POLITECNICO DI MILANO

SOFTWARE ENGINEERING 2

# Requirements Analysis and Specification Document

Mirjam Škarica

Milan, November 2014.

# Table of Contents

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to provide a comprehensive description of the MeteoCal system. It's aim is to communicate what the software should do and identify the capabilities and characteristics of the system being developed. This document is meant for everyone included in the production of the software.

## 1.2 Scope

The aim of this project is to develop and implement MeteoCal, a weather based online calendar for helping people schedule their personal events and avoiding bad weather conditions in case of outdoor activities.

The developed system should allow the registration of new users. Users, once logged in, should be able to:

- create, delete and update events
- accept or decline event invitations from other users
- receive and read notifications about the events they are participating in

Each event should contain information about the location, the time and whether it is an outdoor or an indoor event. Only the user who created the

event (the organizer) will be able to update and delete the said event. He is also the only one who can invite other users to the event upon event's creation. Invited users can only either accept or decline the invitation.

Whenever an event is saved, the system should automatically enrich the event with weather forecast information, if available. Also, in case of bad weather conditions for outdoor events, the system should notify all participants one day before the event. Notifications will be received by the users when they log into the system.

## 1.3 Glossary

The following are the definitions of some commonly used phrases throughout the document:

- **Calendar:** a graphical representation of a chart showing the days, weeks, and months of a particular year and possibly also events taking place on those days
- **Event:** a planned social occasion organized by a user. It is classified either as an outdoor or an indoor activity and has a specified location, start time and duration
- **Guest:** a person accessing a system that has either never registered of hasn't logged in yet. Guest has only two available options, to log in or to register for the first time.
- **Invitation:** organizer's event participation request sent out to other users. A user who receives an invitation can either accept or reject it.
- **Organizer:** (of an event) a user who created that particular event

- **Participant:** (of a specific event) a user who accepted organizer's invitation to that particular event
- **User:** a person already registered and logged into the system.
- **Weather:** the state of the atmosphere at a given time and place. In this document it will be used in conjunction with phrases 'good' and 'bad' to describe suitable and not suitable weather, respectively, for the event in question

# 2. Overall description

## 2.1 Product perspective

MeteoCal is a web application that provides users a service described in section 1.2. The software will be developed using a server-client model. The server side contains the application's logic and is used to interact with permanent storage, serve pages to the client and process user input. The web client consists of dynamic web pages which provide an easy-to-use graphical interface and the web browser through which they are accessed.

Considering MeteoCal is a web app, it is platform independent. The only requirement is users having a web browser installed on the device of their choice.

## 2.2 Identifying stakeholders

There are tree distinct interest groups of people regarding this project:

- The **professors** who have assigned the project and who expect it to be developed in a way which satisfies the given requirements respecting the given deadlines. Their interests go beyond just working software, they are interested in conveying an thorough understanding of the underlying development process

- The **users** who will use the application and rely on it for scheduling and managing the events they are participating in
- The **developer** group, in this case one person, responsible for the actual design and development of the software

## 2.3 User Characteristics

MeteoCal is expected to have users across a wide range of demographics, meaning users of any age, gender and educational background. Still, because of the ubiquitous nature of internet and social media, it is assumed that people using our software do have the basic web browsing skills.

## 2.4 Actors identifying

Two possible actors interacting with our system are the following:

- **Guest:** person accessing a system that has either never registered of hasn't logged in yet. Guest can only access the initial page from where he has only two available options, to log in or to sign up for the first time.
- **User:** a person already registered and logged into the system. User can use all of the features offered by the application

## 2.5 Goals

Having possible users in mind, MeteoCal should have these features:

- registering new users
- managing weather information for all events
- sending weather based notifications regarding an event to all of it's participants
- for each user it should provide a calendar
- for each user it should provide following actions:
  - creating new events
  - modifying and deleting events which they created themselves
  - inviting other users to events which they created themselves
  - accepting or declining event invitations

## 2.6 Domain properties

We suppose that these conditions hold in the analyzed world:

- each event is held in one specific location and at one specific time
- each event is either outdoor or indoor
- a user creating the event knows and provides correct information needed about the event (time, place, indoor/outdoor)
- a user knows the usernames of other users he wishes to invite
- a user creating the event is considered a participant
- a user will want to create only future events
- a person can be only in one place at a specific point in time
- a user can receive a lot of invitations for events happening at the same

time, but can only accept one

- weather forecasts are reliable

## 2.7 Assumptions

Considering that there were some ambiguities in the specification document, the following facts are assumed:

- weather forecast is checked 2 times. First, upon event creation in order to enrich the created event, and second time is one day before the event.
- bad weather means any weather that is unpleasantly cold, damp or windy
- users can accept, reject or leave invitation pending
- users can change their password and email

# 3. Requirements

Assuming the domain properties stated in section 2.6 hold, we can derive software requirements from the goals defined in section 2.5.

## 3.1 Functional requirements

Functional requirements are defined for the system and for each actor identified in 2.4.

- For a **Guest** the system has to provide 2 functionalists:
  - **sign up:** a guest will have to provide some basic information (email, username and password) in order to be registered as a user
  - **log in:** after entering correct identification information ( username and password) the guest will be logged in and redirected to his homepage

- For a **User** the system has to provide the following functionality:
  - **Sign out**
  - **View and browse his calendar :**

    the user has to have the ability to access his calendar at any time, browse through it, and display all the events for a particular day by clicking on the desired date in the calendar

- **Managing events:**
  - create a new event
  - update events which users themselves created
  - delete events which users themselves created
  - invite other users to events which users themselves created
- **Manage invitations:**
  - view received invitations
  - accept or decline invitations
- **Manage notifications:**
  - view weather based notifications about the events the users are participating in
- **Manage settings:**
  - change email
  - change password

- The **system** should :
  - Enrich the event with weather information upon it's creation
  - Update the weather forecast one day before the event and notify all participants in case of bad weather conditions

## 3.2 Non-functional requirements

**User Interface** of the web application should be intuitive so even the non-technically savvy users can use the system as simply and efficiently as

possible.

**Availability**: the application should be available handle user's request at all times using any device with an installed web browser.

# 4. Scenarios

## Event Creation

Jolene has been a software developer for 10 years. Being a female engineer in a male dominated field has compelled her to try help close the gender gap. Since she hasn't got a clear idea how to go about achieving her goal, she wants to invite all of her developer friends and colleagues for a session of brain storming. For that reason she has decided to organize a dinner at her house the following weekend. Considering Jolene and her developer friends are MeteoCal users, Jolene logs in to the system, from her home page she chooses the "Create event" option which redirects her to a page where she can input the desired time and place, specify the event as being indoor or outdoor and invite all of her friends. After entering all needed information, she clicks on the "create" button, at which moment her newly created event is automatically enriched with weather information and she is redirected to a page informing her of a successful event creation.

