REPORT PHP

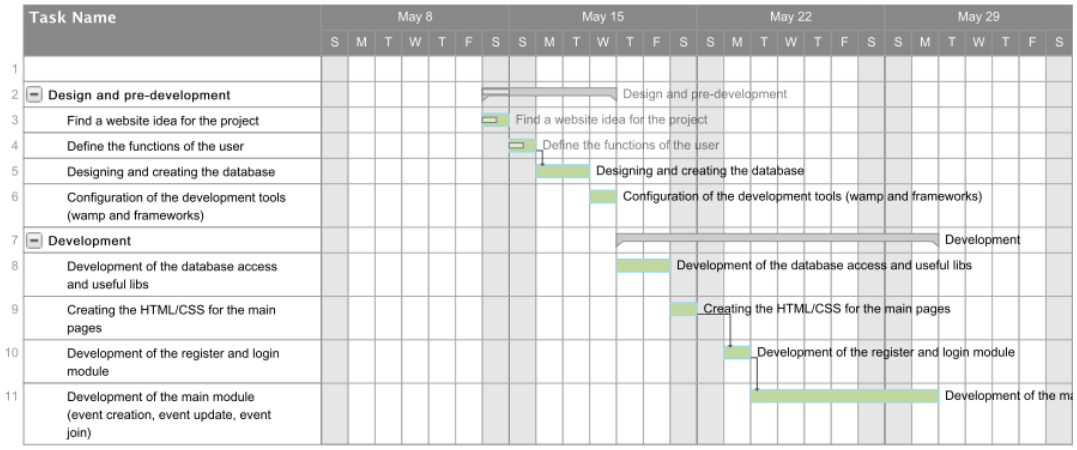
Hichem HAMZA
05/30/2016

Introduction to web development

I- Gantt chart

Organizor

smartsheet



II- Introduction and purposes

This website is a little social network in development, it allows people to organize events so that other people can join them.

The purposes are, of course finding people to join in your events, but also the build new relationship with people, based on a shared experience, which can create very good memories. By doing so, it allows to enlarge your network when it comes to people sharing the same hobbies, whether it be music, sport, or anything else.

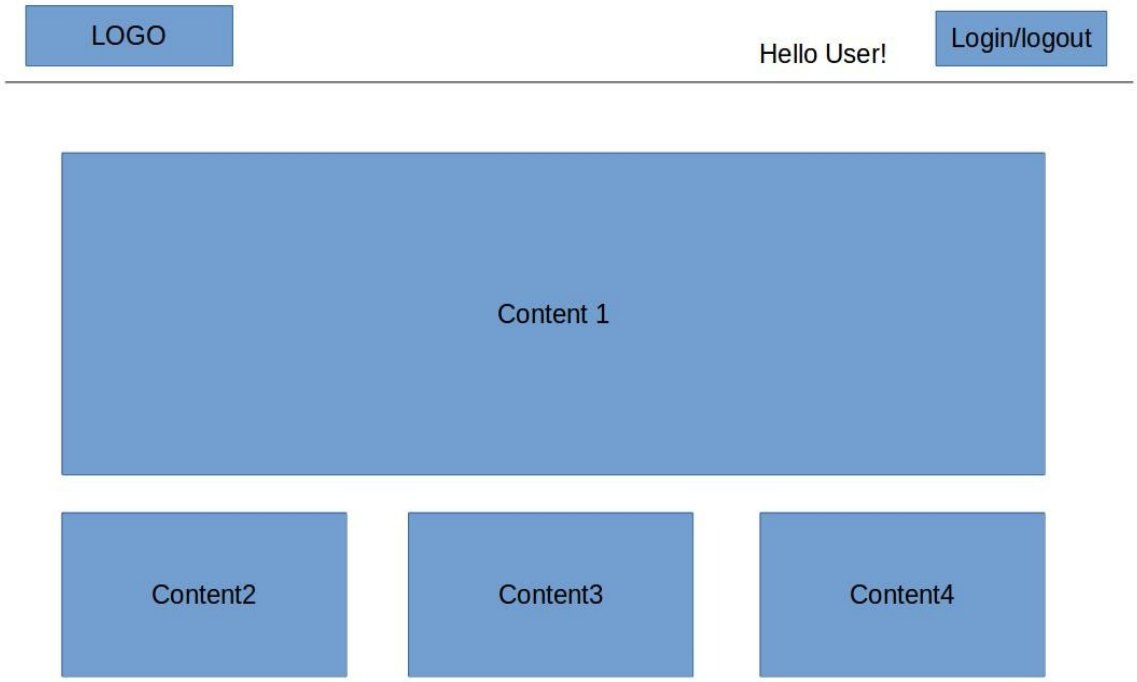
The website is meant to evolve in the future. It's concept makes it very easy to improve with new functionalities, for example chatting system r a comment/rate section.

