Assignment 8 : Banking Website Design

Sarbajit Ghosh [CrS1911]

July 29, 2020

Basic Design

- Login page shown in figure 1.

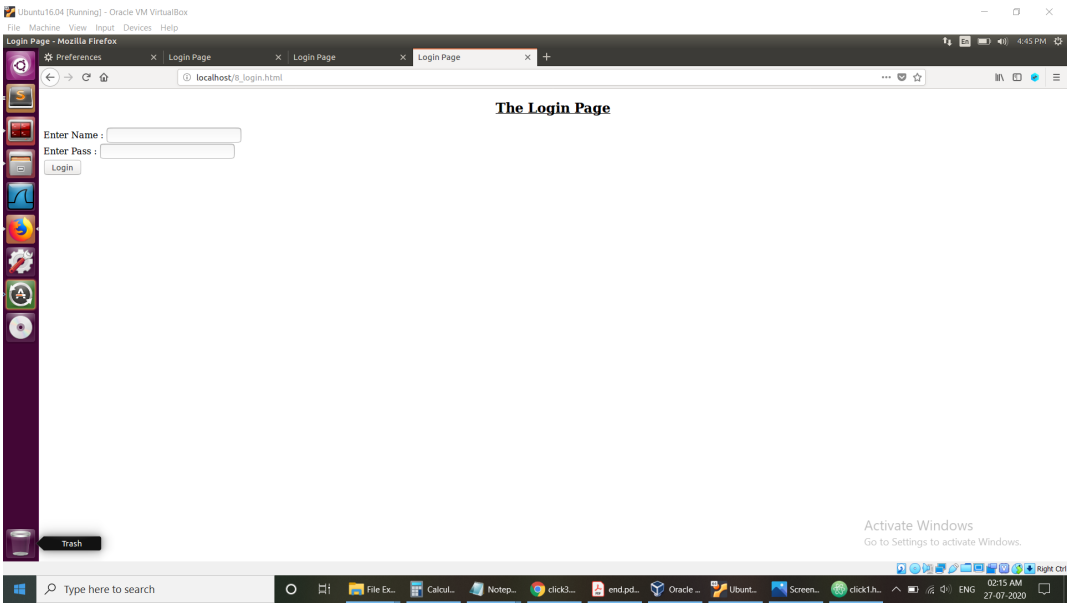

Figure 1: Login Page

- If some one gives wrong user name it give the folloing error messege shown in figure 2
  If some one gives wrong password the page gives the follwing error, shown in figure 3

