

Saya WebServer Mini-project report

Introduction:

The Saya WebServer mini-project is a multipurpose one. One use of it is when a lecturer (of the cs faculty) is at the reception desk and interested in knowing if some other lecturer left a (voice) message, or to ask for a room number of some lecturer , or maybe the lecturer himself wants to leave a message to other lecturers.

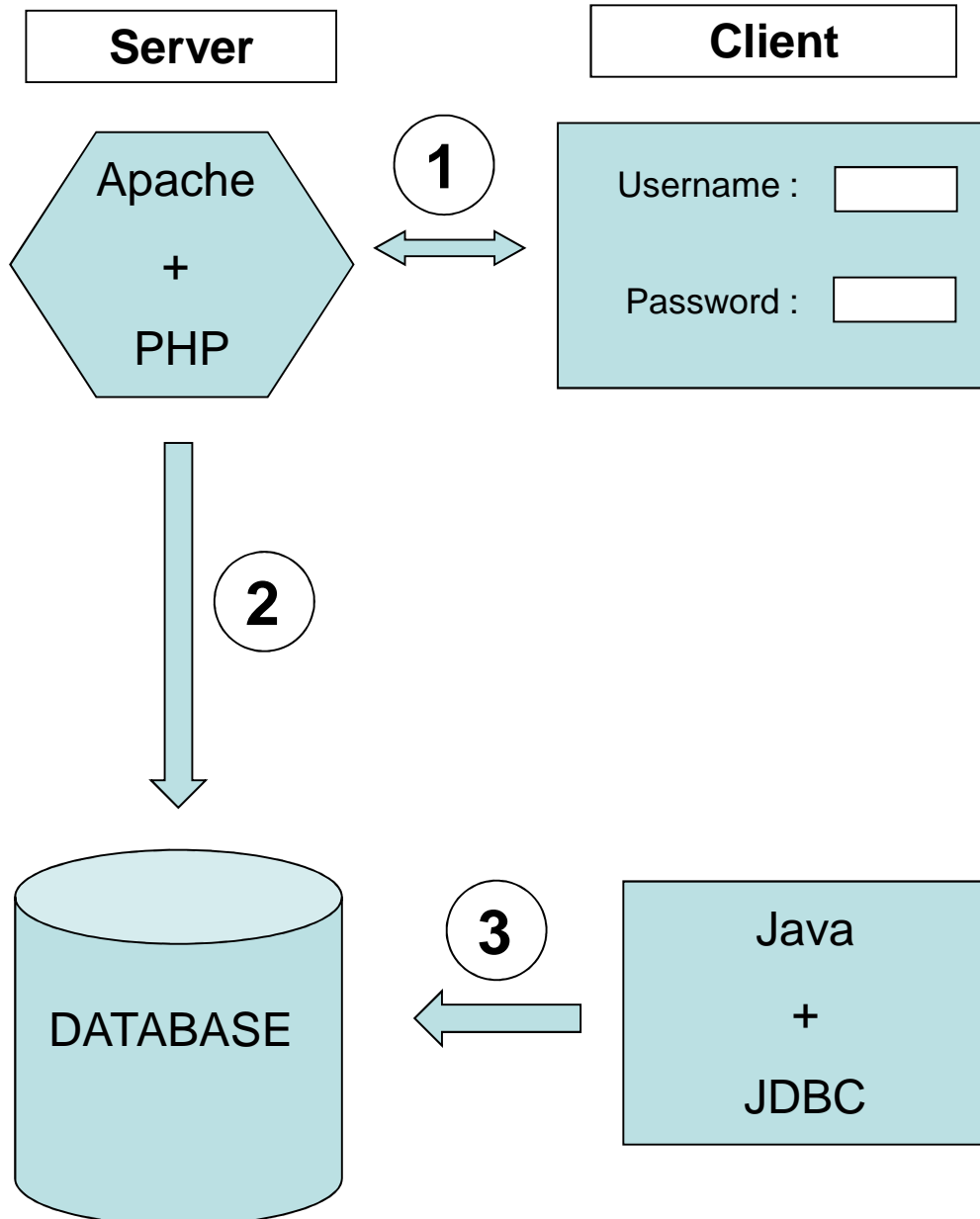
Implemented in Java.

Another use for it can be in case a lecturer is at his office/home and willing to communicate with Saya Database so as to elicit info about himself as listed in the DB archives.

We would want to grant the lecturer the option to make changes to his entry in the DB, such as adding a comment, changing room number ,etc...

Implemented in php.

Webserver Architecture



Description in words

1. Information flow between Client to Server. Client side sending requests and authentication details and Server side returning query results.
2. Information flow between Server to Database. Server makes a connection to Database and issues queries.
- 3 . Information flow between Java and JDBC to Database.
Java uses methods of its JDBC API for querying and updating data in the database.

Implementation:

- Making a connection to Saya Mysql DB:

We use `mysql_connect("webdev.cs.bgu.ac.il","sayaweb2","*****")` in PHP in order to connect to the Mysql DB.

- Migrating Oracle DB -> Mysql DB:

Done only once by running an executable "OracleToMysql.exe".

The original .php follows these steps:

First we establish a connection to the FRODO Host, where the oracle DB is:

`file_get_contents("http://frodo.cs.bgu.ac.il/php/connections/db.php?action=get-group)`

Then we take only the members of that DB that we want, i.e. faculty members, by adding "&group=faculty".

Then we make a connection to our Mysql DB - `include "mysqlLink.php";`

Then we create a table on our Mysql Database with the corresponding columns by using the query `"CREATE TABLE lecturers...col1 col2"`.