The user can create an event, update this event, delete the event. The user can also browse the list of his events or the list of all the events. He can of course join an event created by another user. Once he joined an event he can also leave it whenever he wants. There are no payments involved on this website but the creator of an event can precise the budget necessary for his event.

III- System design

1- Storyboard

The website has a very basic and refined design.



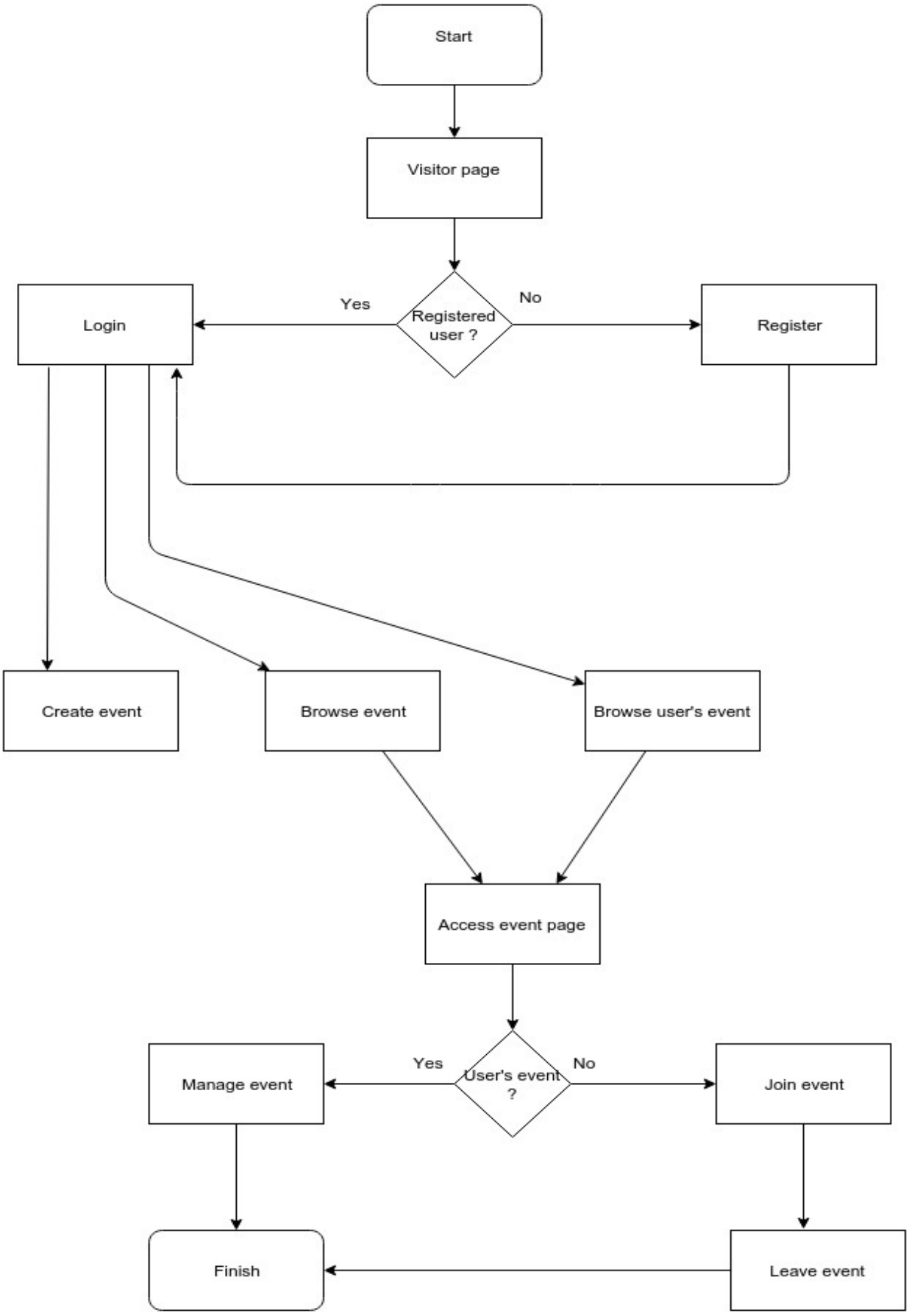
Visitor’s storyboard

The visitor will arrive on his own main page where there information about the website, on the functionalities. There, he can finds links that redirect him to the register page or the login page. This is the only page accessible for the visitor, all the others are made for the User.