### Event Invitations

Jack and his wife Freda are both catholic and both come from very large and loving families. This means they get invited to a lot of weddings, birthday and anniversary parties, holy communion and christening celebrations, the list could go on. Naturally, it can be hard to keep track of all those events, especially the ones they get invited to a couple of months in advance, like it is

usually the case of weddings. Luckily, Jack and Freda being technically savvy, have easily convinced their families to use the MeteoCal system to manage the numerous events and invites. Now the two of them log in to the system each night to check their schedule. This night is no different, Jack navigates the the MeteoCal system from his browser, logs in and sees he has four new invitations for different events. He reads each of them, but accepts only three. The reason being that the rejected event is an invitation for a  distant relative's graduation party and it overlaps with his father's birthday. After accepting the three events, they are added to his calendar.

## Event Notifications

John, being a CEO of a fortune 500 company, gets invited to a a lot of events, anything from found-raisers to grand openings of new locales. Given his importance, his time is precious and has to be managed well. It's a Friday night and upon arriving home late and tired from work, John logs into the MeteoCal system to check his schedule for the next day. He is pleasantly surprised to see a notification on his home page informing about the heavy rain that is expected the following morning, during the time an outdoor charity event was scheduled. John decides not to attend the event due to poor weather conditions. Content with the fact that he will get some well deserved rest, he logs out and goes about is night.

# 5. UML models

## 5.1 Use Case

After stating all the desired features, goals and requirements, and describing possible scenarios we can identify some use cases. The diagrams are sown and described bellow

### 5.1.1 Accessing the application

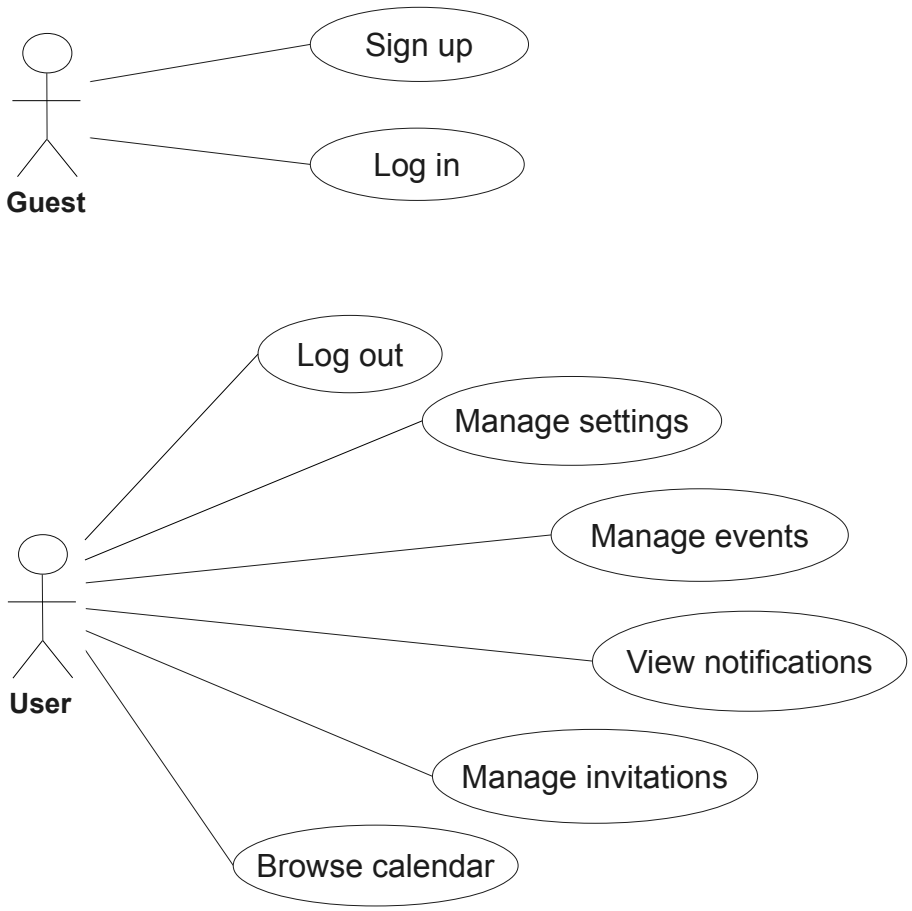


*Figure 5.1*

| **Name** | **Registration** |
|---|---|
| **Actors** | Guest |
| **Entry conditions** | The guest has never registered to the system |
| **Event flow** | • After navigating to the MeteoCal home page, the guest chooses the *Sign up* option<br>• The system serves the guest the sign-up page containing a form asking him to enter basic information, his email, his desired username and password<br>• Upon form submission the system checks if the information entered is valid. i.e. the username entered is not already in use by an existing user and the password field is not blank<br>• If the entered information is valid, the user is automatically logged in to the system and redirected to his personal home page |
| **Exit conditions** | The information about the new user is correctly stored, a welcome email is sent and a calendar is associated with the user |
| **Exceptions** | If he information the user entered is not valid, an appropriate message is displayed explaining what went wrong and the guest is shown the same form to be filled again. |

| **Name** | **Log in** |
|---|---|
| **Actors** | Guest |
| **Entry conditions** | The guest has already registered in the system and knows his username and password |
| **Event flow** | • After navigating to the MeteoCal home page, the guest chooses the *Log in* option<br>• The system requests the guest to enter his username and password<br>• Upon form submission the system checks if the username and the password match an existing user<br>• If the entered credentials match an existing user, the user is automatically logged in to the system and redirected to his personal home page |
| **Exit conditions** | The user is granted access to all of the MeteoCal's functionalities |
| **Exceptions** | If he information the user entered is not valid, an appropriate message is displayed and the guest is asked to enter his username and password again |

### 5.1.2 Managing events

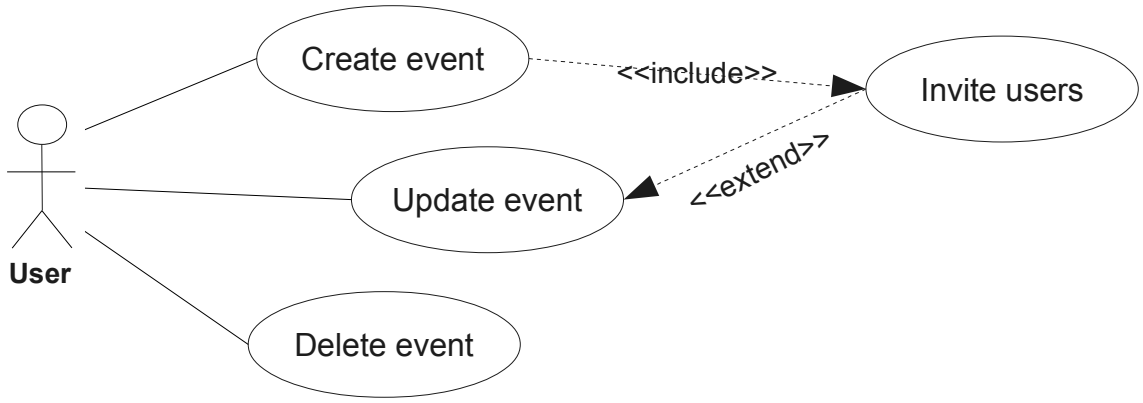


*Figure 5.2*

| **Name** | **Create event** |
|---|---|
| **Actors** | User |
| **Entry conditions** | The user is logged in |
| **Event flow** | • User chooses the *Create event* option<br>• User is redirected to a page where he is asked to provide the following information:<br>◦ start/end time<br>◦ location<br>◦ specify if the event is indoors or outdoors<br>◦ specify other users, if any, to invite (optional)<br>• User submits the filled out form |
| **Exit conditions** | The new event is created, enriched with a weather forecast, if available, and it's shown in the user's calendar. Invitations are send to all invite users, if any |