Then we iterate over the array we got from Oracle and perform:

`"INSERT INTO lecturers (colNames) VALUES (valList)";`

After this last query we have a copy of the original Oracle DB in a new Mysql DB.

- The Webservice:

Technical background:

The WebServer is implemented in PHP + HTML.

The system was tested on Apache Server 2.2.6 + PHP 5.2.4 on Windows XP.

Database was created in MySQL 5.0.45

Step by step Implementation:

Design of pages:

Background color throughout was done by CSS `<style type="text/css">`

Using HTML FORMS for user input. `<FORM action="xxx.php" method="POST">`

Sessions were used to keep data about user such as username. Some relevant code:

```
session_start(); //starting session per current user
$_SESSION["user"] = $_POST['username'] //keeping user input
session_unregister("user"); //logging out
```

Also, sessions were used to prevent user from maliciously trying to access subsequent pages without logging in first:

```
If (!isset($_SESSION["time"]))
{
    header("Location:login.php")
}
Else
{
    //main code
}
```

Regular expressions were used to validate correct user input:

```
preg_match("/^\d{3}$/", $new_room)
```

User can log out at any time.

USER AUTHENTICATION:

User details in Unix-like systems is stored in /etc/passwd file.

We use the command `"ypmatch $user passwd"` to retrieve \$user entry in this file.

Since we are only interested in the encrypted password of the user we have to first break down the retrieved string `:explode(":",$user_details_string)`

And return the correct cell of the resulting array.

Then:

If \$user is not found at all in the directory, we output:

"Sorry, you are not listed as a faculty member" and redirect him to the login page.

If he is listed, we then check if the password entered by user (after applying `crypt()`) and the encrypted password match.

In case of success, the user is granted access to the database and can make queries.

Interacting with Mysql Database:

We use `$_SESSION["user"].'@cs.bgu.ac.il'` to find the corresponding row of the lecturer.

For instance, issuing an update query to mysql that tells it to update the comment field:

```
"UPDATE lecturers SET comment=".$comment.'" WHERE  
email=$_SESSION["user"].'@cs.bgu.ac.il';
```

Java:

First we define a simple interface with basic operations:

```
String get_comment(String name,String family);  
//a lecturer asks saya if there's any message from some other  
//lecturer by the name of name+family  
  
void put_comment(String name,String family, String message);  
//a lecturer want to leave a message  
  
String get_room (String name, String family);  
//a lecturer asks saya for the room number of some other lecturer
```

Implementation:

We use JDBC api to bridge between Java Platform and Mysql Database.
We also need to make use of the Mysql connector/J 5.1 driver to allow the
communication to take place.

Throughout the program we use the java.sql Package.
To establish a connection, we make an instance of the driver:

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

Followed by:

```
DriverManager.getConnection("jdbc:mysql:///test",  
"root", ""); //DB "test" located on 127.0.0.1
```

Then we execute the queries by using createStatement() executeQuery() of
the java.sql package.

