**Program 1a. Develop and demonstrate a XHTML file that includes JavaScript script for the following problems:**

**Input: A number n obtained using prompt Output: The first n Fibonacci numbers.**

**File Name = Pg1a.html**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title> Fibonacci Numbers </title>

</head>

<body>

<h1>Calculating the fibonacci numbers</h1>

<script type="text/javascript">

var n,a=0,b=1,i,c

n=prompt("Enter a number ","")

if(n<=0) alert("Invalid number")

else

{

if(n==1) document.write(a)

else document.write(a+"<br />"+b)

for(i=2;i<n;i++)

{

c=a+b

a=b

b=c

document.write("<br />"+c)

}

}

</script>

</body>

</html><body>

<form name = "f1">

computation : <input type = "text" name = "n" />

<input type = "button" value = "compute" onClick="show(f1.n.value);"/>

<input type = "button" value = "Reset"/>

</br>

</br>

result

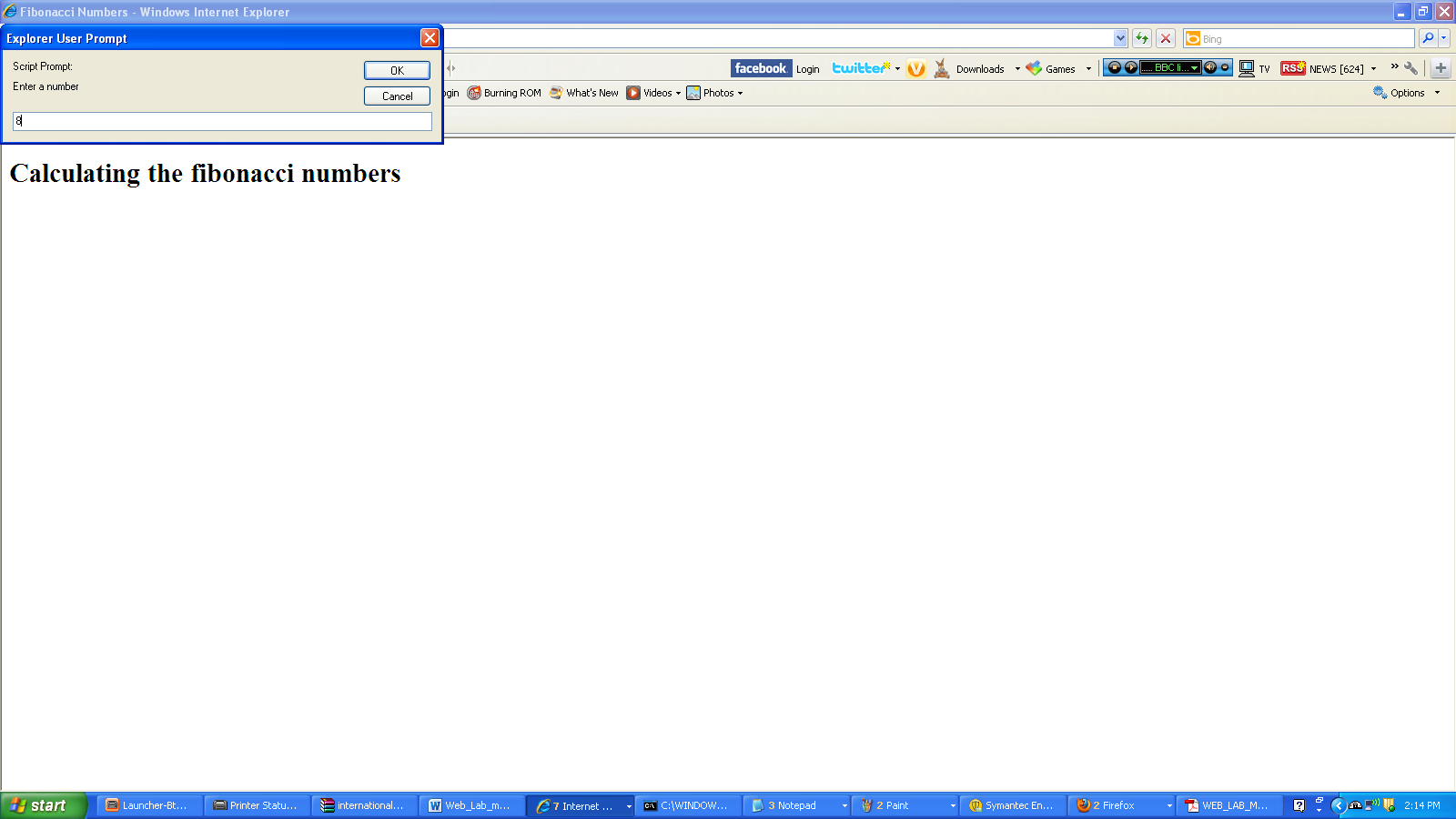
<input type = "text" name = "out" , size = "50"/>

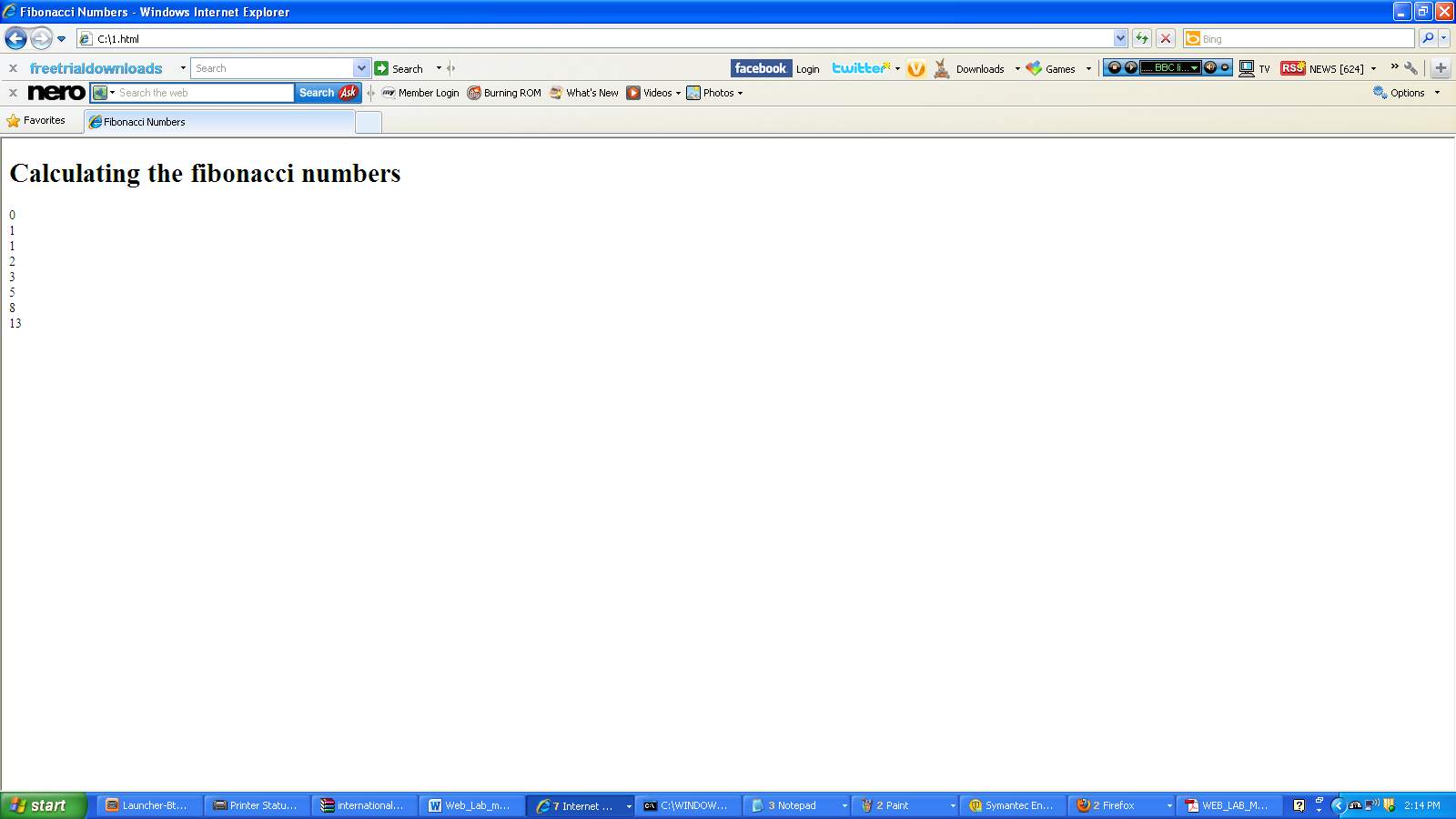
</form>

</body>

</html>

**Output**





**Program 1b Input: A number n obtained using prompt Output: A table of numbers from 1 to n and their squares using alert**

**File Name = pg1b.html**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head> <title>Squares of numbers</title> </head>

<body>

<h1>

Printing numbers &amp; calculating their squares

</h1>

<script type="text/javascript">

var n,i;

n=prompt("Enter a number","");

if(n>0)

{

c="Number | Square"

for(i=1;i<=n;i++) c=(c+"\n"+i+" -----> "+i\*i);

alert(c)

}

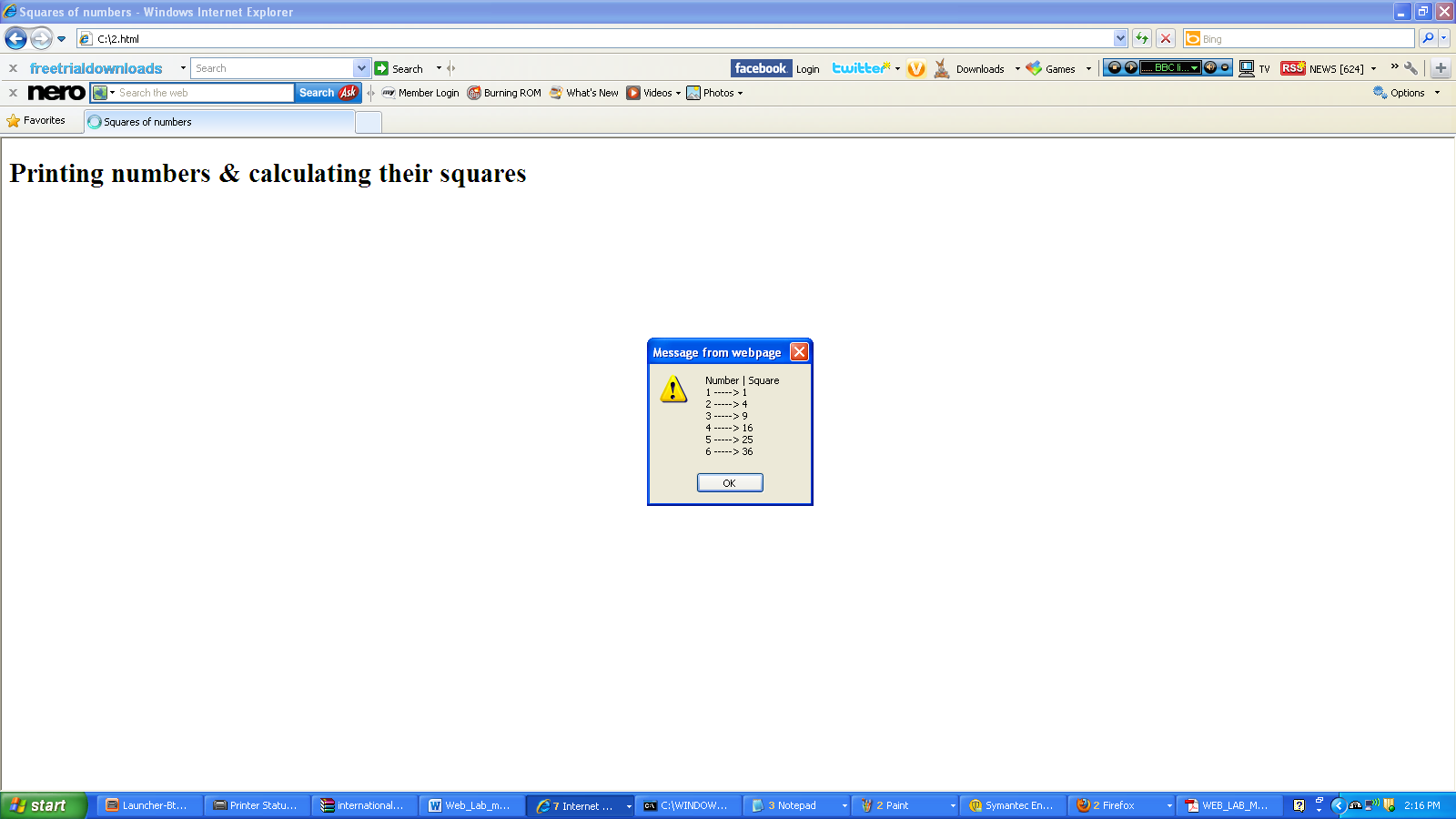
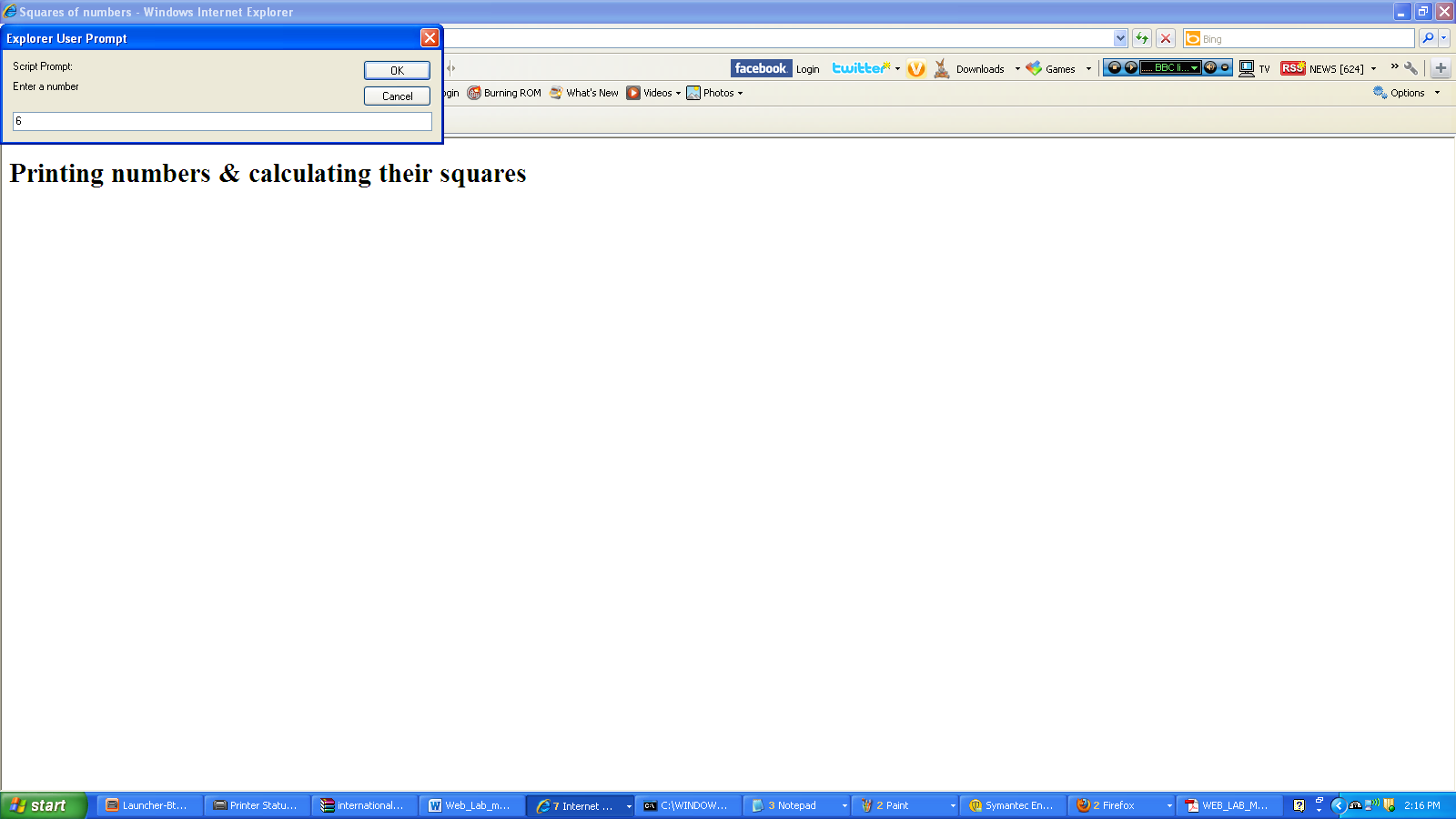
else alert("Enter a number greater than 1.")

</script>

</body>

</html>

**Output**



**Program 2a** **Develop and demonstrate, using JavaScript script, a XHTML document that collects the USN ( the valid format is: A digit from 1 to 4 followed by two upper-case characters followed by two digits followed by two upper-case characters followed by three digits; no embedded spaces allowed) of the user. Event handler must be included for the form element that collects this information to validate the input. Messages in the alert windows must be produced when errors are detected.**

**File name = usn\_2a.html**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head> <title> USN Validation </title> </head>

<body>

<script type="text/javascript">

function func(usn)

{

var pattern1=/^[1-4][A-Z]{2}[0-9]{2}[A-Z]{3}[0-9]{2}$/

if(!usn.value.match(pattern1)||usn.value.length==0)

{

alert("Invalid USN!\nEnter a valid USN")

return false

}

else alert("USN valid!")

