# Introduction

## Problem description

The main scope of the project is to create a web application using JavaEE platform which informs the registered users about the incoming events, helping them to avoid bad weather conditions in case of outdoors activities.

This web application will permit to the registered users to create, delete, update and invite people to the personal events. The calendar permits to the creator of the event to invite other registered users at any time, where the latter can only accept or decline the invitation.

In fact the software will allow to organizer to specify more details about the future event like: place, date and whether it will be indoor or outdoor appointment. Furthermore, each event can be private or public. Public events details can be consulted by any registered users whereas private events are visualized with busy status where all details are disclosed only to the private sphere.

Once the event has been committed every participating user will be informed about forecast condition and other important details.

In case of incoming bad weather conditions the system will notify it to the organizer recommending another better day to choose for the event and a users participating to outdoor events one day before.

## Scope

In order to to guarantee the main functions here below are listed the most important features:

* The system permits only to the owner of the event the creation, deletion, update of the schedule;
* Each registered user can create an event;
* The system is able to send invitations and receive responses from the users;
* The organizer has the complete list of the users that has committed to the event;
* In case of incoming bad weather conditions the system has to notify the forecast situation for outdoor events one day before;
* In case of incoming bad weather conditions the system has to suggest to the creator the closest sunny day (if any) three days before;
* Only on invitation people can view the details of the event;
* The system provides the forecast for all upcoming events;
* The system has to guarantee access only to the registered users;
* Only when users are logged in can view own notifications and personal schedule;
* Registered users can see other users' schedule, being able to check when they are busy and, in case of public event, to see the details of the event;
* The system has to provide a registration form.

# Glossary

User: a person already registered in the system who is able view notifications, create new events, reply to invitations, view details of own events.

Organizer: is a registered user who arranges an activity, which is the only one who can update and delete an event. Also is known as owner.

Event: is a planned activity with a specific date and place.

Public: all events which informations are accessible to every registered user.

Private: information is available only to selected members.

Indoor: all activities happening inside a building or under a covered structure.

Outdoor: opposite of indoor.

Invitation: a notification with a specific request to join an event that can be either accepted or declined.

Notification: is an official message sent by the system to the registered user.

Registered user: is a person who has compiled a registration form and is acknowledged by the system.

# Vision

The initial document of the project presents some not well defined point. In order to avoid ambiguities and misunderstandings in the project, we will give our vision to some crucial points from our point of view:

Bad weather condition– in nature there is a large set of bad weather conditions, therefore for humans it is a subjective point of view, i.e. something related to their planned activities, mood, or other sort of factors. Considering these premises the application shall provide to user when creating an event the opportunity to define what bad weather conditions are.

One day before – usually for people this concept is perceived as 24 hours before a given event, this concept will be used in this project as well.

Three days before – we consider this elapse of time as 72 hours to the event's beginning.

Closest sunny day – we intend such a day that is the closest date with the sunshine and comes after the event's day.

Event creation – a registered user can create an event at least 24 hours before its occurrence.

Notification – are sent immediately after the event creation and no further invitations can be added.

Place: to avoid ambiguities in this case the owner will provide unique identification of the event's location, e.g. zip code.

Modification of the event – in case of one of important fields have been modified (e.g. location, date), the notification shall be sent to all originally participants.

Overlapped events – an user can create an event only if he isn't participating to any other scheduled event at that precise time, i.e. creation only of synchronously disjointed events are permitted.

Acceptance of the event – once an invitation has been either accepted or declined no additional modification is possible.

Calendar – Gregorian.

# Proposed System

As has been stated in the introduction and in assumptions our web application is conceived to be able to allow people to see both personal and others' calendar.

Each new user shall to register through a very simple registration form, in order to be acknowledged by the system. Every user, after logging into the application, will be able to:

* In first place to interact with the personal calendar creating, updating or deleting their personal event through a simple user interface;
* Secondly to retrieving information about forecast conditions in the period of incoming event;
* Furthermore to see other users' schedules;
* Eventually to see the invitation or bad weather notifications;
* Finally thanks to this system users will be able easily to link other people to programmed events.

# Domain

As start point for gathering requirements and creating specifications, Jackson-Zave model has been chosen. This approach helps to identify the main interactions between the world and the machine, in our case registered users or weather forecast and the web application respectively.

The intersection of two domains will give an idea of the shared phenomena, e.g. personal account

Identifying stakeholders

The main stakeholders of the project is Politecnico of Milan which is represented by internal committee of professors. The main needs of this project is to fulfill all requested requirements respecting all deadlines. Furthermore we have to exhibit all stages of the project throughout of the development process.

The other stakeholder is our development team which is focused on (is really into/ be concerned) delivering a high-quality web application. In addition we are concerned with acquiring main soft engineering techniques. Besides what has been stated before, our team is interested especially to get a good mark for the university career.

Finally the main intention is to provide an ease-to-use software, that will accommodate user experience and fulfill all requested needs.

Identifying actors

In this system under consideration, there are basically two main actors:

* Registered user: a person acknowledged by the system, has access to all application's features such as creation, deletion, update events, check up other users' schedules and signed up arrangements with their relative weather forecast.
* Weather forecast: an external service which provides an analysis of the state of the weather in certain area and date.

# Requirements

Requirements engineering includes two main activities;

requirements elicitation, which results in the specification of the system that the client

understands, and analysis, which results in an analysis model that the developers can

unambiguously interpret.

Non functional: he system functionality, the interaction between the user and the system, the errors

that the system can detect and handle, and the environmental conditions in which the system

functions are part of the requirements.