Test case

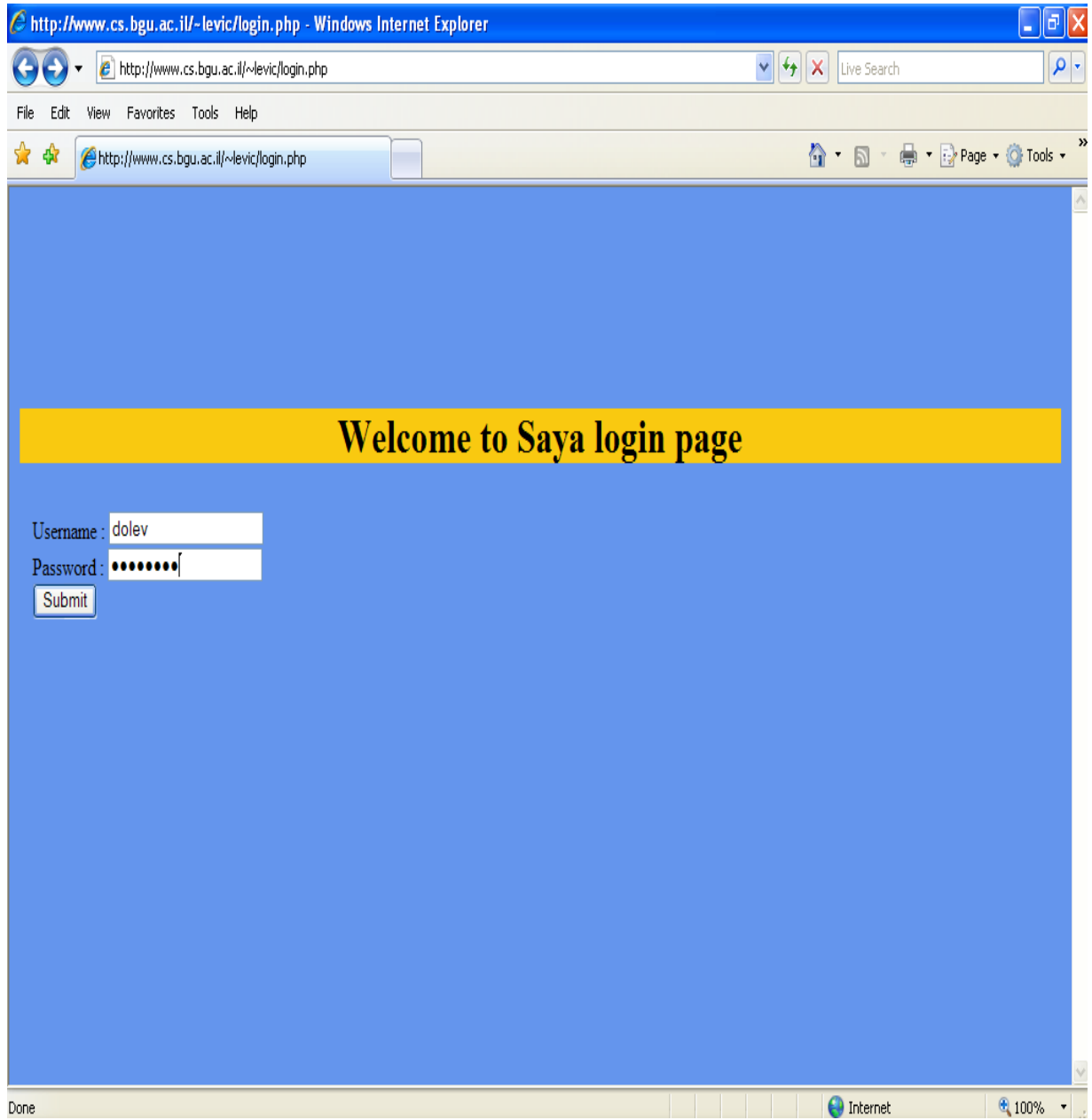


fig. 1 a lecturer enters his username and password to login

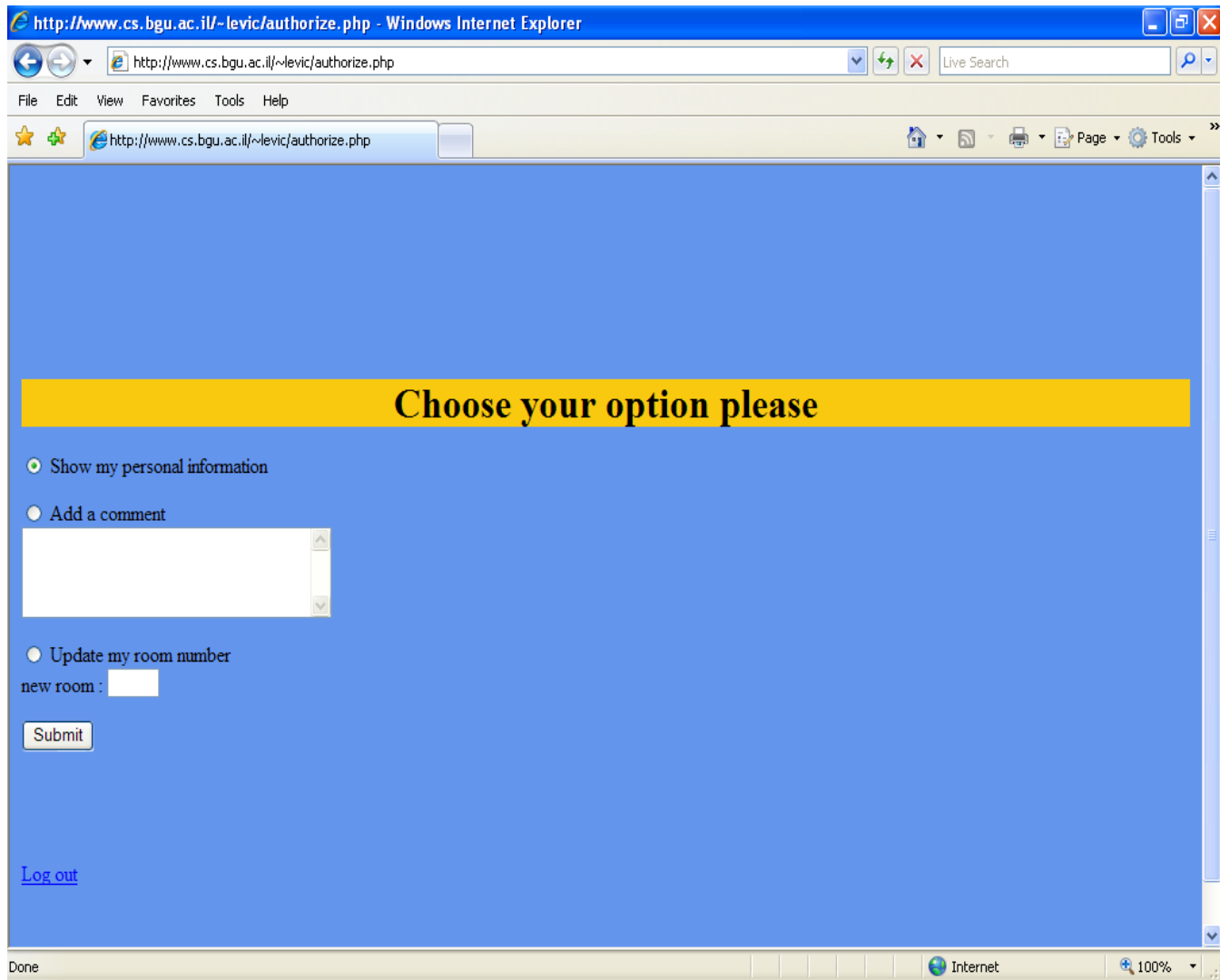


fig. 2_ Query selection page:
a lecturer selects to show his personal details

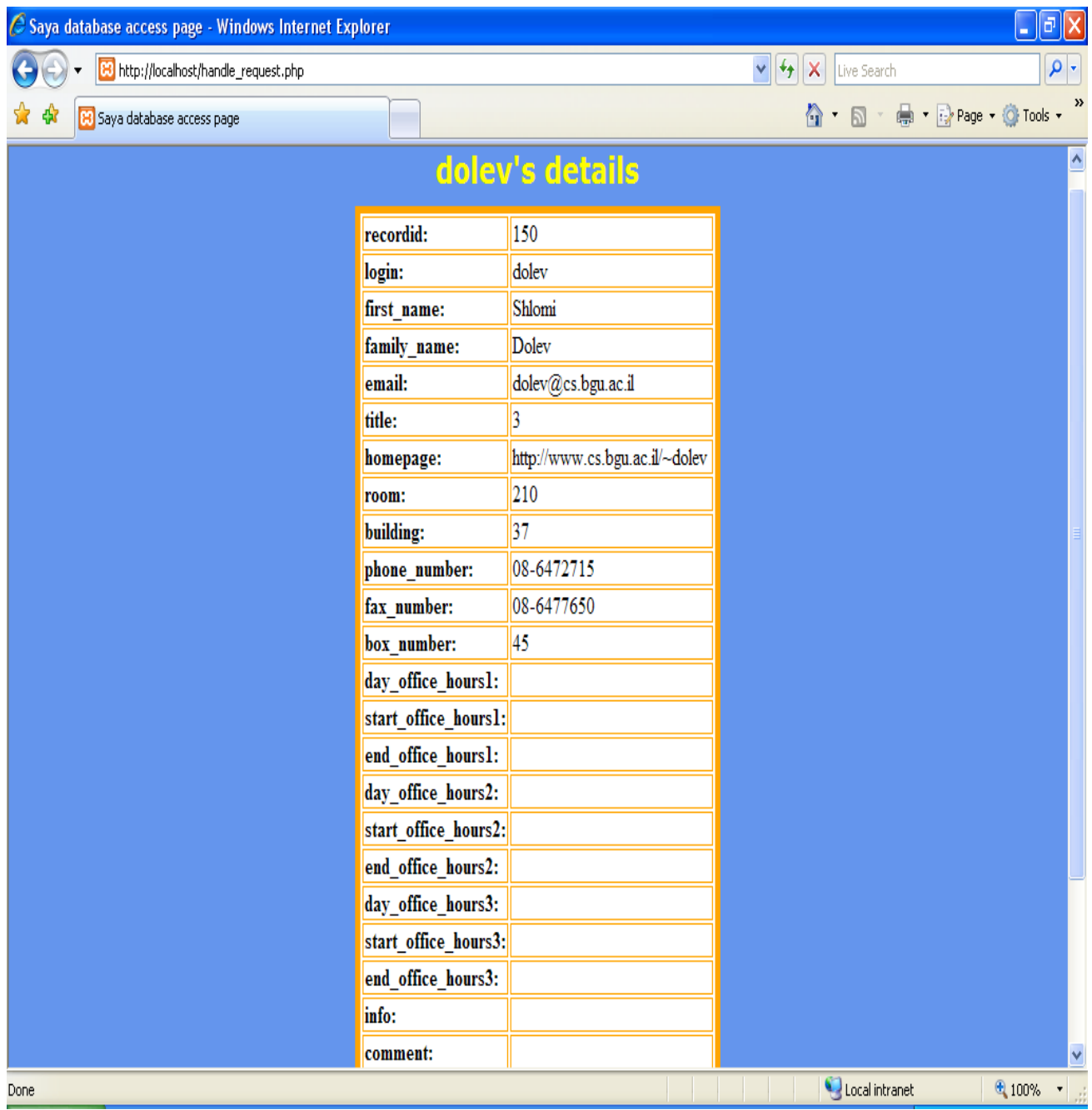


fig. 3_ the lecturer's personal information is retrieved and output to screen

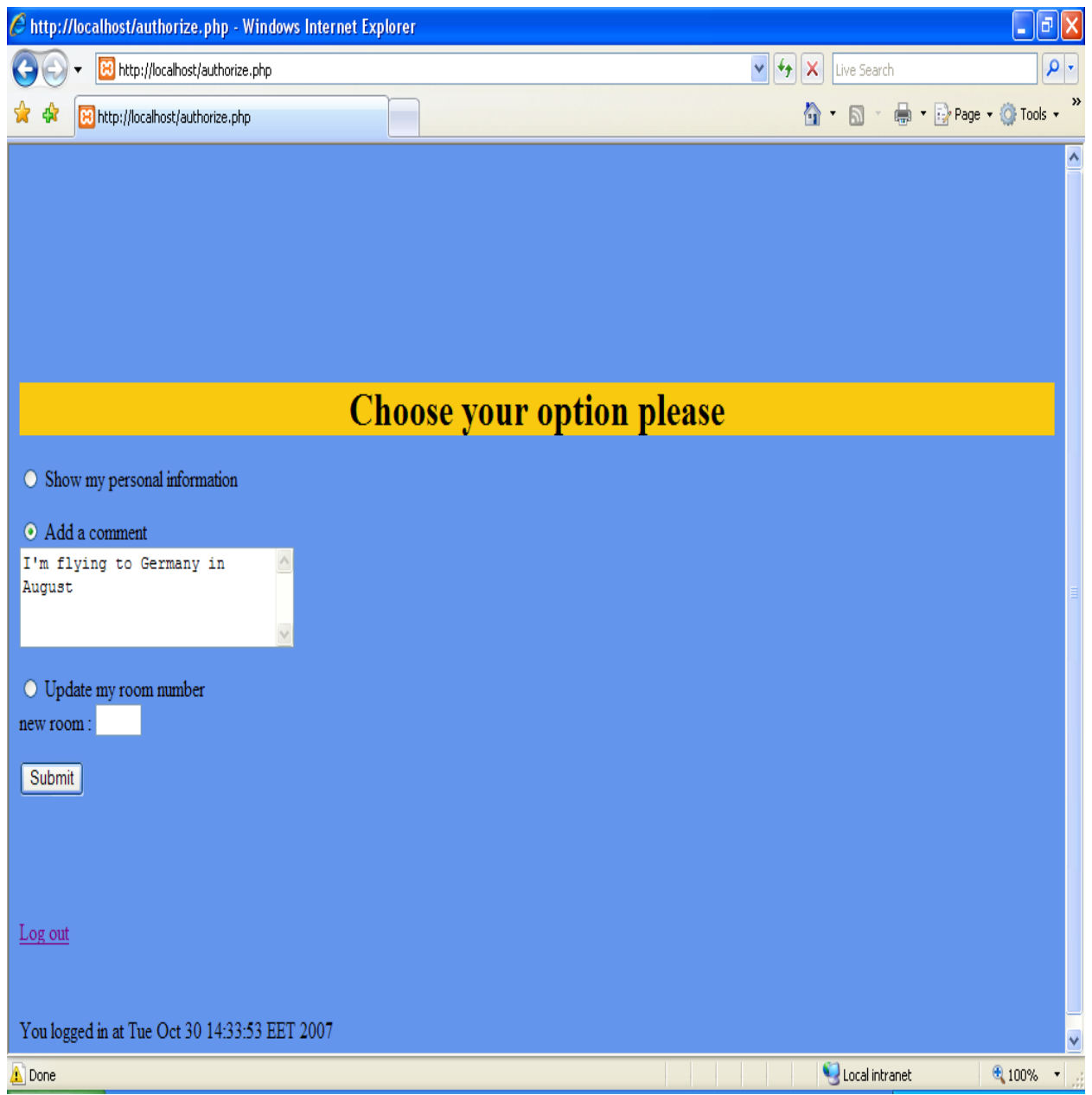