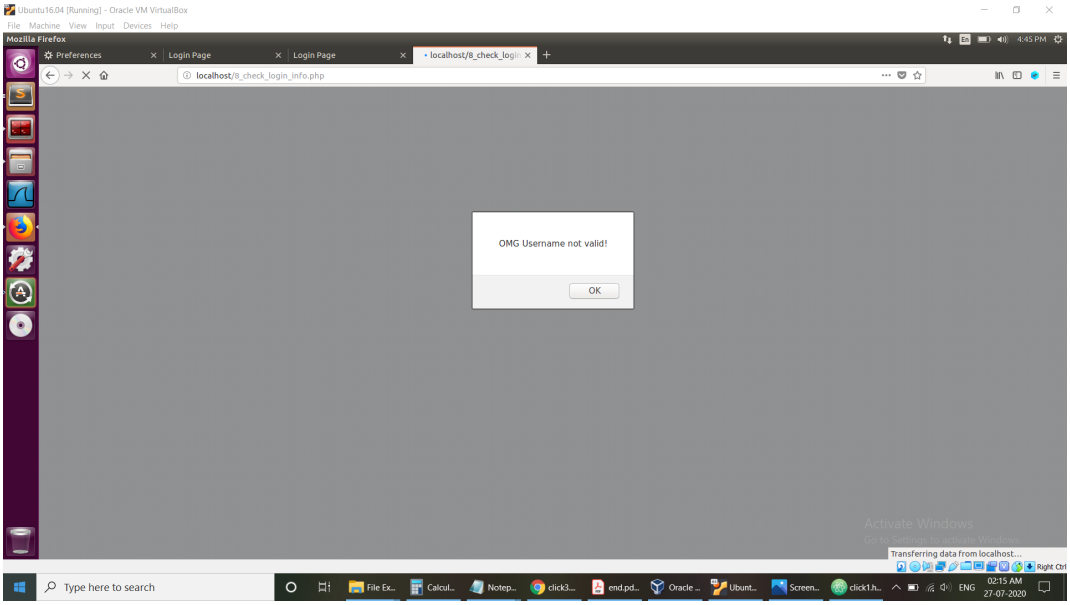

Figure 2: If username wrong

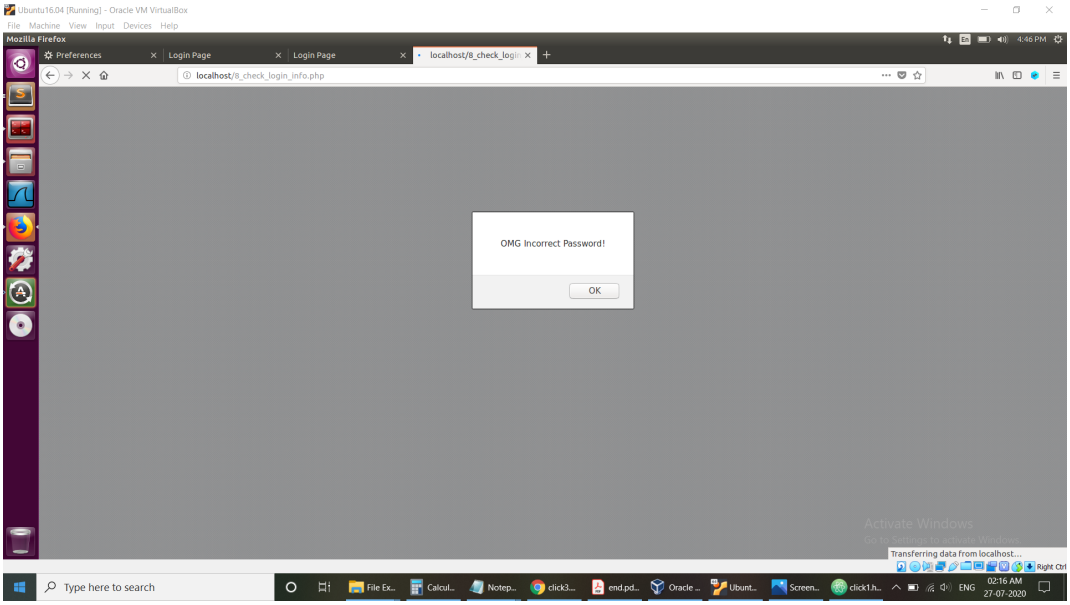

Figure 3: If username wrong

- After login home page of an user apears. This page contains balance check, fund transfer , and last 5 transaction option. Shown in figure 4

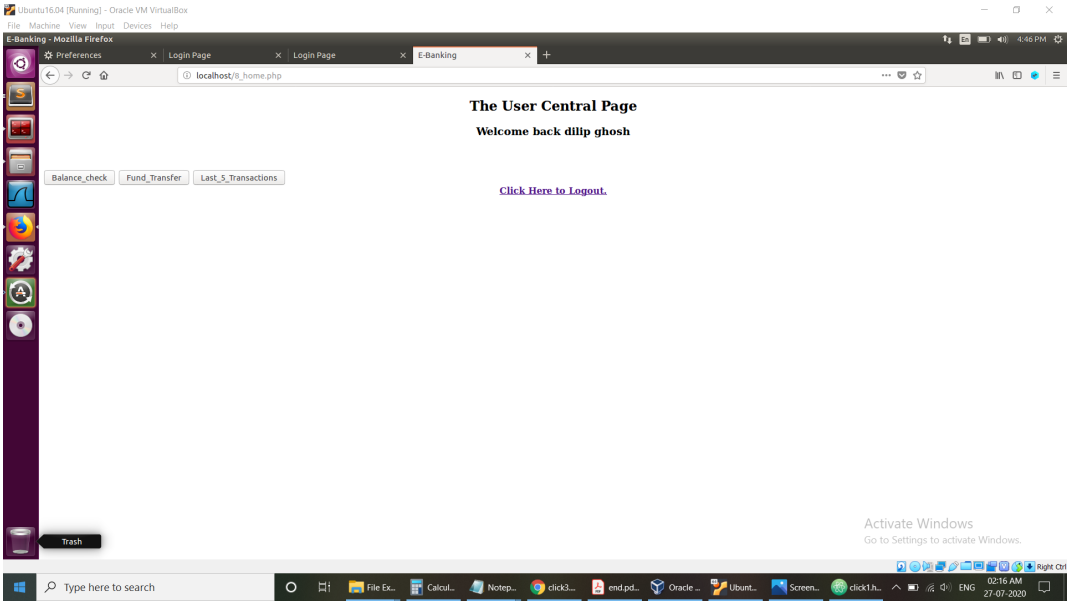

Figure 4: Home page

- View balance page.