}

</script>

<form action="">

<p>

USN: <input type="text" name="usn" /> <br/>

<input type="button" value="Validate" onclick="func(usn)" />

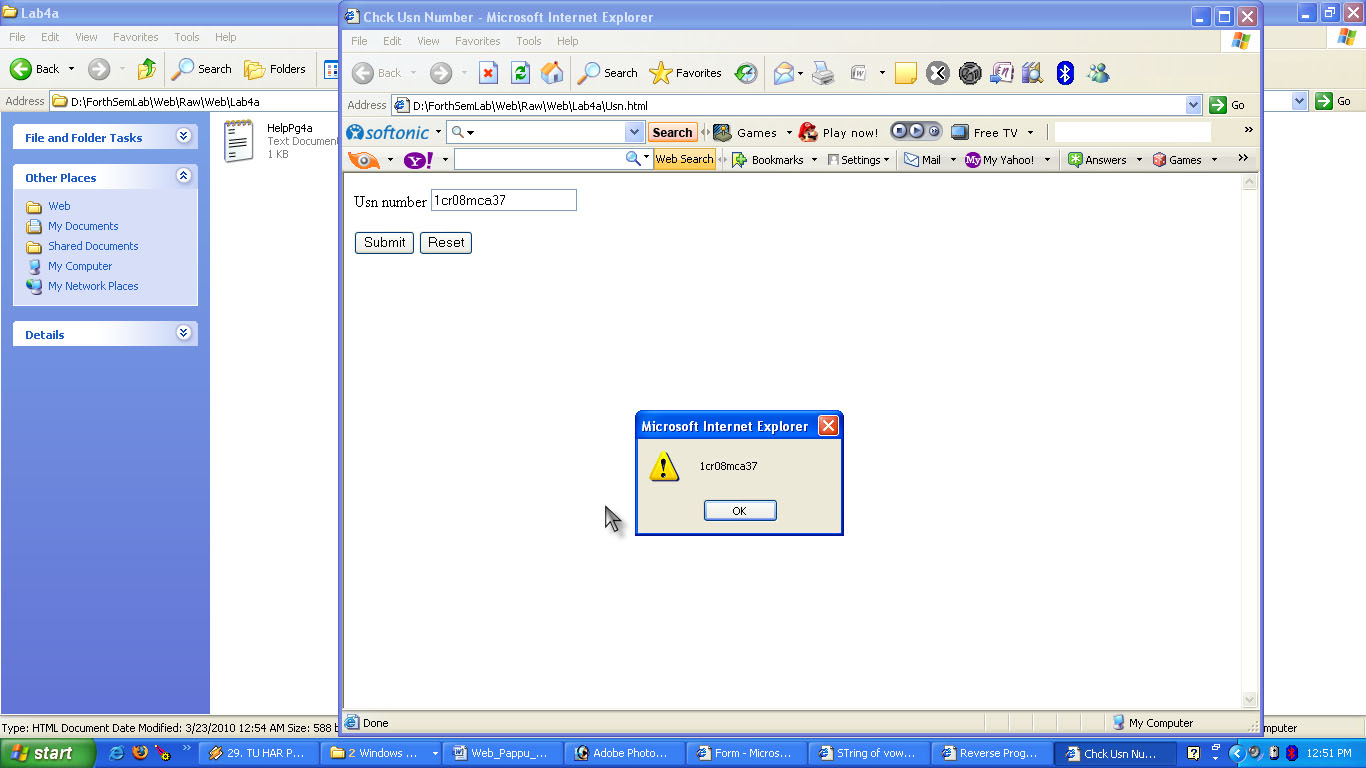
</p>

</form>

</body>

</html>

**Output**



**Program 2b Modify the above program to get the current semester also (restricted to be a number from 1 to 6)**

**File Name = usn\_2b.html**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head> <title>USN and Semester Validation</title> </head>

<body>

<script type="text/javascript">

function disp(usn,sem)

{

var pattern1=/^[1-4][A-Z]{2}[0-9]{2}[A-Z]{3}[0-9]{2}$/

if(!usn.value.match(pattern1)||usn.value.length==0)

{

alert("Invalid USN!\nEnter a valid USN")

return false

}

else alert("USN valid!")

var pattern2=/^[1-6]$/

if(!sem.value.match(pattern2)||sem.value.length==0)

{

alert("Invalid Semester!\nEnter a valid Semester")

return false

}

else alert("Semester valid!")

}

</script>

<form action="">

<p>

USN: <input type="text" name="usn" /> <br/>

Semester: <input type="text" name="sem" /> <br/>

<input type="button" value="Validate" onclick="disp(usn,sem)" />

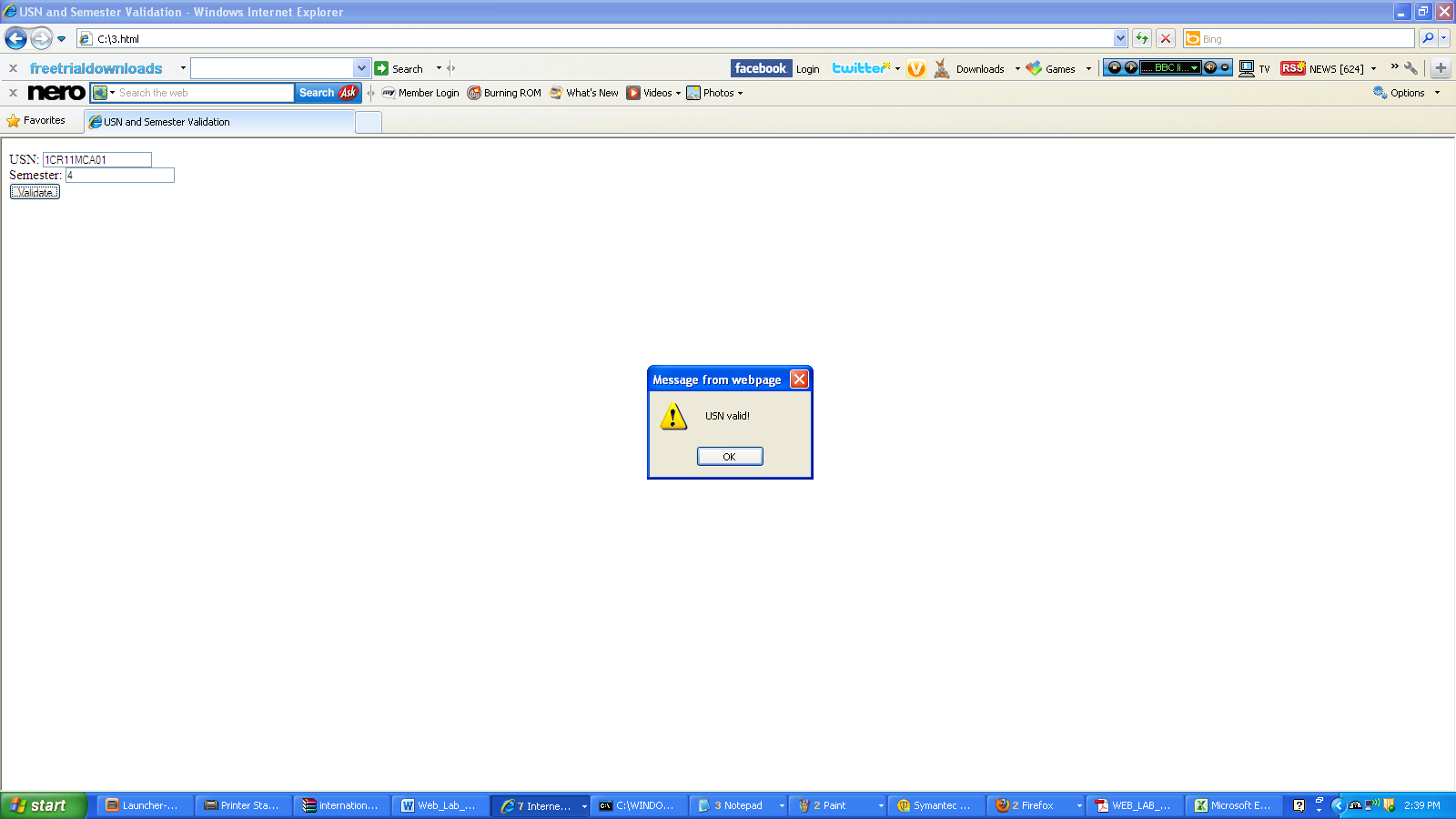
</p>

</form>

</body>

</html>

**Output**



**Program 3a Develop and demonstrate, using JavaScript script, a XHTML document that contains three short paragraphs of text, stacked on top of each other, with only enough of each showing so that the mouse cursor can be placed over some part of them. When the cursor is placed over the exposed part of any paragraph, it should rise to the top to become completely visible.**

**File Name = Pg3a.html**

<html>

<head><title>Stacking</title>

<script type="text/javascript">

function f1(n1)

{

var dom=document.getElementById(n1).style;

dom.zIndex=10;

}

function f2(n2)

{

var dom=document.getElementById(n2).style;

dom.zIndex=0;

}

</script>

<style type="text/css">

.s1 {position:absolute; top:0px; left:0px; z-index:0; background-color:red;}

.s2 {position:absolute; top:20px;left:20px; z-index:0; background-color:blue;}

.s3 {position:absolute; top:30px; left:30px; z-index:0; background-color:cyan;}

</style>

</head>

<body>

<textarea id="t1" class="s1" rows="5" cols="12" onmouseover="f1('t1');" onmouseout="f2('t1');">

This is textarea1..

</textarea>

<textarea id="t2" class="s2" rows="7" cols="15" onmouseover="f1('t2');" onmouseout="f2('t2');">

This is textarea2..

</textarea>

<textarea id="t3" class="s3" rows="9" cols="20" onmouseover="f1('t3');" onmouseout="f2('t3');">

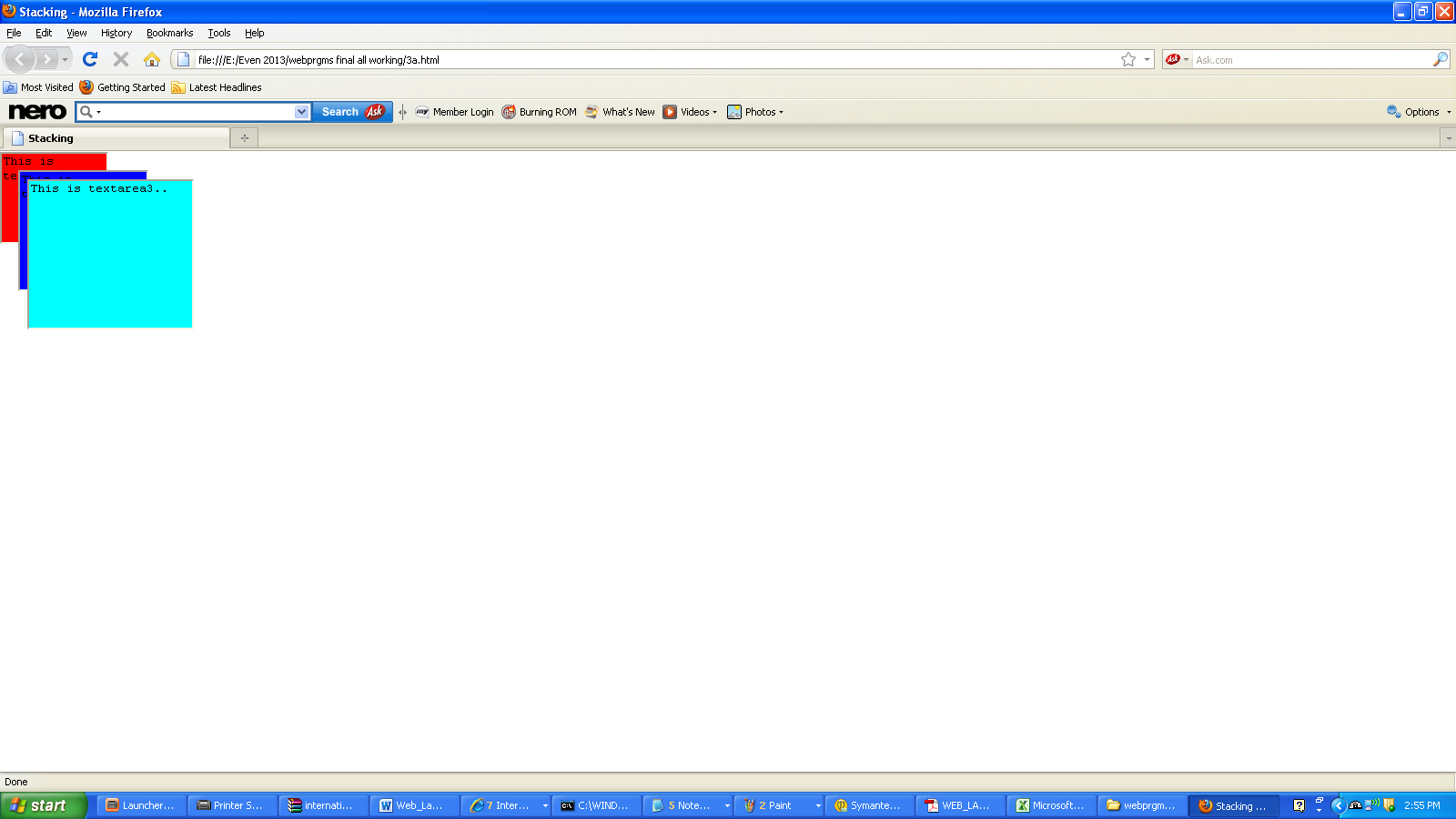
This is textarea3..

</textarea>

</body>

</html>

**Output**



**Program 3b Modify the above document so that when a paragraph is moved from the top stacking position, it returns to its original position rather than to the bottom.**

**File Name = Pg3b.html**

<html>

<head><title>Stacking</title>

<script type="text/javascript">

var top=1;

function f1(n1)

{

var dom=document.getElementById(n1).style;

dom.zIndex=20;

top.zIndex=0;

top=dom;

}

</script>

<style type="text/css">

.s1 {position:absolute; top:0px; left:0px; z-index:0;}

.s2 {position:absolute; top:10px;left:10px; z-index:0;}

.s3 {position:absolute; top:20px; left:20px; z-index:0;}

</style>

</head>

<body>

<textarea id="1" class="s1" onclick="f1('1');">

textarea1

</textarea>

<br/>

<textarea id="2" class="s2" onclick="f1('2');">

textarea2

</textarea>

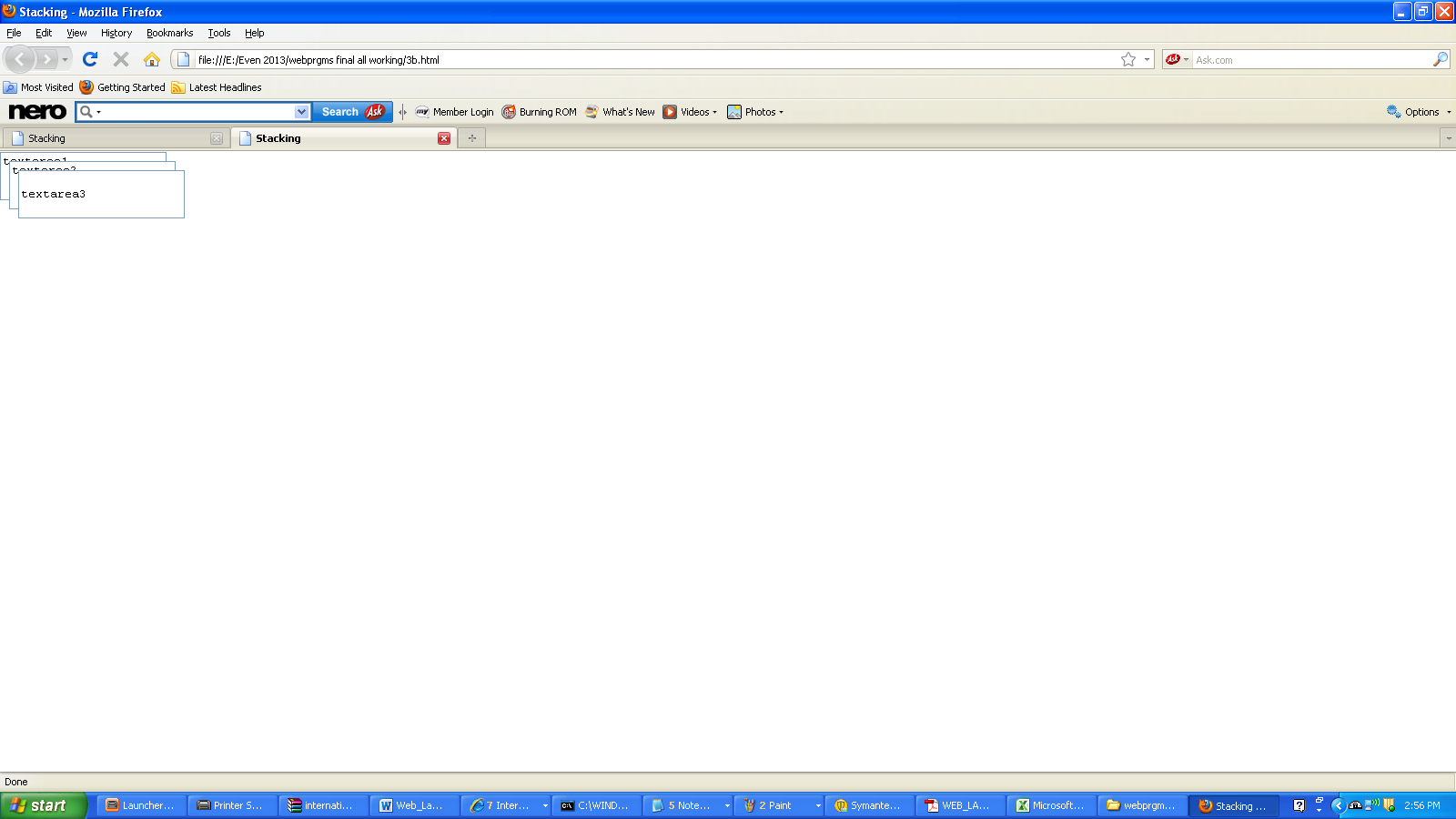
<br/>

<textarea id="3" class="s3" onclick="f1('3');">

textarea3

}

**Output**



**Program 4a** **Design an XML document to store information about a student in an Engg. College affiliated to VTU. The information must include USN, NAME, Name of the College, Branch, Year of Joining and E-Mail-Id. Make up sample data for 3 students .Create a CSS style sheet and use it to display the document.**

**File Name = Student.css**

STUDENT

{

}

DETAILS

{

display:block;

color:"black";

font-size:"30pt";

font-family:"Times New Roman";

font-weight:"bold";

border-style:"dashed";

}

TITLE

{

display:block;

color:"pink";

font-size:"20pt";

text-decoration:"underline";

font-family:"Times New Roman";

text-indent:"1.5in";

}

USN

{

display:block;

font-size:"30pt";

font-family:"Times New Roman";

color:"red";

