WeCalEvent

Requirements Analysis and Specification Document

Software Engineering 2

Authors:

Mert Ergun

Mohamed Mehdi Kaabi

Nazrin Javadova

Milano – 2014

LaTeX

Contents

1 Introduction

1.1 Description

WeCalEvent is a platform where users are able to create and manage their events and invite other users. Information about time and place of the event should be provided while creation of the event. The system alerts the event creator 3 days before the event day in case of weather condition alteration and suggests him the closest day with suitable weather.

Invited users will be able to decline or accept the invitation. All event participants will be notified in case of weather forecast alteration a day before the event.

This platform is sufficient for users who look forward to schedule their events according to the weather conditions.

1.2 Purpose

The purpose of the Requirements Analysis and Specification Document is to clearly state every requirement, functions and features of the product. It is important to predict and sort out how we hope this product will be used. RASD will provide a detailed overview of our software product, its parameters and goals, being useful both for developers and customers. Therefore, this document is written for mainly developers of the system to make sure that all team members has the same vision of requirements. Secondarily, it is written for the Software Engineering 2 course staff to give them general description of our project requirements.

1.3 Scope

The project named WeCalEvent which our team intend to develop is a web application that aims to provide Event management environment in the basis of weather forecast. The users of our system can easily create events invent other users who currently use our system. Basically, our application is capable of dealing with bad conditions. For example, the system notifies and suggest alternative dates with good weather conditions to the user in case of bad weather conditions on the current date of the event.

1.4 Definitions, acronyms, and abbreviations

Weather forecast: a statement saying what the weather will be like the next day, or the next few days.

Calendar: is a table or register with the days of each month and week in a year.

Event: Something that occurs in a certain place during a particular interval of time with the participation of invited people in case of an appropriate weather. A social gathering or activity at the indicated time and in the indicated place .

User: A person registered in the system. Users can create and manage events and be invited to other users events.

Alloy: Alloy is a language for describing structures and a tool for exploring them.

Abbr	**Definition**
UML	Unified Modeling Language
GB	Gigabyte
RAM	Random Access Memory
EJB	Enterprise Java Beans
JSF	Java Server Faces
JVM	Java Virtual Machine

1.5 References

- Star UML Guide, retrieved from http://staruml.sourceforge.net/docs/user-guide(en)/toc.html
- IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

1.6 Overview

There are mainly four more parts of the document. The first part after the Introduction part is the Overall Description. This section describes the general factors that affect our product and its requirements. The Specific Requirements part contains all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies hose requirements. The fourth section is the Data Model and Description part. At this section, the domain of the software is explain with the class diagrams. At the end the conclusion part and the appendix part, which contains the alloy code and corresponding diagrams, take place.

2 Overall description

2.1 Product Perspective

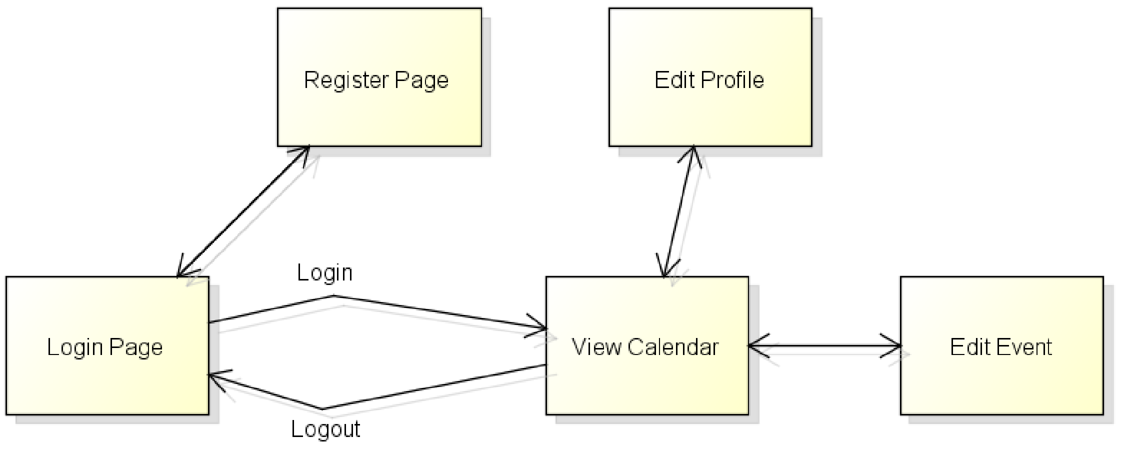
Our product is a web application that mainly focuses on events and their weather forecast information. It is a stand-alone product that is not depend on a larger system or included in one. It mainly consists of two parts: front-end and back-end. While front-end is responsible for providing all product functionalities to the user with useful, smooth and easy interfaces, back-end is responsible for managing all functionalities and communicating with the front-end in order to provide the best experience to the users.

2.1.1 System Interfaces

There are no system interfaces provided by our product.

2.1.2 User Interfaces

There are mainly five different user interfaces, these are register, login, edit event, edit profile, create event, and last but not least view calendar page. These pages will be explained in detail in later sections.



2.1.3 Hardware Interfaces

There are no hardware interfaces provided by our product.

2.1.4 Software Interfaces

Our application will be written with EJB (Enterprise Java Beans) framework, that means it will work with the JVM (Java Virtual Machine). Therefore, there are no specific operating system requirement. The application will work on the Glassfish Server v4.1. The required database system is MySQL with the version number 5.6.17.

2.1.5 Communication Interfaces

The MySQL server and our back-end communicates on the port 3306. Glassfish server uses the 8080 port for HTTP requests. The database and the server locate always at the same physical address.

2.1.6 Memory

The system should have at least 1 GB of primary memory and have at least 10 GB of free secondary memory space.

2.2 Product Functions

Our project provides users an opportunity to schedule their events by means of weather forecast information. By using the WeCalEvent system, users can edit and postpone their events if there exist a weather condition with which they are not willing to host their events. In addition to that, users are able to invite other users of the system to their events. System will notify the every participant user of the event in case of weather forecast change and suggest a better day to the event owner.

The first actor of the project is the user. However, the users may have different roles on the system depending on their relation with the event. There are basically three different roles:

- Event owner: This user can edit the event and can invite other users.
- Invited user: This user can accept or reject the invitation.
- Participant user: This user is able to see the event details even if it is private and receives notifications about the event.

The second actor is the system. System checks periodically whether the forecast is changed or not. It has the responsibility of notifying the users via application and e-mail.