Client’s storyboard

Once logged, the user access to his main page. On this page, there are a few functionalities. First, of course the user can log out which will lead him to the visitor's page. Then he can access to the event creation page. He can also access the event browsing page. He can also access his own list of events. From the two previous pages he can of course access the page of an event. If it is one of his event then he can either update the description or delete the event, he can also check the list of users. If it is an event created by another user, then he can check all the data, and either join or leave the event if he had previously joined.

2- Flow chart


Start
Visitor page
Registered user ?
Yes
No
Login
Register
Create event
Browse event
Browse user's event
Access event page
User's event ?
Yes
No
Manage event
Join event
Finish
Leave event

3- Objectives of the website

The main objective of this website is to create an event management utility. It's always hard to organize an event with only words because we can't be clear enough and we are never sure of who will participate or not. This website allows to create an event with the proper an clear information necessary to attract people. The creator can also see who will participate his events via the list which is updated whenever someone joins or leaves the event.

4- Style design and audience targeted

Style design

The design is very simple, based on bootstrap. The idea is to use an refined design so that the user finds it easy to navigate on the website.

The header is here to display the logo of the website, and access the login page, or logout if already login.

The main containers are incorporated on the background of the website, this gives an impression of a panel.

Audience targeted

Everyone is welcome to use this website since any kind of event can be organized, but of course we will need to create a regulation program in order to control if there aren't any illegal or immoral events being organized. And also check the age of the users for party-like events.

IV- Implementation

1- Example of code: display of all the events.

```php
require_once("../php/useful_lib.php");
require_once("../php/DBConnec.php");
page_start("Events", "style.css");


$events = array();

$pdo = DBConnec::getConnec()->getPdo();
$sql = 'SELECT * FROM EVENT';
$stmt = $pdo->prepare($sql);

try
{
    $stmt->execute();
    $stmt->rowCount() or die('Pas de résultat' . PHP_EOL);
    $stmt->setFetchMode(PDO::FETCH_OBJ);
    while ($result = $stmt->fetch())
    {
        array_push($events, $result);
    }
}
catch (PDOException $e)
{
    echo 'Erreur : ', $e->getMessage(), PHP_EOL;
    echo 'Requête : ', $sql, PHP_EOL;
    exit();
}
```

First we get all the events from the database, we stock the result of the query on an array() so that it is easy for us to manipulate it later.

```php
<?php
    foreach($events as $event) {
?>
<div class="media">
  <div class="media-left">
    <a href="event.php?eventID=<?php echo $event->ID; ?>">
      <img style="height: 120px; width:170px" class="media-object" src="<?php echo 'img/' . getImageType($event->TYPE); ?>" alt="<?php echo getImageType
($event->TYPE); ?>">
    </a>
  </div>
  <div class="media-body">
    <h4 class="media-heading"><?php echo $event->NAME; ?></h4>
    <p><?php echo $event->DESCRIPTION;?></p>
  </div>
</div>

<?php
    }
    page_end();
?>
```

Then we use a loop with HTML code to display all the results with the data stocked in the array.

2- Exemple of code : login user

```php
$pdo = DBConnec::getConnec()->getPdo();

$sql = 'SELECT * FROM USER WHERE nickname = :nickname';
$stmt = $pdo->prepare($sql);
$stmt->bindValue('nickname', $_POST['nickname'], PDO::PARAM_STR);
try
{
    $stmt->execute();
    if($stmt->rowCount() == 0) {
                        header("location: ../login.php?err=0"); // user doesn't exist
                }
                else {
                        $pdo = DBConnec::getConnec()->getPdo();
                        $sql = 'SELECT * FROM USER WHERE NICKNAME = :nickname';
                        $stmt = $pdo->prepare($sql);
                        $stmt->bindValue('nickname', $_POST['nickname'], PDO::PARAM_STR);
                        try
                        {
                            $stmt->execute();
                            if($result = $stmt->fetch()) {
                                        if($result['PASSWORD'] == $_POST['password']) {
                                                session_start();
                                                $_SESSION['sid'] = $result['ID'];
                                                $_SESSION['nickname'] = $result['NICKNAME'];
                                                $_SESSION['email'] = $result['EMAIL'];
                                                header("location: ../index.php");
                                        }
                                        else
                                                header("location: ../login.php?err=1");
                                    }
                        }
```

