the user is already assigned to another event, the system displays a warning message indicating that the desired date/time is unavailable. • The system displays another message after the warning asking the user if he/she would like to change the date/ time. • If the user selects "Yes" the flow starts again from the fourth step. • If the user selects "No", the use case terminates keeping the current date/time allocated.

Table 4. Edit Event Date/Time Use Case

Use Case 5	Edit Event Location
Actors:	User
Goals:	G_3,G_4

Entry Condition:	• The user is logged in. • The event list is not empty.
Flow of Events:	1. The user selects the My Events function from the Main Menu. 2. The user selects the event he/she would like to edit. 3. The system directs user to the date selected on the calendar. 4. The user slides the screen to the left. 5. The system displays the location on the map view. 6. The user selects a new location by entering the address or the specific location name. 7. The system displays a message asking the user "Would you like to save the changes?". 8. The user selects Yes. 9. Travlendar+ displays a confirmation message notifying the user that the changes are saved.
Exit Condition:	• The use case terminates when the new location is assigned to the event. • The event's current date and time interval stays unavailable to be allocated by any other new event to be added in the future.
Exceptions 1:	• The user selects No after the system displays the "Would you like to save the changes?" message. • The use case terminates without changing the location of the event.
Exceptions 2:	• If the new location is unreachable for the selected time according to the system calculations based on the current schedule, the system displays a warning message indicating the event is unreachable. • The system displays another message after the warning asking the user if he/she would like to select another event location. • If the user selects "Yes" the flow starts again from the sixth step. • If the user selects "No", the use case terminates keeping the current location unaltered.

Table 5. Edit Event Location Use Case

Use Case 6	Delete Event
Actors:	User
Goals:	
Entry Condition:	• The user is logged in. • The event list is not empty.
Flow of Events:	1. The user selects the My Events function from the Main Menu. 2. The user holds the event he/she wants to delete and slides

	it to the left. 3. The system displays a message asking the user "Would you like to delete this event?". 4. The user selects Yes. 5. The system removes the event from the list and the calendar.
Exit Condition:	• The use case terminates when the event is removed from the list and the calendar. • The deleted event's date and time interval is now available to be allocated by any other new event to be added in the future.
Exceptions:	• The user selects No after the system displays the "Would you like to delete this event?" message. • The use case terminates without removing the event from the list or the calendar. • The event's date and time interval stay unavailable to be allocated by any other new event to be added in the future.

Table 6. Delete Event Use Case

Use Case 7	Sort Mobility Preferences
Actors:	User
Goals:	G_5
Entry Condition:	• The user is logged in. • The mobility option is active.
Flow of Events:	1. The user selects Mobility Option Preferences from the Main Menu. 2. The system displays all mobility options. 3. The user sorts the mobility options based on his/her preferences (e.g. if the user prefers to walk or ride a bike as much as possible he/she should put these options on the top of the list) by holding the option and swiping it to the desired row on the list or he/she can turn on the "Minimize Carbon Foot Print" option displayed at the end of the list and the options are automatically sorted to fulfill this goal. 4. Travlendar+ displays a confirmation message stating that the preferences are saved.
Exit Condition:	• The use case terminates when the system saves the preferences to be used in the best mobility option algorithm.

Table 7. Sort Mobility Preferences Use Case

Use Case 8	Deactivate Mobility Option
Actors:	User
Goals:	G_5
Entry Condition:	The user is logged in.

Flow of Events:	1. The user selects Mobility Option Preferences from the Main Menu. 2. The system displays all mobility options. 3. The user holds the mobility option he/she wants to deactivate and slides it to the left. 4. Travlendar+ removes it from the preference list and fades it in the Mobility Options List and displays a confirmation message.
Exit Condition:	• The use case terminates when the system saves the mobility option's state to be used in the best mobility option algorithm. • The customization and sorting options are unavailable for the deactivated mobility option.
Exceptions:	

Table 8. Deactivate Mobility Option Use Case

Use Case 9	Activate Mobility Option
Actors:	User
Goals:	G_5
Entry Condition:	The user is logged in.
Flow of Events:	1. The user selects Mobility Option Preferences and then Mobility Option List from the Main Menu. 2. The system displays all mobility options. 3. The user holds the deactivated mobility option he/she wants to activate and slides it to the right. 4. The system brightens the activated option and brings it to the end of the preference list, assigning it the next priority after the active mobility option above it and displays a confirmation message.
Exit Condition:	• The use case terminates when the system saves the mobility option's state to be used in the best mobility option algorithm. • The customization and sorting options are now available for the activated mobility option.
Exceptions:	

Table 9. Activate Mobility Option Use Case

Use Case 10	Customize Mobility Option/ Restrict Undesired Time Interval
Actors:	User
Goals:	G_5
Entry Condition:	• The user is logged in. • The mobility option is active.
Flow of Events:	1. The user selects Mobility Option Preferences from the Main Menu. 2. The system displays mobility preferences list. 3. The user selects the mobility option he/she wants to

	customize. 4. The user selects a start and end time from the "Restricted Time Interval" drop down menu.
Exit Condition:	• The use case terminates when the system saves the custom preference to be used in the best mobility option algorithm.
Exceptions:	

Table 10. Set Restricted Time Use Case

Use Case 11	Customize Mobility Option/ Set Distance Limit
Actors:	User
Goals:	G_5
Entry Condition:	• The user is logged in. • The mobility option is active.
Flow of Events:	1. The user selects Mobility Option Preferences from the Main Menu. 2. The system displays Mobility Preferences List. 3. The user selects the mobility option he/she wants to customize. 4. The user enters the maximum distance limit he/she would utilize the selected mobility option. 5. The user selects the unit of measure from the drop down list. 6. Travlendar+ displays a confirmation message.
Exit Condition:	• The use case terminates when the system saves the custom preference to be used in the best mobility option algorithm.
Exceptions:	

Table 11. Set Distance Limit Use Case

Use Case 12	Edit Mobility Preferences
Actors:	User
Goals:	G_5
Entry Condition:	The user is logged in.
Flow of Events:	1. The user selects Mobility Option Preferences from the Main Menu. 2. The system displays the previously sorted and customized mobility preferences list. 3. The user edits his/her preferences (e.g. resorts the list, changes the customizations or activates a deactivated mobility option.) 4. Travlendar+ displays a confirmation message.
Exit Condition:	• The use case terminates when the system saves the new preferences to be used in the best mobility option algorithm.