fig. 4 A lecturer asks to insert a comment to the database

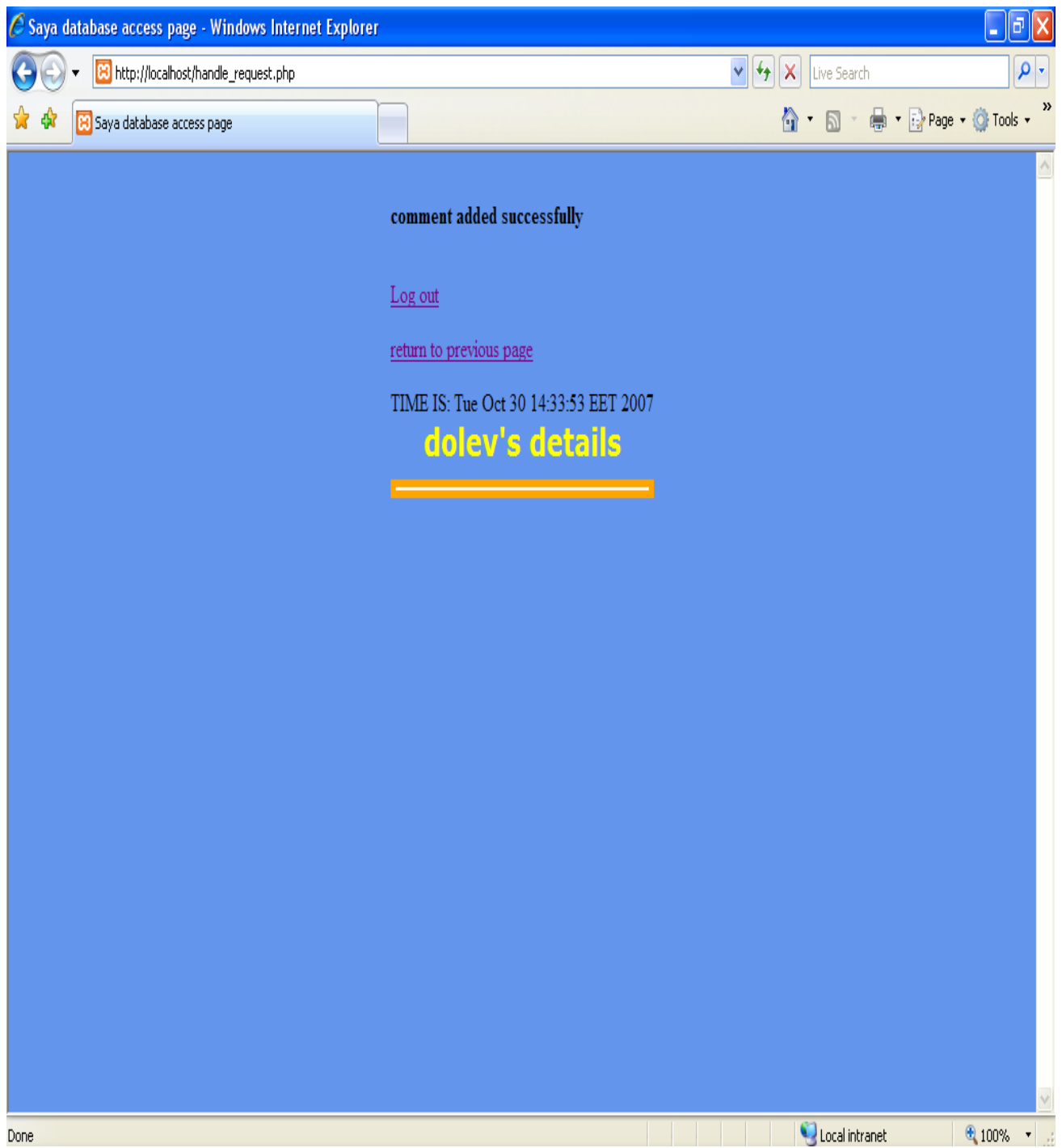


fig.5 the user is notified about the changes made to the database

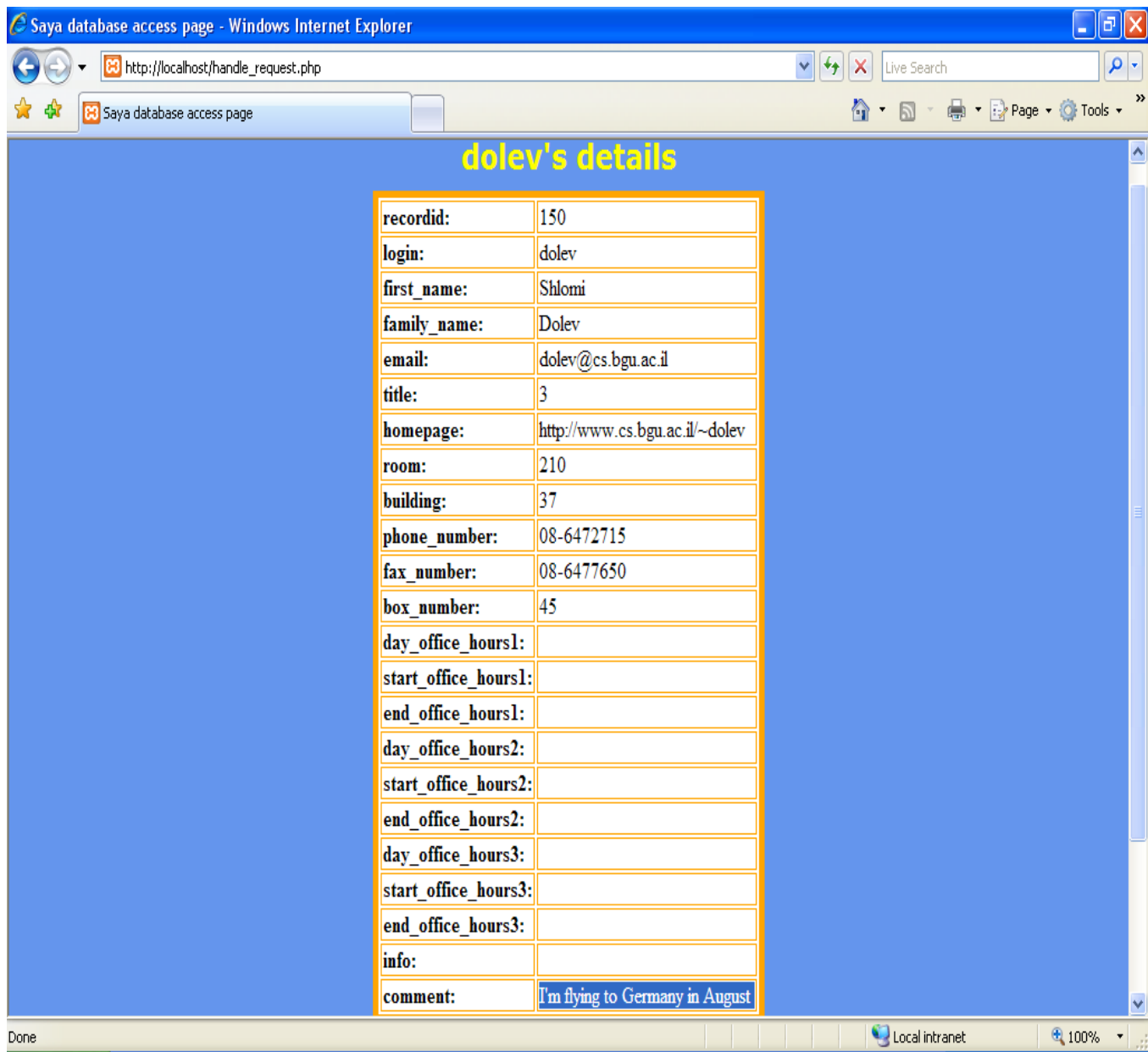