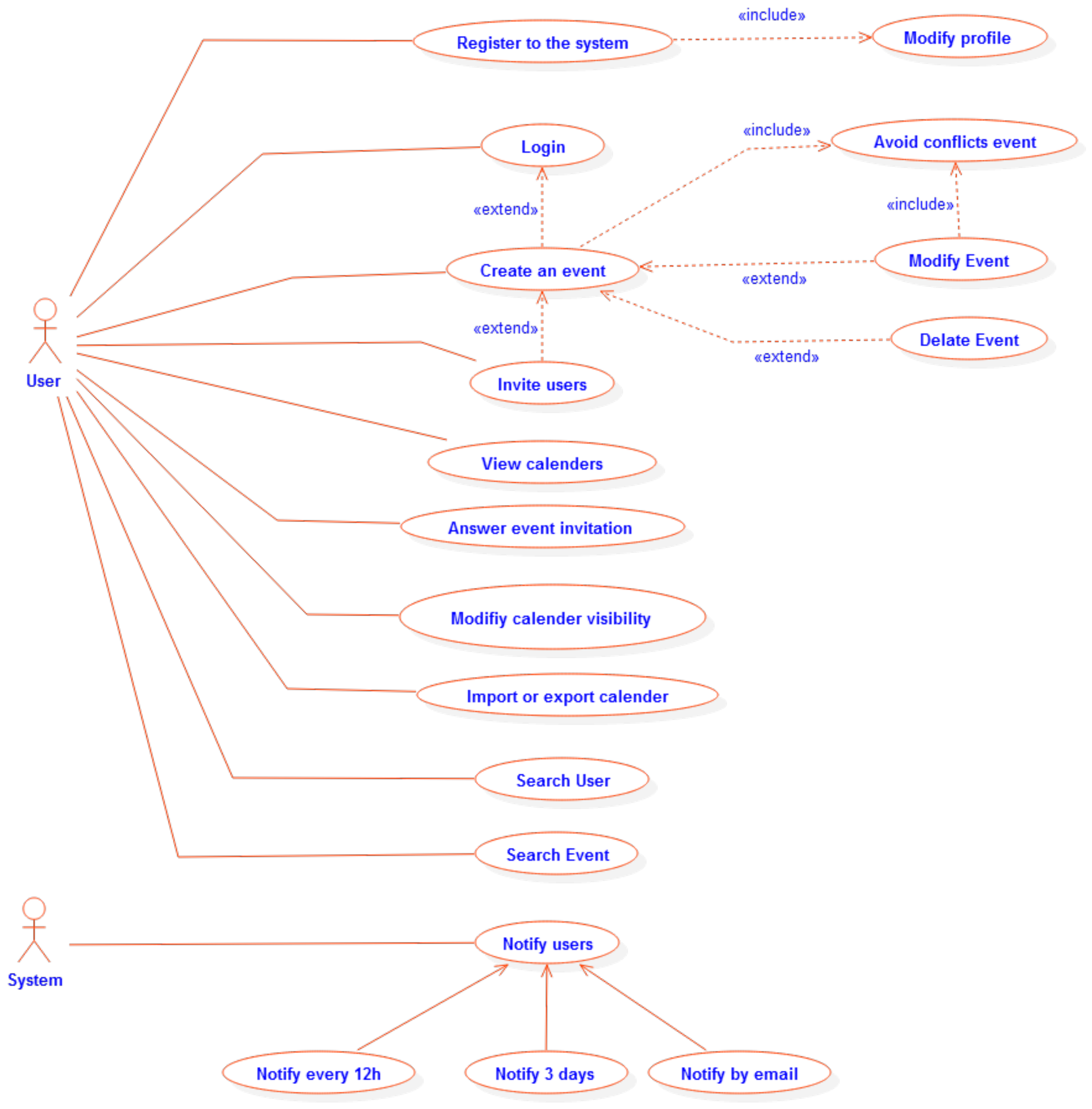


Figure 1: Use Cases

2.3 Constraints

The system is constrained by the server machine on which our application works. The forecast information is constrained by the external system used for weather forecast. Furthermore, the internet constraints the application because the weather data is derived from the internet as the web interface is requested via HTTP Request through Internet.

2.4 Assumptions

The weather forecast is provided for the next 16 days. If a user is creating an event within this period of time and the weather is not good for the event on the chosen date the user will still be able to create an event on this date. In this case he will be notified that the weather conditions do not match the needed ones.

The event organiser will be notified if the weather is "bad" 3 days before the event. If there is no "good" weather within next 13 days we notify user but we cannot suggest any other day .

The event creator can choose which weather conditions to be warned about. The weather conditions list is: cloudy, rainy, snowy, clear, unknown.

The event organiser will be able to invite users after the event creation.

Only event starting date's weather forecast will be considered.

Users are able to import only new events.

3 Requirements

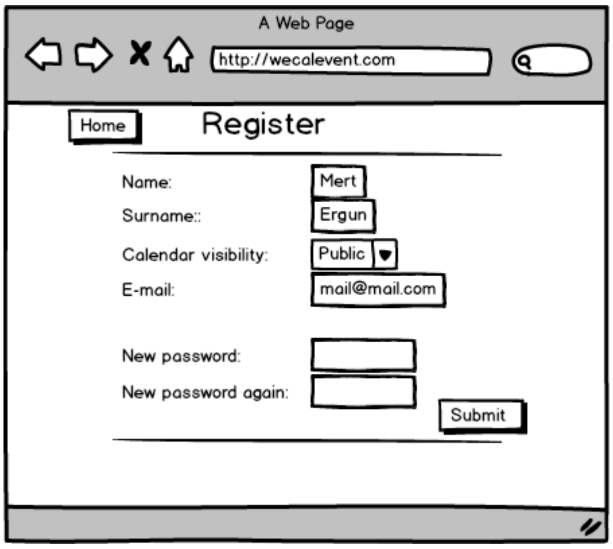
3.1 External Interface Requirements

3.1.1 Registration screen

This mockup is related to the registration use case. It explains the registration process. To register to the system a user should fill following fields:

- Name
- Surname
- Calendar visibility
- E-mail
- Password

After filling in all te fields a user will have to press "submit" button.



Name: Mert
Surname:: Ergun
Calendar visibility: Public
E-mail: mail@mail.com
New password:
New password again:

Figure 2: Registration screen

3.1.2 Login screen

This mockup is related to the login to the system use case. It explains the login process. Registered user can login into the system filling in "e-mail" and "password" fields.

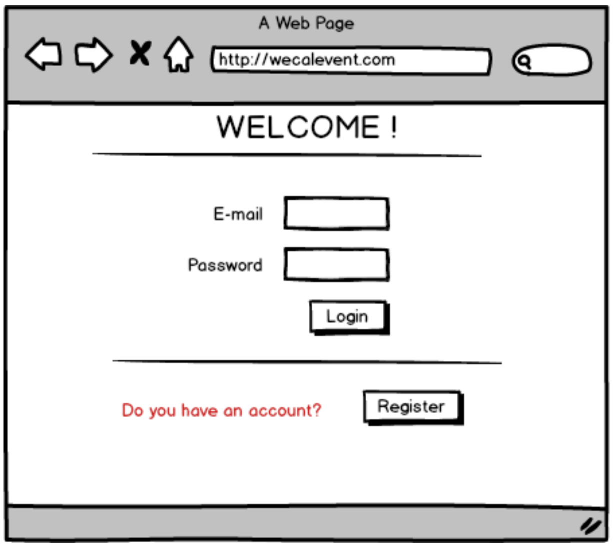
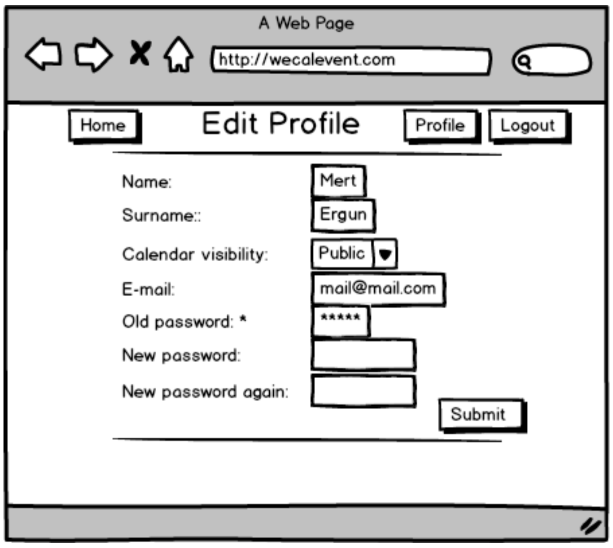


Figure 3: Login screen

3.1.3 Edit profile screen

This mockup is related to the edit profile use case. It explains the edit profile process. Previously registered users are able to edit all the fields of their profile filled in during registration and calendar visibility.

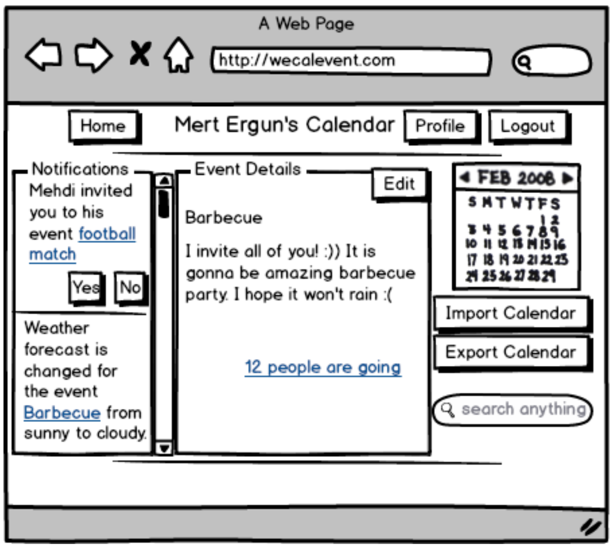


Name: Mert
Surname:: Ergun
Calendar visibility: Public
E-mail: mail@mail.com
Old password: * *****
New password:
New password again:

Figure 4: Edit profile screen

3.1.4 View calendar screen

This mockup is related to the "View calendars", "Import or Export calendar", "Search event", "Search user" use case. It explains the view calendar process. Users will be able to see events they are going to participate in and events they have been invited to in a view calendar screen. On its left side users will see notification about the events. By clicking on any event related to this user she/he will see detailed information of this event and link to the list of users which have already accepted the event. By choosing anyone from this list the user will see a calendar of this person in case if it's public. The user will be able to export or import her/his calendar and search for another user or event.



Mehdi invited you to his event football match

Weather forecast is changed for the event Barbecue from sunny to cloudy.

Barbecue

I invite all of you! :)) It is gonna be amazing barbecue party. I hope it won't rain :(

12 people are going

Figure 5: View calendar screen

3.1.5 Create an event screen

This mockup is related to the "Create an event" use case. It explains the event creation process. In event creation screen (Figure 5), information about the event which will be created is given by the user. These information are event name, event description, time, type, visibility (public, private), location, "bad" weather conditions and invited people list .