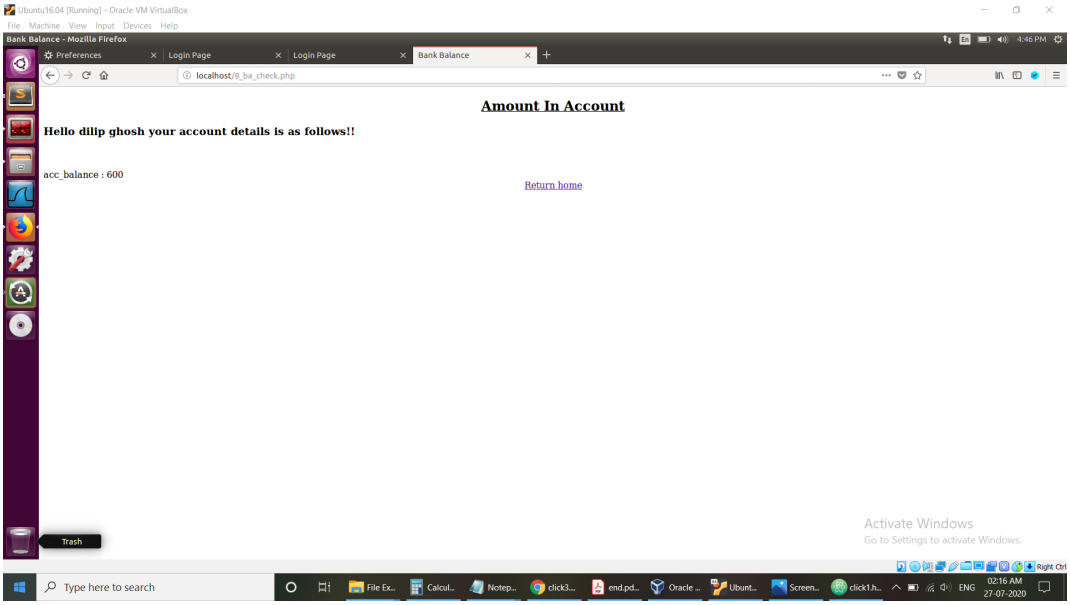

Figure 5: Balance page

- Fund transfer page

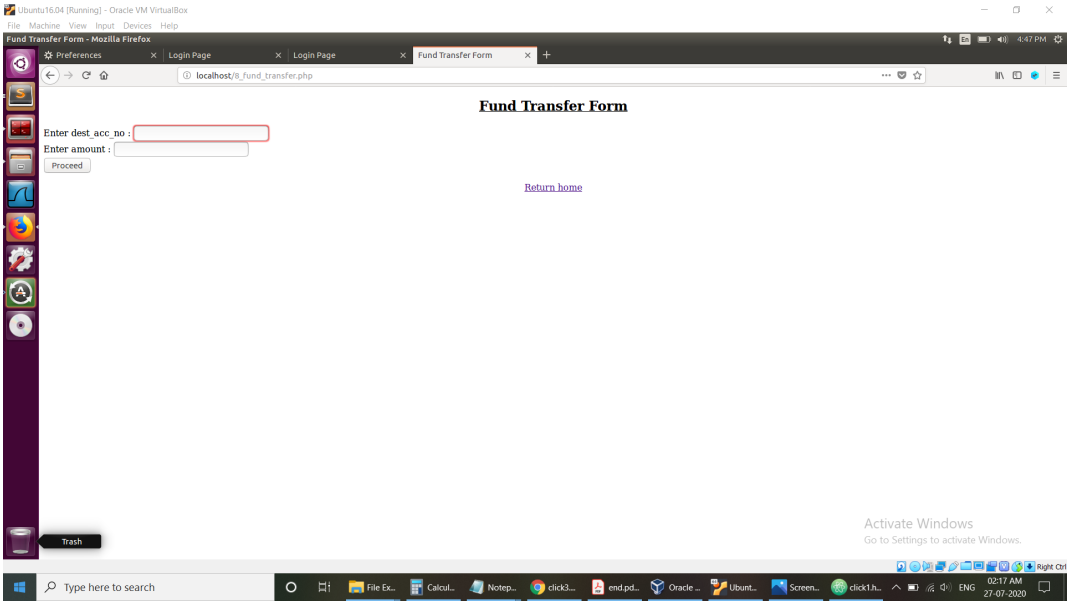

Figure 6: Balance page

- If some one enters wrong dest_ac_no or insert transferable amount mare than his account balance, the following account balance will pop out.