| | |
|---|---|
| **Exceptions** | If the user submits the form with at least one non-optional field empty, an appropriate message will be displayed and the user will be asked to fill out the the missing information |

| **Name** | **Update event** |
|---|---|
| **Actors** | User |
| **Entry conditions** | The user is logged in and is the organizer of the event in question |
| **Event flow** | • User selects the event and chooses the *Update event* option<br>• User is redirected to a page where he is will e able to modify the following information:<br>◦ start/end time<br>◦ location<br>◦ specify if the event is indoors or outdoors<br>◦ specify other users, if any, to invite (optional)<br>• User submits the filled out form |
| **Exit conditions** | The event is modified, and invitations are send to all newly invited users, if any |
| **Exceptions** | If the user submits the form with at least one non-optional field empty, an appropriate message will be displayed and the user will be asked to fill out the the missing information |

| **Name** | **Delete event** |
|---|---|
| **Actors** | User |
| **Entry conditions** | The user is logged in and is either the organizer or a participant of the event in question |
| **Event flow** | • User selects the event and chooses the *Delete event* option<br>• Two cases are possible:<br>1. If the user is the organizer, the event is removed from his calendar and from the calendar of all of it's participants<br>2. If the user is a participant, the event is deleted only from his calendar |
| **Exit conditions** | The event is deleted and it is removed from the users calendar |

### 5.1.2 Managing invitations

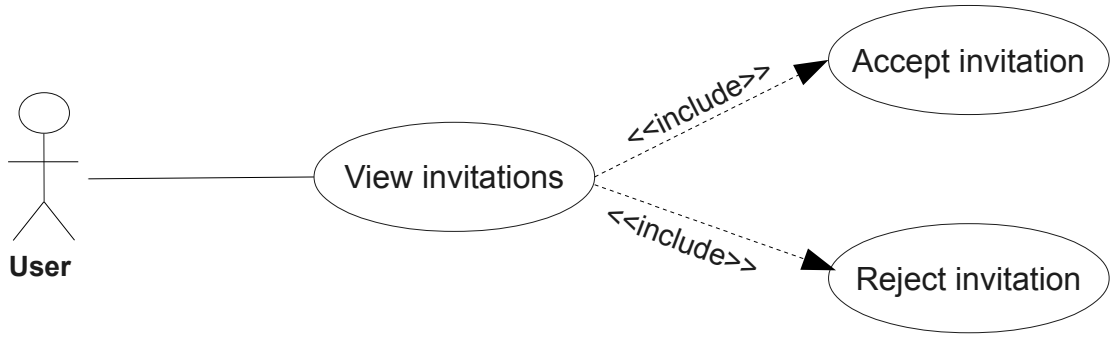


*Figure 5.3*

| **Name** | **Accept event invitation** |
|---|---|
| **Actors** | User |
| **Entry conditions** | The user is logged in and has received an invitation from another user |
| **Event flow** | • User selects the invitation after which he is presented with two choices, either to accept or decline<br>• User accepts the invitation |
| **Exit conditions** | The notification is removed, and the event is added to the user's calendar |

| Name | Decline event invitation |
| --- | --- |
| **Actors** | User |
| **Entry conditions** | The user is logged in and has received an invitation from another user |
| **Event flow** | • User selects the invitation after which he is presented with two choices, either to accept or decline<br>• User declines the invitation |
| **Exit conditions** | The notification is removed |

### 5.1.3 Managing settings

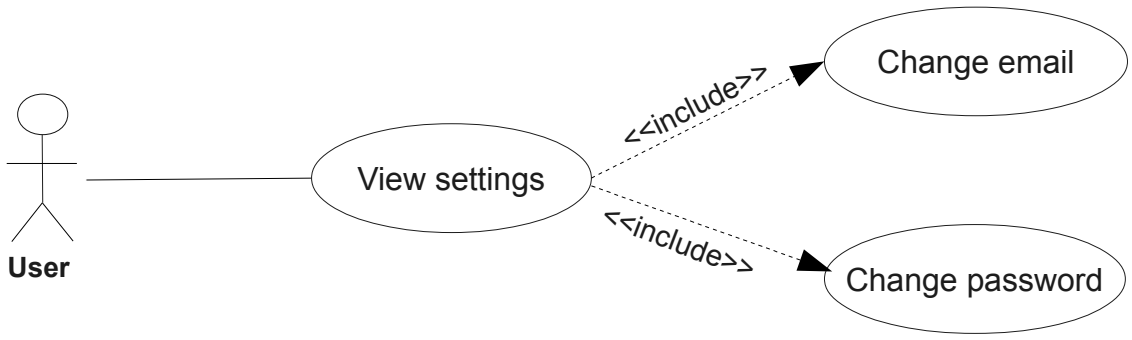


*Figure 5.4*

| **Name** | **Change email** |
|---|---|
| **Actors** | User |
| **Entry conditions** | The user is logged in |
| **Event flow** | • User chooses the *View settings* option<br>• User is redirected to a page where his username and email are shown along with two options<br>◦ Change email<br>◦ Change password<br>• User selects the *Change email* option, enters the new email, and submits the change |
| **Exit conditions** | The new email is correctly stored and the old one is deleted |
| **Exceptions** | If the user submits the form leaving the email field empty, an appropriate message will be displayed and the user will be asked to fill out the the missing information |

| **Name** | **Change password** |
|---|---|
| **Actors** | User |
| **Entry conditions** | The user is logged in |
| **Event flow** | • User chooses the *View settings* option<br>• User is redirected to a page where his username and email are shown along with two options<br>◦ Change email<br>◦ Change password<br>• User selects the *Change password* option, enters the new password, and submits the change |
| **Exit conditions** | The new password is correctly stored and the old one is deleted |
| **Exceptions** | If the user submits the form leaving the password field empty, an appropriate message will be displayed and the user will be asked to fill out the the missing information |