B.2.3 Use Case Diagram

B.2.3.1 Visitor

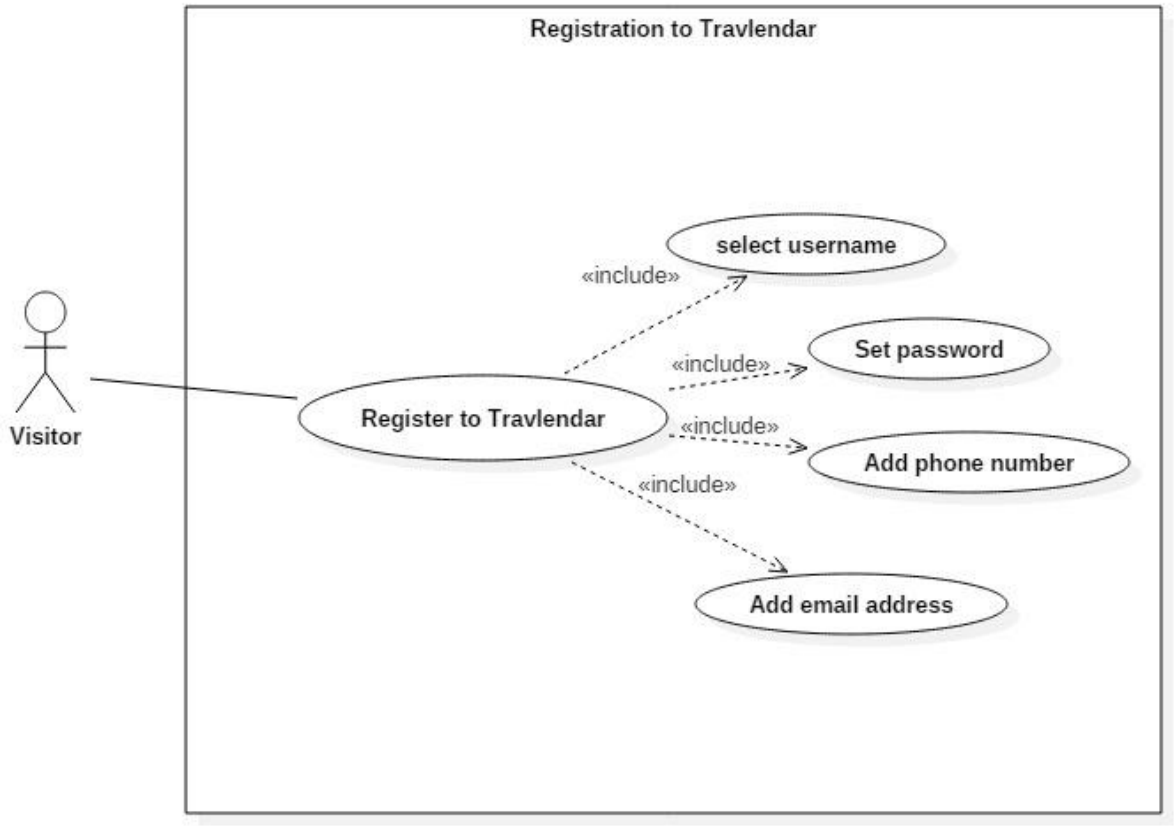


Figure 9. Registration Use Case Diagram

B.2.3.2 User

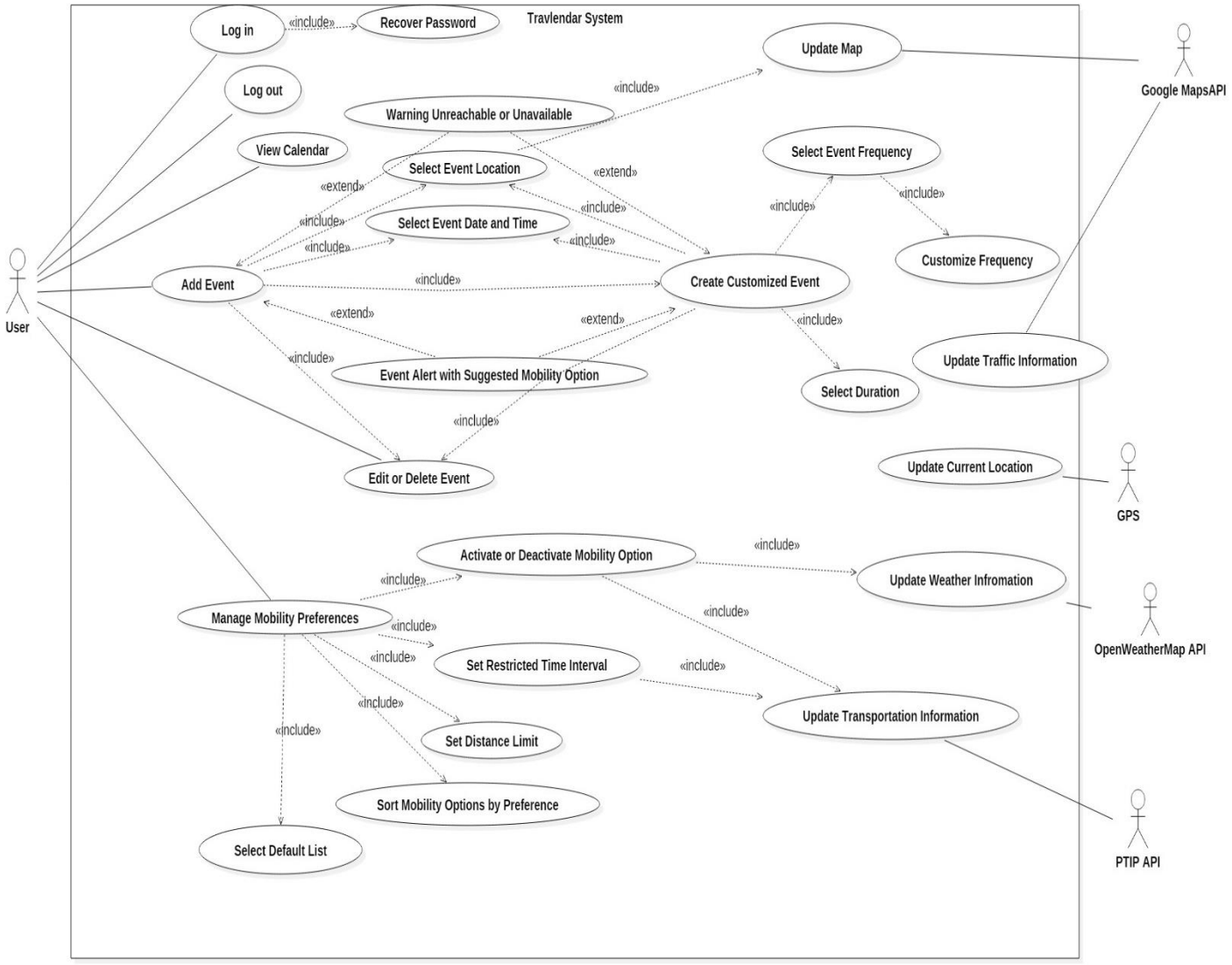


Figure 10. Registered User Use Case Diagram

B.3 Sequence Diagrams

B.3.1 Visitor Create New Account

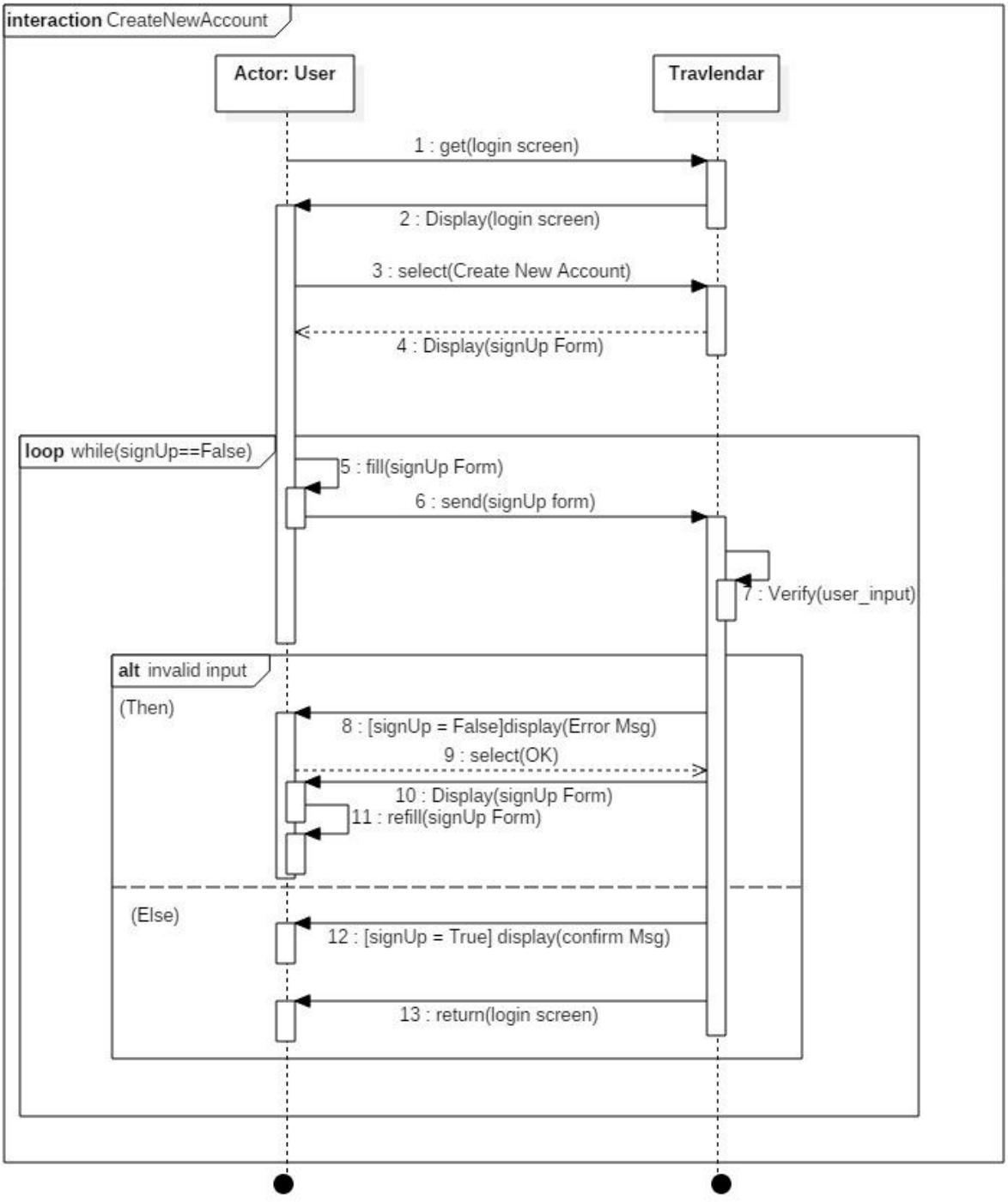


Figure 11. Create New Account Sequence Diagram

B.3.2 User Login

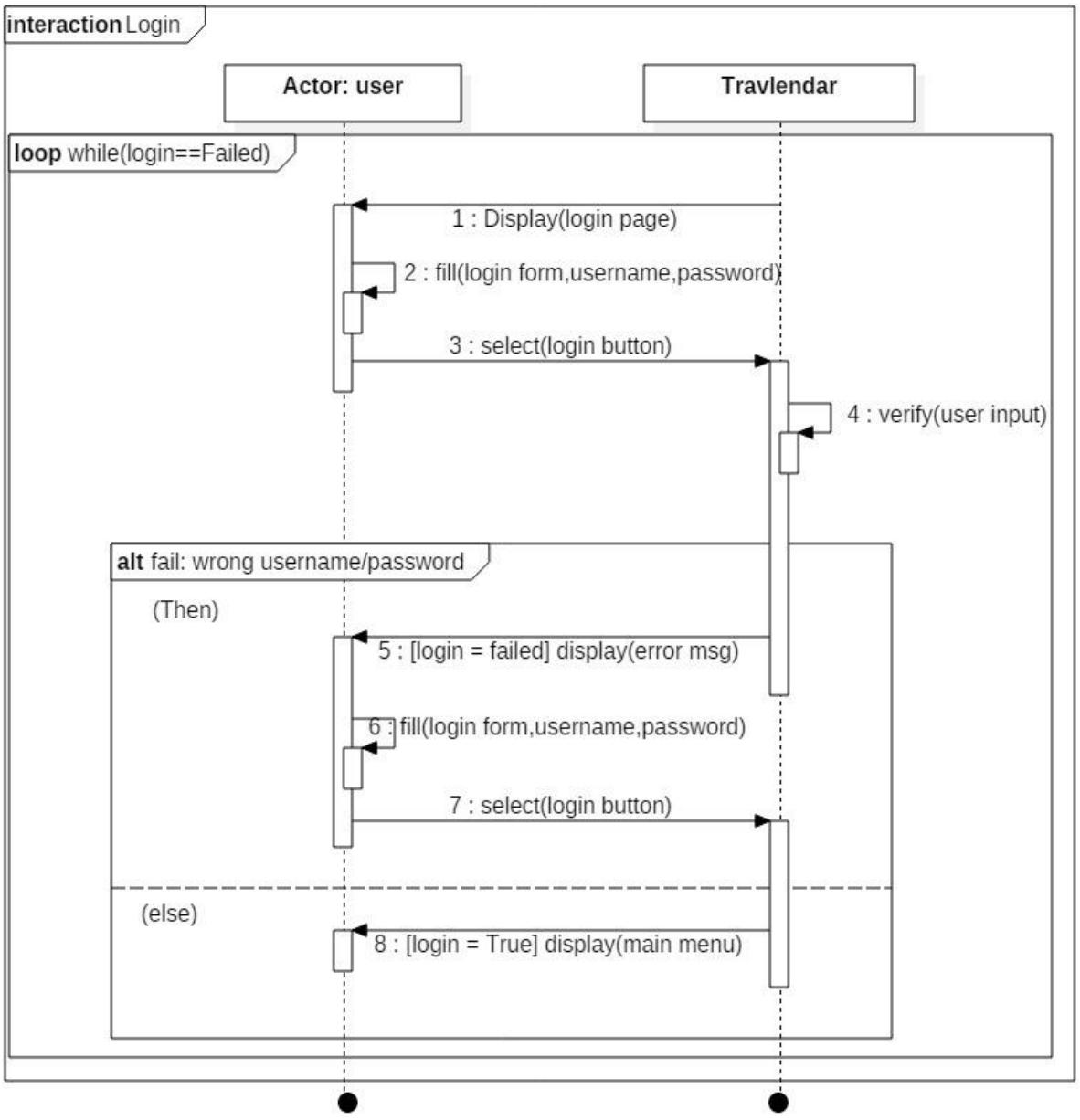


Figure 12. Login Sequence Diagram

B.3.3 Add Event

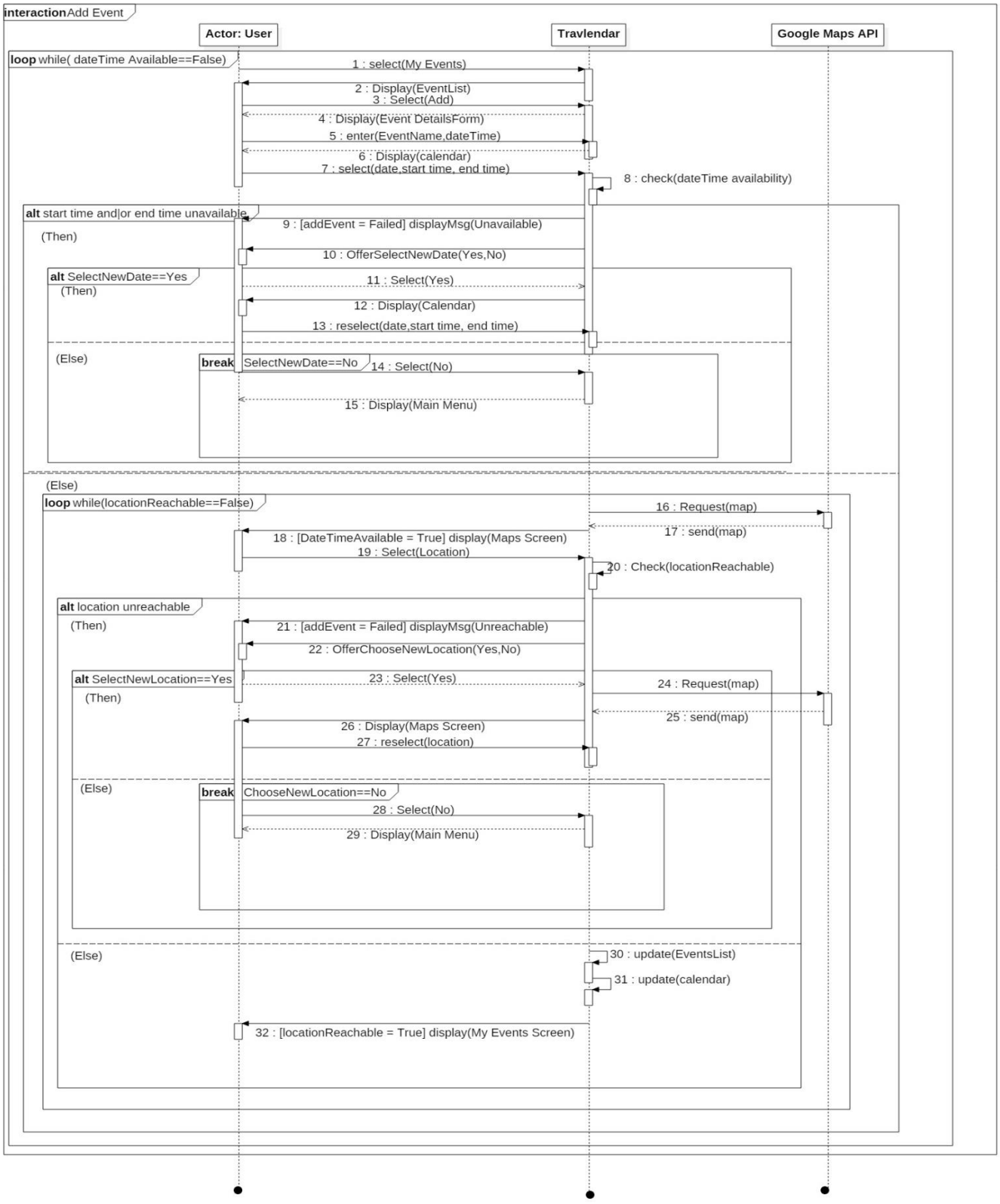


Figure 13. Add Event Sequence Diagram

B.3.4 Manage Mobility Preferences

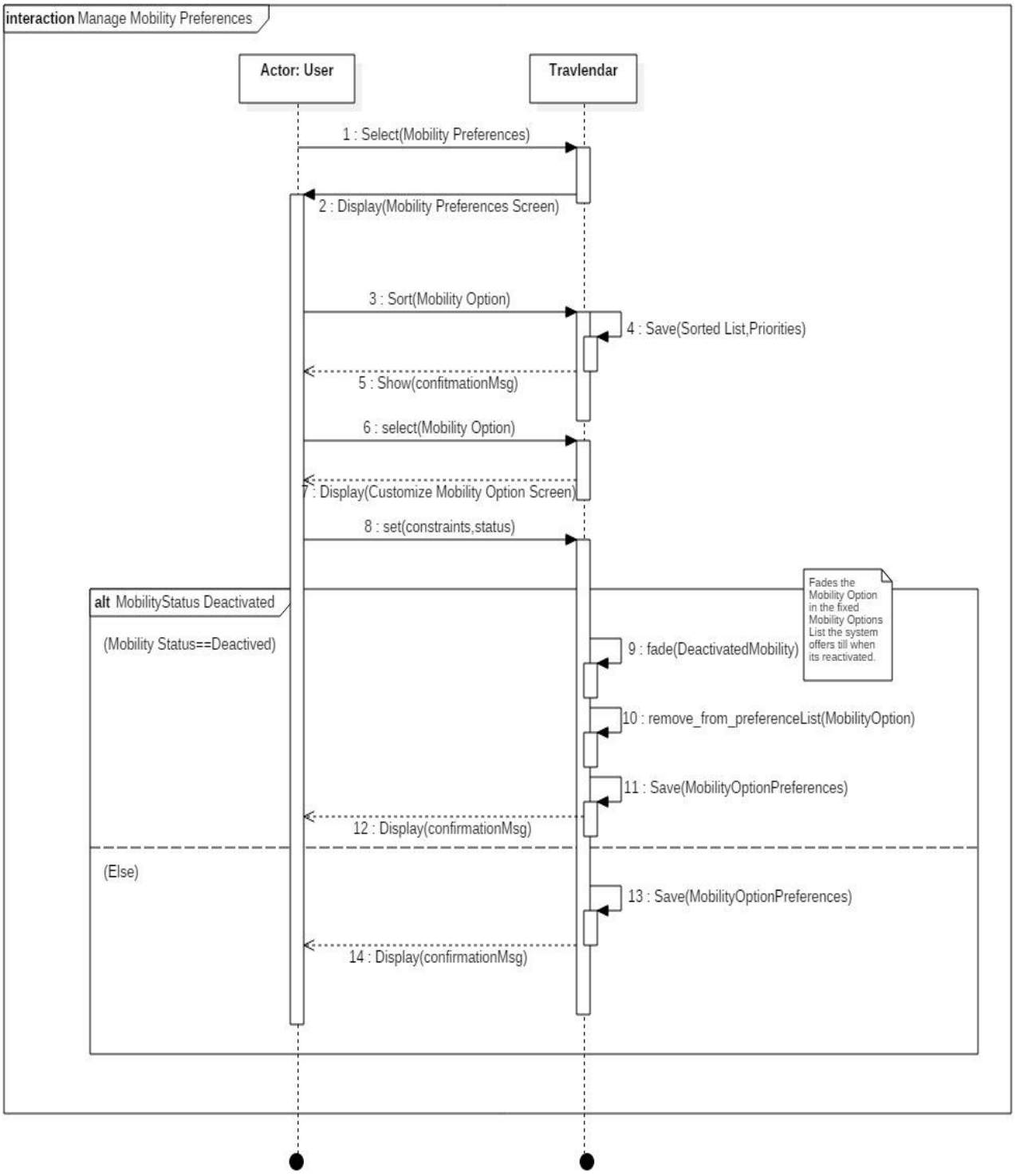


Figure 14. Manage Mobility Preference Sequence Diagram

B.3.5 Add Break