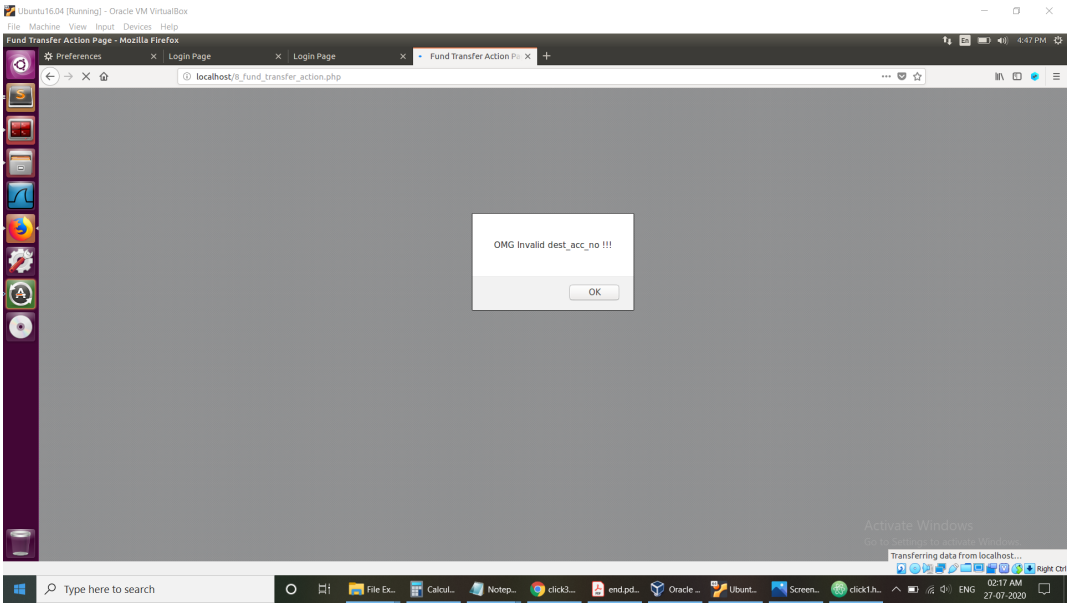

Figure 7: Err Msg on wrong dest_acc_no

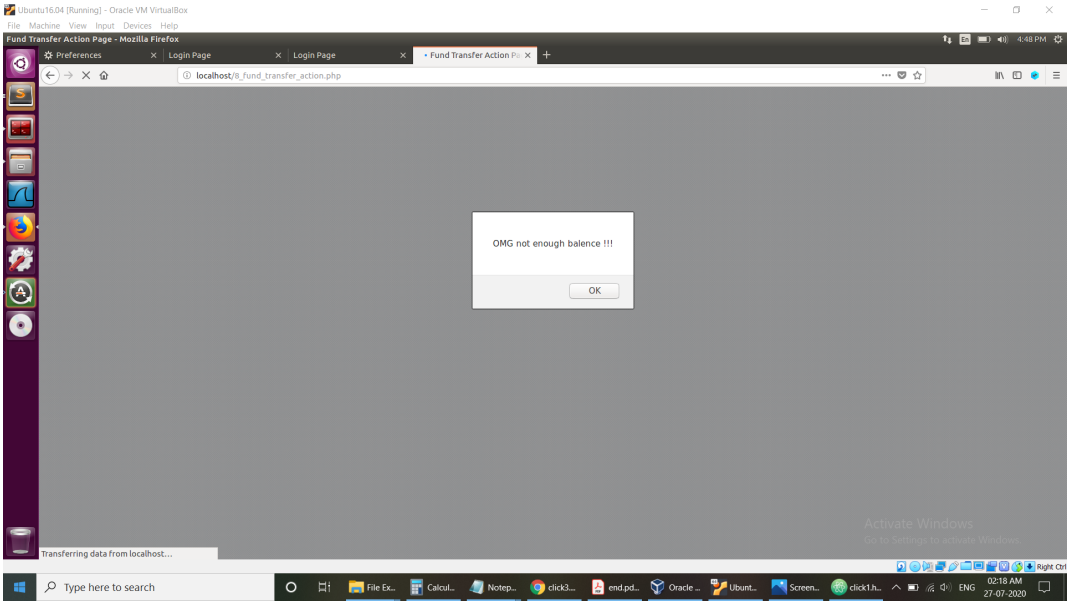

Figure 8: Err Msg on transferable amout more than acc balance

- On sucessfull transction it shows the following meg.

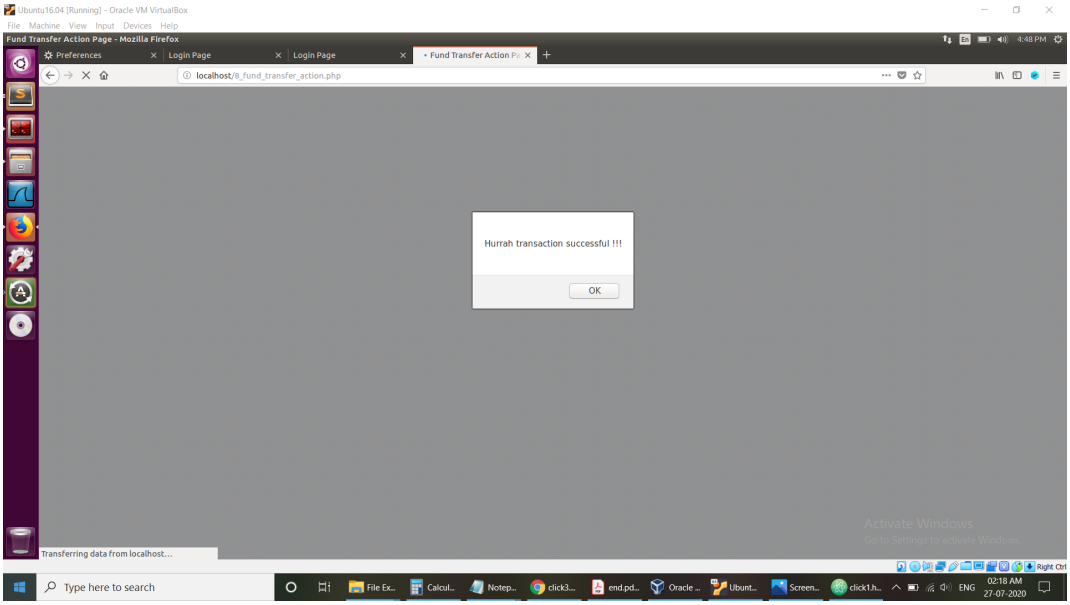

Figure 9: On sucessfull transaction

- Last five transction page.