## 5.2 Class diagram

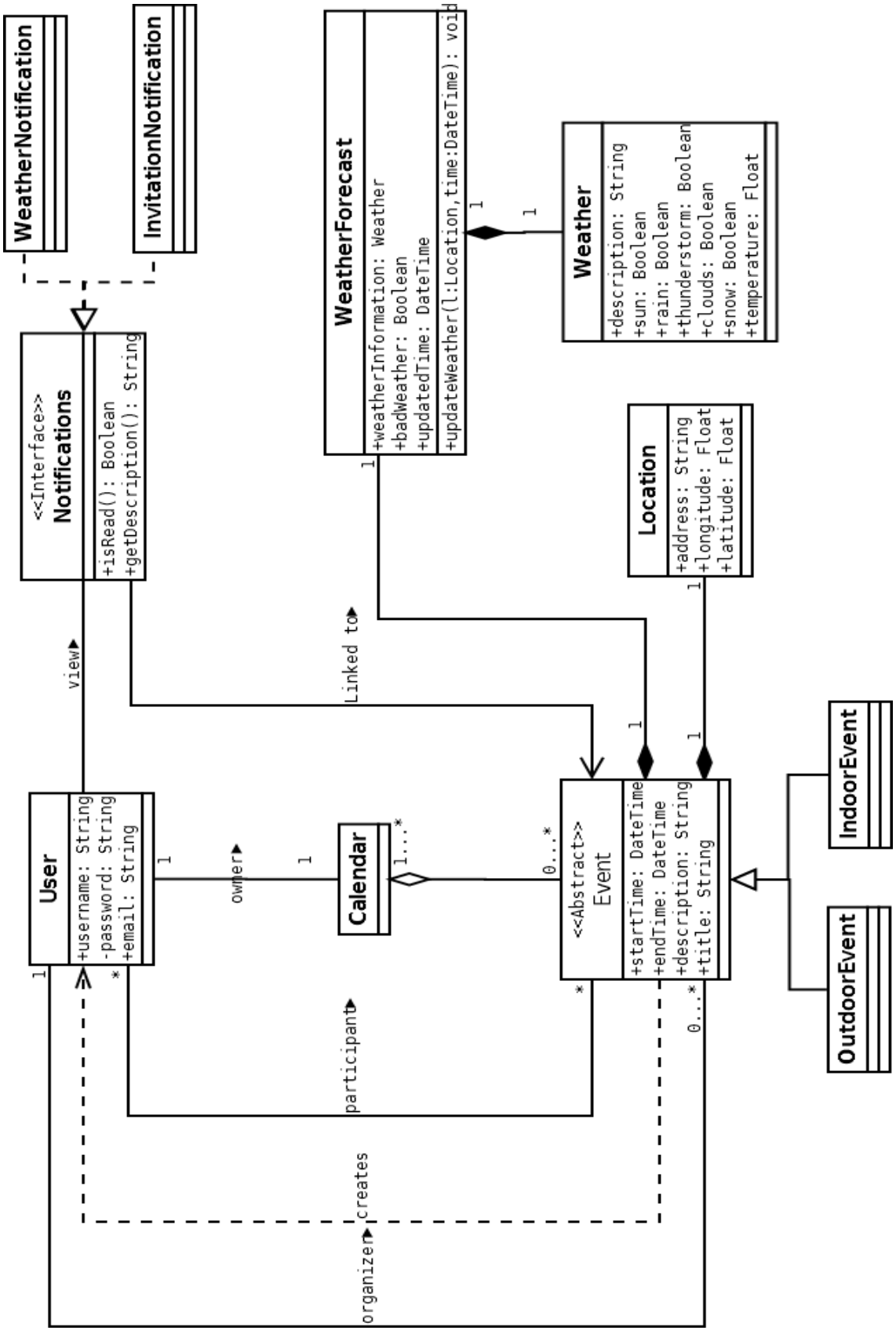


*Figure 5.5*

## 5.3 Sequence diagram

### 5.3.1 User Registration

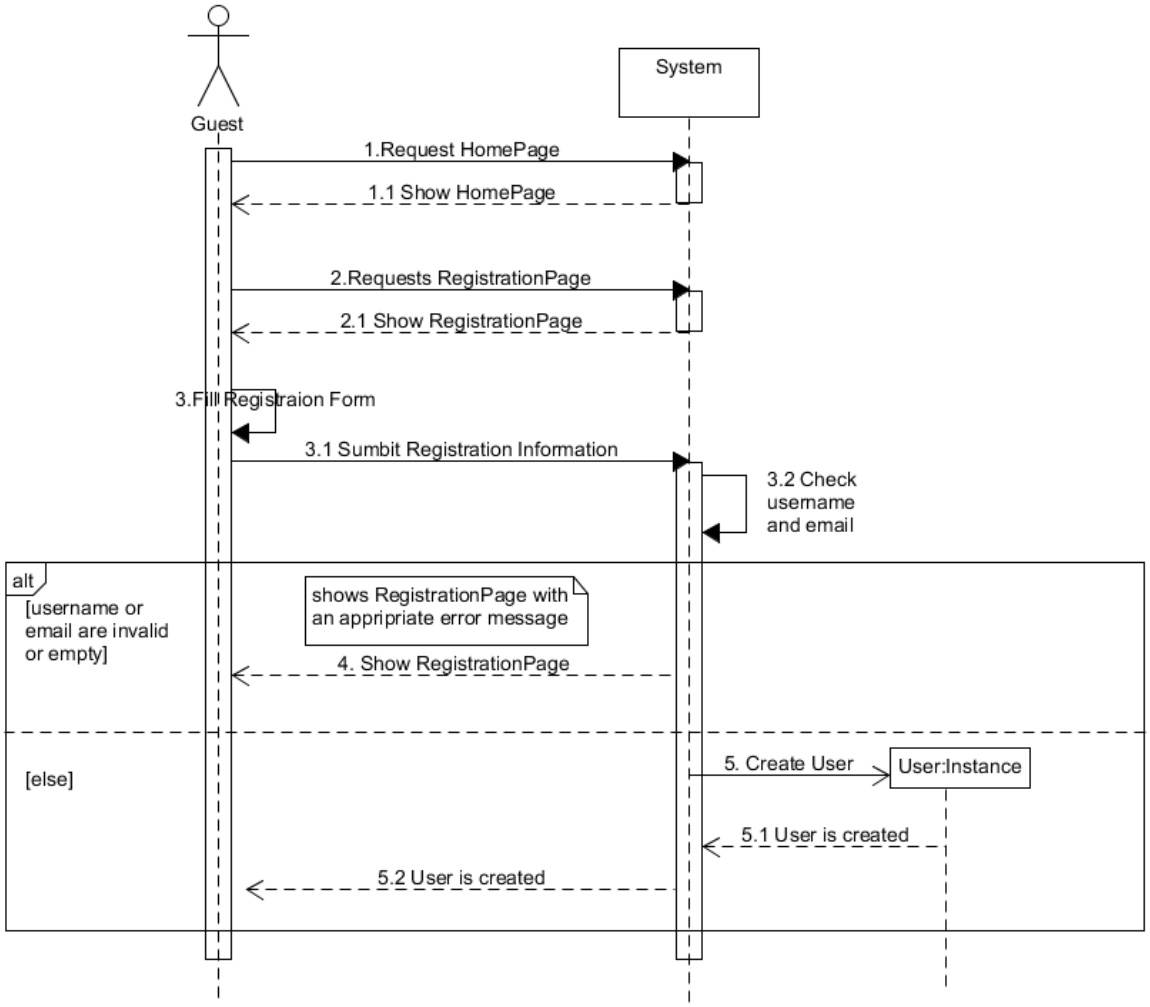


*Figure 5.6*

### 5.3.2 User Log in

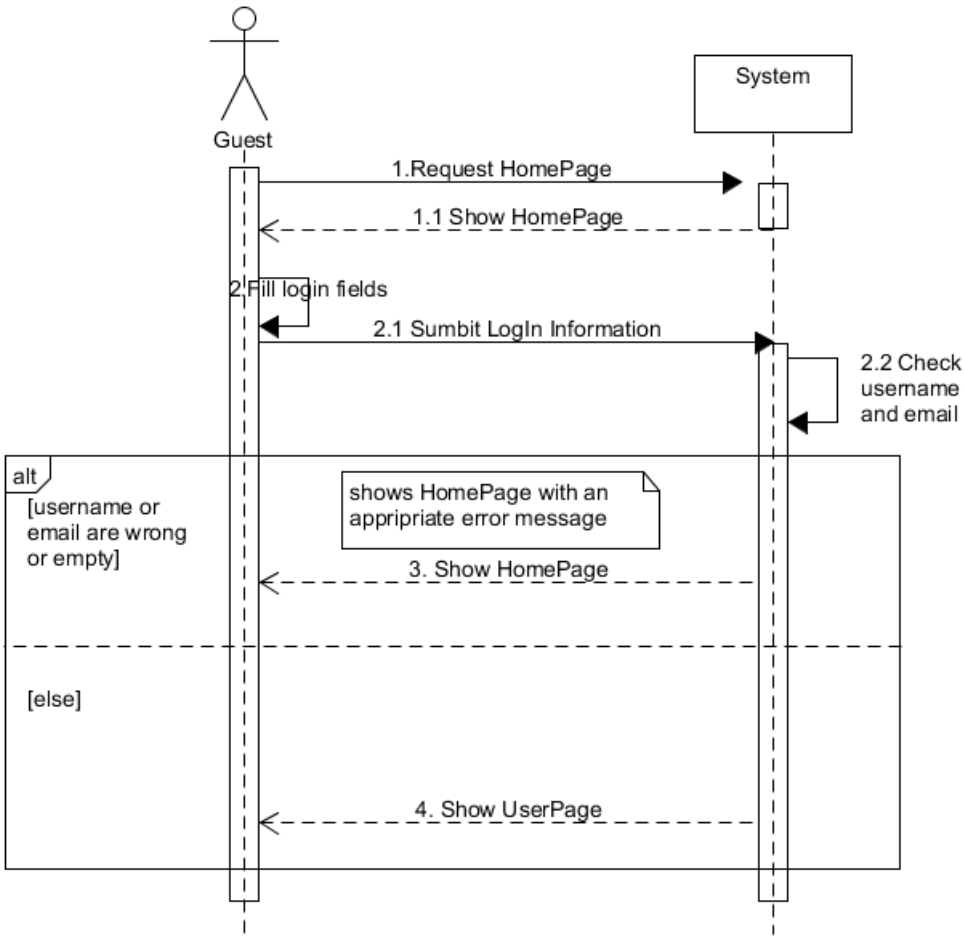


*Figure 5.7*

### 5.3.3 Change Password

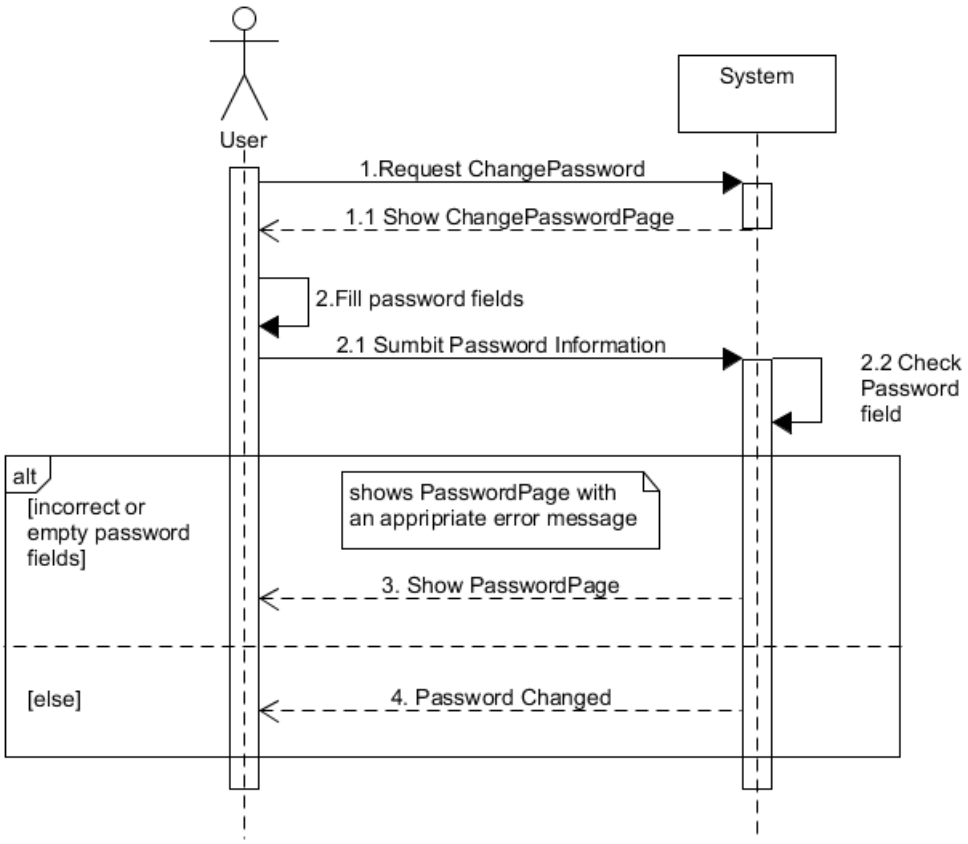


*Figure 5.8*

### 5.3.4 Create event

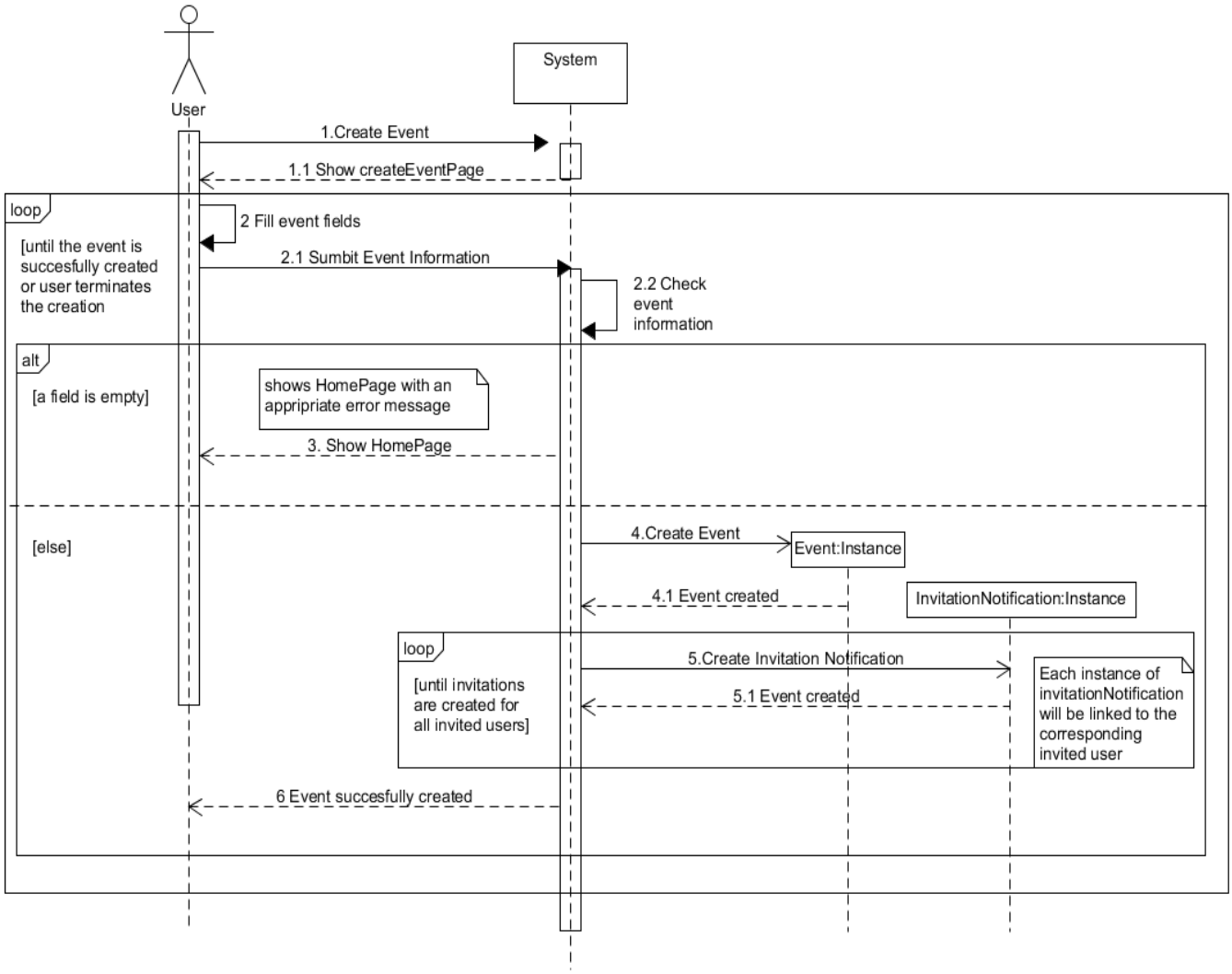


*Figure 5.9*

### 5.3.5 Manage invitations

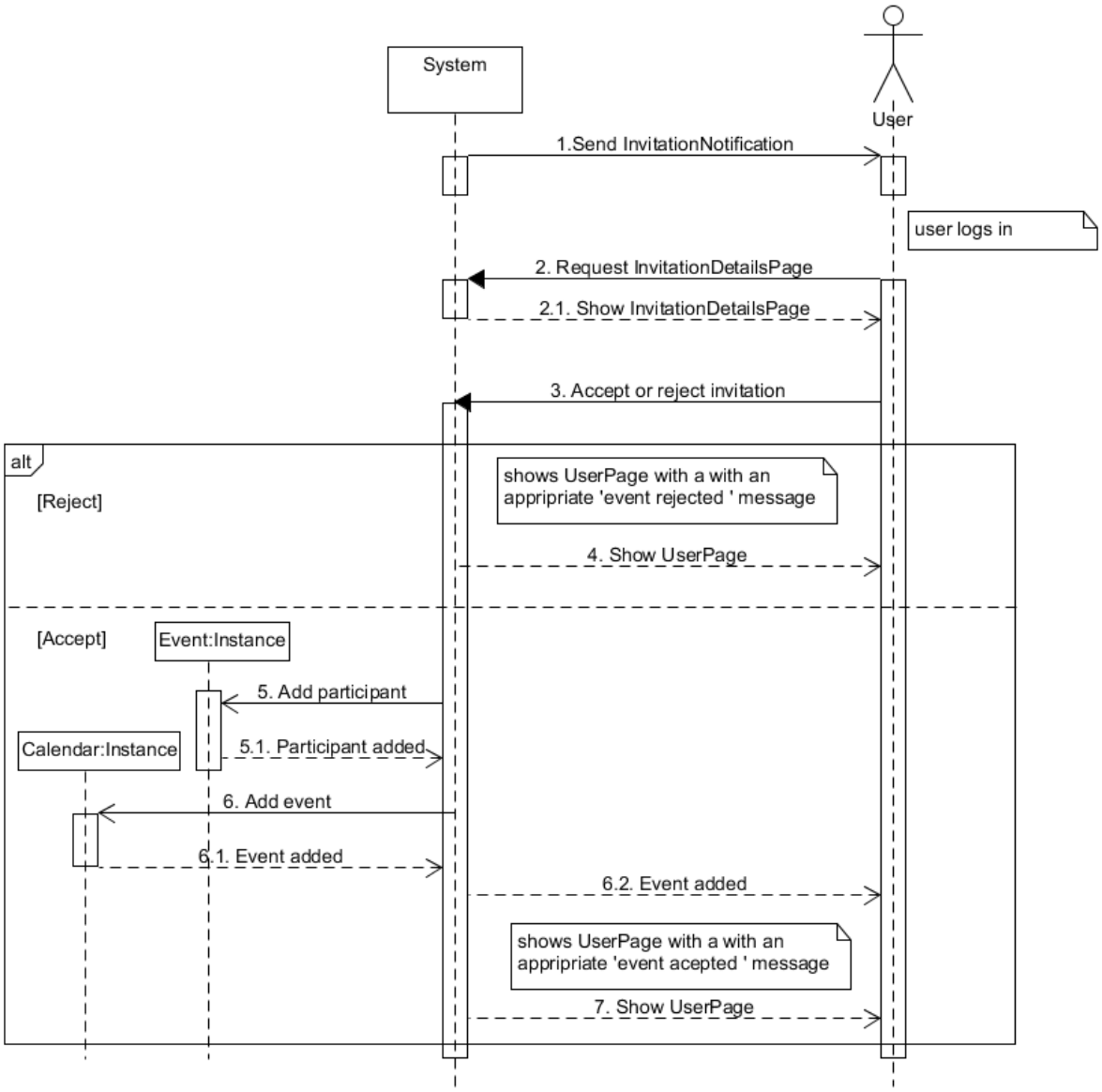


*Figure 5.10*

## 5.4 State chart diagrams

### 5.4.1 Event's lifecycle

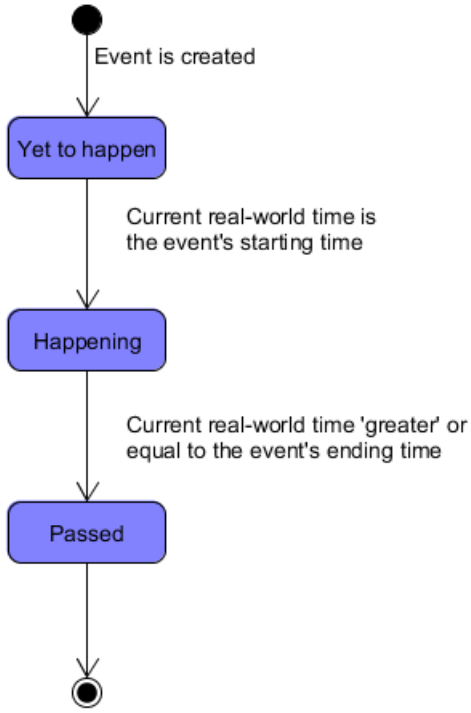


*Figure 5.11*

### 5.4.2 Invitation's lifecycle

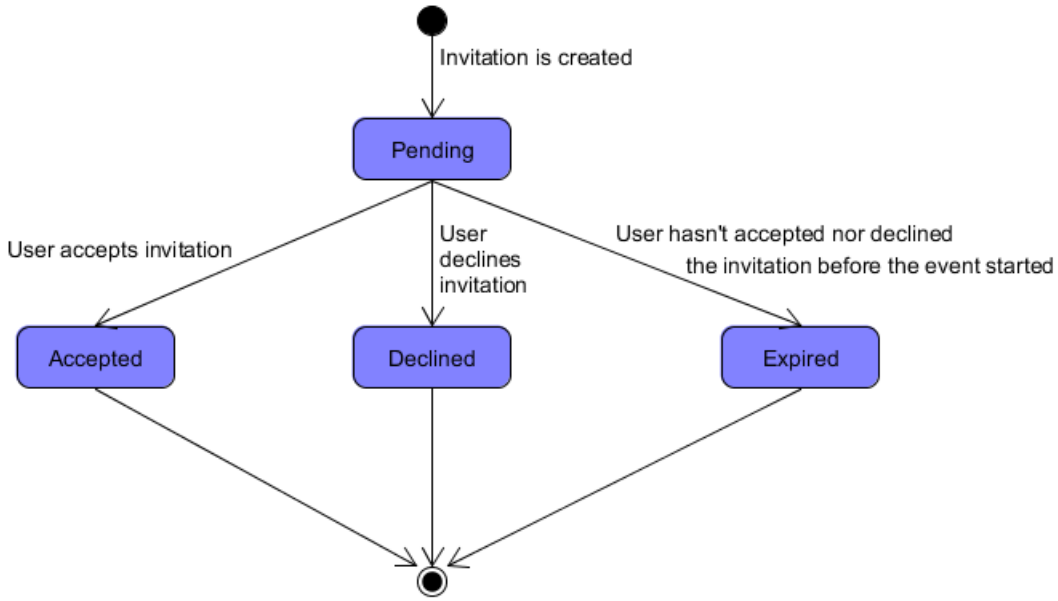


*Figure 5.12*

# 6. Alloy

```
sig User {
	calendar: one Calendar,
	notifications: set Notification
}

sig Calendar {
	events: set Event
}