}

NAME

{

display:block;

font-size:"18pt";

font-family:"Times New Roman";

color:"black";

}

COLLEGE

{

display:block;

font-size:"18pt";

font-family:"Times New Roman";

color:"black";

}

BRANCH

{

display:block;

font-size:"18pt";

font-family:"Times New Roman";

color:"black";

}

YEAR

{

display:block;

font-size:"18pt";

font-family:"Times New Roman";

color:"black";

}

EMAIL-ID

{

display:block;

font-size:"18pt";

font-family:"Times New Roman";

color:"black";

}

**File Name = Student.xml**

<?xml version ="1.0" encoding="utf-8"?>

<?xml-stylesheet type="text/css" href="student.css"?>

<STUDENT>

<DETAIL>

<TITLE> STUDENT INFORMATION </TITLE>

</DETAIL>

<DETAILS>

<USN><A>1CR08MCA50</A></USN>

<NAME>PAPPU</NAME>

<COLLEGE>CMR INSTITUTE OF TECHNOLOGY</COLLEGE>

<BRANCH>4TH SEM</BRANCH>

<YEAR>2010</YEAR>

<EMAIL-ID>PAPPU@YAHOO.COM</EMAIL-ID>

</DETAILS>

<DETAILS>

<USN>1CR08MCA27</USN>

<NAME>FARUKH</NAME>

<COLLEGE>CMR INSTITUTE OF TECHNOLOGY</COLLEGE>

<BRANCH>4TH SEM</BRANCH>

<YEAR>2010</YEAR>

<EMAIL-ID>FAROOQS\_13@YAHOO.COM</EMAIL-ID>

</DETAILS>

<DETAILS>

<USN>1CR08MCA56</USN>

<NAME>CHINTU</NAME>

<COLLEGE>CMR INSTITUTE OF TECHNOLOGY</COLLEGE>

<BRANCH>4TH SEM</BRANCH>

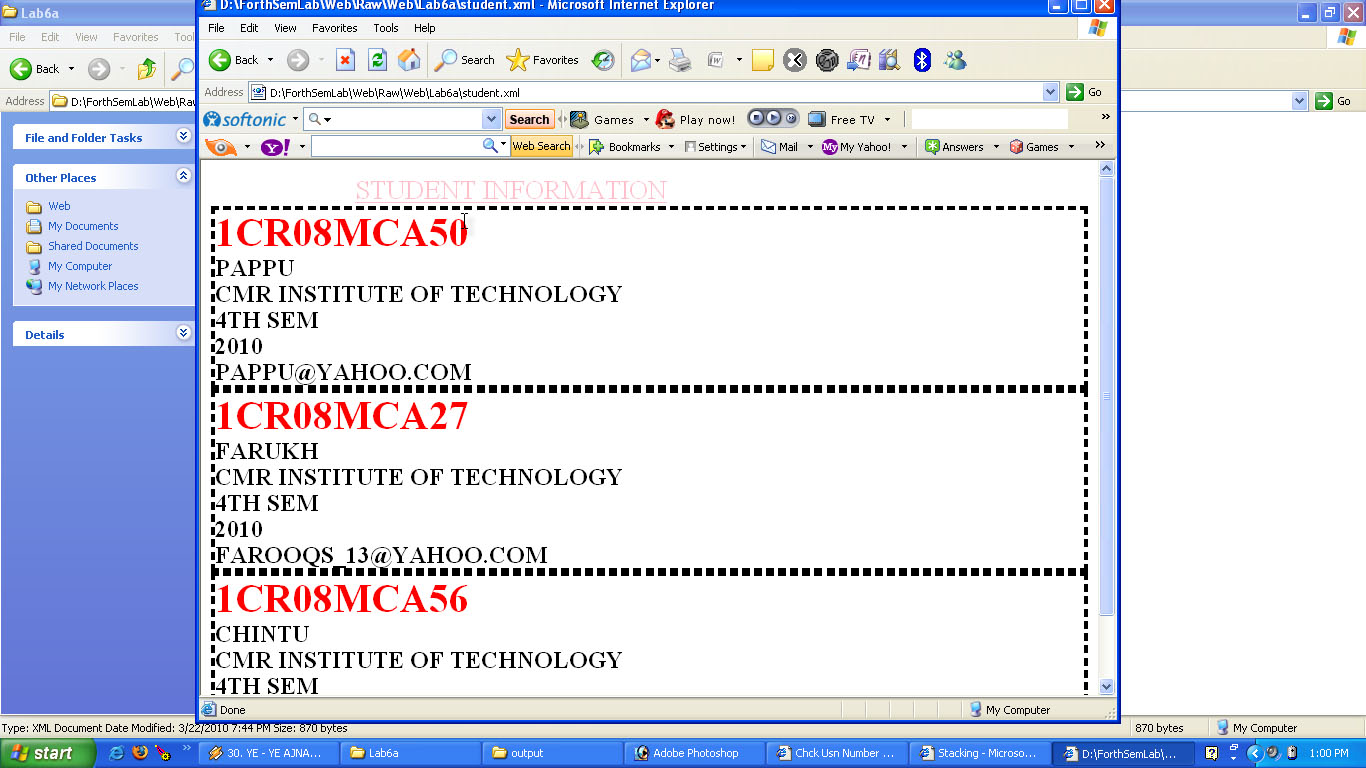
<YEAR>2010</YEAR>

<EMAIL-ID>Ramu@YAHOO.COM</EMAIL-ID>

</DETAILS>

</STUDENT>

**Output**



**Program 4b Create an XSLT style sheet for one student element of the above document and use it to create a display of that element**.

**File Name = 4b.xsl**

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

xmlns="http://www.w3.org/1999/XHTML">

<xsl:template match="STUDENT">

<html>

<head>

<title>xsl document</title>

</head>

<body>

<center>STUDENT INFORMATION</center>

<table border="3" align="center">

<tr>

<th>

<span style="font-style:italic;font-size:25;">USN</span>

</th>

<th>

<span style="font-style:italic;font-size:25;">NAME</span>

</th>

<th>

<span style="font-style:italic;font-size:25;">NAME</span>

</th>

</tr>

<xsl:for-each select="DETAILS">

<tr>

<td>

<xsl:value-of select="USN"/>

</td>

<td>

<xsl:value-of select="NAME"/>

</td>

</tr>

</xsl:for-each>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

**File Name = Sample.xml**

<?xml-stylesheet type="text/xsl" href="6b.xsl"?>

<STUDENT>

<DETAILS>

<USN>1cr08mca37</USN>

<NAME>Pappu</NAME>

<COLLEGE>cmrit</COLLEGE>

<BRANCH>MCA</BRANCH>

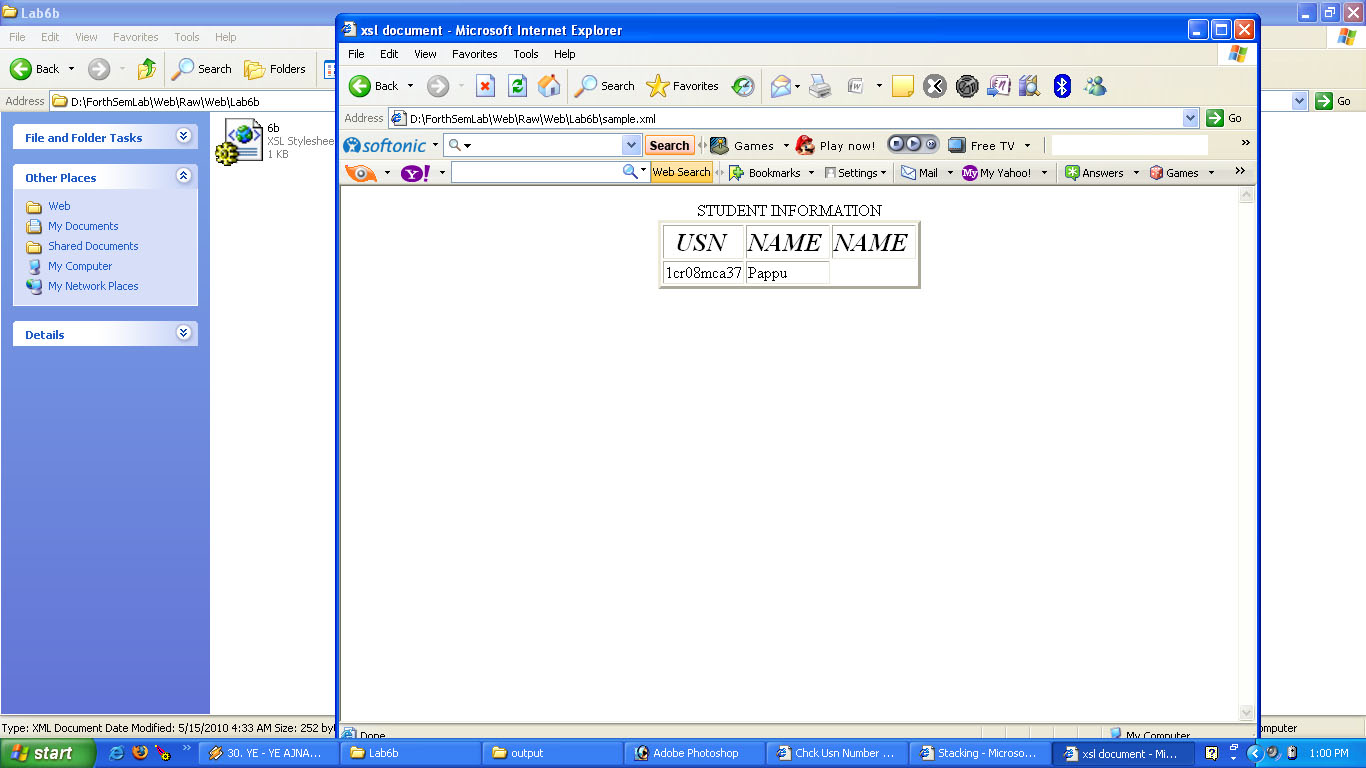
<YEAR>2010</YEAR>

<EMAILID>pappu\_gaya21198@yahoo.co.in</EMAILID>

</DETAILS>

</STUDENT>

**Output**



# Setting Up Perl and CGI For Wamp Server WAMP(P)

**Step 1:**

First you need to download Wamp Server from [www.wampserver.com](http://www.wampserver.com/en/download.php) and install Wamp Server on your machine. The default installation directory is ‘C:\wamp” and here I am using the default options for installation. To complete the installation you have to set the host name for your mail server and your email address, here you can leave the default option again. That will do no harm.

The current Wamp Server will install Apache 2.2.11, PHP 5.2.9-2 + PECL, MySQL 5.1.33, SQLitemanager and PhpMyadmin.

**Step 2:**

Now you have to download ActivePerl (currently 5.10.0) from [www.activestate.com](http://www.activestate.com/activeperl/) and install it. The default installation directory is “C:\Perl“, but for simplicity and ease of use I use different directory. I create a new folder name “perl” inside “C:\wamp\bin“. So I install Active Perl in “C:\wamp\bin\perl” directory. The next thing you need to do is configure the Apache web server to execute Perl and CGI script.

**Step 3:**

This is the most important part here. You need to edit the Apache configuration file. Now go to “C:\wamp\bin\apache\Apache2.2.11\conf” directory and open “httpd.conf” file. Edit the httpd.conf file as below.

1. Inside httpd.conf, look for the line that says “<Directory “c:/wamp/www/”>“, just a few lines below this you’ll find the line that says “Options Indexes FollowSymLinks“. Add “Includes ExecCGI” in the line just next to FollowSymLinks, thus it look like this

[**view plaincopy to clipboardprint?**](http://chromicdesign.com/2009/05/setting-up-perl-for-wampp.html)

1. Options Indexes FollowSymLinks Includes ExecCGI

Options Indexes FollowSymLinks Includes ExecCGI

This will enable CGI script inside your www folder.

2. Now look for the line “AddHandler cgi-script .cgi“, this line is commented out. You need to enable this by un-comment this line, to do that remove the # character at the beginning of this line. This will add handler for files with .cgi extension. If you want to use .pl file extension in your server add “AddHandler cgi-script .pl” just below the above line. Now you will be able to execute CGI and Perl script with .cgi and .pl, extension.

Lines to add

[**view plaincopy to clipboardprint?**](http://chromicdesign.com/2009/05/setting-up-perl-for-wampp.html)

1. AddHandler cgi-script .cgi
2. AddHandler cgi-script .pl

AddHandler cgi-script .cgi

AddHandler cgi-script .pl

3. To add directory index file, look for the line “DirectoryIndex index.php index.php3 index.html index.htm“. Add index.cgi and index.pl in this line.

Lines to add

[**view plaincopy to clipboardprint?**](http://chromicdesign.com/2009/05/setting-up-perl-for-wampp.html)

1. DirectoryIndex index.php index.php3 index.html index.htm index.cgi index.pl

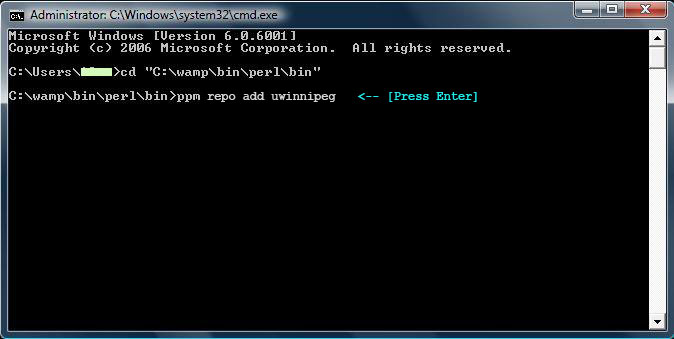
DirectoryIndex index.php index.php3 index.html index.htm index.cgi index.pl

Alternative: If you do not want to waste your time doing the above 3 steps, you can download the edited configuration file [httpd.conf](http://www.chromicdesign.com/downloads/httpd.conf) here. Replace the one inside your apache directory with this one.

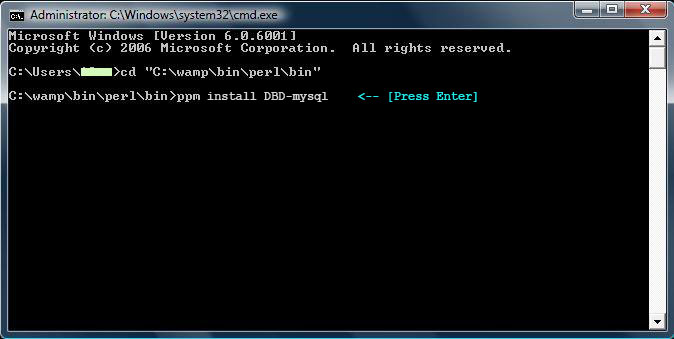
**Step 4:**

Your server is now configured and ready to run perl and cgi script. Next thing you might need to do is to configure perl to use mysql database. You need to download and install mysql driver to enable database connection through your perl script. You have to grab the driver from the ActivePerl package repository. However, mysql driver module is not available in the default ActivePerl Package Repository. So, you need to add additional repository and install from that repository. Follow the steps below:

1. Open command prompt [type cmd in run], then type “ppm repo add uwinnipeg” and press enter.

[](http://www.chromicdesign.com/wp-content/uploads/2009/05/perl-ppm-command-line1.jpg)

2. After the “uwinnipeg” repository is added successfully, you can install DBD-mysql by typing this command “ppm install DBD-mysql” and hit enter.

[](http://www.chromicdesign.com/wp-content/uploads/2009/05/perl-ppm-command-line2.jpg)

***Note:*** The ActivePerl default package repository contains DBD-mysqlPP module. If you install that one, you will get an error in your SQL SELECT query and especially when you use the WHERE clause. Your localhost will hang if you run this kind of query with the WHERE clause, so to get it work you need to install the package that I mentioned above only.