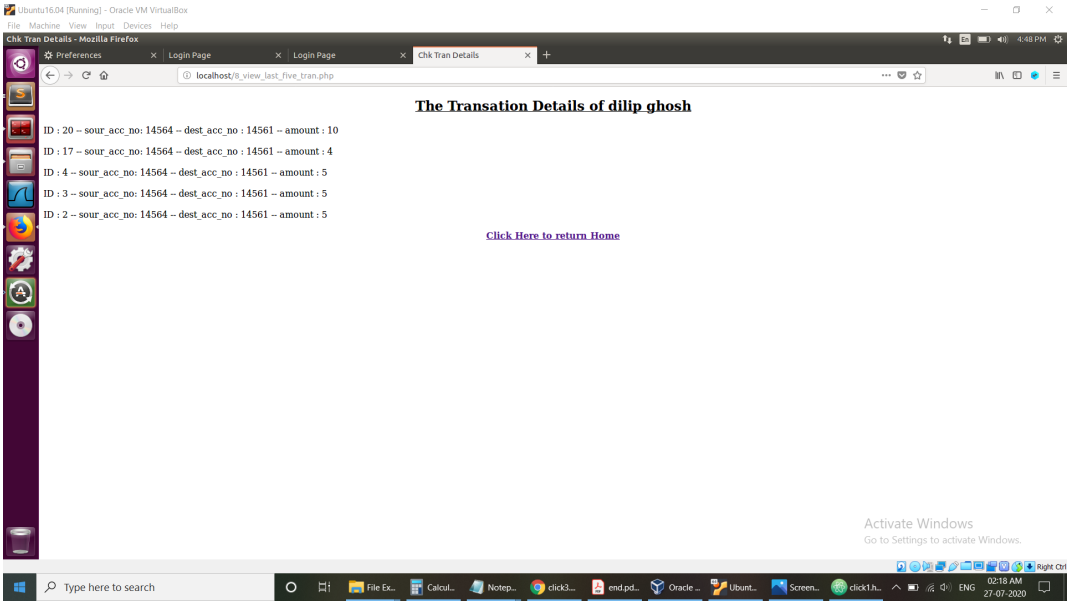

Figure 10: On sucessfull transaction

Back-end Database

For back end database wehave created three tables.
log_cred this table contains Name, password, account number.
ba_info this table contains account number, bank balane, and name.
tran_details this table contains ID [auto incremented], source account, destination account and transaction amount.
Shown in figure 11 and figure 12.

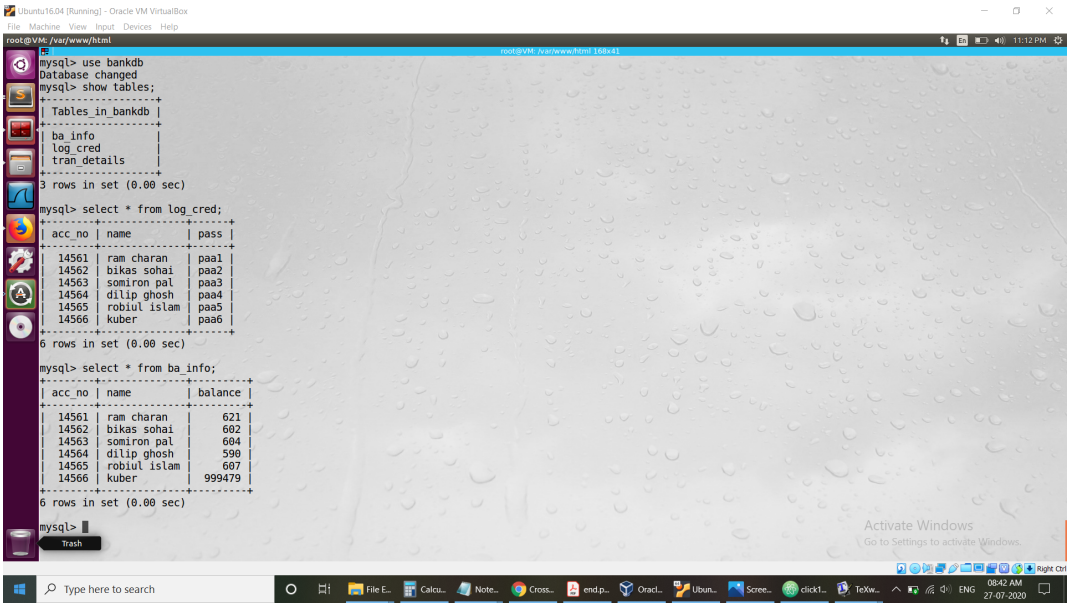

Figure 11: Bank end database 1

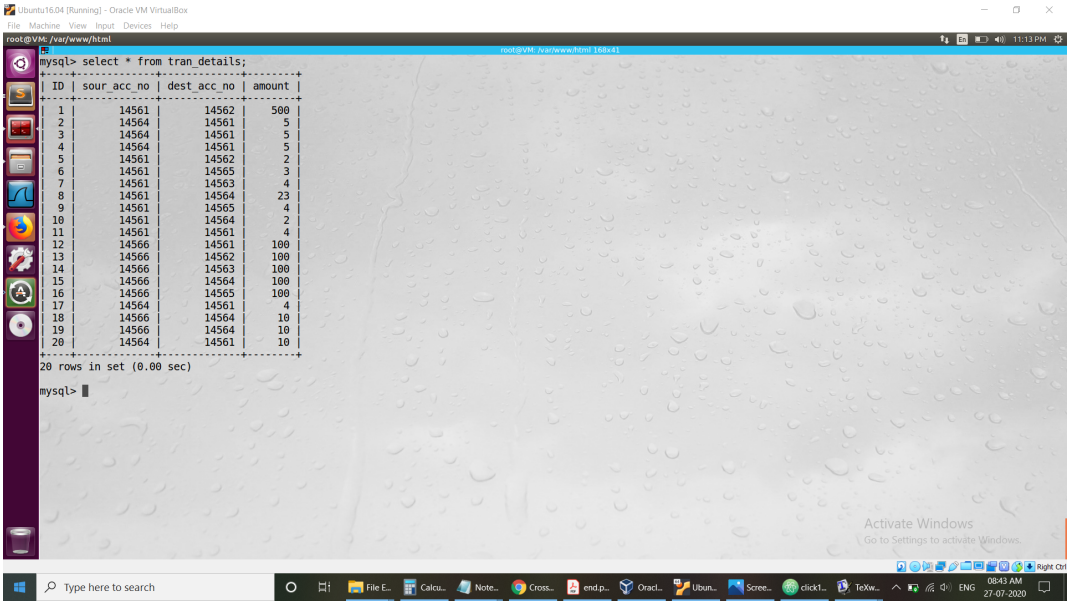

Figure 12: Bank end database 2

ClickJack attack

For click jack attack we have created a malicious page, whose apearence is exact same as our login page. An attacker can set this page on top of bank website page, with its opacity set to zero. If some naive user inserts her/his log_cred into this website this webpage saves this confidention information in a backend database.