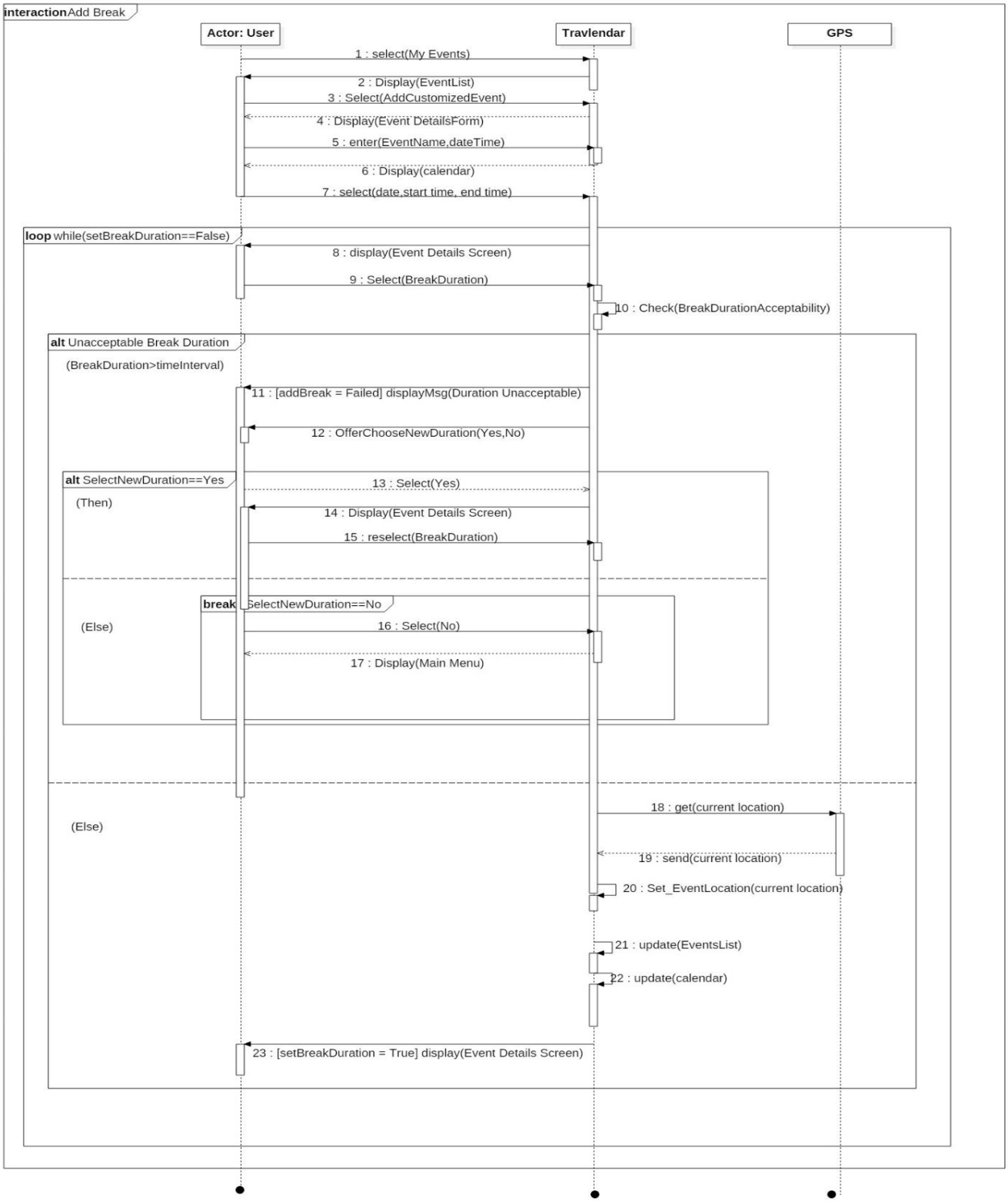


Figure 15. Add Customized Event Sequence Diagram

B.4 Statechart Diagram

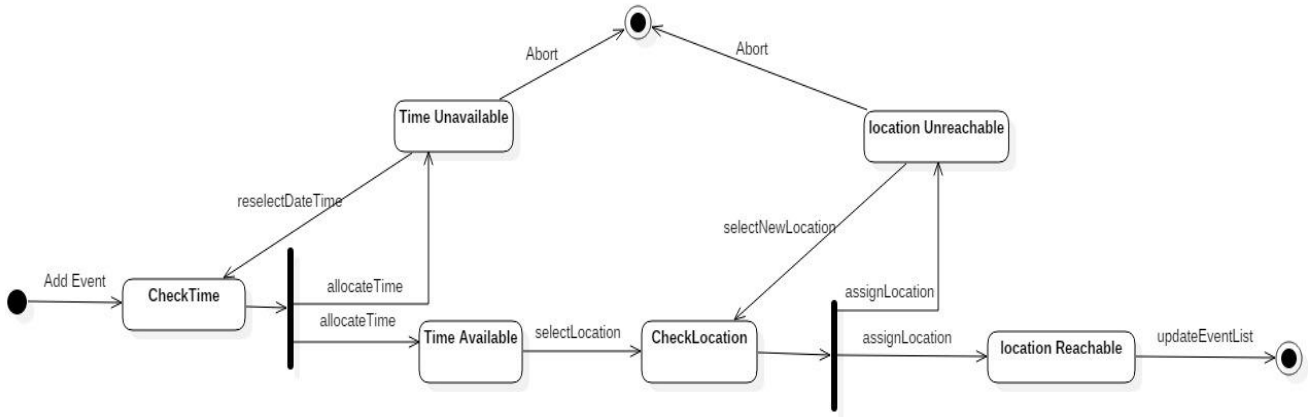


Figure 16. Add Event Statechart Diagram

B.5 Class Diagram

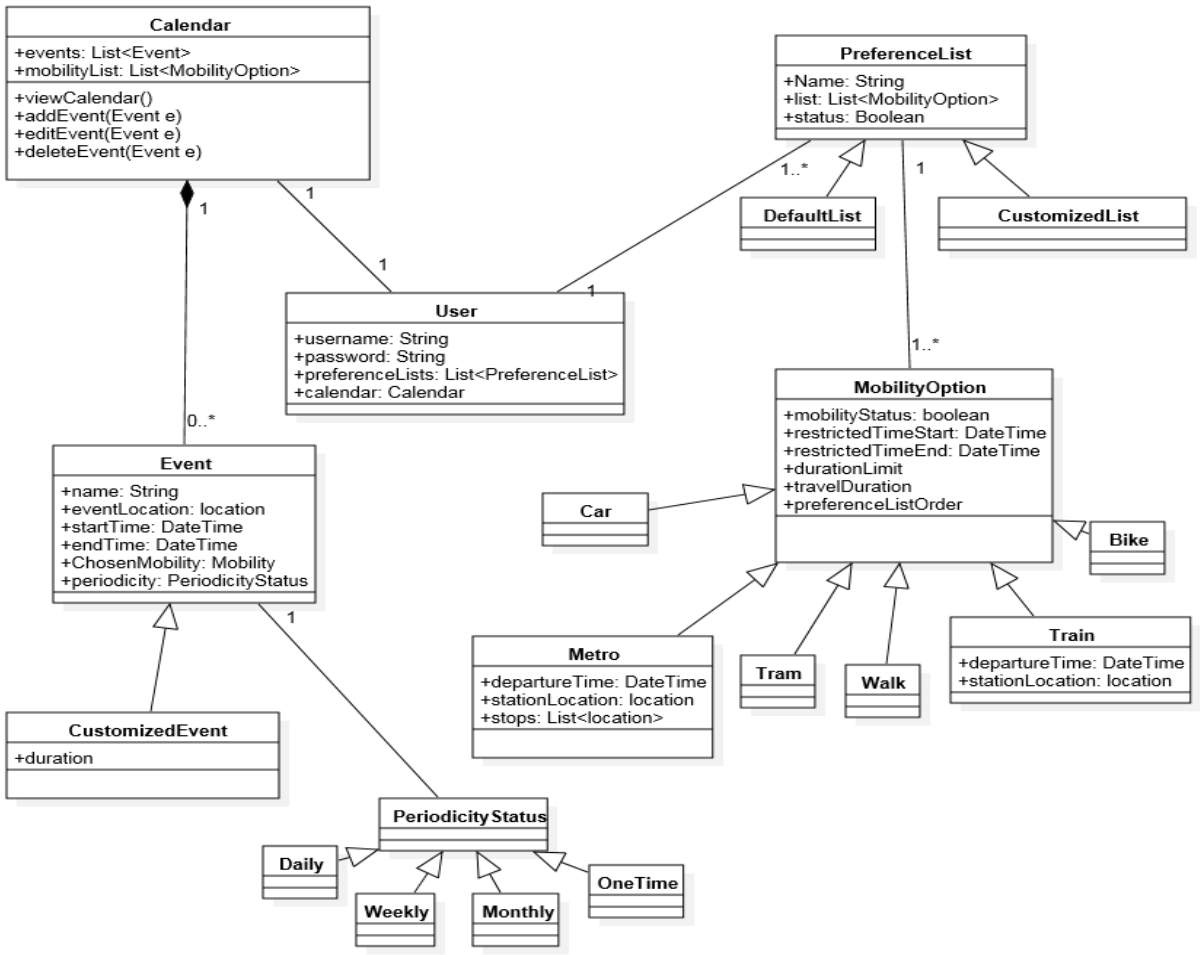


Figure 17. Class Diagram

C. Performance Requirements

The system must enable simultaneous access to user accounts and their database by the different users. Until 10000 user access, the system must be sustainable and able to provide any service without any delay due to application itself.

D. Design Constraints

D.1 Hardware limitations

For the mobile application, user needs a device with:

- At least 3G Internet connection
- GPS Connection
- Compatible operating system and device (IOS or Android smartphone)
- Space for application

E. Software System Attributes

E.1 Reliability

The proposed system collects information from APIs that always present accurate information. For public transportation, official information on transportation means' stops and schedules is received by our application from the related APIs and for the localization, Google Maps and deployed GPS device provide most recent maps and accurate current location information.

E.2 Availability

The application does not require any human operator or similar dependency. Therefore, it can be available 7/24 if the Internet connection exists and is stable.

E.3 Security

The application should be usable for only registered users and no communication or information sharing is possible between users through the application. Also, the system developers guarantee that no personal information is published or used for any commercial usage. Personal information can be only accessed by entering user credentials.

E.4 Maintainability

The implementation and design should be documented with clear comments and presents all the updates in chronological order. Also, the system should provide modularity between each subsystem that facilitates the coding and debugging processes.

E.5 Portability

The application should be downloaded and installed in any device with Android or IOS operating systems.