**Program 5a. Write a Perl program to display various Server informations like Server Name, Server Software, Server Protocol, CGI Revision, etc.**

**File name =Pg5a.cgi**

#!c:/wamp/bin/perl/bin/perl.exe

print "Content-type: text/html", "\n\n";

print "<HTML>", "\n";

print "<HEAD><TITLE>About this Server</TITLE></HEAD>", "\n";

print "<BODY><H1>About this Server</H1>", "\n";

print "<HR><PRE>";

print "Server Name: ", $ENV{'SERVER\_NAME'}, "<BR>", "\n";

print "Running on Port: ", $ENV{'SERVER\_PORT'}, "<BR>", "\n";

print "Server Software: ", $ENV{'SERVER\_SOFTWARE'}, "<BR>", "\n";

print "Server Protocol: ", $ENV{'SERVER\_PROTOCOL'}, "<BR>", "\n";

print "CGI Revision: ", $ENV{'GATEWAY\_INTERFACE'}, "<BR>", "\n";

print "<HR></PRE>", "\n";

print "</BODY></HTML>", "\n";

exit (0);

**Output**

|  |  |
| --- | --- |
|  |  |

**Program 5b. Write a Perl program to accept UNIX command from a HTML form and to display the output of the command executed.**

**File Name = Pg5b.html**

<html>

<title>UNIX Command</title>

<body>

<h2>UNIX or Windows Command</h2>

<form action='http://localhost/cgi-bin/cmd.cgi'>

<input type='text' name='cmd'/><br/><br/>

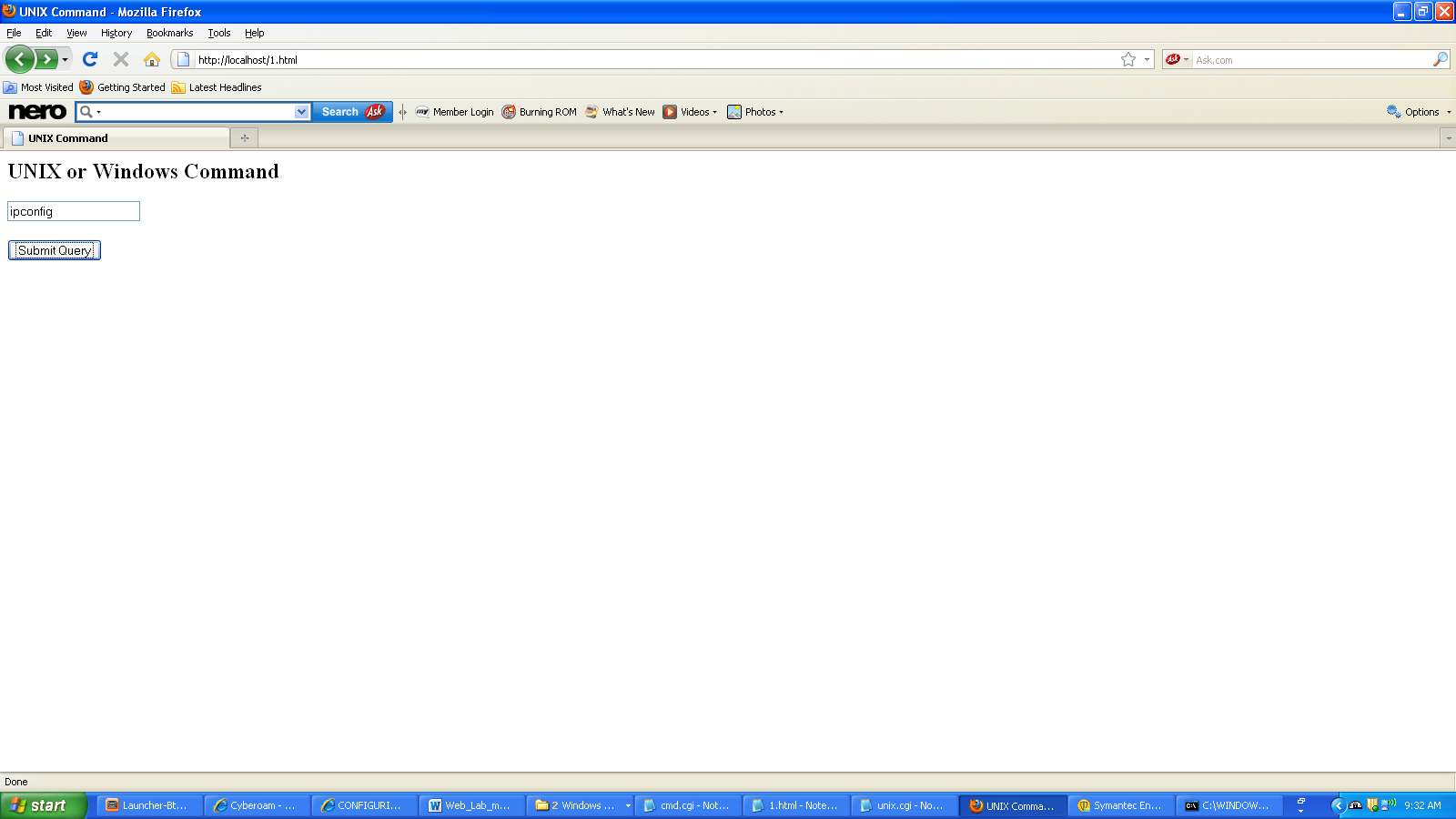
<input type='submit'/>

</form>

</body>

</html>

**Output**



**File Name = Pg5b.cgi(Windows Command)**

#!C:\wamp\bin\perl\bin\perl.exe

use CGI ':standard';

print "Content-type:text/html\n\n";

$a=param('cmd');

system($a);

*#!/usr/bin****/***perl**(Unix Command)**

use CGI':standard';

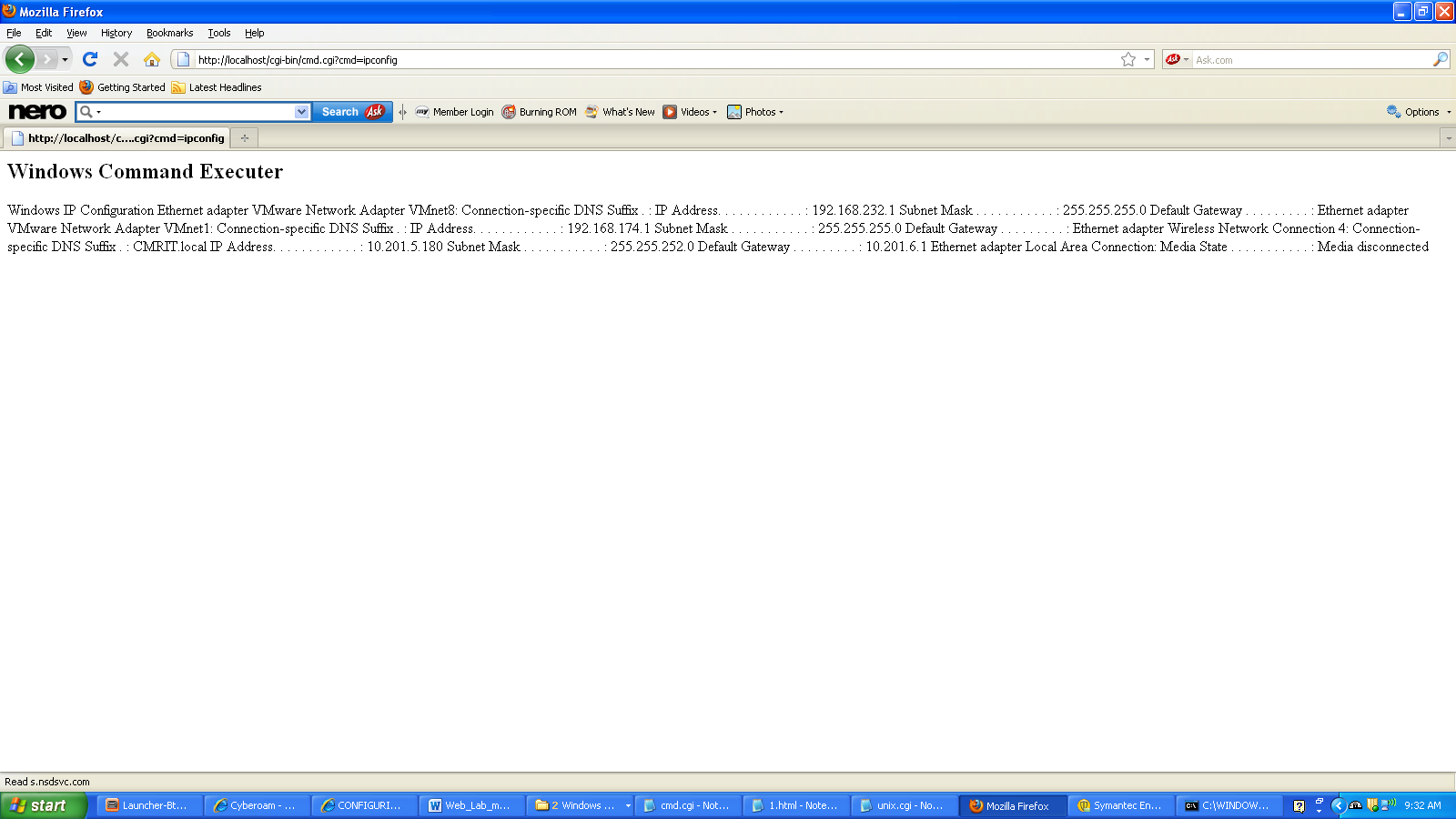
print "content-type:text/html\n\n";

$c=param('cmd');

system($c);

exit(0);

**Output**



|  |  |
| --- | --- |
|  |  |

**Program 6a Write a Perl program to accept the User Name and display a greeting message randomly chosen from a list of 4 greeting messages.**

**File Name = Greeting.html**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head> <title>Accept user name</title> </head>

<body>

<form action ="http://localhost/cgi-bin/Greetings.cgi" method="post">

<p> Username: <input type="text" name="username" />

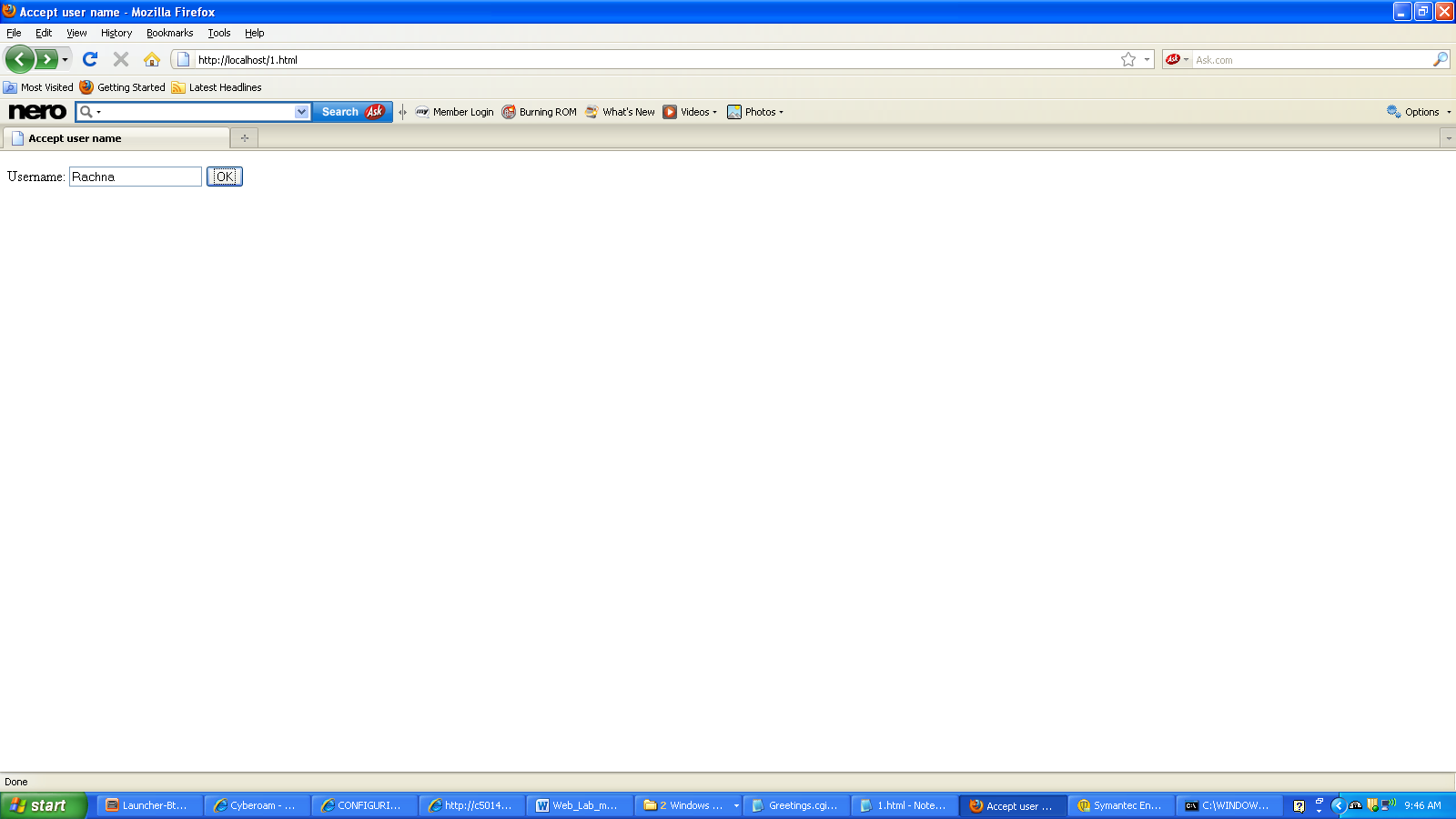
<input type="submit" value="OK" /> </p>

</form>

</body>

</html>

**Output**



**File Name = Greetings.cgi**

#!C:\wamp\bin\perl\bin\perl.exe

use CGI ':standard';

print "Content-type:text/html\n\n";

@greet=("Good morning","Welcome","How are you doing?","Hello!");

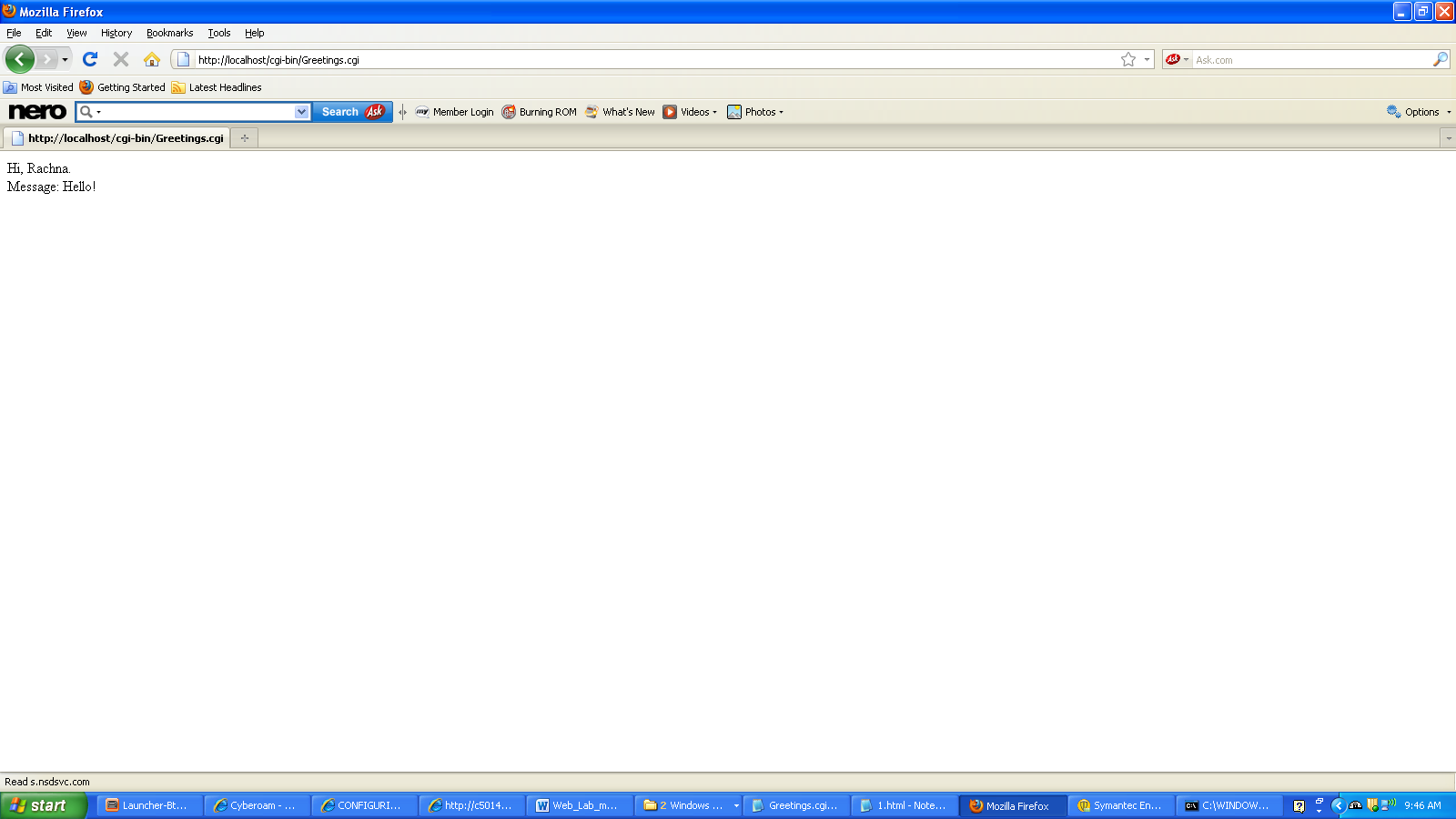
$a=int(rand(4));

$b=param('greet');

$input=param('username');

print "Hi, $input.<br>Message: " , $greet [$a],$b;

**Output**



**Program 6b**. **Write a Perl program to keep track of the number of visitors visiting the web page and to display this count of visitors, with proper headings.**

**File Name = Pg6b.html**

<html>

<head>

<title> check perl with pl </title>

</head>

<form action="http://localhost/cgi-bin/pg6b.cgi">

<body>

<p> Number of Visits to this page</p>

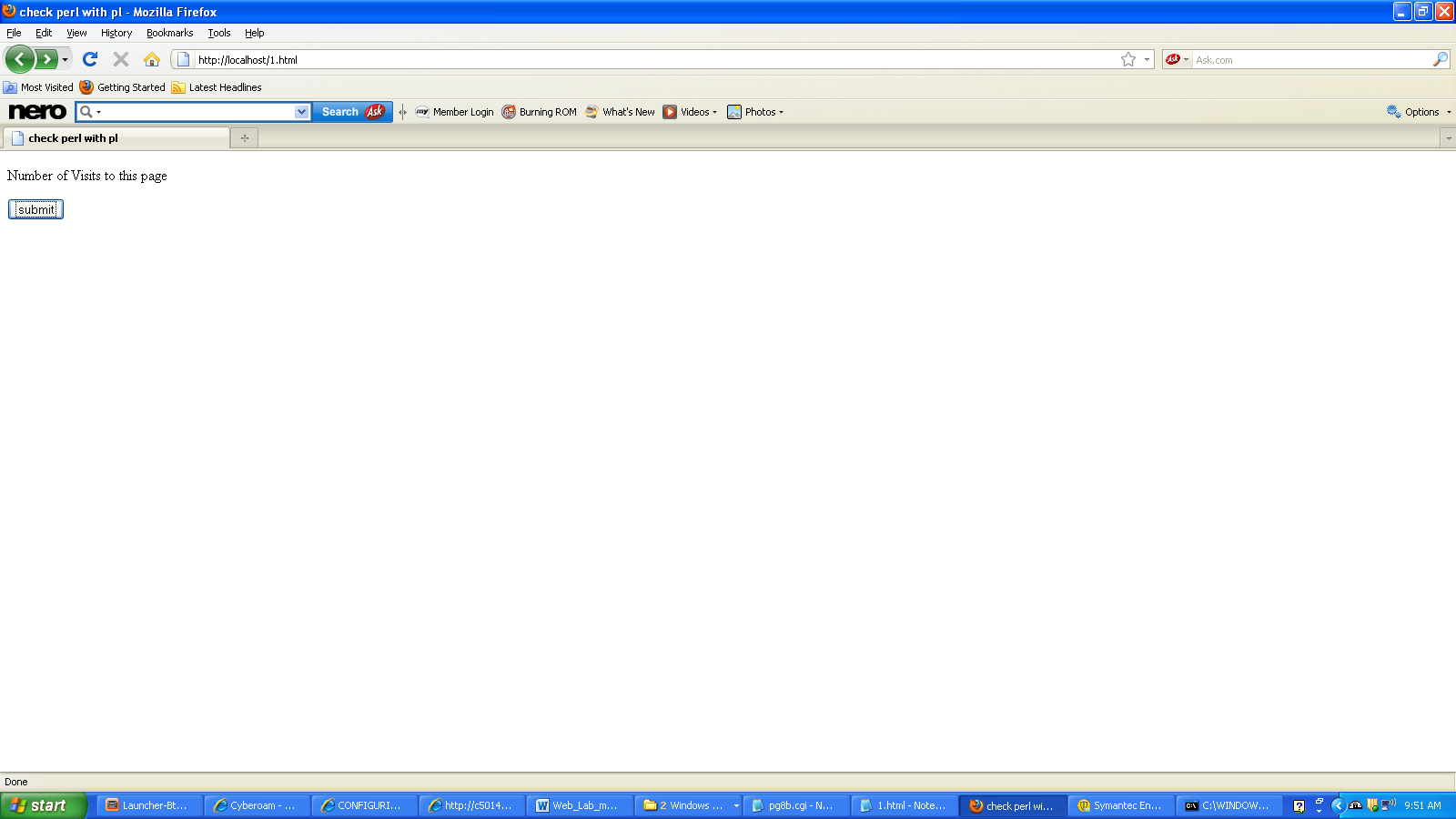
<input type="submit" value="submit"/>

</body>

</form>

</html>

**Output**



**File name = Pg6b.cgi**

#!C:\wamp\bin\perl\bin\perl.exe

use CGI':standard';

print "content-type:text/html \n\n";