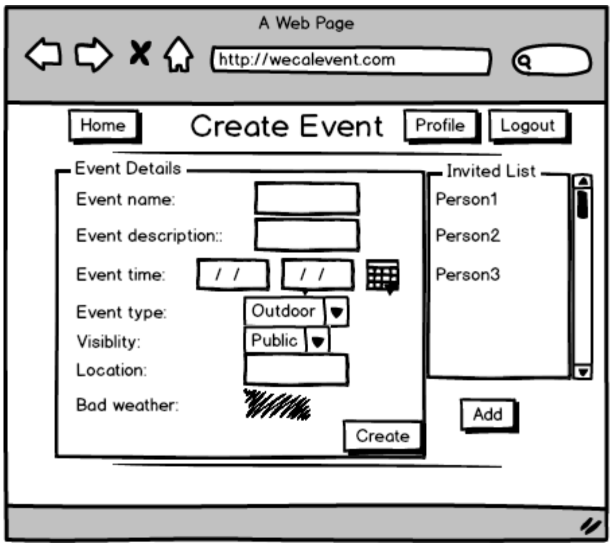


Figure 6: Create event screen

3.1.6 Edit event screen

This mockup is related to the "Edit an event" use case. It explains the event edition process. User can invite other users to the events she/he has created even after creation.

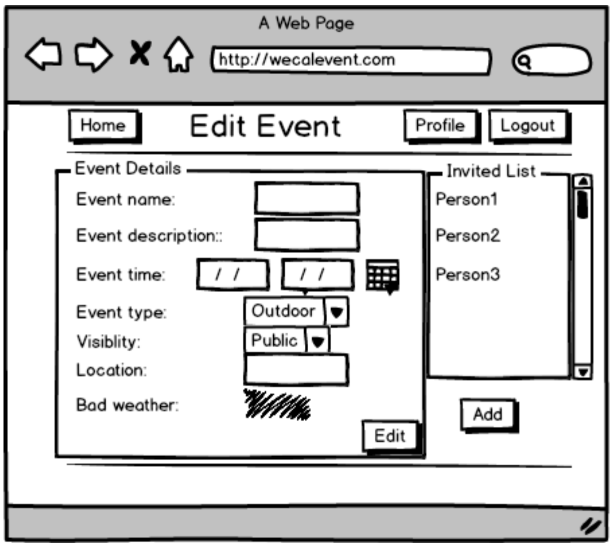


Figure 7: Manage event screen

3.2 Functional Requirements

3.2.1 Registration

Name	Register to the system
Actor	User
Entry conditions	There are no entry conditions.
Flow of events	• The user opens the web page. • The system shows the web page to user. • The user clicks on “Register” button. • The system shows the form page to user. • The user fills the form. • The user clicks on "submit" button.
Exit conditions	• The system shows a success message. • The system loads the home page.
Exceptions	• The user enters wrong inputs in some fields, then an error message is shown. • The user leaves compulsory fields empty, then an error message is shown. • The user refreshes the page without submit, then data is lost.

3.2.2 Login

Name	Login to the system
Actor	User
Entry conditions	User must be registered into the system .
Flow of events	• The user opens the web page. • The system shows the home page to user. • The user enters his email and password in the input form. • The user clicks on "Login" button. • The system shows the calendar page.
Exit conditions	There are no exit conditions.
Exceptions	The user enters wrong email and/or password, then an error message is shown.

3.2.3 Create an event

Name	Create an event
Actor	User
Entry conditions	The user must be logged in.
Flow of events	• The users clicks on "create an event" button. • The system loads event creation page. • The user fills the fields with event information. • The system warns the user in case of conflict with an another event and suggests to user free time. • The user invites other users. • The user clicks on "create" button.
Exit conditions	• The system shows an alert message. • The calendar page is refreshed with new event.
Exceptions	• The user enters wrong inputs in some fields, then an error message is shown. • The user leaves compulsory fields empty, then an error message is shown. • The user refresh the page without submit, then data is lost.

3.2.4 Edit event

Name	Edit event
Actor	User
Entry conditions	• The user must be logged in. • The event must be created.
Flow of events	• The event creator clicks on an event. • The system loads event creation page with existing information. • The event creator updates the event information. • The system warns the user in case of conflict with an another event and suggests to user free time. • The user clicks on "Edit" button.
Exit conditions	• The system shows an alert message. • The calendar page is refreshed with new event.
Exceptions	• The user enters wrong inputs in some fields, then an error message is shown. • The user leaves compulsory fields empty, then an error message is shown. • The user refreshes the page without submit, then data is lost.

3.2.5 Delete an event

Name	Delete an event
Actor	User
Entry conditions	• The user must be logged in. • The event must be created.
Flow of events	• The event creator clicks on an event. • The system loads event creation page with existing information. • The event creator clicks on “Delete” button. • The system shows a dialog box. • The user clicks on "yes" button.
Exit conditions	• The system shows an alert message. • The calendar page is refreshed without the event.
Exceptions	• The user clicks on "NO" button.

3.2.6 Invite users

Name	Invite users
Actor	User
Entry conditions	• The user must be logged in. • The event must be created. • The user must be the event owner.
Flow of events	• The user selects one of his events. • The user clicks on "invite people" button. • The user selects users to invite from the list.
Exit conditions	There are no exit conditions
Exceptions	There are no exceptions.

3.2.7 Answer event invitation

Name	Answer event invitation
Actor	User
Entry conditions	• The user must be logged in. • The event must be created. • The user must be invited.
Flow of events	The user accepts/declines to participate in event(s)
Exit conditions	There are no exit conditions
Exceptions	There are no exceptions.

3.2.8 Search user

Name	Search user
Actor	User
Entry conditions	The user must be logged in.
Flow of events	The user inputs the name to search for.
Exit conditions	User found.
Exceptions	No user found with that name.

3.2.9 Search event

Name	Search event
Actor	User
Entry conditions	The user must be logged in.
Flow of events	The user input the event to search for.
Exit conditions	Event found.
Exceptions	No event found with that name.

3.2.10 Edit profile

Name	Edit profile
Actor	User
Entry conditions	The user must be logged in.
Flow of events	• The event creator clicks on "Edit profile". • The system loads personal information page with existing information. • The user updates his information. • The user clicks on "submit" button. • The system shows the alert message. • The user clicks on "yes" button.
Exit conditions	• The system shows an alert message. • The profile page is refreshed with new personal information.
Exceptions	• The user enters wrong inputs in some fields, then an error message is shown. • The user leaves compulsory fields empty, then an error message is shown. • The user refresh the page without submit, then data is lost.

3.2.11 View Calendars

Name	View Calendars
Actor	User
Entry conditions	The user must be logged in.
Flow of events	• The user selects another user. • The system shows the calendar of that user.
Exit conditions	There are no exit conditions
Exceptions	The calendar is private.

3.2.12 Manage Calendar Visibility

Name	Manage Calendar Visibility
Actor	User
Entry conditions	The user must be logged in.
Flow of events	• The user clicks on "edit profile" button. • The user chooses to put his calendar public/private.
Exit conditions	There are no exit conditions
Exceptions	There are no exceptions.

3.2.13 Import or Export Calendar

Name	Import or Export Calendar
Actor	User
Entry conditions	The user must be logged in.
Flow of events	• The user click on "import calendar" or "export calendar" buttons. • If the user wants to export his calendar, the user clicks the export button. Otherwise, the user provides a valid document to import events to his calendar and the system saves the provided calendar..
Exit conditions	There are no exit conditions
Exceptions	If the calendar is not in the requested format, the system gives an error.

3.2.14 Notify Users

Name	Notify Users
Actor	System
Entry conditions	• The user must be logged in. • The user must be owner of event. • The user must be participant in that event. .
Flow of events	• There are three conditions at which the user is notified via e-mail and a notification in the application when they log in. – The weather forecast will be controlled every 12 hours. In case of a forecast change for an event, every user that participates in the particular event will be notified. – The system checks exactly 3 days before the event how is the weather on the event time. If there is a bad weather condition, the system suggests the closest possible day with a good weather condition to the owner of the event. – The system checks exactly 1 day before the event how is the weather on the event time. If there is a bad weather condition, the system notifies every user that participates in that event.
Exit conditions	There are no exit conditions
Exceptions	There are no exceptions.