4. FORMAL ANALYSIS USING ALLOY:

4.1 MODEL

```
//******************** Signatures for the overall system ************************//
// All the signatures are defined based on class diagram
// For the timestamps and duration operations, addition/subtraction and all types of comparisons
// are needed. Therefore, "Natural" type is employed for this purpose in this alloy model.
open util/integer
// Since an order is needed for user preference list of mobility options, Priority signature is used
// with ordering utilization
open util/ordering[Priority] as po
// Necessary Data structures for string
sig Str{}

// Signature for events
sig Event{
   location: one Str,
   StartTime: one Int,
   EndTime: one Int,
   ChosenMobility: one Mobility,
   periodicity : one PeriodicityStatus,
   name: one Str
}{
one StartTime
StartTime > 0
one EndTime
EndTime  > 0
// Start time is always before end time
lt[StartTime , EndTime]
}
abstract sig PeriodicityStatus{ }
lone sig OneTime,Daily,Weekly,Monthly extends PeriodicityStatus{ }
```

```
// Extension for customized events
sig CustomizedEvent extends Event {
   Duration: one Int
}
{
   one Duration
   // Duration for customized event is always greater than zero and less than or equal to time between start and endtime
   gt[ Duration,0]
   lte[Duration , sub[sub[EndTime,StartTime],ChosenMobility.TravelDuration] ]
}

// Signatures for related Mobility Options

abstract sig mobilityStatus {}
one sig Activated extends mobilityStatus{}
one sig Deactivated extends mobilityStatus{}
lone sig Car,Tram,Train,Metro,Bus,Bike,Walk extends Mobility{}
sig Priority{}

abstract sig Mobility{
   // No mobility can be used in between restrictedStartTime and restrictedEndTime
   restrictedStartTime: one Int,
   restrictedEndTime: one Int,
   status: one mobilityStatus,
   // Maximum allowed duration by user
   durationLimit: lone Int,
   TravelDuration: one Int,

}
{
   restrictedStartTime > 0
   restrictedEndTime > 0
   durationLimit > 0
   TravelDuration > 0
   gt[restrictedEndTime,restrictedStartTime]
}

// Signature for Calendar
sig Calendar {
   EventList: some Event,
   MobilityList = EventList.ChosenMobility
}

// Signature for User
sig User {
   username: one Str,
   password: one Str,
   calendar: one Calendar,
   mlist: one PreferenceList
}
{
// One username,password and calendar for each users
   one username
   one password
   one calendar
   one mlist
}
```

```
// Signatures for related PreferenceList
sig PreferenceList {
   listName : one Str,
   MobilityList: some Mobility,
   priorityList:some Priority,
   mplist : ( MobilityList -> one priorityList)
}{
   #priorityList > 0
   #MobilityList > 0
   #priorityList = # MobilityList
   no disjoint m1,m2: MobilityList | mplist[m1] =mplist[m2]
}
// Default is predefined lists that are focused on particular subjects such as footprint minimization, cost minimization etc.
sig Default extends PreferenceList {}
sig CustomizedList extends PreferenceList {}

// ********************  Facts start here ********************** //
fact uniqueUsername {
   no disjoint u1,u2:  User | u1.username = u2.username
}

fact uniquePreferenceList {
   no disjoint u1,u2:  User | u1.mlist = u2.mlist
}
fact uniqueCalender {
   no disjoint u1,u2:  User | u1.calendar = u2.calendar
}
fact uniqueEvents {
   no disjoint c1,c2: Calendar,e:Event | e in c1.EventList and e in c2.EventList
}
// one calendar and preference list for each user
// No unowned calendar or preference list occur
fact Cardinality {

   #Calendar = #User
   #PreferenceList = #User
   #Event=#PeriodicityStatus
}

// Choose top mobility option in user preferences
fact chooseTopMobilityOption {
   all  u:User ,
       m : u.calendar.EventList.ChosenMobility|
       m = ((u.mlist).mplist).(min[u.mlist.priorityList])
}
fact NoDeactiveMobilityOptionInPreferenceList {
   no u:User, m:Mobility | m.status=Deactivated and m in u.mlist.MobilityList
}
// No overlapping event occur
fact noOverlappingEventInCalendar {
   // If the events are not customized, then start time of any event cannot be between start time and end time of any other event
   all e, e2 : Calendar.EventList  | (e != e2  and e != CustomizedEvent and
      e2 != CustomizedEvent) implies
      (e.StartTime != e2.StartTime or e.EndTime != e2.EndTime)

   all e, e2 : Calendar.EventList  | (e != CustomizedEvent and e2 != CustomizedEvent and
                                          gt[e.StartTime, e2.StartTime] ) implies
                                          gte[sub[e.StartTime,e.ChosenMobility.TravelDuration]
                                          ,e2.EndTime]
   // If one event is customized and others are not, there should be enough time for customized event duration between two events
   all e1,e2,ce : Calendar.EventList |(  e1 != CustomizedEvent and e2 != CustomizedEvent and
                                             ce = CustomizedEvent and gt[e1.StartTime , e2.EndTime]
                                             and lte[ce.StartTime, e1.EndTime] and gte[ce.EndTime ,
                                             e2.EndTime] ) implies lte[add[ce.Duration,
                                             ce.ChosenMobility.TravelDuration] ,
                                             sub[sub[e1.StartTime,e1.ChosenMobility.TravelDuration],
                                             e2.EndTime] ]


// If there are one event and one customized event, total duration of both should be less than time between start time of previous one and end time of next one
   all e1,ce: Calendar.EventList |( (e1 != CustomizedEvent and ce = CustomizedEvent) and
      (lte[e1.StartTime,ce.StartTime] ) ) implies( gt[ce.EndTime,e1.EndTime]