This is isn't the complete code, just an extract. Here the first thing is to check if the user exists in the database, then once done, we get all his data, If the

password matches the one in the database then we can start a session and put the informations we want in the $_SESSION variable. In case the user doesn't exist or if the passwords doesn't match then we send back to the login page with an error code in $_GET. On this page those errors are handled.

3- Database connection

As for the database connection, I decided to implement a class database as a singleton which will be the only database for the whole project, in order to avoid creating a new connection anytime I want to use the database. Here is the class in question. I also deciced to use PDO. Since I want to create a class for the database, PDO is more comfortable than Mysqli, because it is more object oriented.

```php

class DBConnec {

    private static $inst = null;
    private $pdo;

    private function __construct() {
        try
        {
            // Conection to the database
            $dsn = 'mysql:host=localhost;dbname=db_organizer';
            $this->pdo = new PDO($dsn, 'root', 'headshot2');
            $this->pdo->exec('SET CHARACTER SET utf8');
            $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }
        catch(PDOException $e)
        {
            // Diplaying errors
            die('Erreur : ' . $e->getMessage());
        }
    }

    public function getPdo() {
        return $this->pdo;
    }

    public static function getConnec() {
        if(is_null(self::$inst)) {
            self::$inst = new DBConnec();
        }
        return self::$inst;
    }


}
```

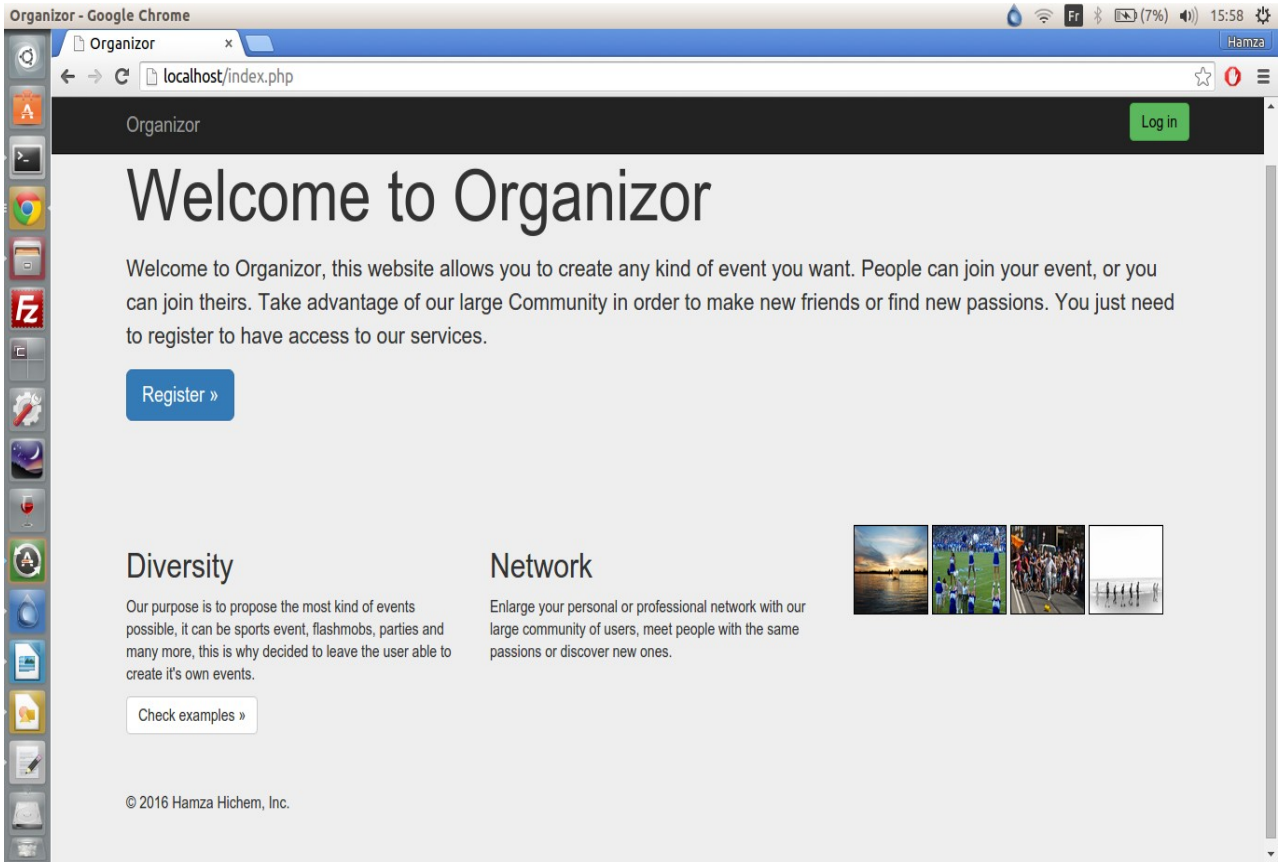
This is a classic singleton with private constructor and static instance which can be called thanks to the static function getConnec(). This function creates the connexion the first time we call it and then call it anytime we use the function again.