# Requires a file 'count.dat' to pre-exist with the content '0'

open FILE, "<count.dat";

my $count = <FILE>;

close(FILE);

$count++;

open Handler, ">count.dat";

print Handler $count ;

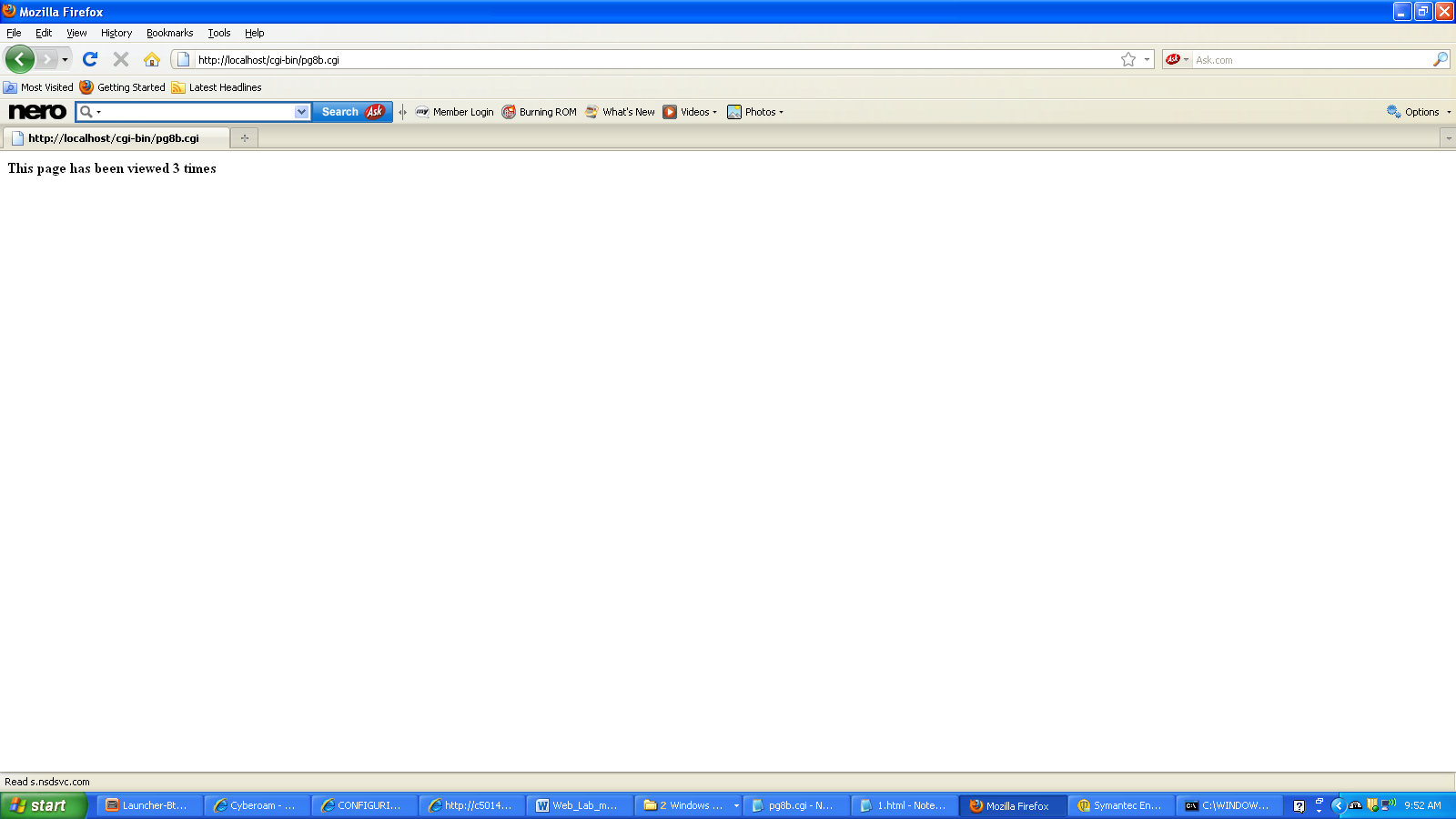
close Handler;

open FILE, "<count.dat";

my $count2 = <FILE>;

close(FILE);

print b("This page has been viewed $count times");print "Visitors Count: $c";



**Program 7. Write a Perl program to display a digital clock which displays the current time of the server.**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Digital Clock Display Program</title>

</head>

<body>

<form action="http://localhost/cgi-bin/DigitalClock.cgi" method="post">

<p>

Digital Clock <input type="submit" value="Click Here" />

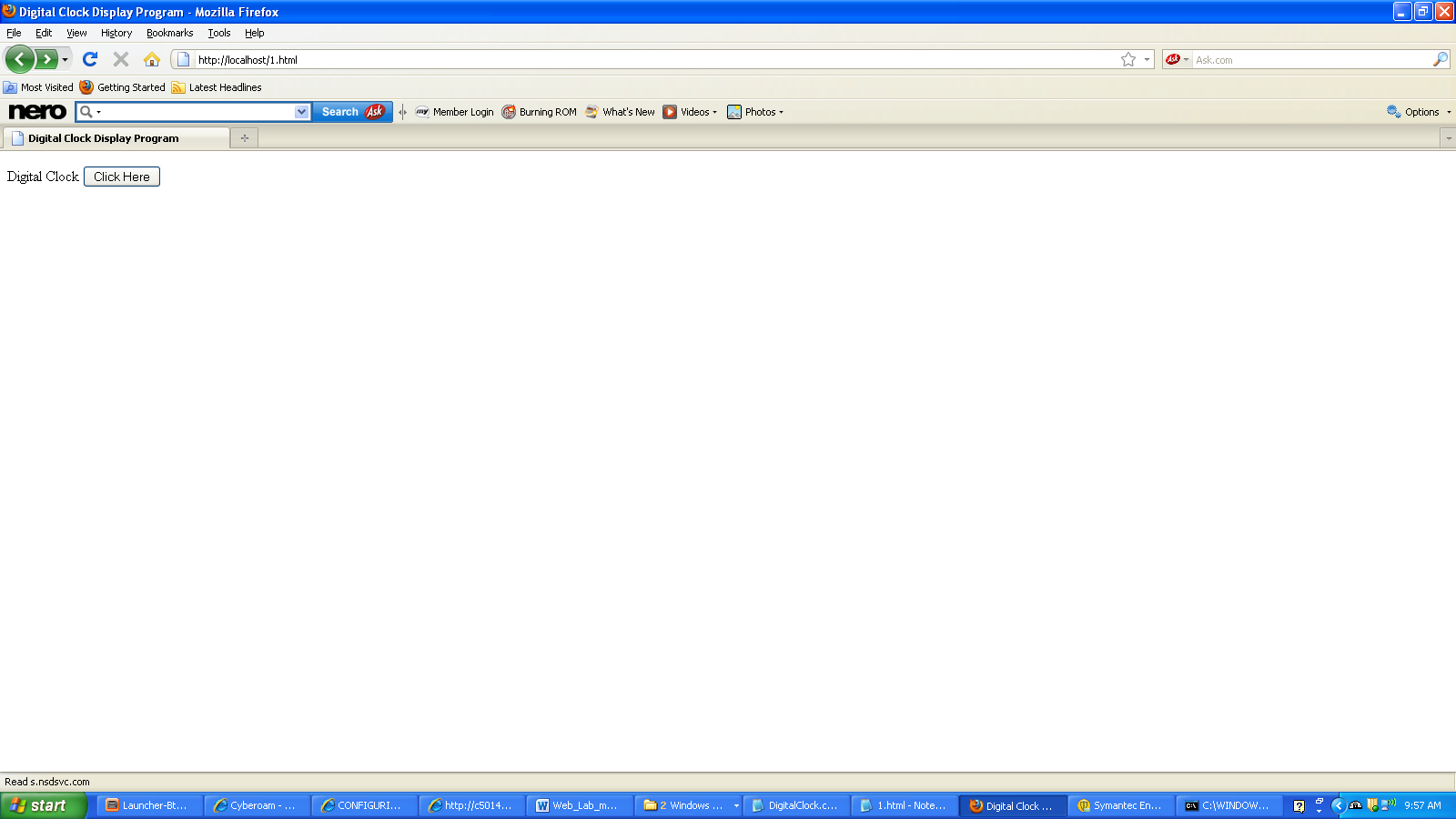
</p>

</form>

</body>

</html>

**Output**



#!C:\wamp\bin\perl\bin\perl.exe

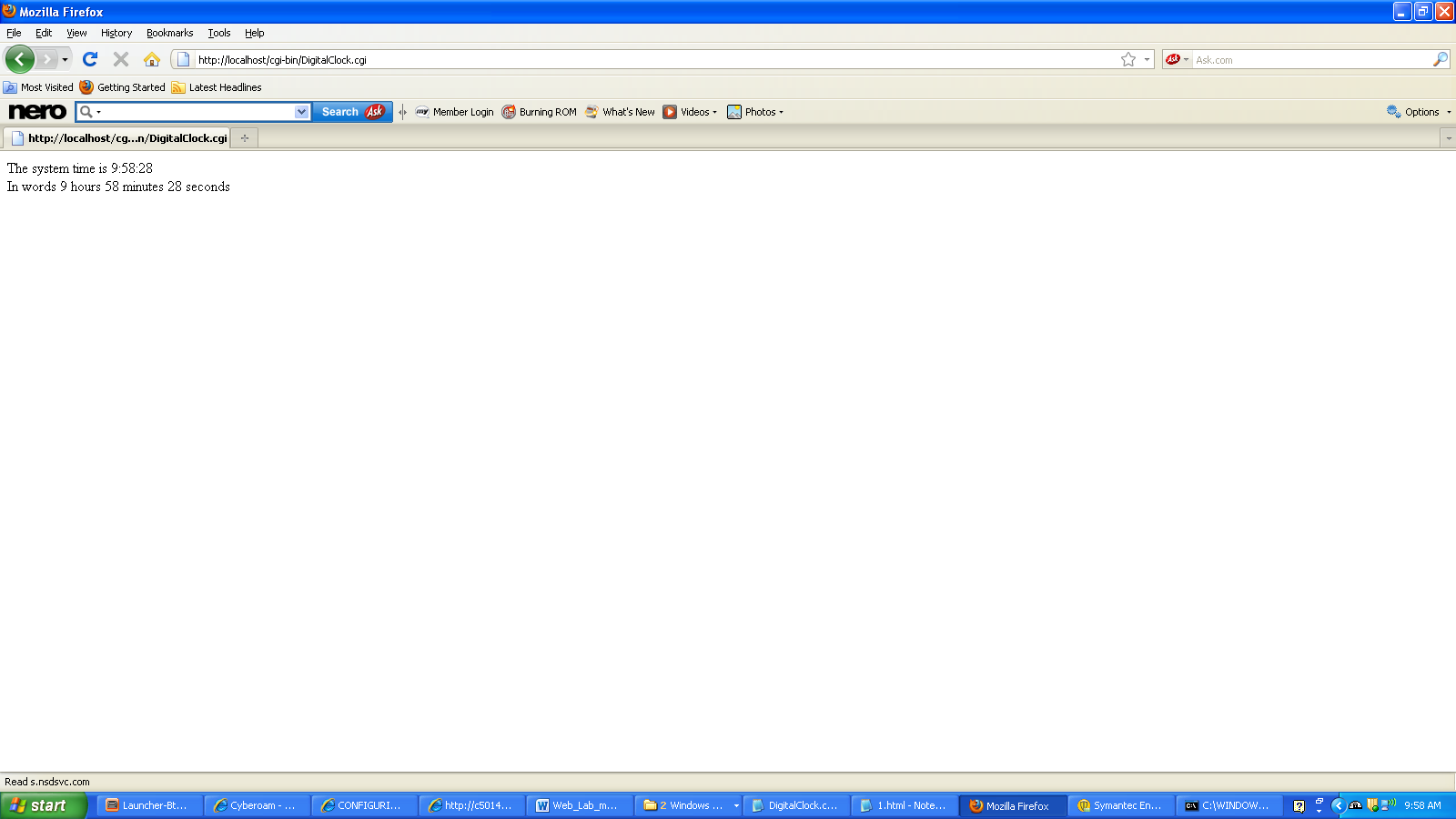
use CGI':standard';

print "content-type:text/html \n\n";

print " "; ($s,$m,$h)=localtime(time);

print "The system time is $h:$m:$s", "<br>"; print "In words $h hours $m minutes $s seconds";

**Output**



**Program 8. Write a Perl program to insert name and age information entered by the user into a table created using MySQL and to display the current contents of this table.**

**File Name = databasepl.html**

<html>

<head><title>Program 8</title>

</head>

<body>

<form action="http://localhost/cgi-bin/database1.cgi">

USN:

<input type="text" name="n1"/>

Name:

<input type="text" name="a1"/>

<input type="submit" value="submit"/>

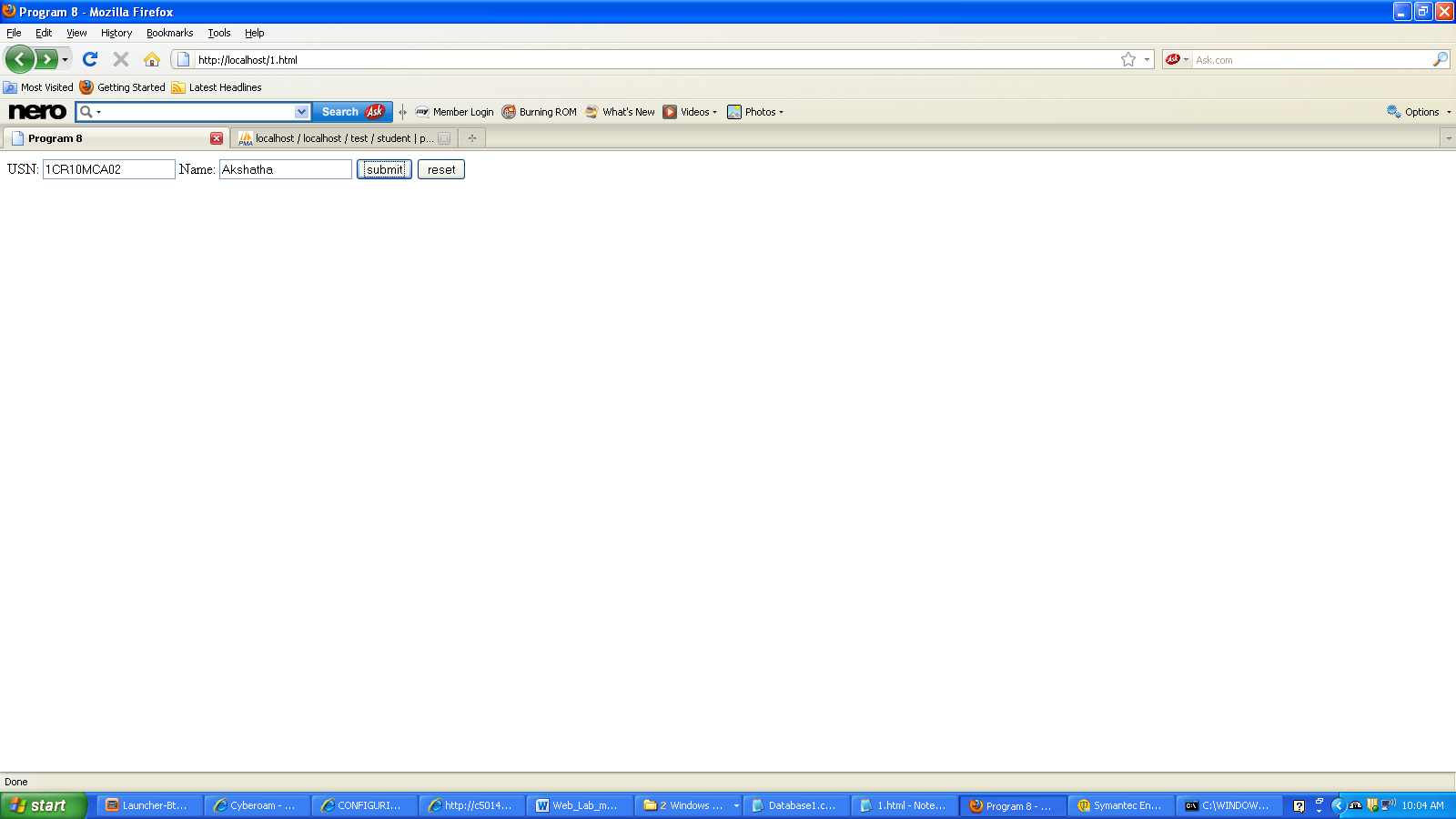
<input type="reset" value="reset"/>

</form>

</body>

</html>

**Output**



**File Name = Database1.cgi**

#!C:\wamp\bin\perl\bin\perl.exe

use CGI ':standard';

use DBI

print "Content-type:text/html\n\n";