      and lte[add[add[sub[e1.EndTime,e1.StartTime],ce.Duration],ce.ChosenMobility.TravelDuration], sub[ce.EndTime,e1.StartTime]])
   all e1,ce: Calendar.EventList | ((e1 != CustomizedEvent and ce = CustomizedEvent) and
      (lt[ce.StartTime,e1.StartTime] )) implies lte[add[add[sub[e1.EndTime,e1.StartTime],ce.Duration],e1.ChosenMobility.TravelDuration], sub[e1.EndTime,ce.StartTime]]


   // If there are two customized events, total duration of both,total duration of both should be less than time between start time of previous one and end time of next one
   all ce1,ce2: Calendar.EventList |(ce1 != ce2 and (ce1 = CustomizedEvent and ce2 = CustomizedEvent) and
      lte[ce1.StartTime,ce2.StartTime]) implies lt[add[add[ce1.Duration,ce2.Duration],ce2.ChosenMobility.TravelDuration], sub[ce2.EndTime,ce1.StartTime]]


}
```

```
--CHOOSING MOBILITY
--deactivated Mobility cannot be on the PreferenceList
fact onlyActiveMobilitiesOnTheList{ all m:Mobility,p:PreferenceList|
                                        m in p.MobilityList <=> m.status=Activated }

--Mobility that is not on Preference List cannot be chosen
fact ChosenMobilityMustBeOnPreferenceList{all e:Event,p:PreferenceList|e.ChosenMobility
                                            in p.MobilityList}

fact NoTravelDurationIfMobilityIsNotChosen{all m:Mobility, e:Event| lt[m.TravelDuration, 0] =>
                                            m!=e.ChosenMobility}

fact DeactivatedIfTravelInRestrictedTime {
   all u:User,e : u.calendar.EventList , m:u.mlist.MobilityList | (gte[m.restrictedStartTime,
      sub[e.StartTime,e.ChosenMobility.TravelDuration]] and
      lte[m.restrictedStartTime ,e.StartTime] ) or (gte[m.restrictedEndTime,
      sub[e.StartTime,e.ChosenMobility.TravelDuration]] and
      lte[m.restrictedEndTime ,e.StartTime]) <=> m.status = Deactivated
}

// If Travel duration is more than duration limit, the mobility is deactivated
// For instance, if it lasts 1 hour to walk somewhere then,
// however the limit for walking is 30 minutes
// walk option is deactivated
fact DeactivatedIfTravelDurationIsMoreThanLimit
{
   all m:Mobility | gt[m.TravelDuration, m.durationLimit] => m.status = Deactivated
}
pred RegisterNewUser[u,up,unew:User] {
   up = u + unew
}
pred editEvent [c,cp:Calendar,e ,ep:Event] {

   deleteEvent[c,cp,e]
   addEvent[c,cp,ep]
// Edit an event by deleting previous event and adding altered one
}
pred addEvent[c,c':Calendar, e:Event ] {
   c'.EventList = c.EventList + e
}
pred deleteEvent[c,c':Calendar, e:Event ] {
   c'.EventList = c.EventList - e
}
pred showAddEvent[c,c':Calendar, e:Event ] {
   #User =1
   #User.calendar.EventList = 3
   #CustomizedEvent = 1
   addEvent[c,c',e]

}
pred showDeleteEvent[c,c':Calendar, e:Event ] {
   #User =1
   deleteEvent[c,c',e]

}
assert RegisterUser{
   all u,up,unew:User | unew not in u and RegisterNewUser[u,up,unew] implies unew.username not in u.username
}

assert AddDeleteUndoEvent {
   all c,c',c'' : Calendar, e : Event | e not in c.EventList and
   addEvent[c,c',e] and deleteEvent[c',c'',e]
   implies c.EventList = c''.EventList
}

assert AddTwoEventInSameTimeInterval{
// Same event cannot be added two times
   no e,ep: Event,u:User, c,c':u.calendar | ep=e and e not in c.EventList and addEvent[c,c',e] and  addEvent[c,c',ep]

}

assert EditEvent {
// Compare two calendar after edit on a event
   all u:User,cp,c: u.calendar, e:c.EventList ,ep: Event| editEvent [c,cp,e ,ep]  implies c != cp
}
```

```
// Checks wheteher chosen mobility option by an event is the top of user preference list
assert MobilityOptionPriority {
   all u:User, m1:u.mlist.MobilityList,m2:u.calendar.EventList.ChosenMobility | lte[u.mlist.mplist[m2],u.mlist.mplist[m1]]
}
// Checks Mobility option is feasible various scenarios
assert MobilityOptionFeasibility {
   all u:User, c:u.calendar ,e1 ,e2: c.EventList  |e1 != e2 and e1!=CustomizedEvent and e2!=CustomizedEvent and
                              gt[e1.StartTime,e2.StartTime] implies gte[sub[e1.StartTime,e1.ChosenMobility.TravelDuration],e2.EndTime]

   all u:User, c:u.calendar ,e1 ,e2: c.EventList  |e1 != e2 and e1=CustomizedEvent and e2!=CustomizedEvent and
                              gt[e1.StartTime,e2.StartTime] implies gte[sub[e1.EndTime,e2.StartTime] ,
                              add[add[sub[e2.EndTime,e2.StartTime],e1.Duration],e1.ChosenMobility.TravelDuration]]

   all u:User, c:u.calendar ,e1 ,e2: c.EventList  |e1 != e2 and e1=CustomizedEvent and e2=CustomizedEvent and
                              gt[e1.StartTime,e2.StartTime] implies gte[sub[e1.EndTime,e2.StartTime] , add[add[e2.Duration,e1.Duration],e1.ChosenMobility.TravelDuration]]

}
--PREDICATES
--pred reachable
--pred IsCalendarFeasible
pred show{
   #User = 1
   #Event = 3
   #CustomizedEvent > 1
   all e:Event | e in User.calendar.EventList
   all p:PeriodicityStatus | p in User.calendar.EventList.periodicity
   all m:Mobility | m.status = Activated
}
pred DeactivateMobilityOptions{
// For Instance, due to weather , bike and walk is deactivated
Bike.status = Deactivated
Walk.status = Deactivated
#User =1
#User.calendar.EventList = 3
#CustomizedEvent = 1
}
pred MultipleUsers {
   #User > 1
}

run show for 7 Int
run DeactivateMobilityOptions for 7 Int
run showAddEvent for 7
run showDeleteEvent for 7
run editEvent for 7 Int
run MultipleUsers for 7

check AddDeleteUndoEvent
check EditEvent
check AddTwoEventInSameTimeInterval
check RegisterUser
check MobilityOptionPriority for 7 Int
check MobilityOptionFeasibility for 7 Int
```

Executing "Run show for 7 int"
Solver=sat4j Bitwidth=7 MaxSeq=4 SkolemDepth=1 Symmetry=20
113480 vars. 5058 primary vars. 362202 clauses. 742ms.
Instance found. Predicate is consistent. 1497ms.