3.3 Sequence diagrams

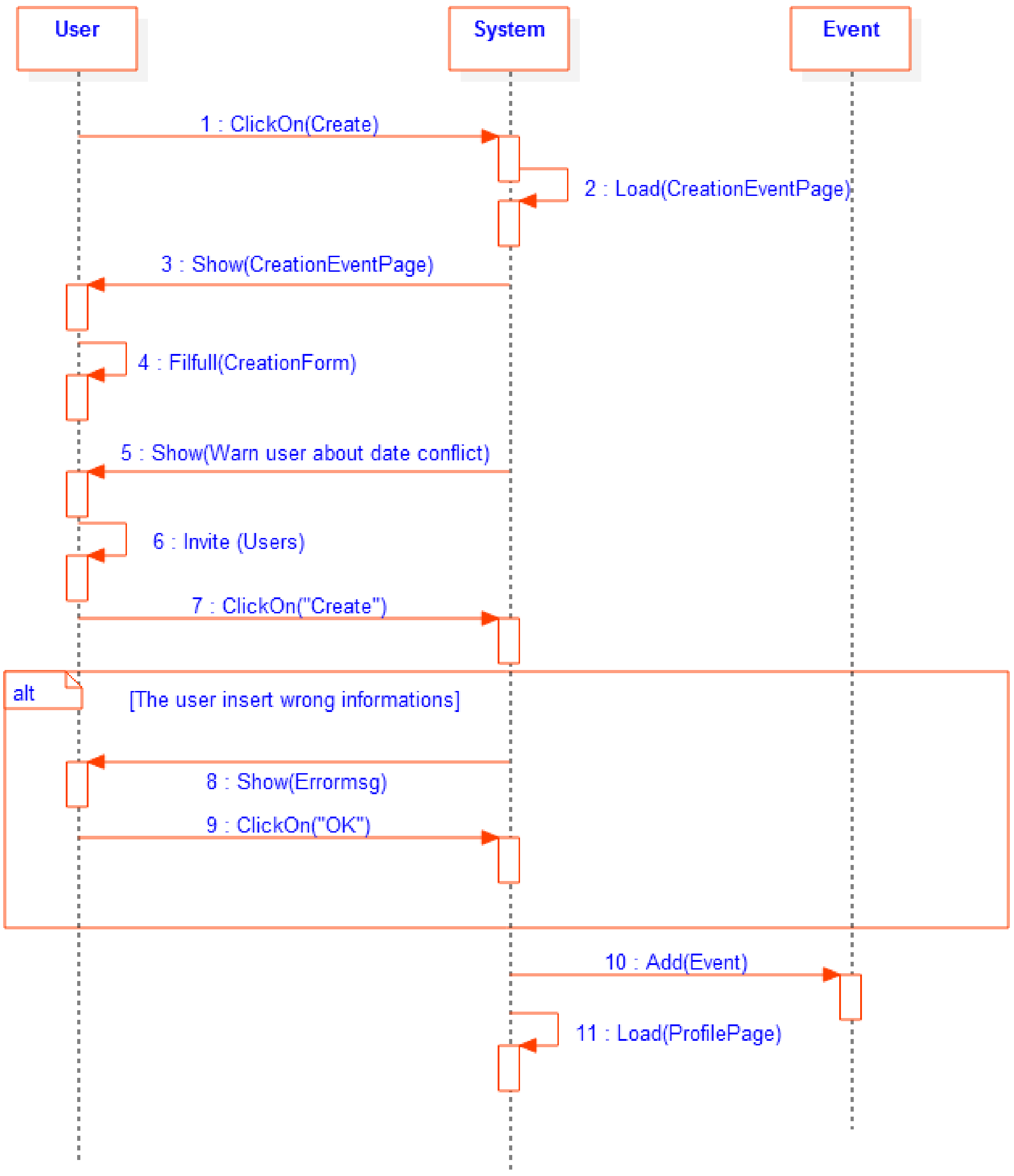

User
System
Event
1 : ClickOn(Create)
2 : Load(CreationEventPage)
3 : Show(CreationEventPage)
4 : Filfull(CreationForm)
5 : Show(Warn user about date conflict)
6 : Invite (Users)
7 : ClickOn("Create")
alt
[The user insert wrong informations]
8 : Show(Errormsg)
9 : ClickOn("OK")
10 : Add(Event)
11 : Load(ProfilePage)