$usn=param('n1');

$name=param('a1');

$con=DBI->connect("DBI:mysql:test");

$q=$con->prepare("INSERT INTO student(usn,name) VALUES('$usn','$name')");

$q->execute;

print "Successfully Inserted!";

print "<br/>Data in DB<br/><br/>";

$q=$con->prepare("SELECT \* FROM student");

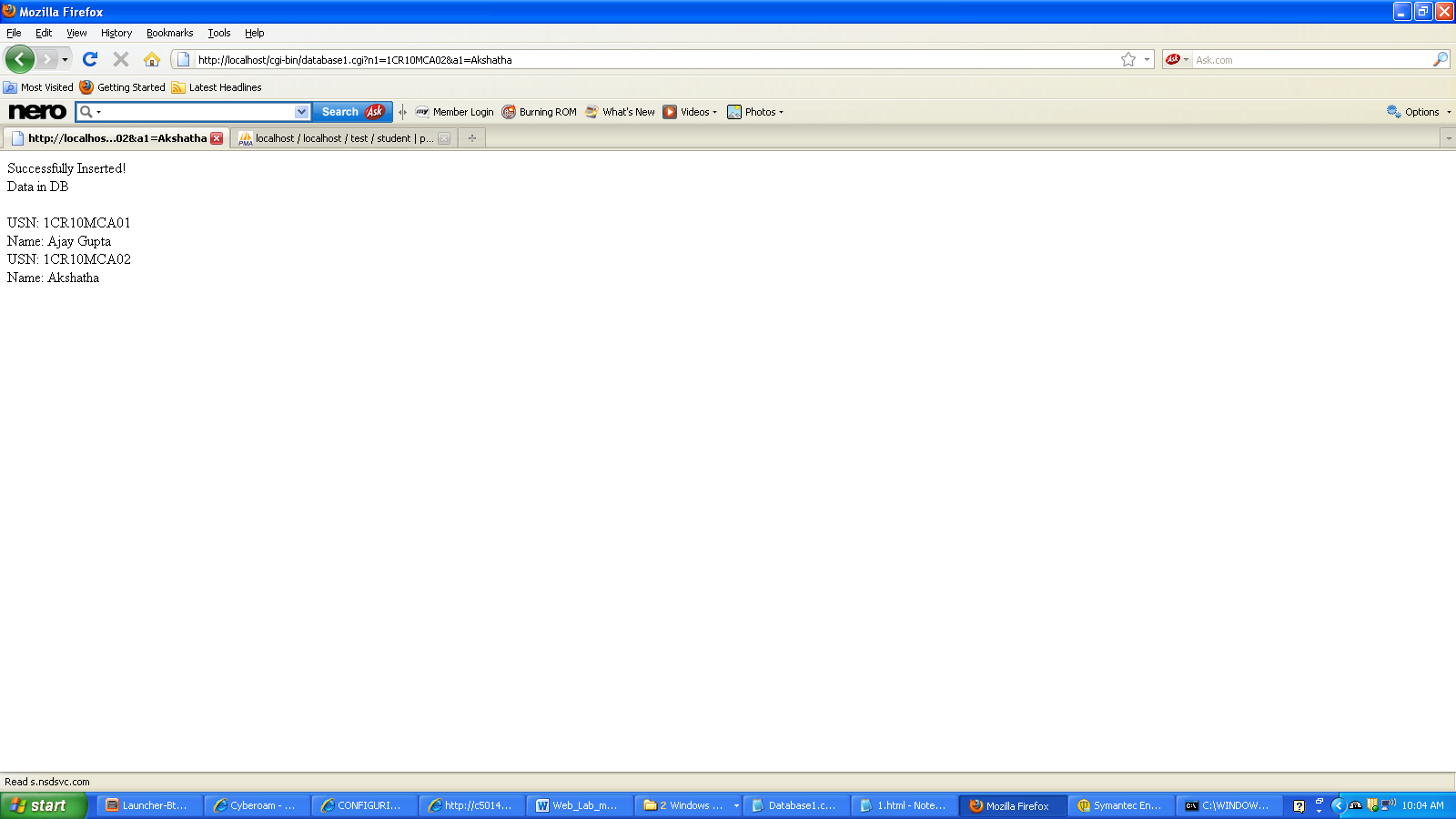
$q->execute();

while(($usn,$name)=$q->fetchrow()){

print "USN: $usn<br/>Name: $name<br/>";

}

**Output:**



**Program 9. Write a PHP program to store current date-time in a COOKIE and display the Last visited on’ date-time on the web page upon reopening of the same page**

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head> <title>Cookies</title> </head>

<body>

<form action="Cookies.php" method="post">

<p>

The last visited time was <input type="submit" name="Display Now"/>

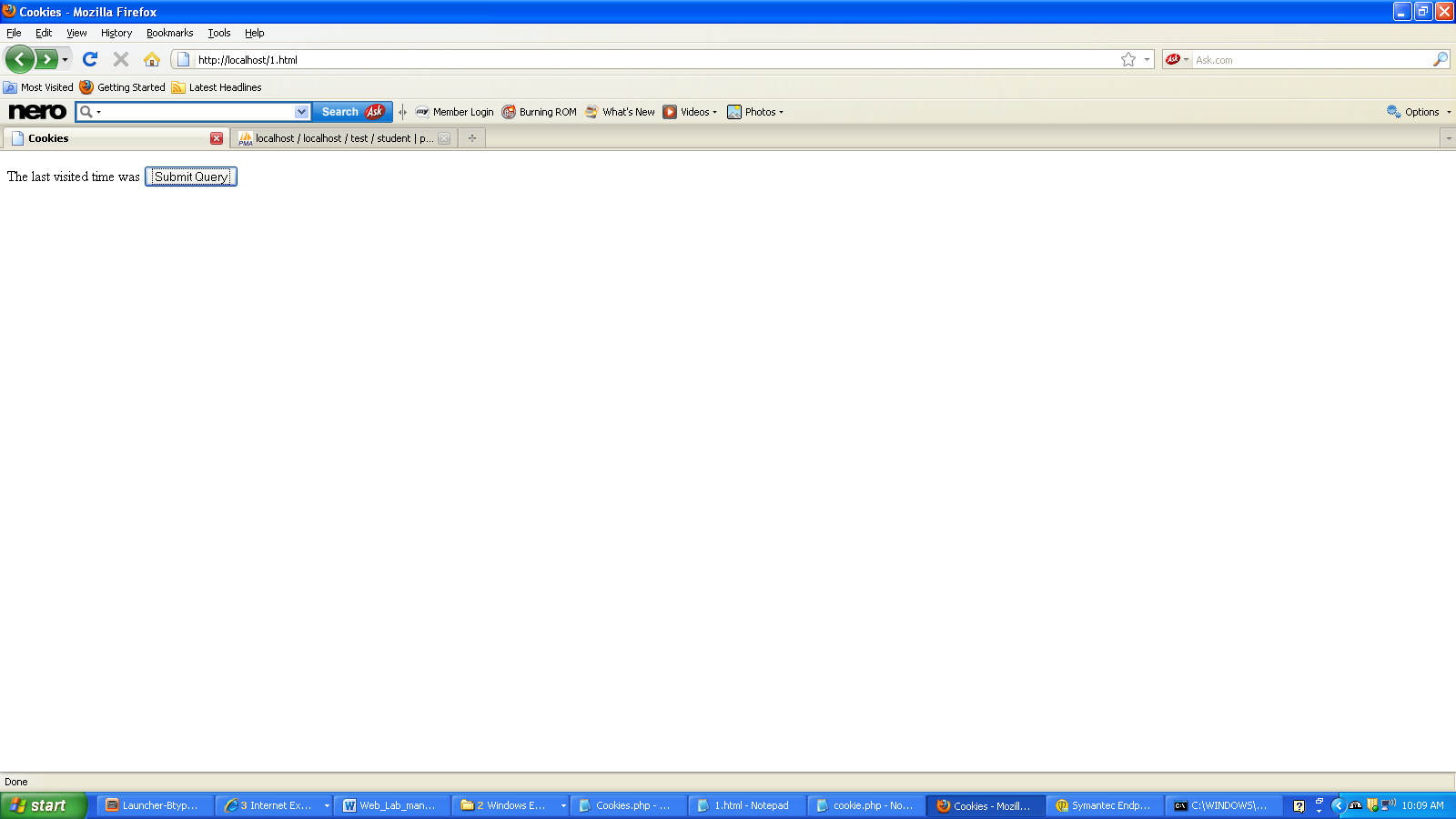
</p>

</form>

</body>

</html>

**Output**



<?php

session\_start();

$date=date('d-m-Y h:i:s');

setcookie('count',$date);

echo "Last Visit: ".$\_COOKIE['count'];

if(isset($\_SESSION['visit'])){

$\_SESSION['visit']++;

echo "<br/>No. of Visits: ".$\_SESSION['visit'];

}

else{

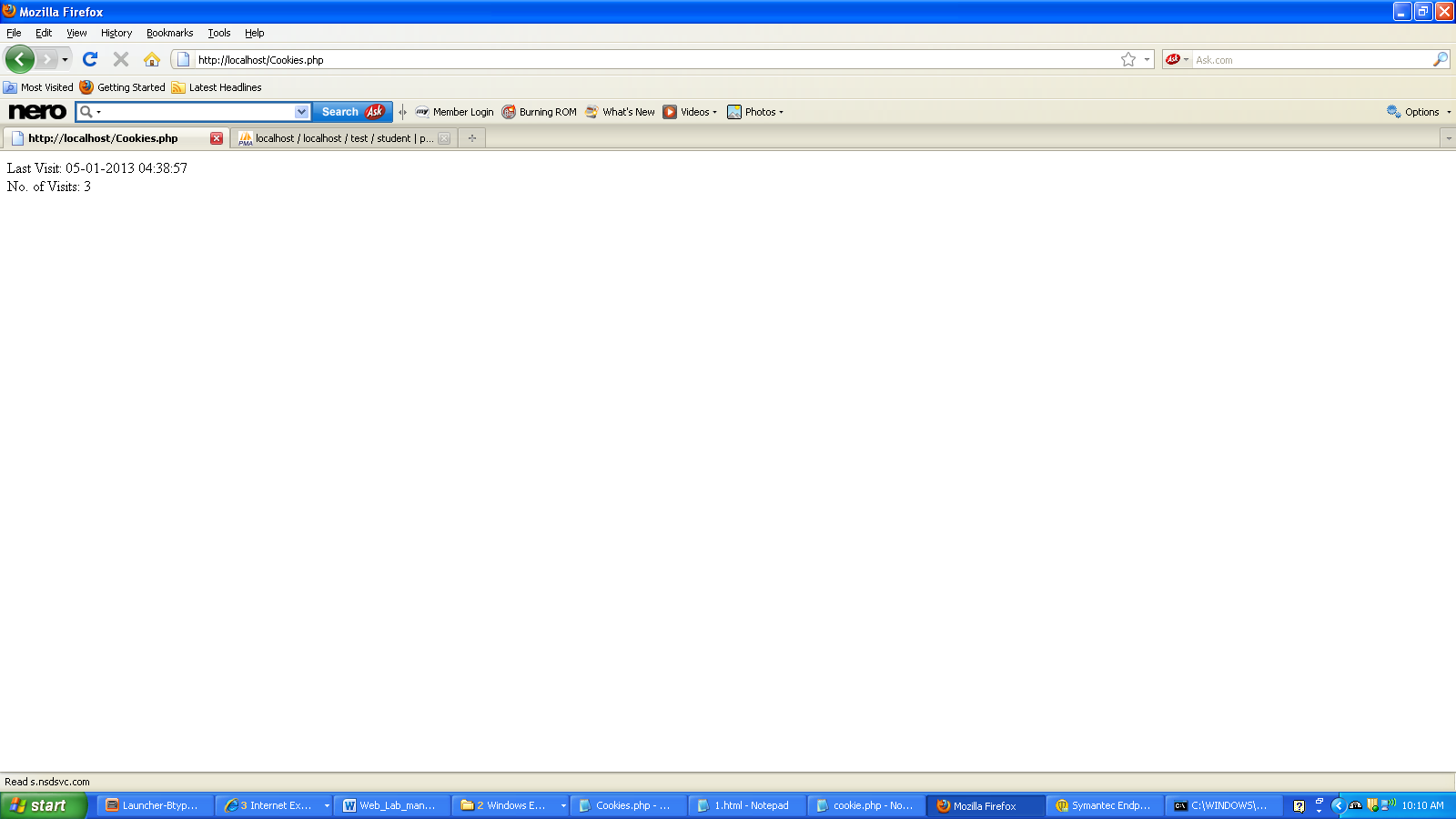
$c=0;

$\_SESSION['visit']=$c;

}

?>

**Output**



**Program 10. Write a PHP program to read student data from an XML file and store into mysql database ,retrieve and display.**

**File Name = 10.xml**

<?xml version="1.0" encoding="ISO-8859-1"?>

<student\_info>

<student>

<usn>1CR10MCA01</usn>

<name>Ajay</name>

</student>

<student>

<usn>1CR10MCA02</usn>

<name>Akshatha</name>

</student>

<student>

<usn>1CR10MCA58</usn>

<name>Piyush</name>

</student>

<student>

<usn>1CR10MCA59</usn>

<name>Taj</name>

</student>

</student\_info>

**File Name = 10.php**

<?php

$con = mysql\_connect("localhost","root","");

if (!$con)

{

die('Could not connect: ' . mysql\_error());

}

mysql\_select\_db("my\_db1", $con);

$lib = simplexml\_load\_file("10.xml");

//if data is already existing then delete

$i="delete from stu";

$result=mysql\_query($i);

foreach($lib as $stu){

$usn= $stu->usn;

$name=$stu->name;

$i="insert into stu values('$usn','$name')";

mysql\_query($i);

}

$result = mysql\_query("SELECT \* from stu");

//Use the mysql\_fetch\_array function on the resource returned by the mysql\_query

$row = mysql\_fetch\_array($result);

//Display the results

echo "<table border='1'><tr><th>USN</th><th>Name</th></tr>";

while($row = mysql\_fetch\_array($result)){

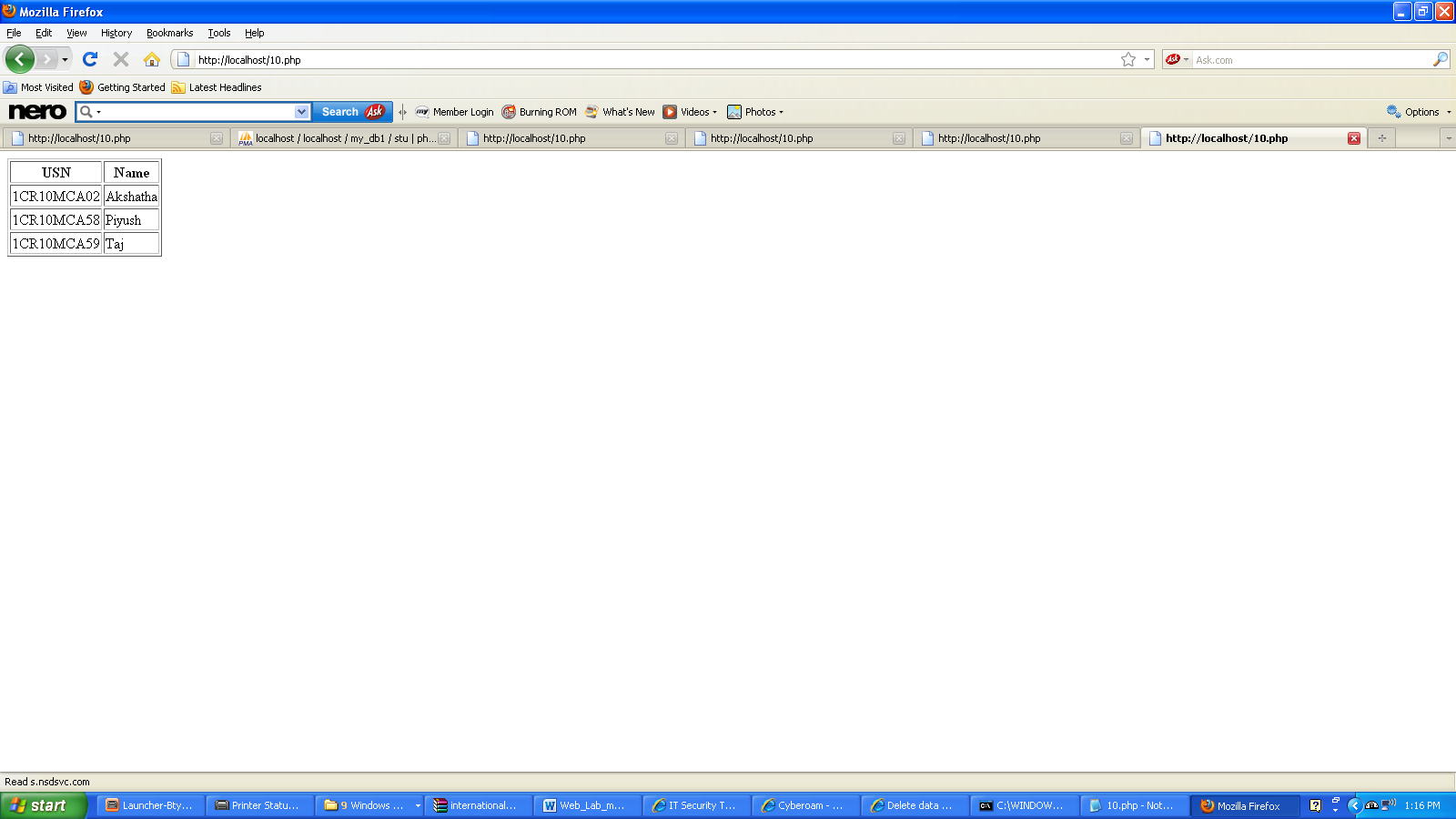
echo "<tr><td>" . $row['USN'] . "</td><td>" . $row['Name'] . "</td></tr>";

}

echo "</table>";

?>

**Output**



**Program 11. Create a XHTML form with Name, Address Line 1, Address Line 2, and E-mail text fields. On submitting, store the values in MySQL table. Retrieve and display the data based on Name.**