* Identify actors
* Scenarios: during this activity, developers observe users and develop a set of detailed scenarios for typical functionality provided by the future system. Scenarios are concrete examples of the future system in use. Developers use these scenarios to communicate with the user and deepen their understanding of the application domain.
* Use cases
* Relationship among use cases
* Non functional requirements: During this activity, developers, users, and clients agree on aspects that are visible to the user, but not directly related to functionality. These include constraints on the performance of the system, its documentation, the resources it consumes, its security, and its quality.

## Functional requirements

| User | System |
| --- | --- |
| * + Each registered user can create an event and view personal notifications, invitations and schedule.   + Each registered user can access to other users' calendar, seeing details of events if they're public.   + Registered user can decline or accept the invitation before its deadline, once expired no possible actions can be performed;   + Registered user can change his mind about participation to an event before the deadline.   + During the creation of an event the user has to fill in mandatory information fields, otherwise it won't be possible to proceed with the creation;   + Organizer defines the meaning of “bad weather”;   + The organizer can invite registered users for own event.   + Updates and cancellation of event can be performed by the organizer at any time.   + Only the organizer is allowed to add more users to the event.   + The member of the event can view event summary; | * + Any interaction with the calendar, is possible only when user is logged into the system;   + All calendar events are based on Gregorian calendar and on local time.   + Any created event has to be acknowledged as “committed event” by the system otherwise the user has to repeat the creation procedure or fulfill missing requirements.   + The system doesn't allow any creation of a new event if it is overlapped with an event previously created by the same organizer.   + Each event has a summary log at least with essential details: weather forecast, date, place, name of activity, and its type (i.e. outdoor or indoor);   + Once the event has been committed the notifications are automatically sent to all involved members.   + If any updates or modifications are submitted new invitations and details should be sent again to original guests.   + Each event has a deadline to reply to an invitations;   + All expired invitations are considered as declined.   + The system is synchronized to a weather forecast system;   + Weather forecast for next 14 days is considered trustworthy otherwise is not available;   + In case of upcoming bad weather conditions a reminder is sent to all participants in outdoor activities one day before.   + In case of upcoming bad weather conditions the system suggest the closest sunny day (if any) for the outdoor events. |

## Non functional requirements

### Usability

* minimal
* user-friendly
* universal accepted symbols for each function
* hints on the icons
* guided procedures
* login page, calendar main (only preview), notifications, upcoming events (shows details)

### Reliability

Since we are dealing with an academic project and our customer hasn't specified any particular request about reliability, we will assume that:

* The project runs on reliable hardware therefore any hardware faults cann't happen;
* The project deals in harmless environment so no cyber attacks can occur;
* There aren't any catastrophic scenarios that may damage the environment with irreversible consequences.

### Performance

No specifications has been given to this point as well, but still some assumptions with common sense cann't be ignored:

* The response time of functions should not annoy the user;
* Should support most of common browsers;
* Can be accessible from low-medium latency connections.

### Supportability

* Our system will not provide any support for addoptability or integration with external module, that is, this projects has been concieved as stand alone project.
* The maintanability will be supported with a rich set of documentation:
  + RASD
  + DD
  + Alloy
  + UML diagrams
  + Testing
* The portability will be ensured through JavaEE platform and Java Virtual Machine, which is supported with almost all common server devices.

|  |  |
| --- | --- |
| **Title** | *Registration of a new user* |
| **Goal** | *Create shared events and/or program personal schedules* |
| **Assumptions** | * *Person is not yet registered in the system* |
| **Scenario** | |
|  | |

­

|  |  |
| --- | --- |
| **Title** | *Creation of a new event and invite other users.* |
| **Goal** | *The organizer can easily create a new event and link other users to it through an invitation.* |
| **Assumptions** | * *[A1] User is already registered in the system.* * *[A2] Invitation can be only sent to registered members.* * *Organizer is a member of the event by default.* * *The organizer hasn't created any event yet.* |
| **Scenario** | |
|  | |

|  |  |
| --- | --- |
| **Title** | *Update of already existing event.* |
| **Goal** | *The organizer is willing to add new users and modify some data about the event.* |
| **Assumptions** | * *[A1] [A2]* * *[A3] The event is already present in the system and is consistent.* * *Main data about event are present in the system.* * *Some invitations have already been sent.* |
| **Scenario** | |
|  | |

|  |  |
| --- | --- |
| **Title** | *Check out of schedule and notifications* |
| **Goal** | *Each user is able to decline or accept incoming invitations, view details of programmed schedules.* |
| **Assumptions** | * *[A1][A3]* |
| **Scenario** | |
|  | |

|  |  |
| --- | --- |
| **Title** | *Creation of a consistent event* |
| **Goal** | *The user is willing to create a new event.* |
| **Assumptions** | * *[A1]* |
| **Scenario** | |
|  | |

|  |  |
| --- | --- |
| **Title** | *Overlapped events* |
| **Goal** | *The user is willing to create a new event. Participating in different events.* |
| **Assumptions** | * *[A1]* * *User has already created an event.* * *Accepted some invitations.* * *Some events are overlapped.* |
| **Scenario** | |
|  | |

|  |  |
| --- | --- |
| **Title** | *Cancel a participation to an event* |
| **Goal** | *User can change own mind at any time about participation.* |
| **Assumptions** | * *[A1]* * *User has already accepted at least one invitation.* * *The weather doesn't have any influence on the user's decision.* |
| **Scenario** | |
|  | |