fig.6 Here we can see that the comment was indeed added to the database

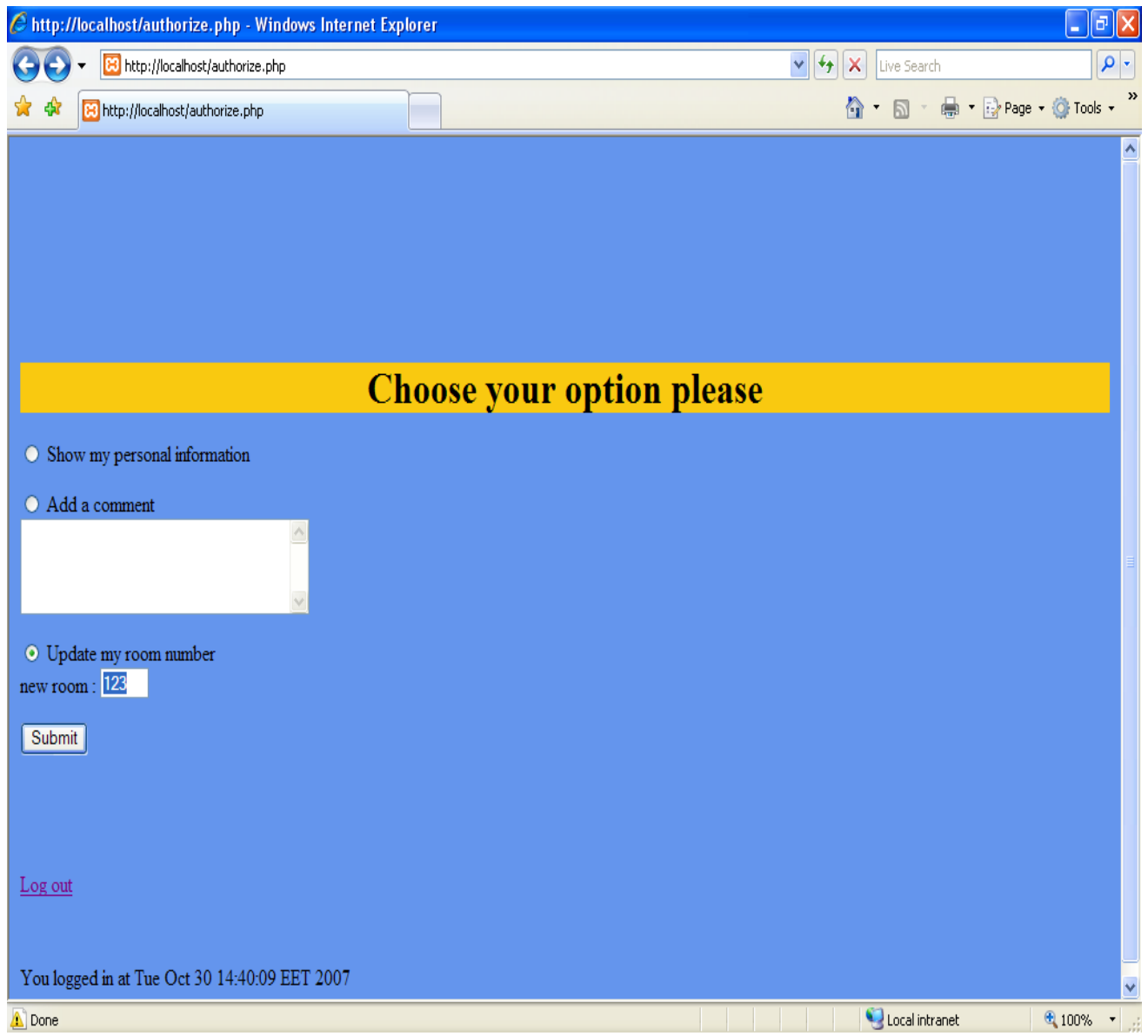


fig. 7 a lecturer is interested to update his room number

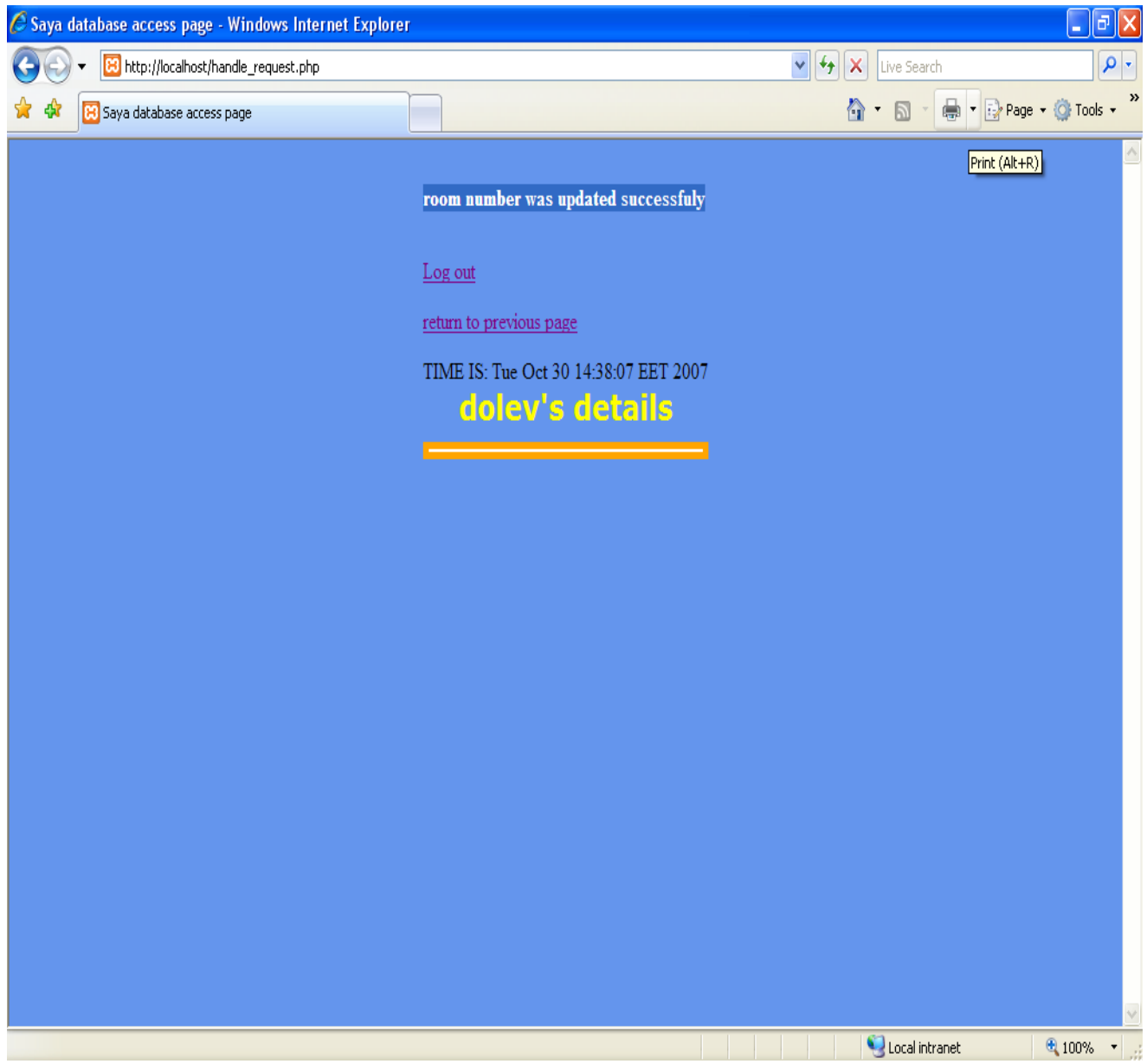