abstract sig Event {
	organizer: one User,
	participants: some User,
	location: one Location,
	weatherForecast: one WeatherForecast
}

sig OutdoorEvent extends Event {}
sig IndoorEvent extends Event  {}

sig Location {}
sig Weather  {}

sig WeatherForecast {
	weatherInformation: one Weather
}

abstract sig Notification {
	linkedTo: one Event
}

sig WeatherNotification extends Notification{}
sig InvitationNotification extends Notification{}


//facts
fact everyCalendarHasOneUser{
	all c: Calendar | one u:User | c = u.calendar
}

fact everyUserHasAUniqueCalendar{
	no disj u1, u2: User | u1.calendar = u2.calendar
}


fact everyCalendarContainsOnlyParticipatingEvents{
	all u: User, e: Event |
		(e in u.calendar.events) <=> ( u in e.participants)
}
```

```
fact eveyEventHasAOneOrganizer{
      all e:Event | one u:User | u = e.organizer
}

fact organizerIsAlsoParticipant{
      all  e: Event | e.organizer in e.participants
}

fact eachLocationIsLinkedToAnEvent{
      all l: Location | some e: Event | l = e.location
}
fact eachWeatherForecastIsLinkedToAnEvent{
      all wf: WeatherForecast | one e: Event | wf = e.weatherForecast
}

fact eachWeatherIsAsociatedWithOneWF{
      all w: Weather | one wf: WeatherForecast | w = wf.weatherInformation
}

fact allNofiticationssAreMeantForAUser{
      all i: Notification | one u: User | i in u.notifications
}

fact organizerCannotGetInvitationForOwnEvent{
      all e: Event,  i:InvitationNotification |
            ( e = i.linkedTo ) =>( i not in e.organizer.notifications )
}

fact participantCannotGetReinvitated{
      all e: Event,  i: InvitationNotification |
            ( e = i.linkedTo) => ( i not in e.participants.notifications )
}

//predicates
pred show() {}

//a. User cannot get 2 invitations to the same event
//b. User cannot get 2 weatherNotifications about the same event
//c. User cannot get weather notification about event he has
//   a pending invitation (not a participant)
fact onlyOneNotifAboutAsingleEventPerUser{
      all u: User, disj n1, n2: Notification |
      (n1.linkedTo = n2.linkedTo) =>( (n1 + n2)  not in u.notifications)
}
fact allWeatherNofiticationsAreMeantForAParticipant{
      all i: WeatherNotification | one u: User |
            (i in u.notifications)<=> (u in i.linkedTo.participants)
}
//predicates
pred show() {}
run show for 3
run show for 3 but 4 InvitationNotification
run show for 3 but 3 Event, 5 WeatherNotification
```

## 6.1 Worlds generated

In this section some worlds generated by Alloy Analyzer will be display in order to prove the model's consistency.