Figure 8: Create

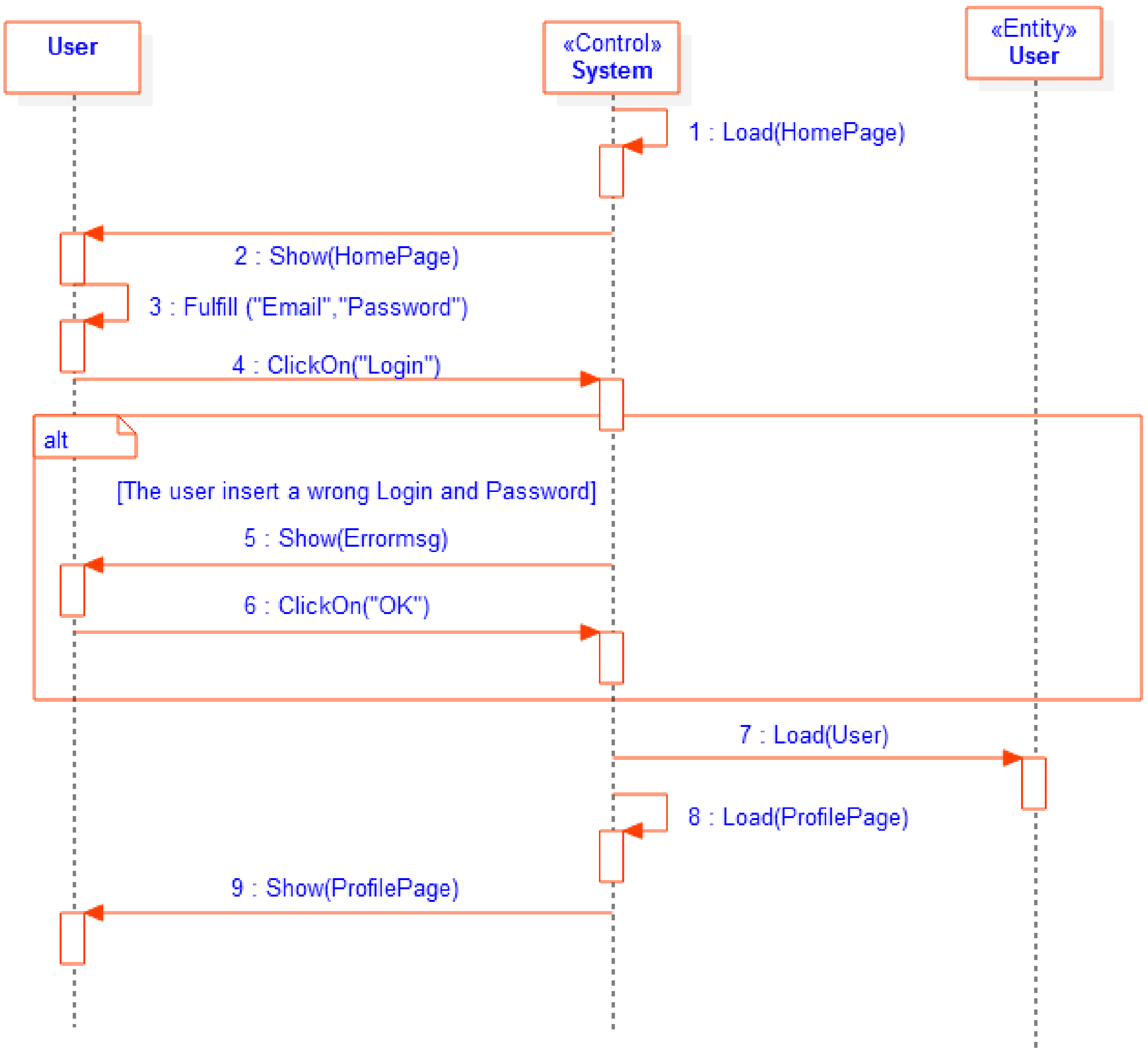


Figure 9: Login

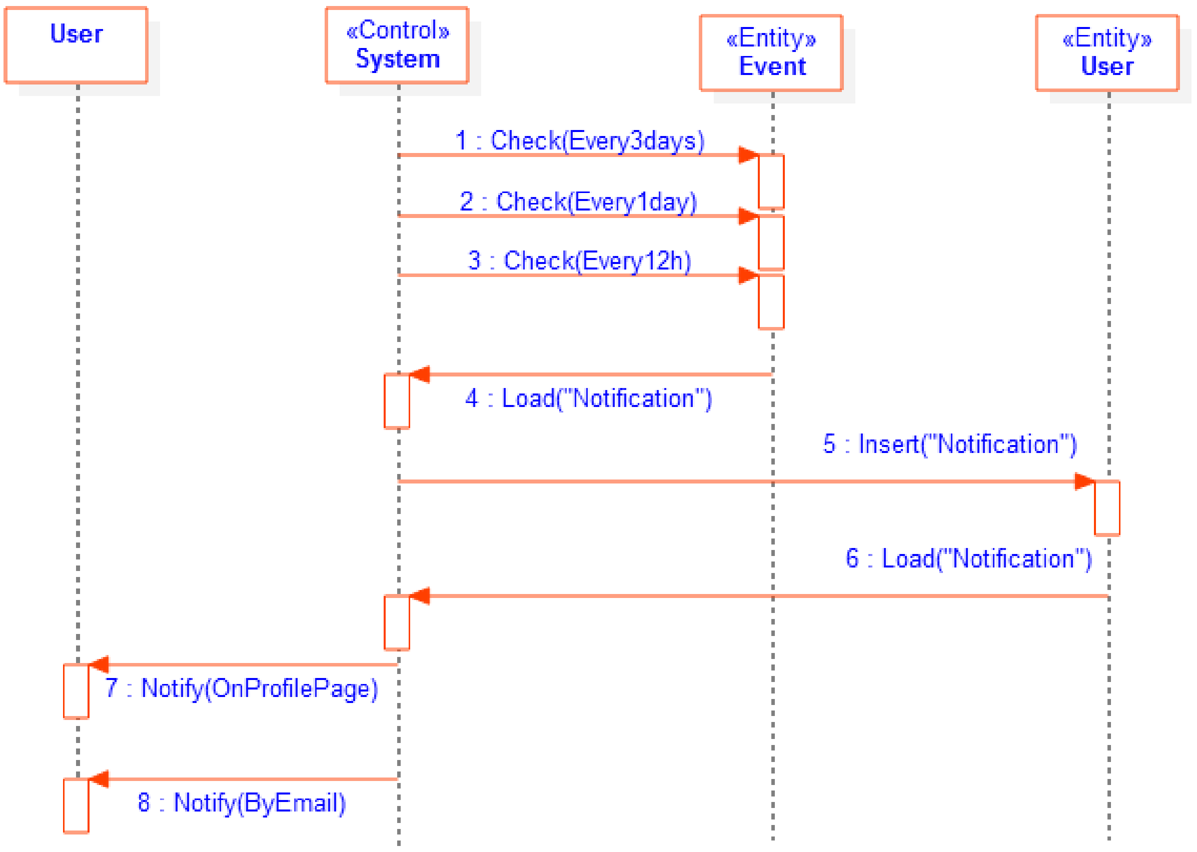


Figure 10: Notify

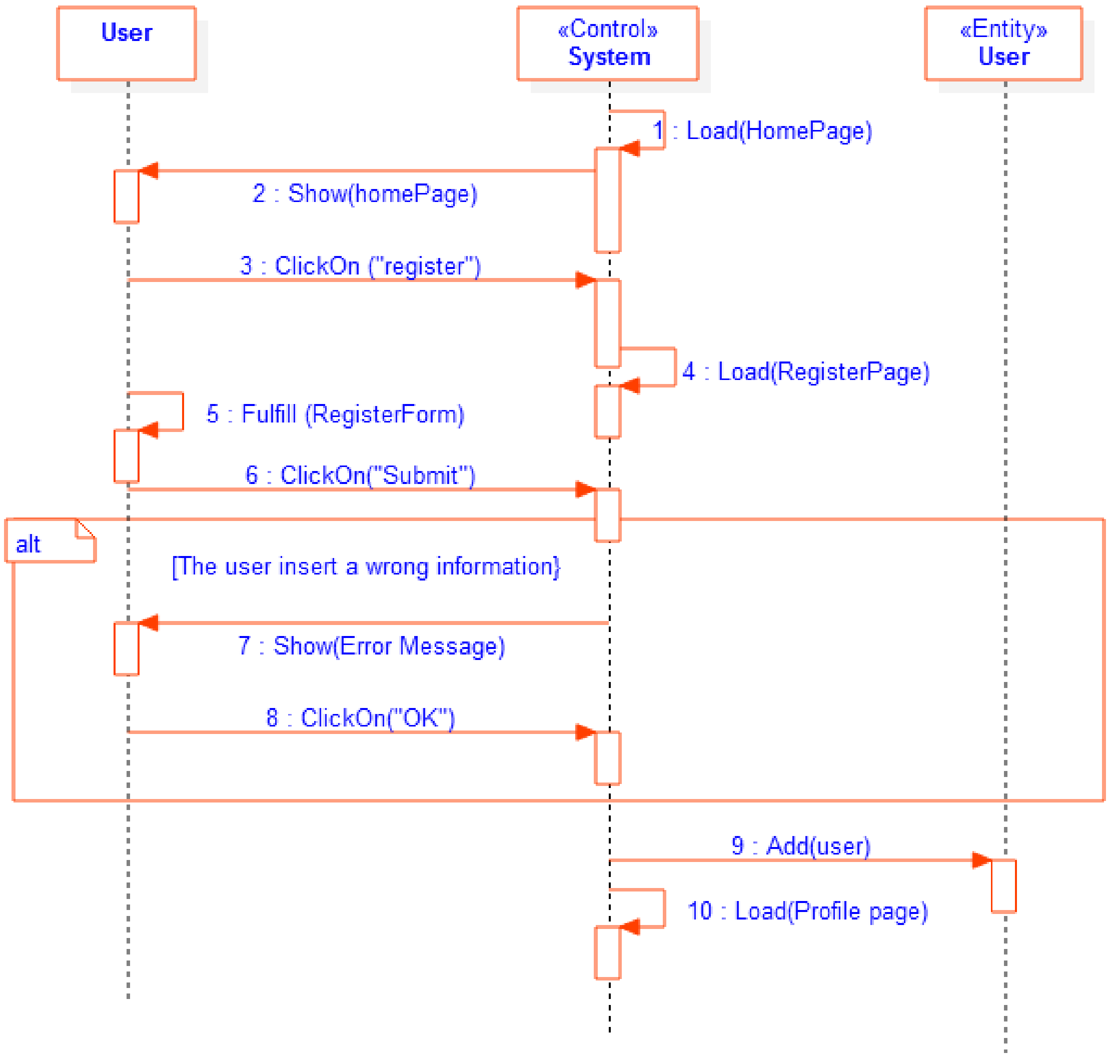


Figure 11: Register

3.4 StateChart diagrams

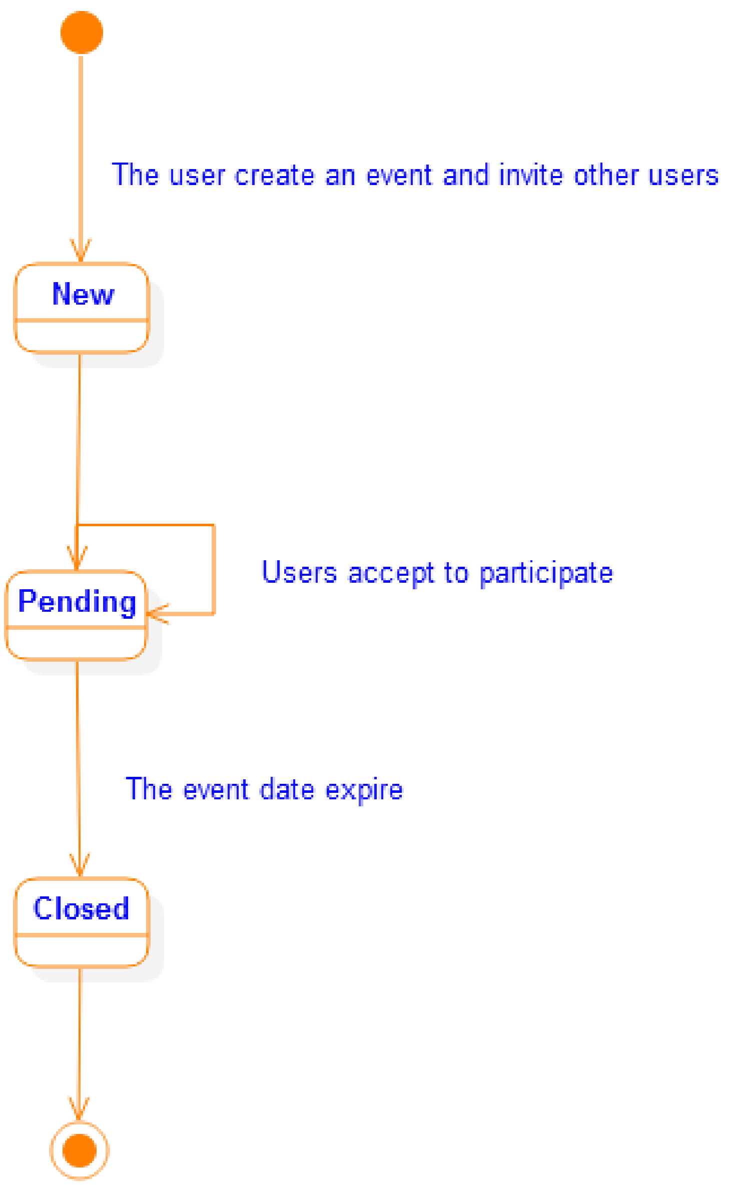

The user create an event and invite other users
New
Users accept to participate
Pending
The event date expire
Closed