Executing "Run DeactivateMobilityOptions for 7 int"
Solver=sat4j Bitwidth=7 MaxSeq=4 SkolemDepth=1 Symmetry=20
113483 vars. 5058 primary vars. 362256 clauses. 642ms.
Instance found. Predicate is consistent. 1735ms.

Executing "Run showAddEvent for 7"
Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
58848 vars. 1899 primary vars. 132277 clauses. 434ms.
Instance found. Predicate is consistent. 294ms.

Executing "Run editEvent for 7 int"
Solver=sat4j Bitwidth=7 MaxSeq=4 SkolemDepth=1 Symmetry=20
113541 vars. 5070 primary vars. 362317 clauses. 638ms.
Instance found. Predicate is consistent. 738ms.

Executing "Run MultipleUsers for 7"
Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
58385 vars. 1878 primary vars. 130917 clauses. 335ms.
Instance found. Predicate is consistent. 445ms.

Executing "Check AddDeleteUndoEvent"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
18404 vars. 926 primary vars. 43376 clauses. 76ms.
No counterexample found. Assertion may be valid. 5ms.

Executing "Check EditEvent"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
18419 vars. 929 primary vars. 43379 clauses. 72ms.
No counterexample found. Assertion may be valid. 2ms.

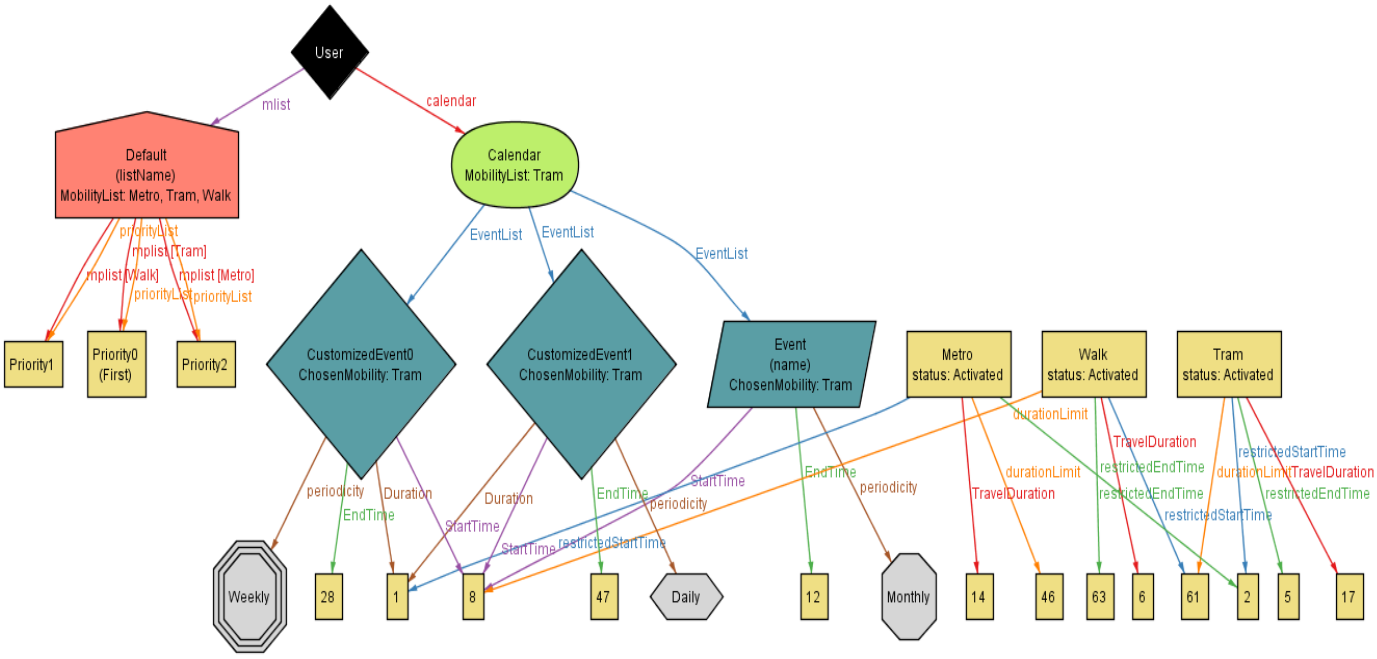
Executing "Check AddTwoEventInSameTimeInterval"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
18424 vars. 929 primary vars. 43388 clauses. 65ms.
No counterexample found. Assertion may be valid. 3ms.

Executing "Check RegisterUser"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
18354 vars. 923 primary vars. 43233 clauses. 64ms.
No counterexample found. Assertion may be valid. 2ms.

Executing "Check MobilityOptionPriority for 7 int"
Solver=sat4j Bitwidth=7 MaxSeq=4 SkolemDepth=1 Symmetry=20
113774 vars. 5075 primary vars. 362922 clauses. 683ms.
No counterexample found. Assertion may be valid. 320ms.

Executing "Check MobilityOptionFeasibility for 7 int"
Solver=sat4j Bitwidth=7 MaxSeq=4 SkolemDepth=1 Symmetry=20
113676 vars. 5058 primary vars. 364002 clauses. 1363ms.
No counterexample found. Assertion may be valid. 24ms.

4.2 GENERATED WORLD



5. EFFORT SPENT:

5.1 Hours of Work

5.1.1 Pelinsu Çelebi

DATE	TASK	HOURS
7.10.2017	Use Case Diagrams	3
10.10.2017	Goals & Assumptions	1,5
14.10.2017	UI	5
17.10.2017	Use Cases	3
20.10.2017	Use Cases	2,5
23.10.2017	Alloy	2
24.10.2017	Alloy	5,5
25.10.2017	Alloy & Scenarios	4
26.10.2017	Sequence & Statechart Diagrams	4
27.10.2017	RASD	3
29.10.2017	RASD	2
TOTAL:		35,5

5.1.2 Yiğit Pilavcı

DATE	TASK	HOURS
3.10.2017	Specifications & Goals	1
8.10.2017	Goals & Assumptions	1
10.10.2017	Requirements	2
15.10.2017	Use Cases	2
16.10.2017	RASD	1
20.10.2017	Class Diagram	3
21.10.2017	Alloy	5
22.10.2017	Various	1
24.10.2017	Alloy & Sequence Diagrams	5,5
25.10.2017	Alloy & RASD	5
27.10.2017	Alloy	4
28.10.2017	Alloy	5
29.10.2017	RASD	1
TOTAL:		36,5

6. REFERENCES:

- Specification Document: "AA 2017-2018 Software Engineering 2—Mandatory Project.pdf".
- Alloy Tutorial Examples "http://alloy.mit.edu/alloy/tutorials/online/index.html".

7. APPENDIX:

7.1 Used Tools

- Alloy Analyzer 4.2
- Star UML 2.8.0
- Pencil 3.0.4
- Github