V- Main section

1- User guide and sample screen

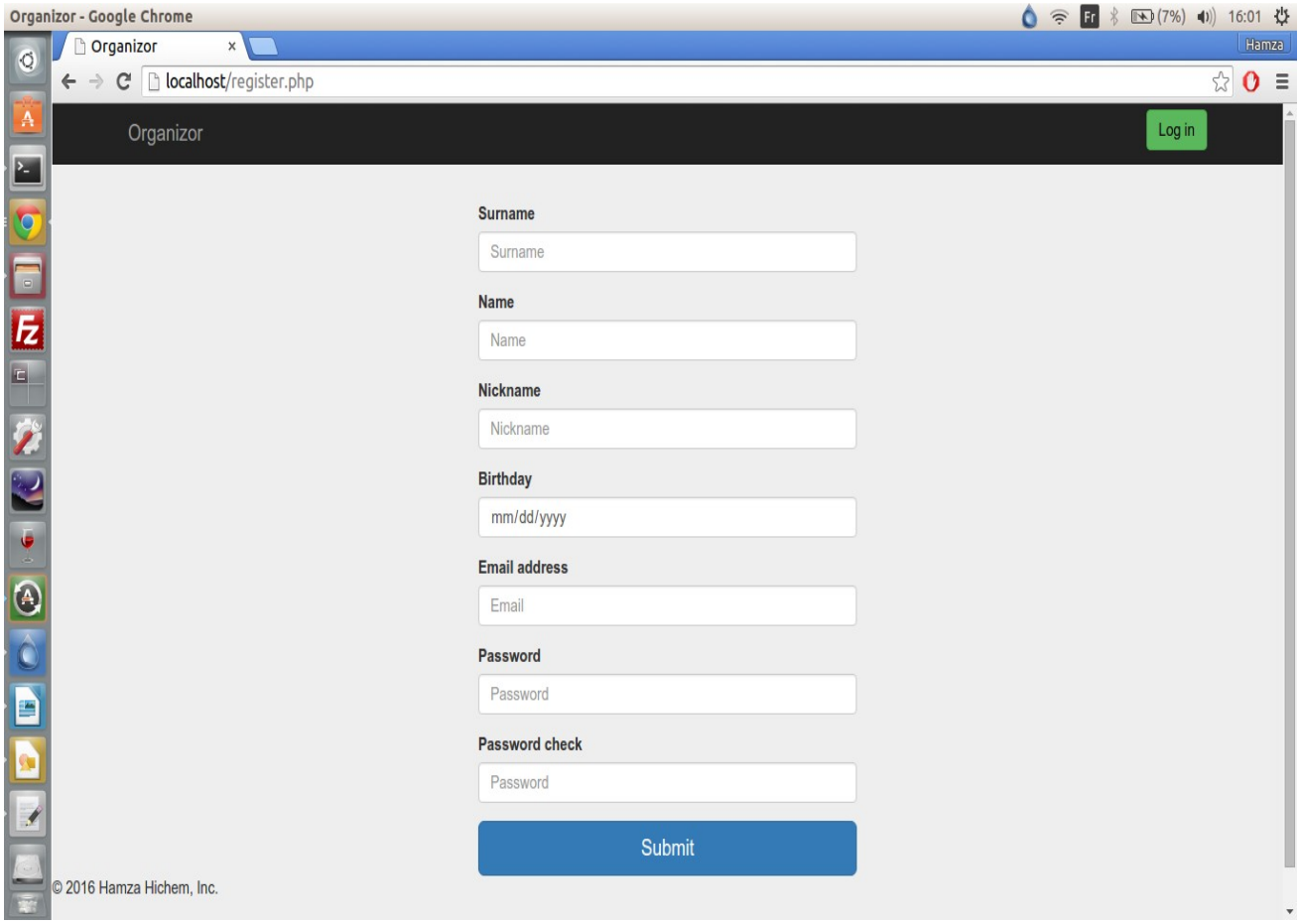
Visitor

main page