**File Name = Prog11.html**

<html>

<head>

<title>New document</title>

</head>

<body bgcolor="cyan">

<form method="post" action="http://localhost/php11.php"><br><fonr color="blue">

<h1><center>Personal Infomation</h1>

<table align=center>

<tr>

<td>Enter Name</td>

<td><input type="text" name="name"></td>

</tr>

<tr><td>Enter Address1</td>

<td><input type="text" name="add1"></td></tr>

<tr><td>Enter Address2</td><td><input type="text" name="add2"></td></tr>

<tr></tr>

<tr></tr>

<tr></tr>

<tr></tr>

</table>

<br>

<br>

<input type="reset" value="CLEAR">

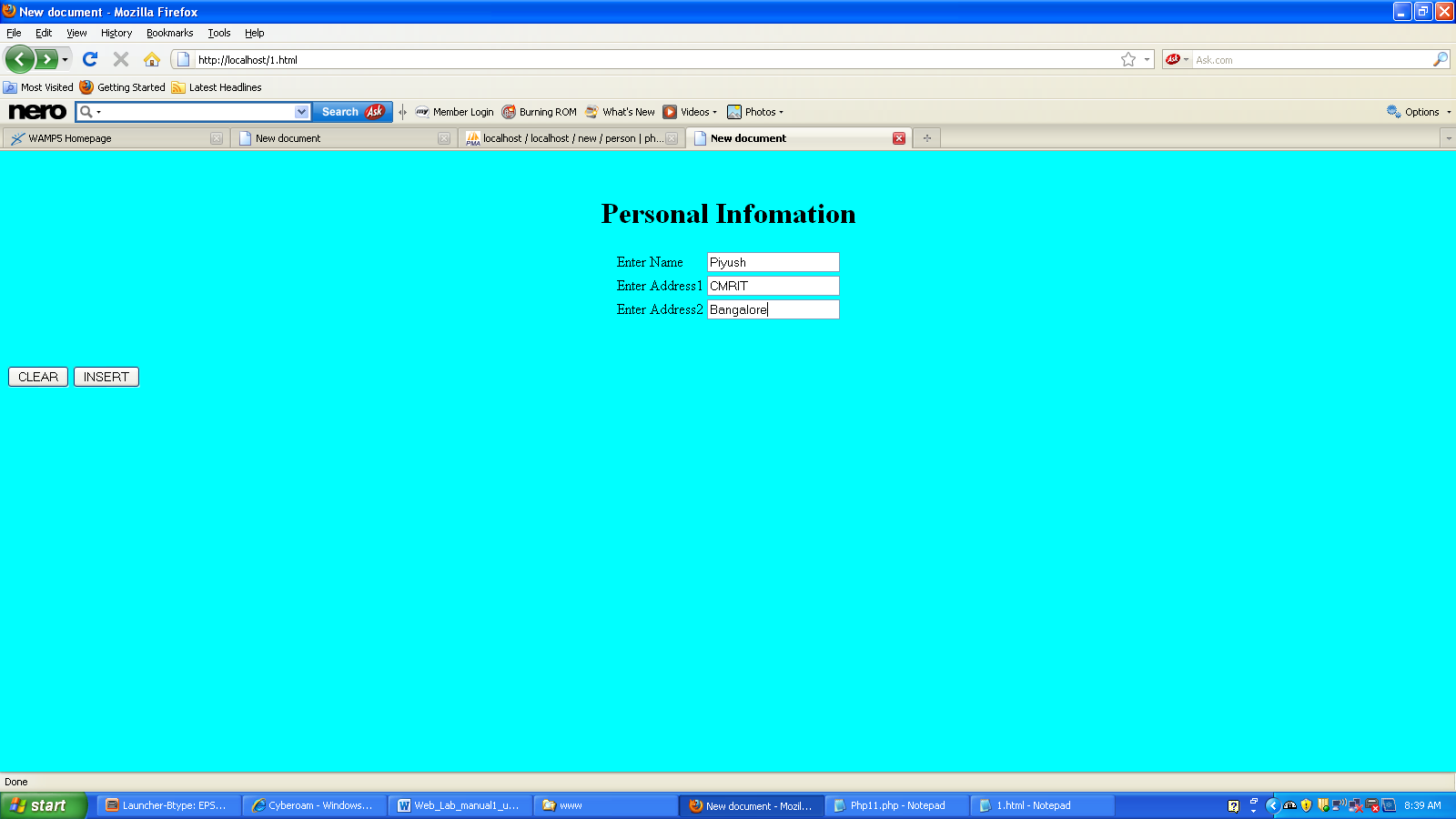
<input type="submit" value="INSERT">

</form>

</body>

</html>

**Output**



**File name = Php11.php**

<?php

mysql\_connect("localhost","root","");

mysql\_select\_db("new");

$name=$\_POST['name'];

$add1=$\_POST['add1'];

$add2=$\_POST['add2'];

$result=mysql\_query("INSERT INTO person(name,add1,add2) VALUES('$name','$add1','$add2')");

$result = mysql\_query("SELECT \* from person");

//Use the mysql\_fetch\_array function on the resource returned by the mysql\_query

$row = mysql\_fetch\_array($result);

//Display the results

echo "<table border='1'><tr><th>Name</th><th>Address1</th><th>Address2</th></tr>";

while($row = mysql\_fetch\_array($result)){

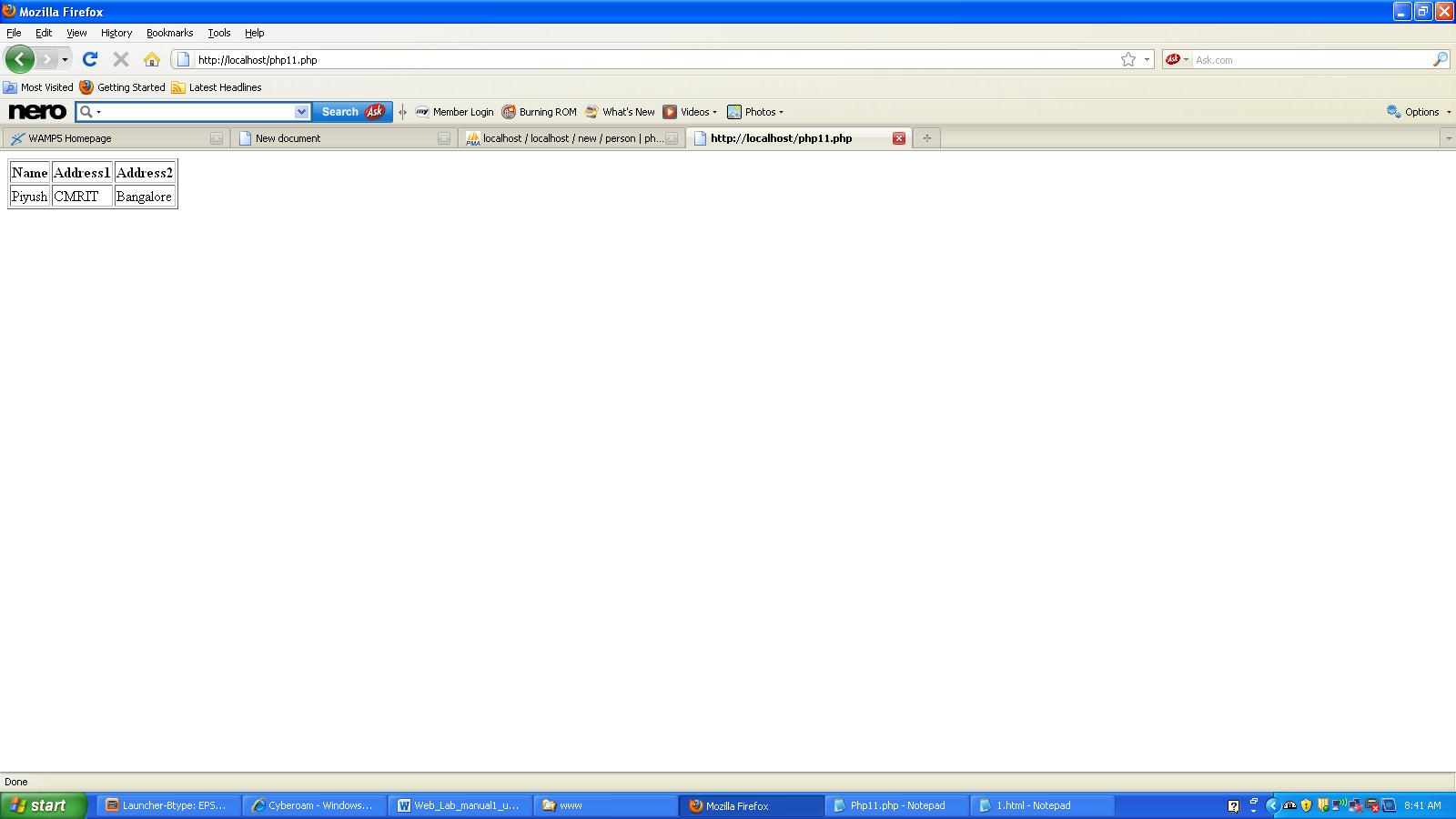
echo "<tr><td>" . $row['name'] . "</td><td>" . $row['add1'] . "</td><td>" . $row['add2'] . "</td></tr>";

}

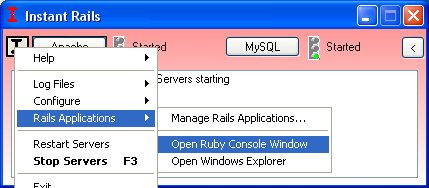
echo "</table>";

?>

**Output**



**Program 12 Build a Rails application to accept book information viz. Accession number, title, authors, edition and publisher from a web page and store the information in a database and to search for a book with the title specified by the user and to display the search results with proper headings.**

1. Starting InstantRails will start the web server and database and take you to the console.  
     
   Open a Rails command prompt by clicking the "I" menu in the upper left corner and then selecting Rails Applications-Open Ruby Console Window.  
     
   The Rails command prompt will automatically set up the PATH to use ruby and rails and place you in the rails\_apps directory where your new application will be created.
2. Create an application by name studbook

C:\rails\rails\_apps>**rails studbook**

create

create app/controllers

create app/helpers

create app/models

create app/views/layouts

create config/environments

create config/initializers

create db

create doc

create lib

create lib/tasks

create log

create public/images

create public/javascripts

create public/stylesheets

create script/performance

create script/process

create test/fixtures

create test/functional

create test/integration

create test/mocks/development

create test/mocks/test

create test/unit

create vendor

create vendor/plugins

create tmp/sessions

create tmp/sockets

create tmp/cache

create tmp/pids

create Rakefile

create README

create app/controllers/application.rb

create app/helpers/application\_helper.rb

create test/test\_helper.rb

create config/database.yml

create config/routes.rb

create public/.htaccess

create config/initializers/inflections.rb

create config/initializers/mime\_types.rb

create config/boot.rb

create config/environment.rb

create config/environments/production.rb

create config/environments/development.rb

create config/environments/test.rb

create script/about

create script/console

create script/destroy

create script/generate

create script/performance/benchmarker

create script/performance/profiler

create script/performance/request

create script/process/reaper

create script/process/spawner

create script/process/inspector

create script/runner

create script/server

create script/plugin

create public/dispatch.rb

create public/dispatch.cgi

create public/dispatch.fcgi

create public/404.html

create public/422.html

create public/500.html

create public/index.html

create public/favicon.ico

create public/robots.txt

create public/images/rails.png

create public/javascripts/prototype.js

create public/javascripts/effects.js

create public/javascripts/dragdrop.js

create public/javascripts/controls.js

create public/javascripts/application.js

create doc/README\_FOR\_APP

create log/server.log

create log/production.log

create log/development.log

create log/test.log

Rails generates an entire framework for the application. Change into the newly created studbook directory and get to work.

C:\rails\rails\_apps>**cd studbook**

C:\rails\rails\_apps\studbook>**dir**

11/20/2008 01:39 PM <DIR> .

11/20/2008 01:39 PM <DIR> ..

11/20/2008 01:39 PM <DIR> app

11/20/2008 01:40 PM <DIR> config

11/20/2008 01:39 PM <DIR> db

11/20/2008 01:39 PM <DIR> doc

11/20/2008 01:39 PM <DIR> lib

11/20/2008 01:40 PM <DIR> log

11/20/2008 01:40 PM <DIR> public

11/20/2008 01:39 PM 307 Rakefile

11/20/2008 01:39 PM 8,819 README

11/20/2008 01:40 PM <DIR> script

11/20/2008 01:39 PM <DIR> test

11/20/2008 01:39 PM <DIR> tmp

11/20/2008 01:39 PM <DIR> vendor

You should have many subdirectories that have been created in support of our application.

## Setting up the Model and Database Table

At this point many web frameworks would have to use database commands and DDL's to create the table we need to hold our book inventory data, but thanks to Rails tight coupling between the data and the application we can use Rails to create and manage the tables our project will need. In the Model-View-Controller pattern of application design it's the model that regulates access to the data.

Rails can create the database and tables needed for the studbook project. Look at the file studbook\config\database.yml, in it you can see the structure of the databases used in Rails:  
development:

adapter: mysql

database: studbook\_development

username: root

password:

host: localhost

test:

adapter: mysql

database: studbook\_test

username: root

password:

host: localhost

production:

adapter: mysql

database: studbook\_production

username: root

password:

host: localhost

You can see that there are separate tables for development testing and production. This separation helps in the development and maintenance of Rails projects.

In a difference from earlier Rails versions, Rails 2.0 will create the databases needed with the command:

C:\rails\rails\_apps\studbook>**rake db:create:all**

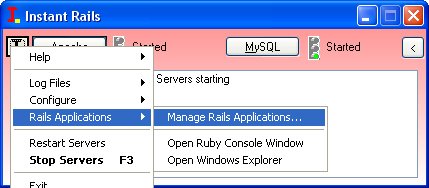
(in C:/rails/rails\_apps/studbook)

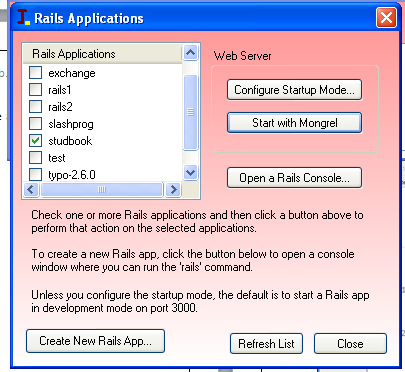
"db/development.sqlite3 already exists"

"db/production.sqlite3 already exists"

"db/test.sqlite3 already exists"

## Starting the application server

ActiveRails includes its own application server, so let's fire it up and see if we have everything working so far. To start the rails appserver, mongrel, on the InstantRails console, click the "I" button and select Rails Applications-Manage Rails Applications.  
  
You will probably have three Rails Applications. InstantRails includes the two sample applications, cookbook and typo-2.6.0. We want to start our studbook application, so check its box and Start with Mongrel.

  
  
Another window should appear and you should see something like the following.

\*\* Starting Mongrel listening at 0.0.0.0:3000

\*\* Starting Rails with development environment...

\*\* Rails loaded.

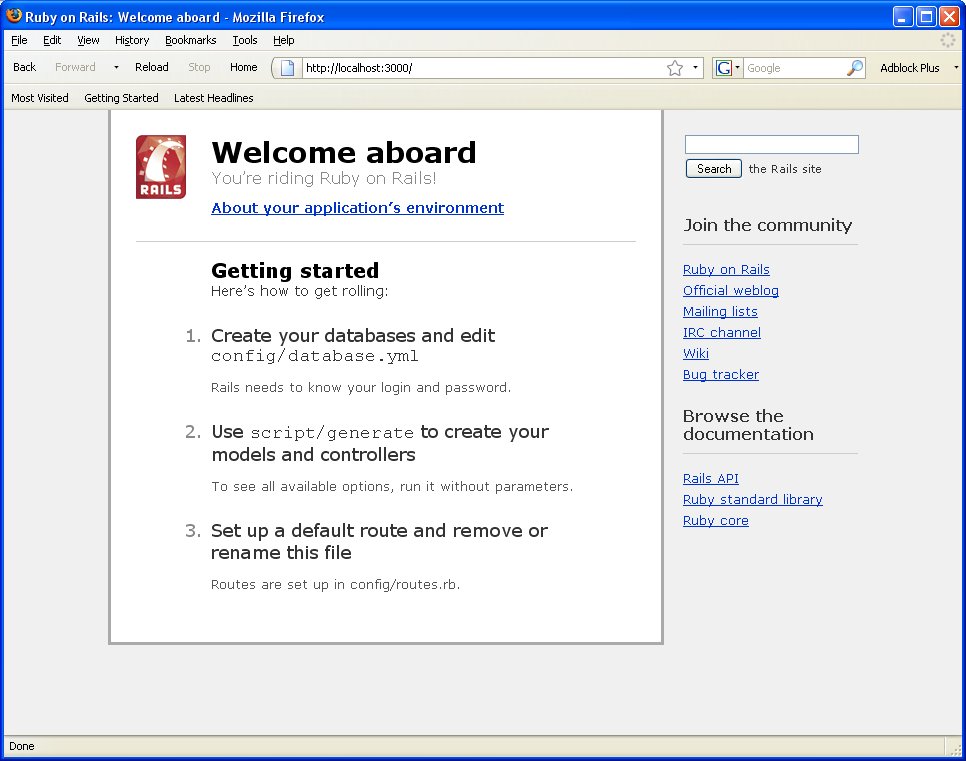
\*\* Loading any Rails specific GemPlugins

\*\* Signals ready. INT => stop (no restart).