- A pic of superposed websites, with hacking page's opacity set to zero. [Here frame size is not set to screen size]. Shown in figure 13.

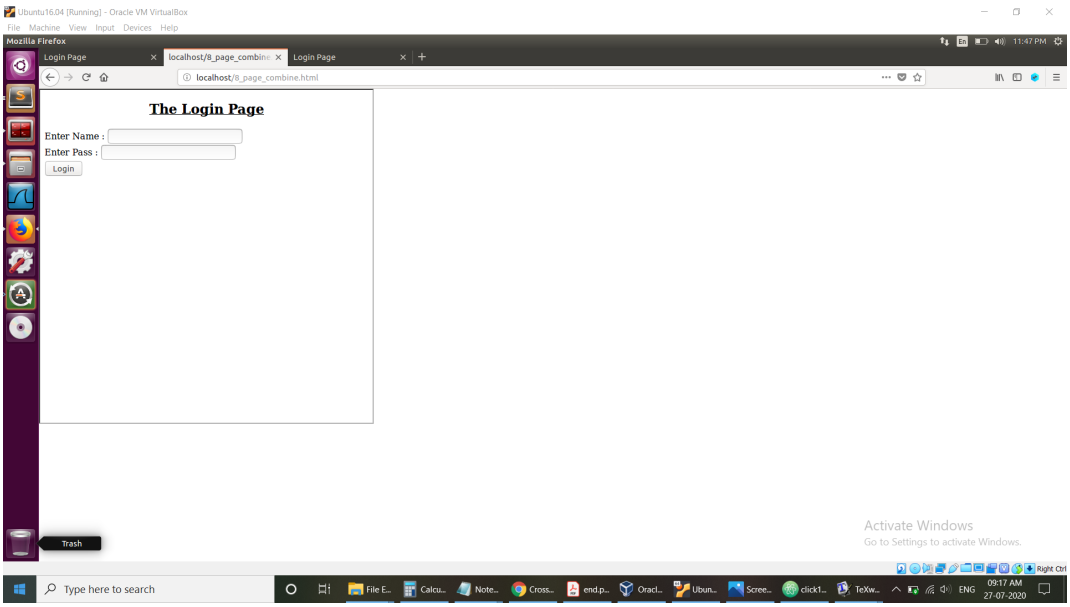

Figure 13: ClickJack Page Superposed on Login Page With Opacity Zero

- A pic of superposed websites, with hacking page's opacity set to 0.4. We can see the 'hack' word. [Here frame size is not set to screen size]. Shown in figure 14.

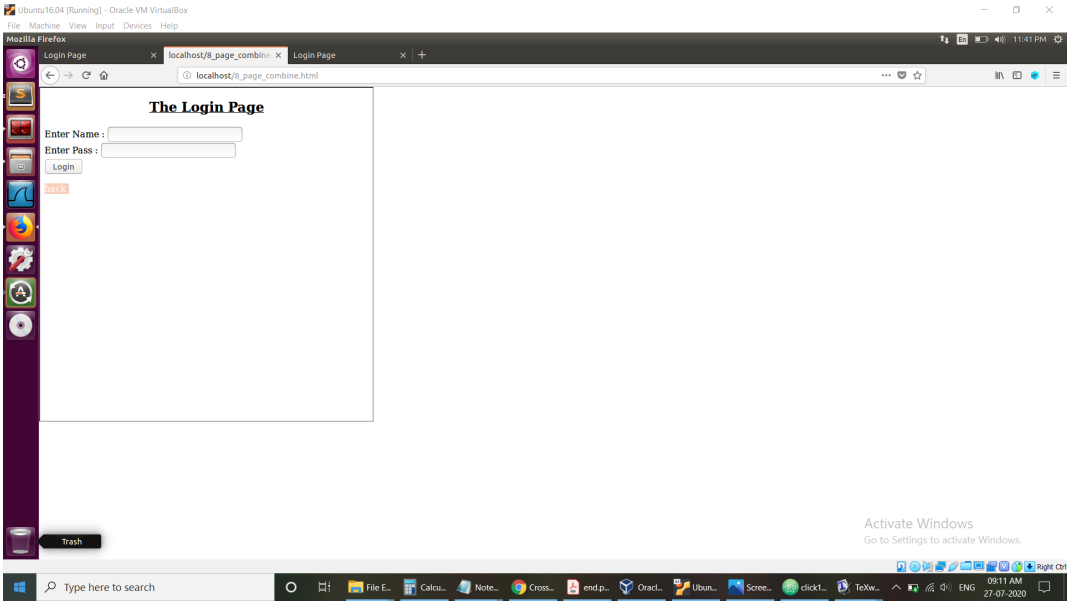

Figure 14: ClickJack Page Superposed on Login Page With Opacity 0.4

- If some user enter her/his login_cred this page shows the above msg. In figure figure 15