fig.8 the lecturer is notified about the changes in the database

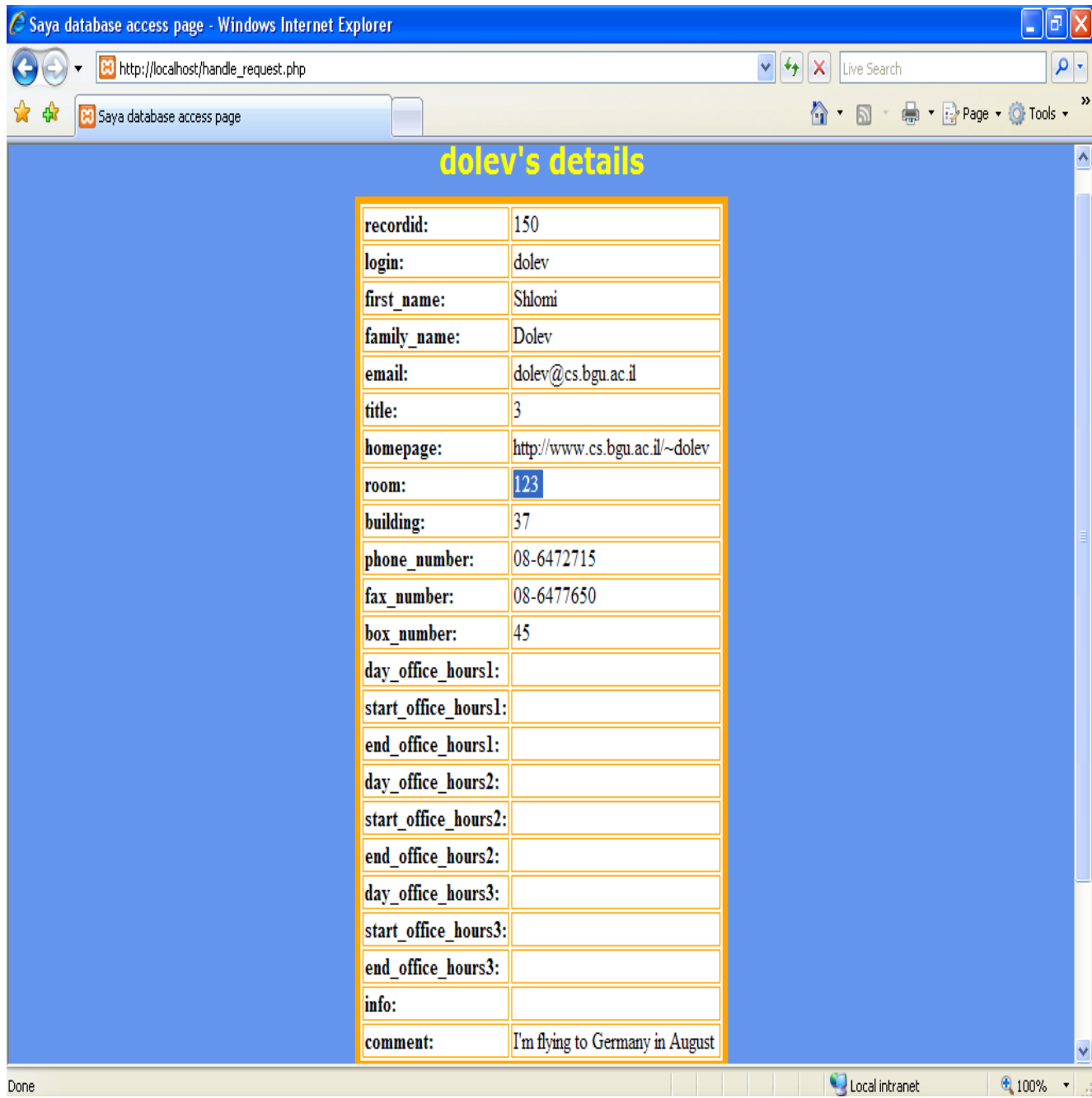


fig. 9 the lecturer can see that his room number has been changed

Conclusion:

Future improvements may include any of the following:

- Support in additional queries by Lecturer, both in Java and PHP.
- Improved Security – using SSH and other protocols
- Adding Javascript code to .php pages to allow easier browsing