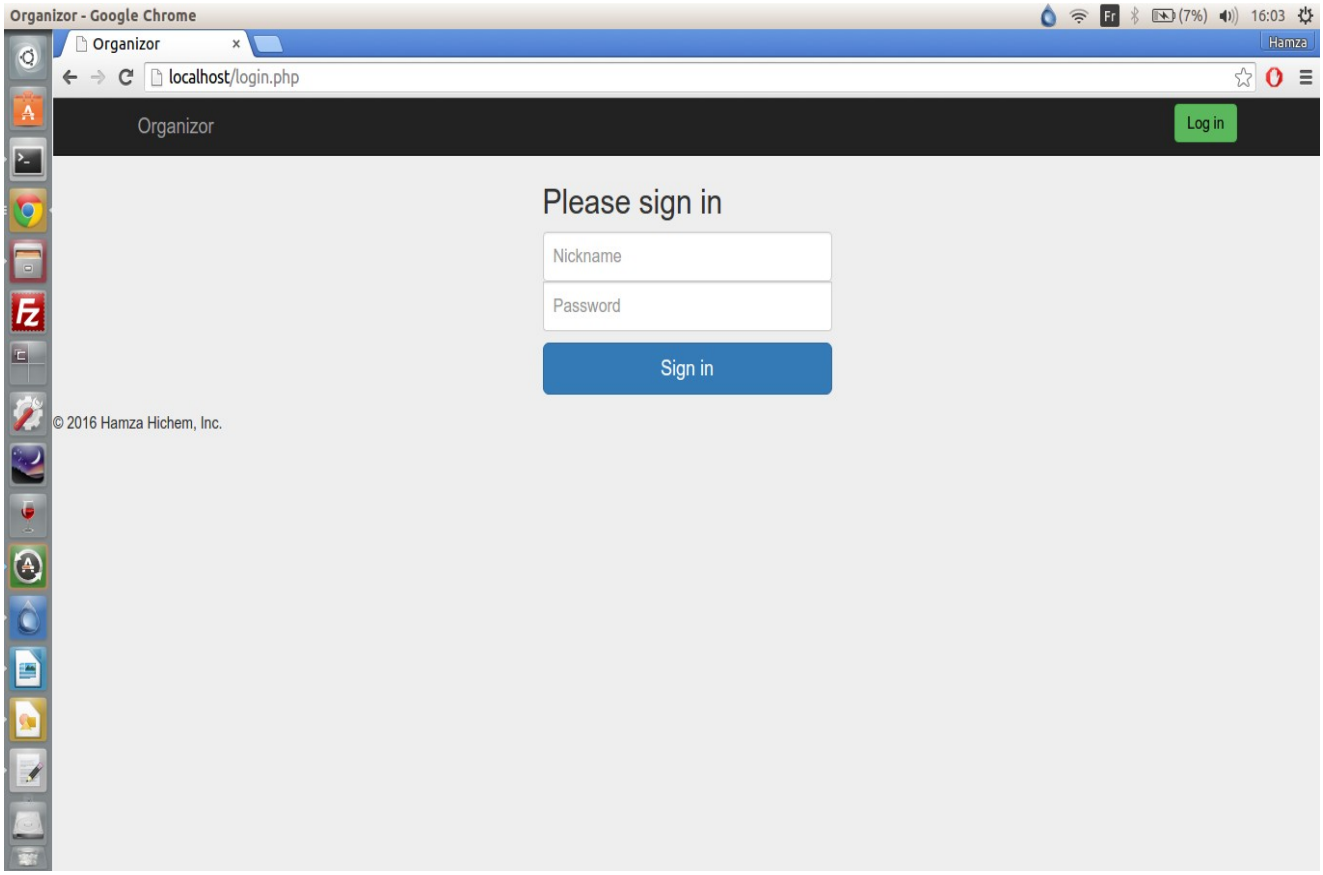
From this page, the visitor can access the registration page or the login page.