### 6.1.2 General world

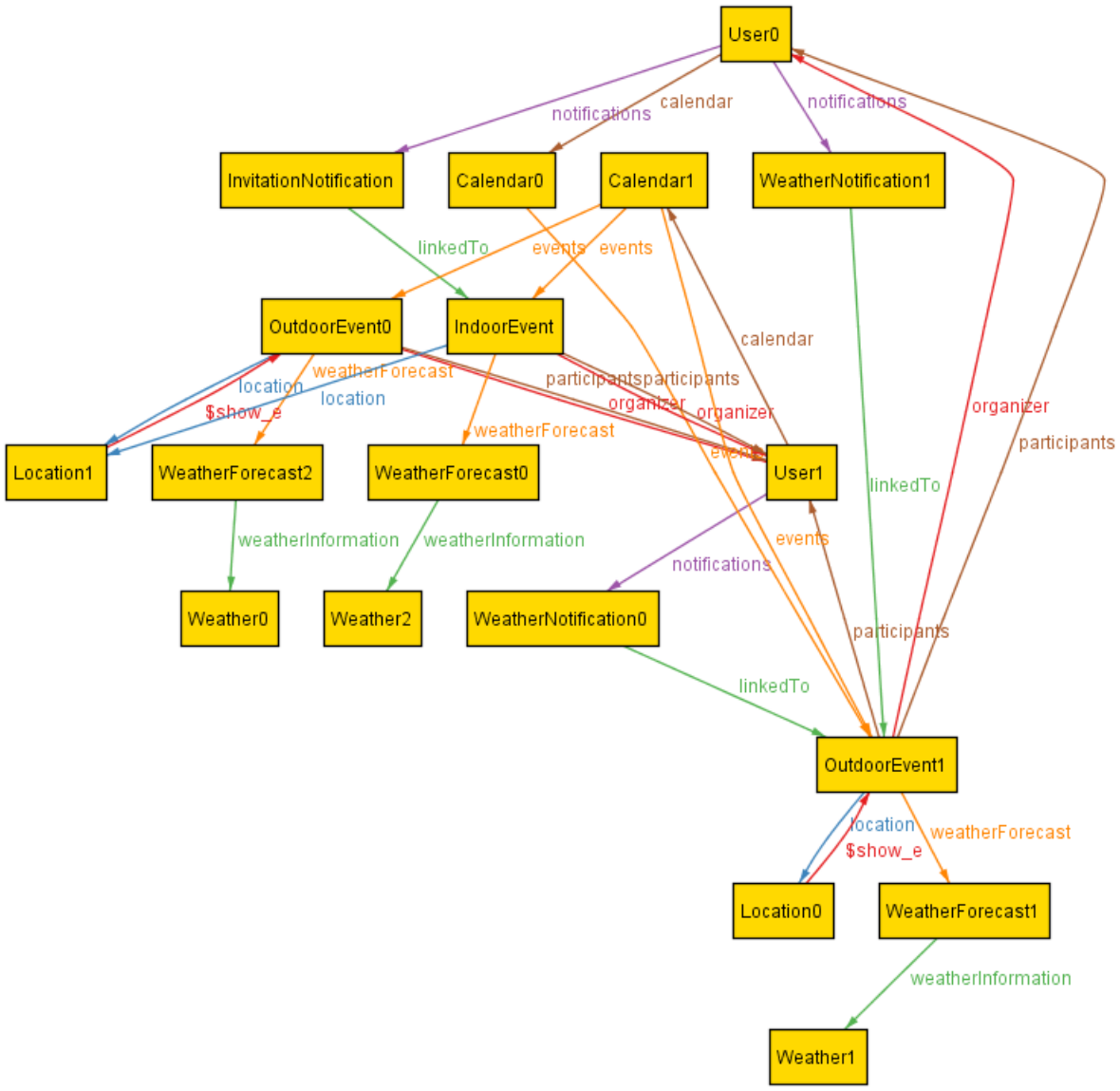


*Figure 6.1*

### 6.1.2 Invitation consistency

A world is shown in *Figure 6.2* in which only a non-participating user can receive a single event invitation.

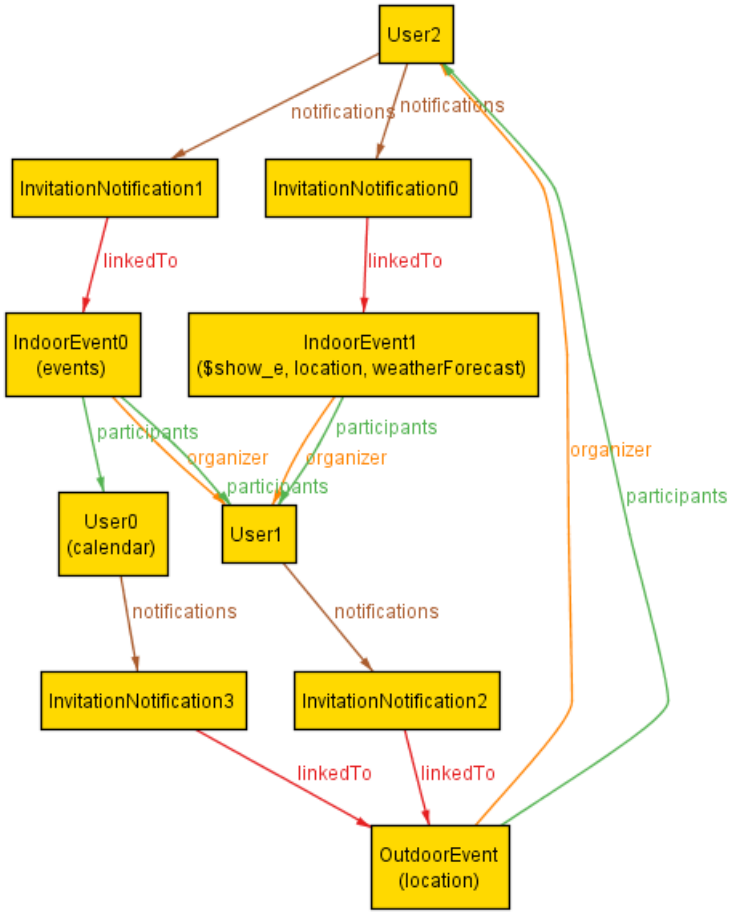


*Figure 6.2*

### 6.1.3 Weather notification consistency

A world is shown in *Figure 6.3* in which only a participating user can receive a single weather notification.

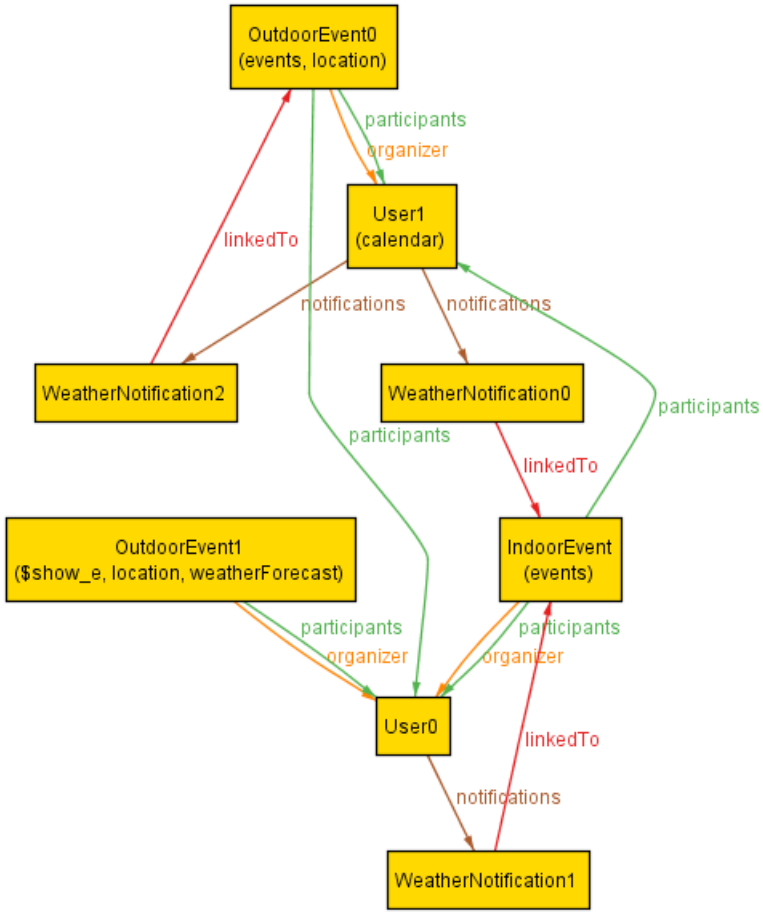


*Figure 6.3*