Figure 12: Create Event

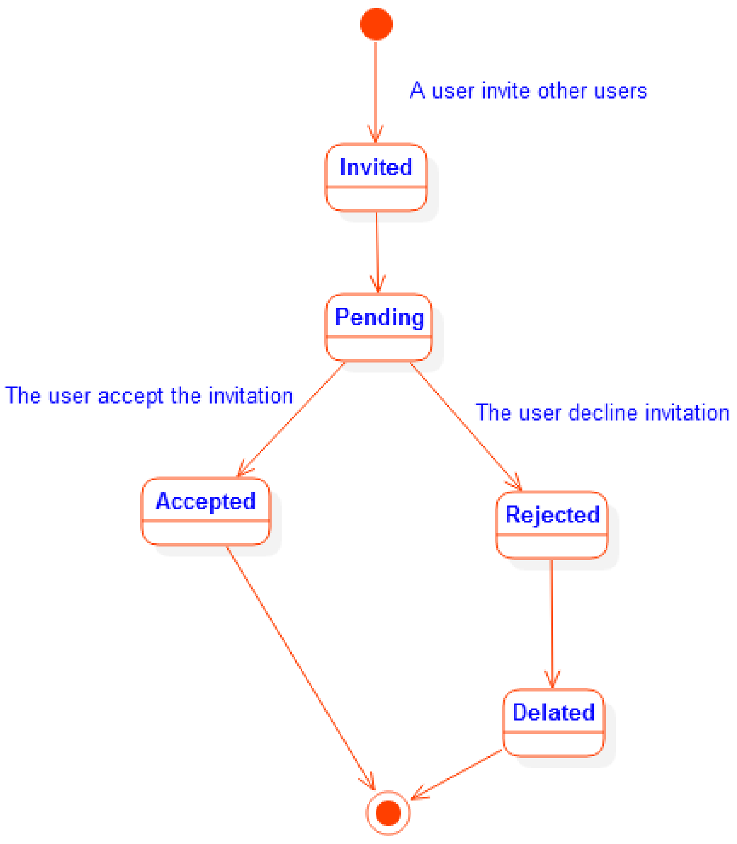

A user invite other users
Invited
Pending
The user accept the invitation
The user decline invitation
Accepted
Rejected
Delated

Figure 13: Invite User

3.5 Non-functional Requirements

3.5.1 Reliability

The system should be able to behave consistently in a user-acceptable manner when operating within the environment for which the system was intended.

3.5.2 Availability

The system should be able to deliver services when requested.

3.5.3 Security

The system should be able to protect itself against accidental or deliberate intrusion.

3.5.4 Maintainability

Maintainability is the parameter concerned with how the system in use can be restored after a failure. The system should periodically have back ups in order not to loose any information.

3.5.5 Portability

The application should easily be transferred from one computer environment to another.

4 Data Model and Description

There are four main data classes which are Event, User and Weather classes.

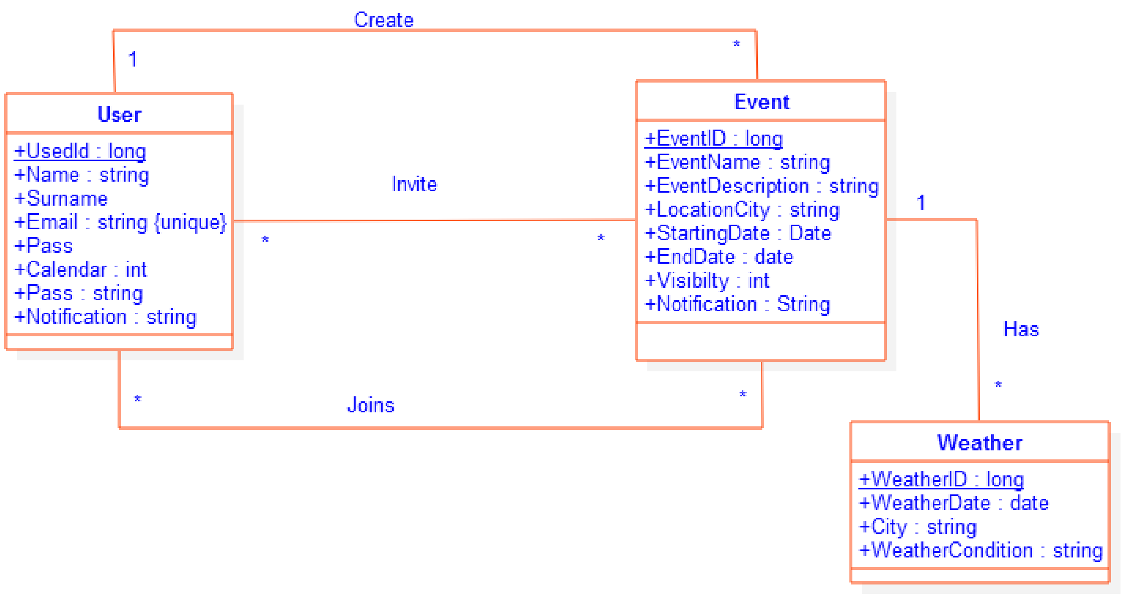


Figure 14: Class Diagram

4.1 User

User class stores user details, calendar visibility and notifications received.

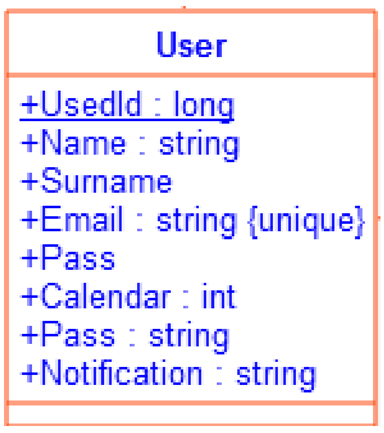


Figure 15: User Class Diagram

4.2 Event

This class stores the event details, event visibility and the notifications for this event to its participant users. There are three different relations between this class and the User class. Two Many to many relation called invite and joins and one one to many relation called create.

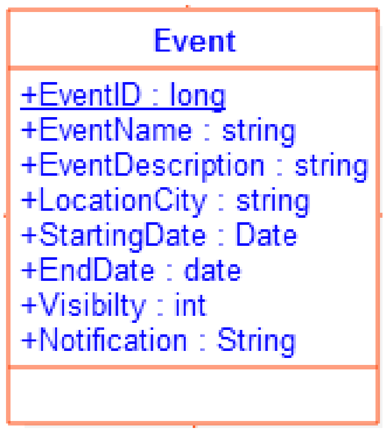


Figure 16: Event Class Diagram

4.3 Weather

This class stores the weather information for events. Every event has at least one weather information so, there is a one to many relation.

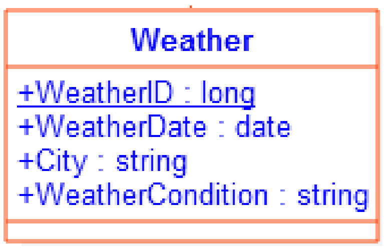


Figure 17: Weather Class Diagram

5 Planning

Our team consists of three students. Since it is a small team it is possible to decide and develop everything together. However, ever member has her/his responsibilities stated below with the working hours until the end of requirement document.

Mohamed Mehdi Kaabi: Back-end development, 20h

Nazrin Javadova: Documentation, 22h

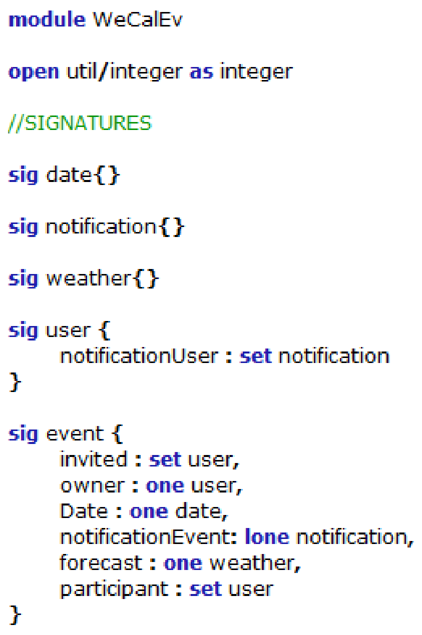
Mert Ergun: Front-end design, 20h

6 Conclusion

To sum up, all required information about the requirements of this project has been given. This project will provide an event management interface in the weather forecast basis with its excellent and usable interfaces to its users. Now the project is ready to go on with determining more detailed design of the product.