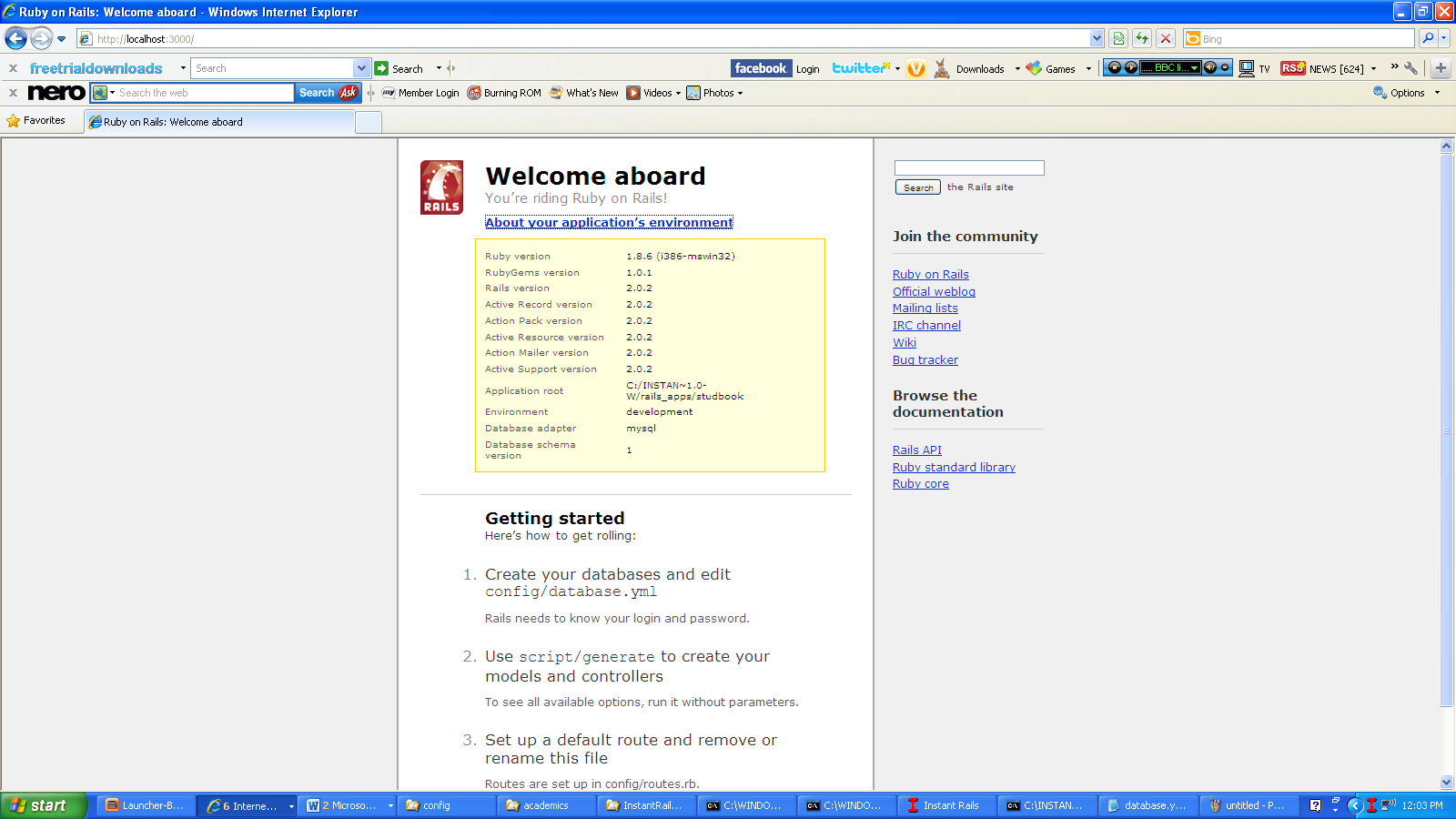
\*\* Mongrel 1.1.2 available at 0.0.0.0:3000

\*\* Use CTRL-C to stop.

This means that your application is now active.

Open your favorite browser and point it torwards the URL <http://localhost:3000>  
You should see something like:  
[](http://plaza.obu.edu/matochaj/ruby/rails/part1/rails-root_page.jpg)

Clicking on the "About your application's environment" link will activate a little piece of AJAX code that lists the particulars of your rails application.

  
Notice that the default environment is development and not testing or production. This is just what we want during our development phase!

## Old vs. New

The next step is to create a model whose job will be to manage the data stored in the database.

The following command will generate the model, plus scaffolding, and the database migration script needed as well as a controller, helper, and testing support files:

C:\rails\rails\_apps\studbook>**ruby script/generate scaffold Book Acc\_No:integer Title:string Authors:string Edition:integer Publisher:string**

exists app/models/

exists app/controllers/

exists app/helpers/

create app/views/books

exists app/views/layouts/

exists test/functional/

exists test/unit/

create app/views/books/index.html.erb

create app/views/books/show.html.erb

create app/views/books/new.html.erb

create app/views/books/edit.html.erb

create app/views/layouts/books.html.erb

create public/stylesheets/scaffold.css

dependency model

exists app/models/

exists test/unit/

exists test/fixtures/

create app/models/book.rb

create test/unit/book\_test.rb

create test/fixtures/books.yml

create db/migrate

create db/migrate/001\_create\_books.rb

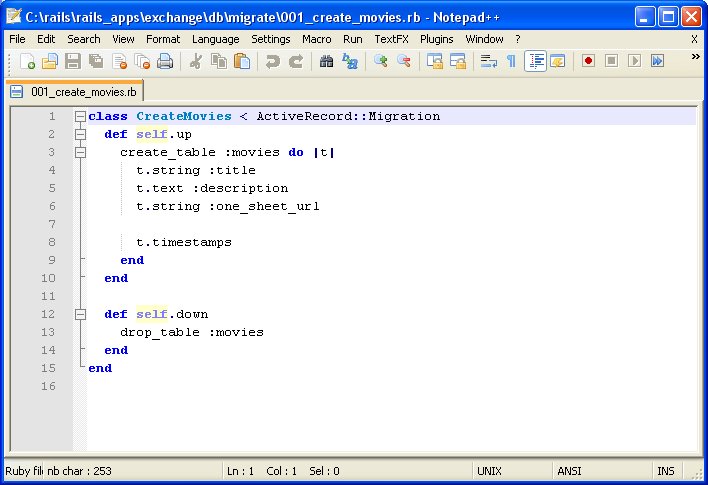
create app/controllers/books\_controller.rb

create test/functional/books\_controller\_test.rb

create app/helpers/books\_helper.rb

route map.resources :books

## Making Books

The table will get created by the file in db\migrate\001\_create\_books.rb. Let's look at the file now:  
[](http://plaza.obu.edu/matochaj/ruby/rails/part1/migration1.jpg)  
This file will create a table called books that will be tied to the model Book. This is a Rails naming convention. A table people would match a model Person. A table cars would match a model Car. You can also see how the parameters we fed the script/generate command show up as table columns and types in this migration file.

Apply this migration to actually create the table with the command:

C:\rails\rails\_apps\studbook>**rake db:migrate**

(in C:/rails/rails\_apps/studbook)

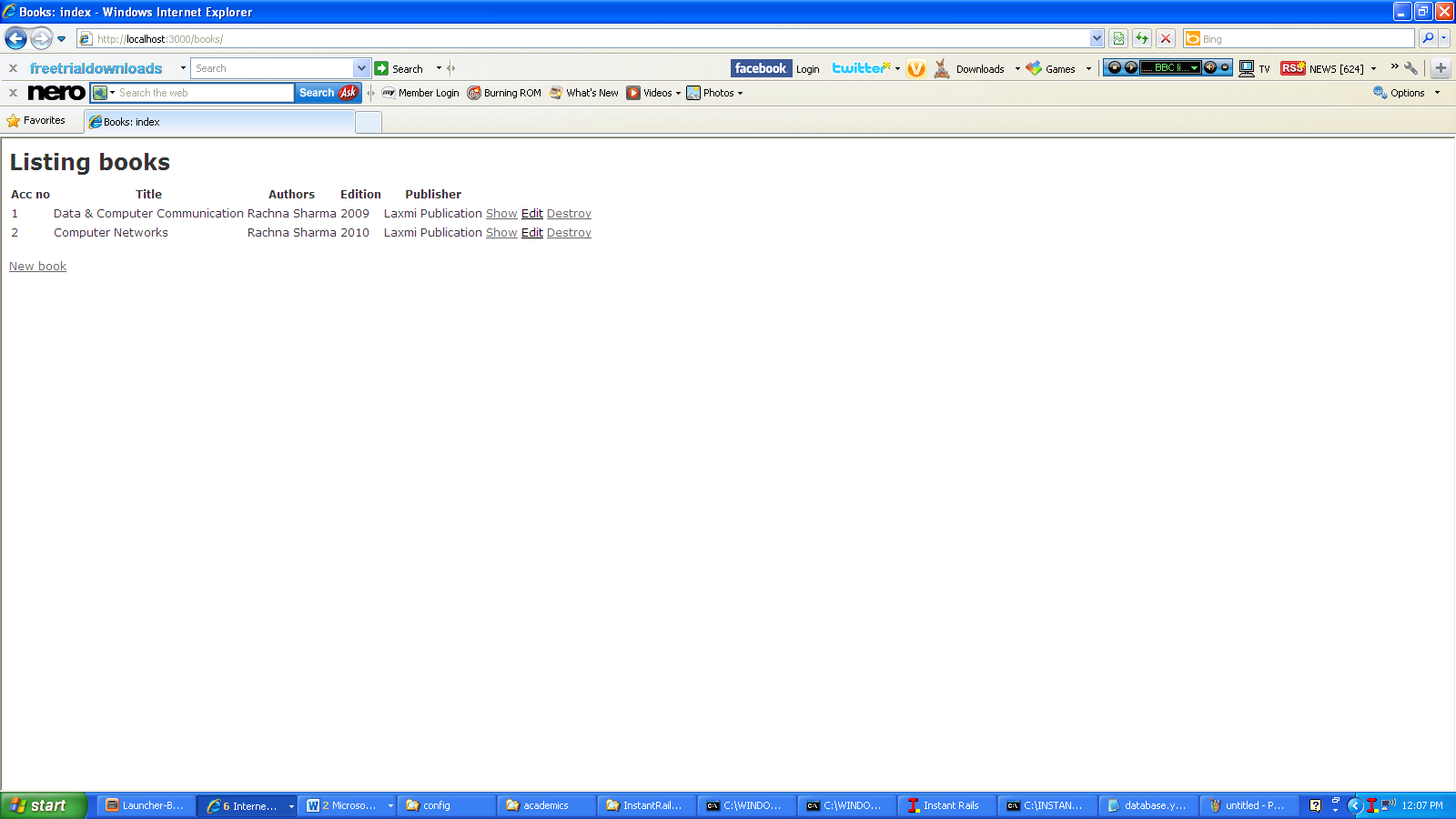
== 1 CreateBooks: migrating ==================================================

-- create\_table(:books)

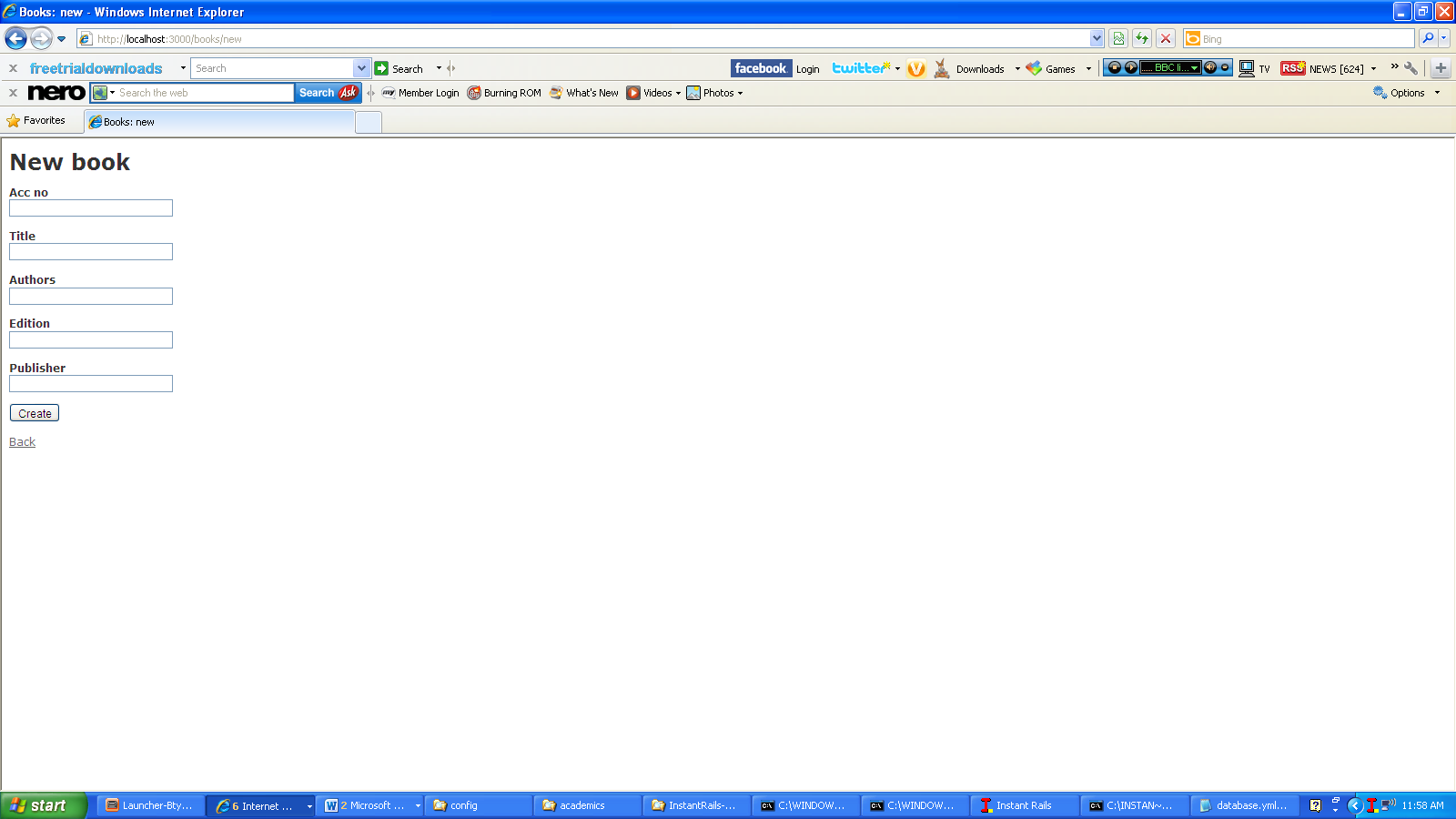
-> 0.0780s

== 1 CreateBooks: migrated (0.0780s) =========================================

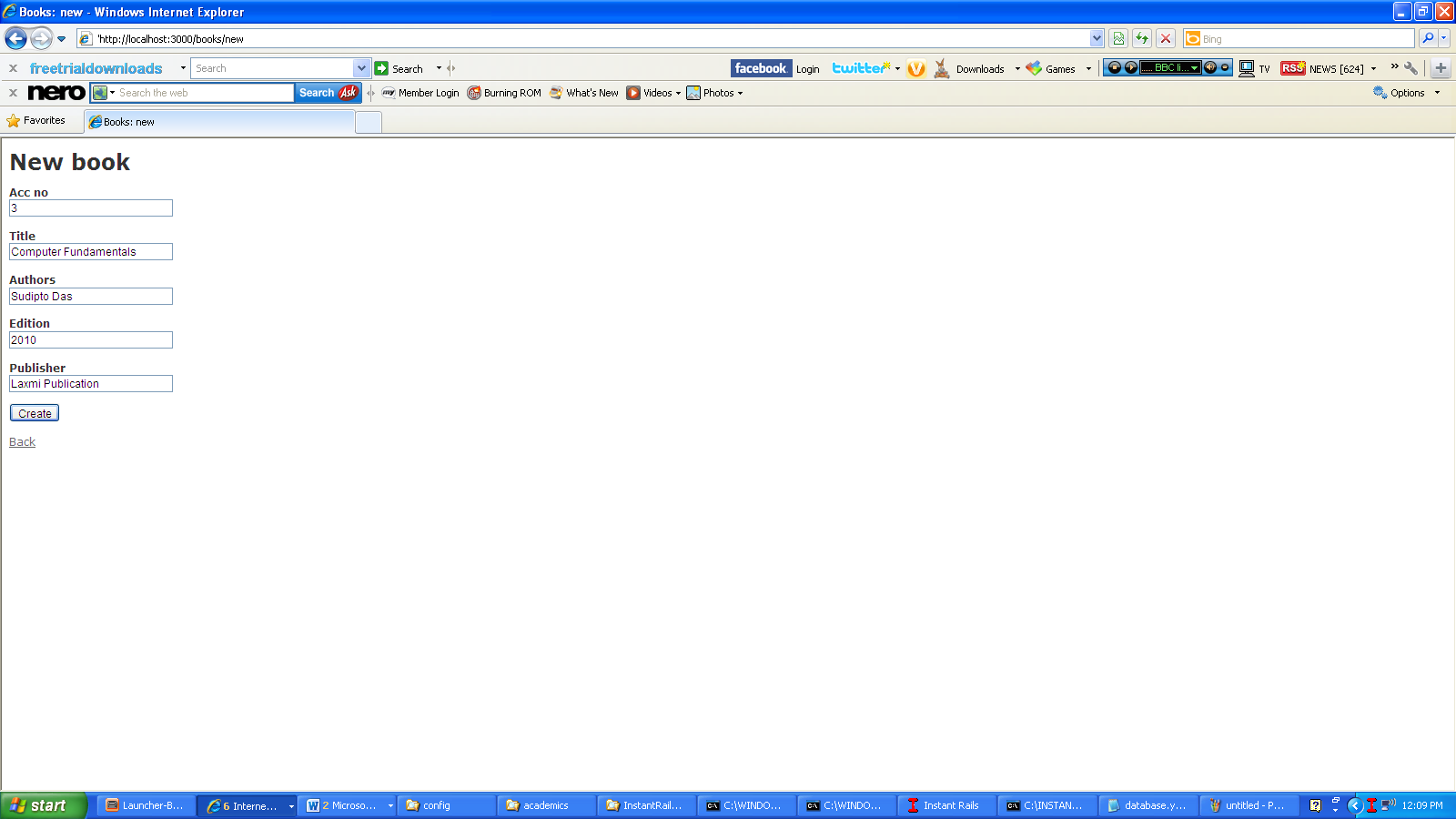
To see what our work so far has produced, point your web browse to the URL [http://localhost:3000/books](http://localhost:3000/movies) and look at what we have created. (If you killed the Mongrel server with Control-C, restart the server from the InstantRails console by clicking the "I" button, selecting Rails Applications-Manage Rails Applications, and starting the Mongrel server for your application.)



It looks pretty bare, but we don't have any inventory yet. Click on the 'New Book' link to start adding some books to the inventory.



Add the book information and click Create.



Click the button labeled 'Back' (not the browser back arrow) to return to the main listing and add another book.