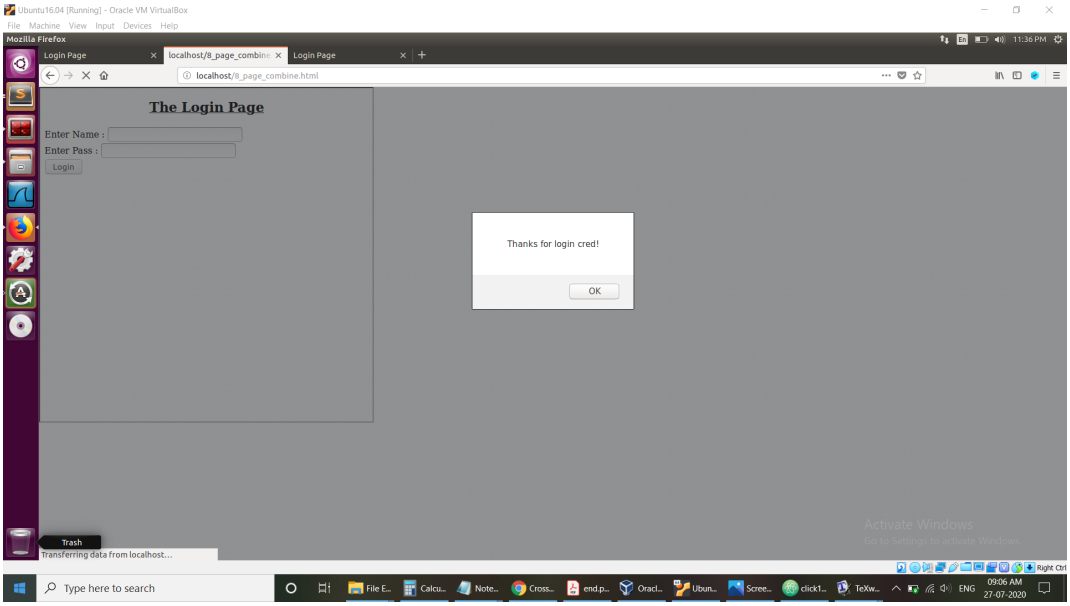

Figure 15: If someone enters data

- We can see a user has enter his data. And it is in row 20 of hackdb database. figure 16

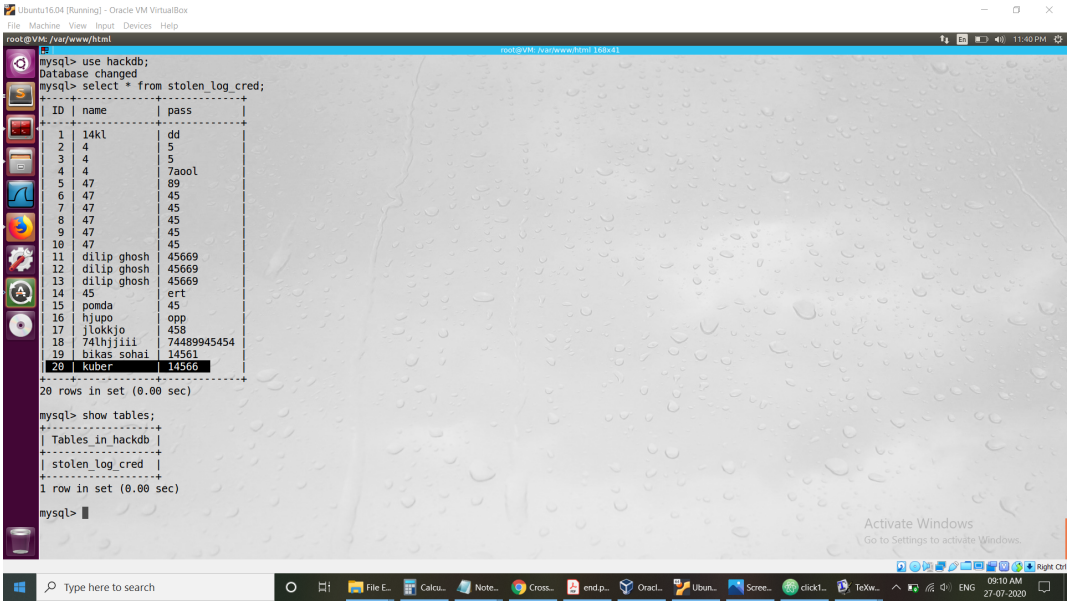

Figure 16: Backend database of clickjack page

Counter measure of ClickJack: To prevent the attack we have inserted the following piece of code into the bank's original login page. It will check whether the website is on the top position or not.

```
<script>
if(self != top) {
top.location = self.location;
}
</script>
```

CSRF Attack

If some one clicks a link of malicious website which is build on, CSRF attack principle, while she/he is logged into her/his bank account. Then csrf attack pagw can malicously stole the user's session variables to do some malicious activity. Based on this principle we have created a page viz. csrf.php if some user, with logged in status click's into that link some amount will be deducted from his account and will be added into account number '14564' in our database.

- First we note that one user is logged into the account. Now lured by the cerf_page the user check the page csrf.php. Then he sees that 'transaction sucessfull' messege pops out. And some amount deducts from account. Shown in figure 17 and figure 18. This attack is based on active session of user, and on the fund transfer page.