Registration page



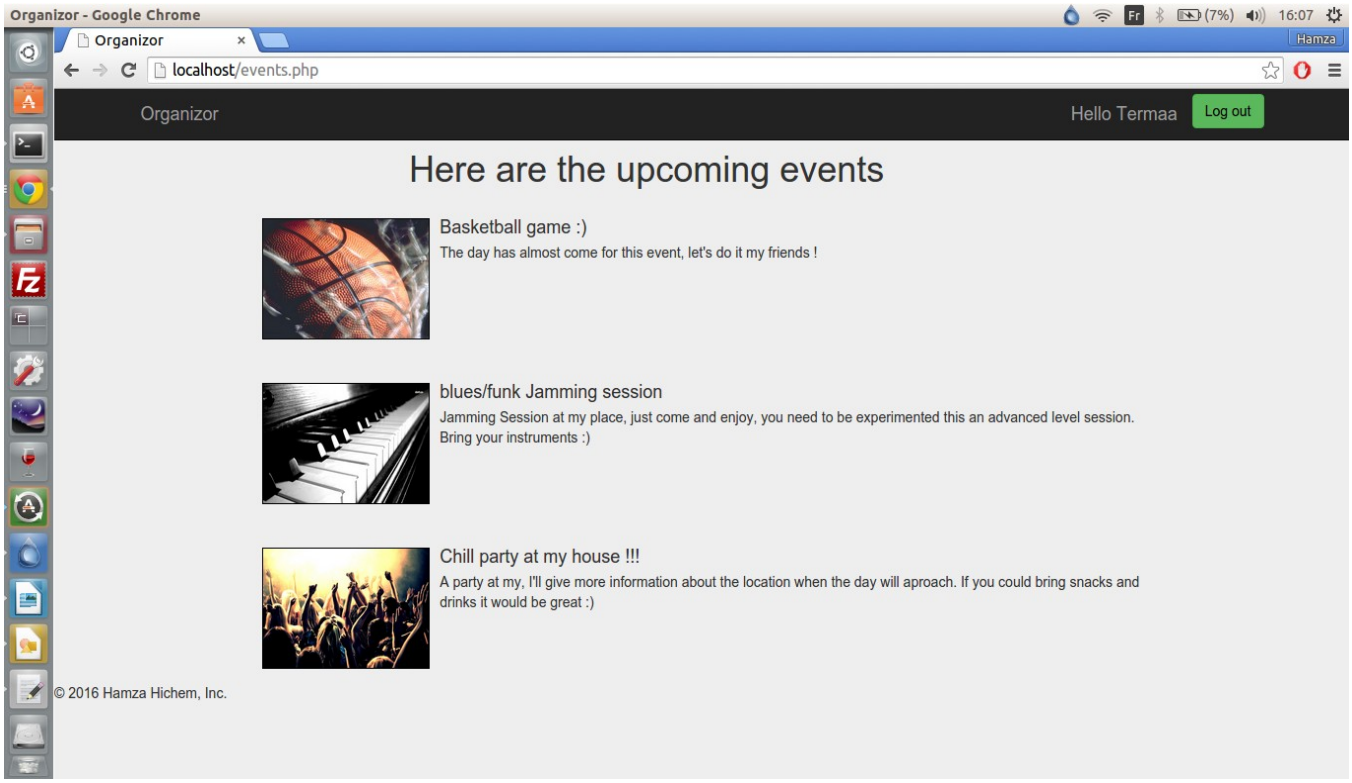
Here the user has to complete all the inputs if he wants to register, if some of the inputs are empty then an error message will appear, the other other cases of errors display are if the nickname already exists or if the two passwords doesn't match.

Login page



A simple page for a basic function, this was explained before on the coding part.

List of events



Here we can see list of available events, If we click on one of them we access to it's page where we can join or manage the event.