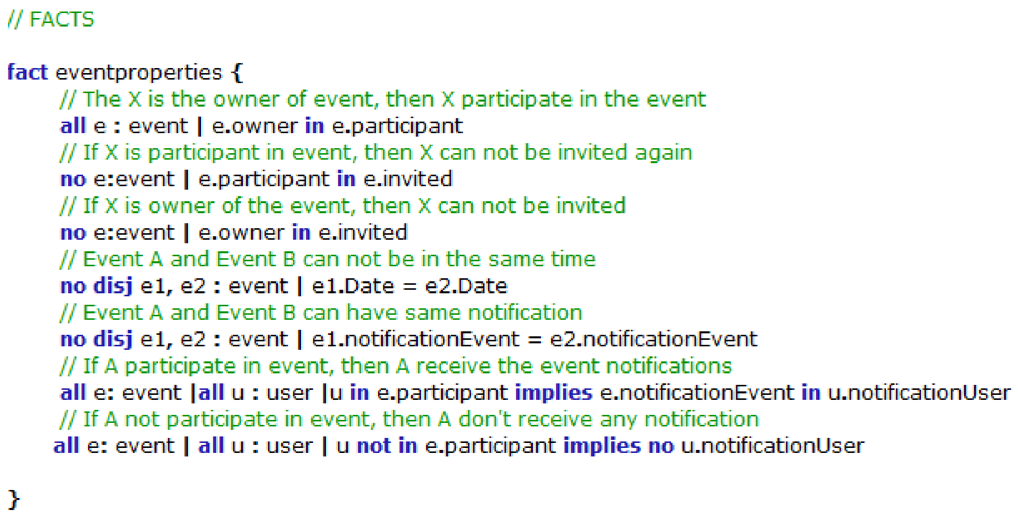
7 Alloy

In this section, the relations, constrains and rules between elements of our data model are given. Here the alloy signatures and facts follow:



```
module WeCalEv

open util/integer as integer

//SIGNATURES

sig date{}

sig notification{}

sig weather{}

sig user {
        notificationUser : set notification
}

sig event {
        invited : set user,
        owner : one user,
        Date : one date,
        notificationEvent: lone notification,
        forecast : one weather,
        participant : set user
}
```

}

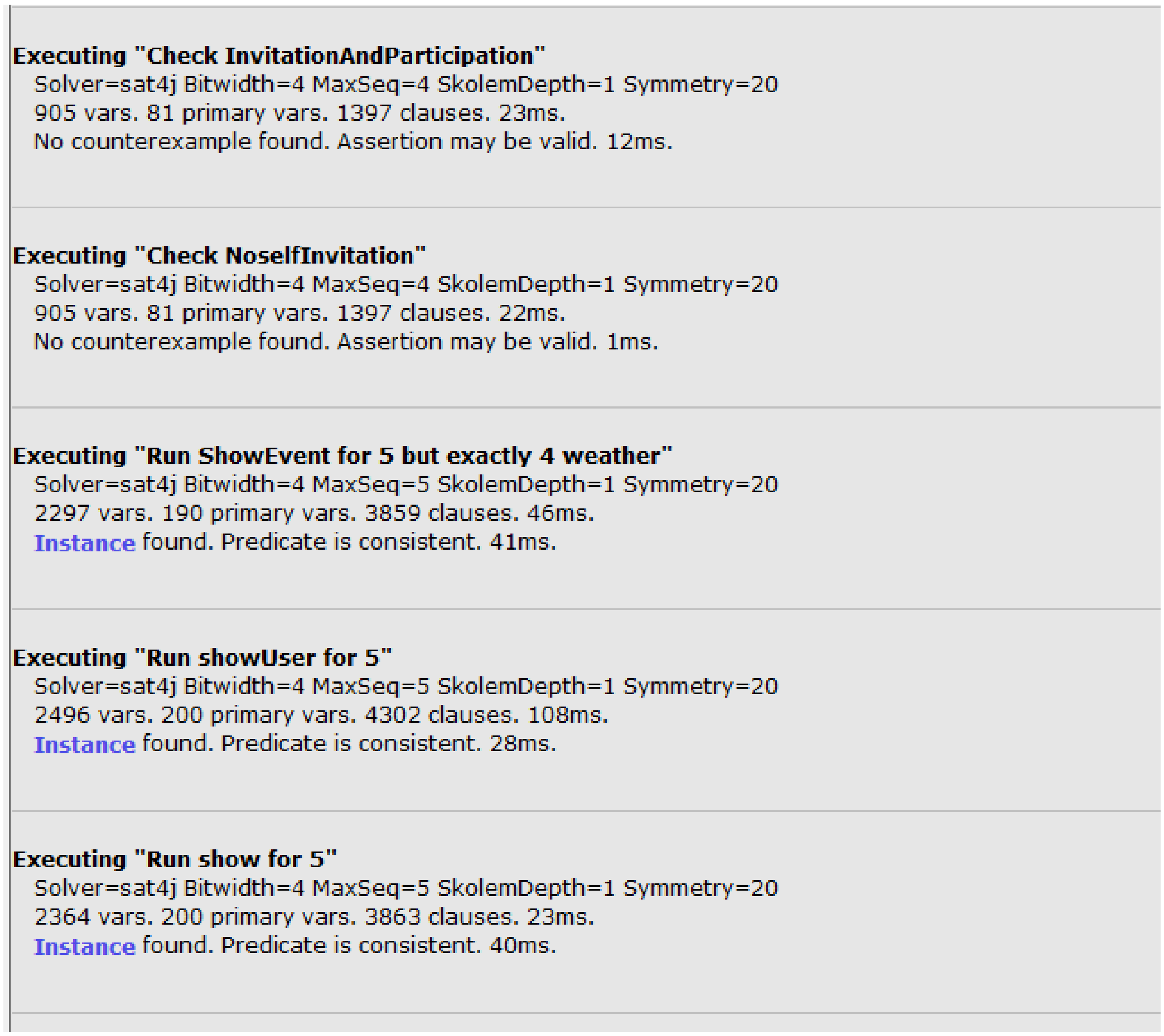


```
// FACTS

fact eventproperties {
        // The X is the owner of event, then X participate in the event
        all e : event | e.owner in e.participant
        // If X is participant in event, then X can not be invited again
        no e:event | e.participant in e.invited
        // If X is owner of the event, then X can not be invited
        no e:event | e.owner in e.invited
        // Event A and Event B can not be in the same time
        no disj e1, e2 : event | e1.Date = e2.Date
        // Event A and Event B can have same notification
        no disj e1, e2 : event | e1.notificationEvent = e2.notificationEvent
        // If A participate in event, then A receive the event notifications
        all e: event |all u : user |u in e.participant implies e.notificationEvent in u.notificationUser
        // If A not participate in event, then A don't receive any notification
       all e: event | all u : user | u not in e.participant implies no u.notificationUser

}
```

}

```
//ASSERTIONS

assert InvitationAndParticipation {
	no e:event | e.participant in e.invited
}
check InvitationAndParticipation

assert NoselfInvitation {
	no e:event | e.owner in e.invited
}
check NoselfInvitation


//PREDICATES
pred ShowEvent() {
	#date = #event
}

run ShowEvent for 5  but exactly 4 weather

pred showUser(){
	#user > 1
	#event > 1
	#weather>1
	#weather<5
	#date = #event
	#event >= #notification
}

run showUser for 5

pred show(){

}

run show for 5
}
```

The execution log of the above code for every run configurations is given below:



```
Executing "Check InvitationAndParticipation"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   905 vars. 81 primary vars. 1397 clauses. 23ms.
   No counterexample found. Assertion may be valid. 12ms.

Executing "Check NoselfInvitation"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   905 vars. 81 primary vars. 1397 clauses. 22ms.
   No counterexample found. Assertion may be valid. 1ms.

Executing "Run ShowEvent for 5 but exactly 4 weather"
   Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
   2297 vars. 190 primary vars. 3859 clauses. 46ms.
   Instance found. Predicate is consistent. 41ms.

Executing "Run showUser for 5"
   Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
   2496 vars. 200 primary vars. 4302 clauses. 108ms.
   Instance found. Predicate is consistent. 28ms.

Executing "Run show for 5"
   Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
   2364 vars. 200 primary vars. 3863 clauses. 23ms.
   Instance found. Predicate is consistent. 40ms.
```

This static diagram is called metadata model. It shows all possible relations between elements without considering the facts.

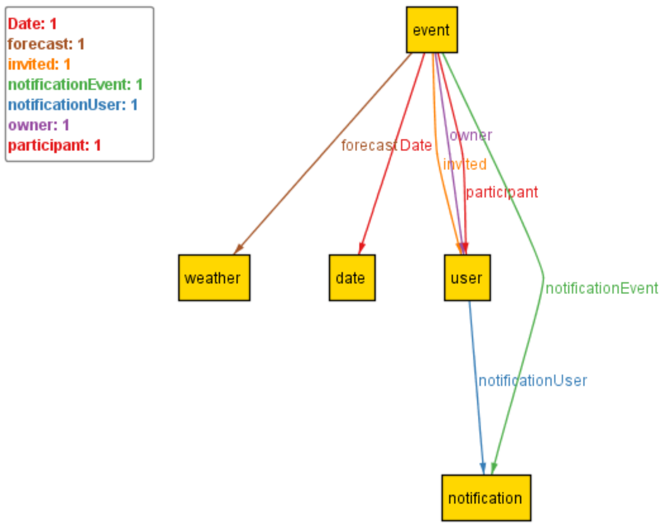
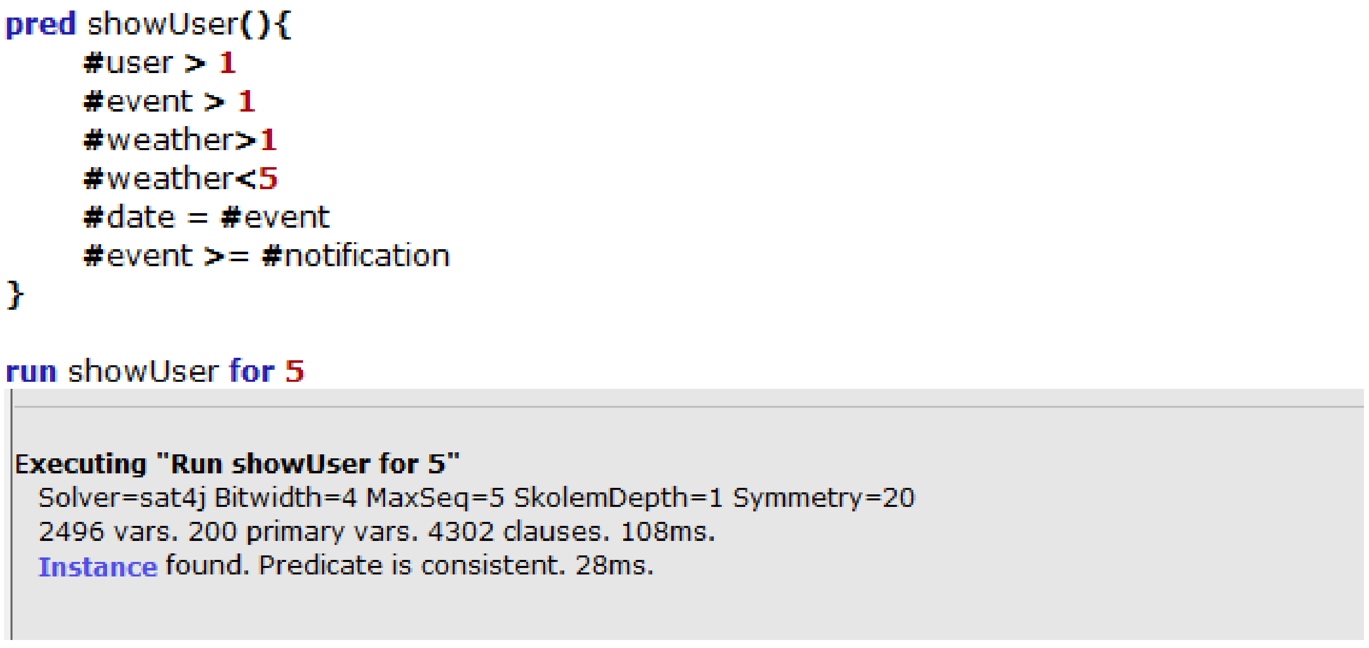


Figure 18: WeCalEvent metamodel



```
pred showUser(){
	#user > 1
	#event > 1
	#weather>1
	#weather<5
	#date = #event
	#event >= #notification
}

run showUser for 5
```

```
Executing "Run showUser for 5"
   Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
   2496 vars. 200 primary vars. 4302 clauses. 108ms.
   Instance found. Predicate is consistent. 28ms.
```

We can see below the possible states:

- If a user is invited to an event, he/she cannot be invited again and cannot be in the participant list.

- The owner of an event is also a participant.

- Two events can not happen on the same date.

- Each event has a weather forecast.

- User must be a participant of the event to receive a notification.

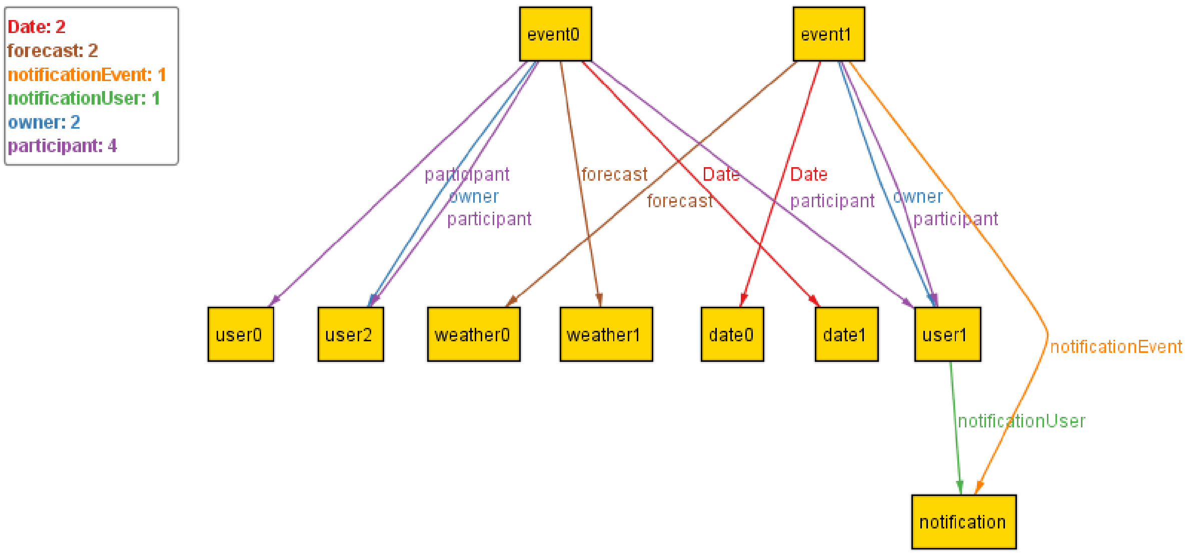


Figure 19: WeCalEvent Instance(simplified)

This diagram provides an example of more complex relations with a couple elements more:

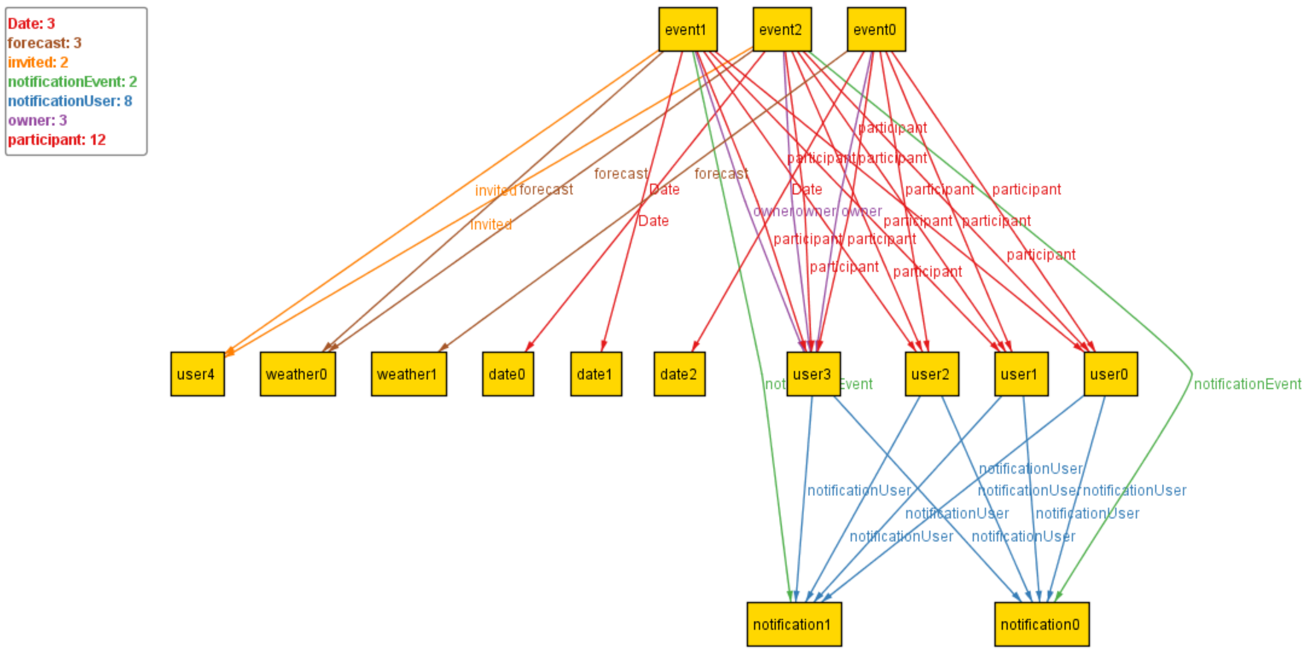


Figure 20: WeCalEvent Instance complex view