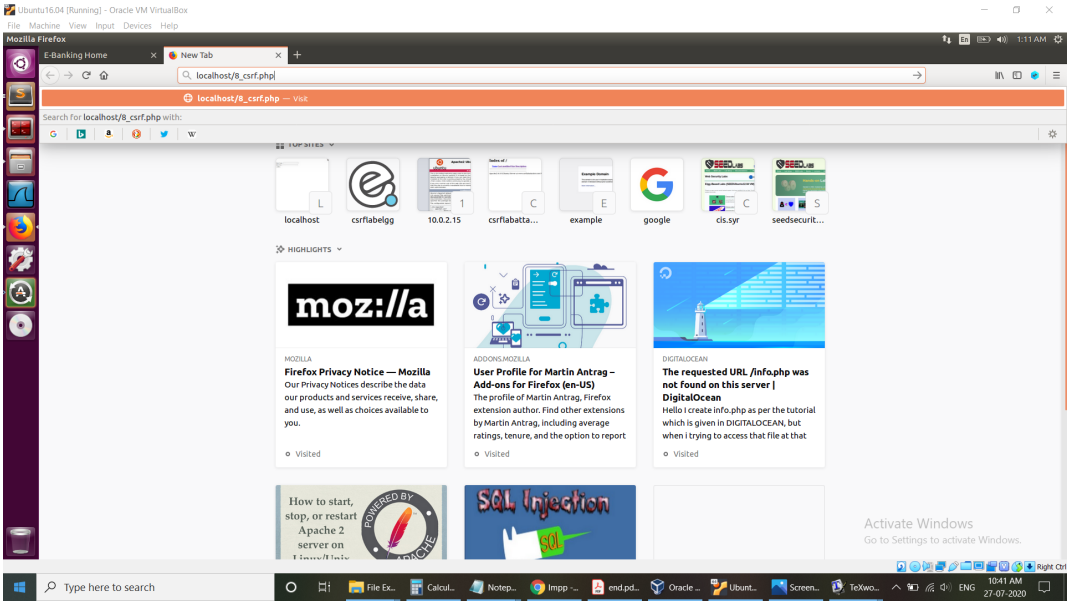

Figure 17: CSRF Demo Part 1

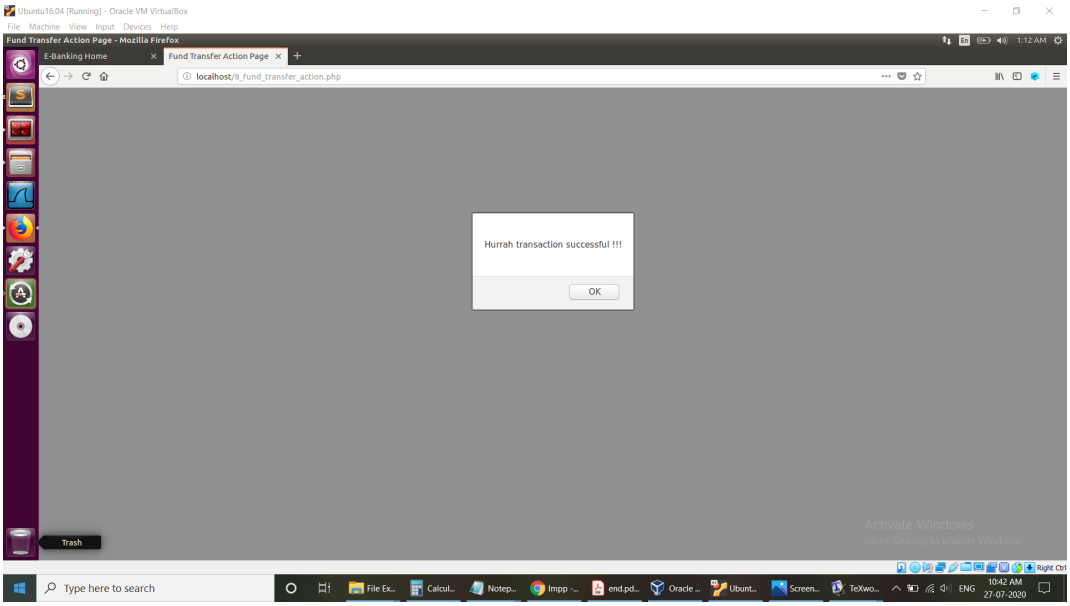

Figure 18: CSRF Demo Part 2

- **Counter measure of CSRF:** To prevent CSRF attack we added the folloing counter measures based on CSRFToken. The basic idea is that in 8_fund_transfer.php page, together with user inserted vales, before passing the control to the 8_fund_transfer_action.php page, we will inset some token hiddenly. By checking the value of the token 8_fund_transfer_action.php page will decide whether the fund transfer page come from valid page or any other website.

  We have implemented following counter measures:
  In 8_fund_transfer.php page:

  ```
  <input type="hidden" name="CSRFToken" value="jshdjdhksfshhsf">
  ```

  In 8_fund_transfer_action.php page:

```
$token = $_POST['CSRFToken'];
if($token != 'jshdjdhksfshhsf')
{
Print '<script>alert("honesty is the best policy !!!");</script>';
session_destroy();
Print '<script>window.location.assign("8_login.html");</script>';
exit();
}
```

After that if some user clicks the link his account won't be vulnerable. Shown in figure figure 19 and figure 20

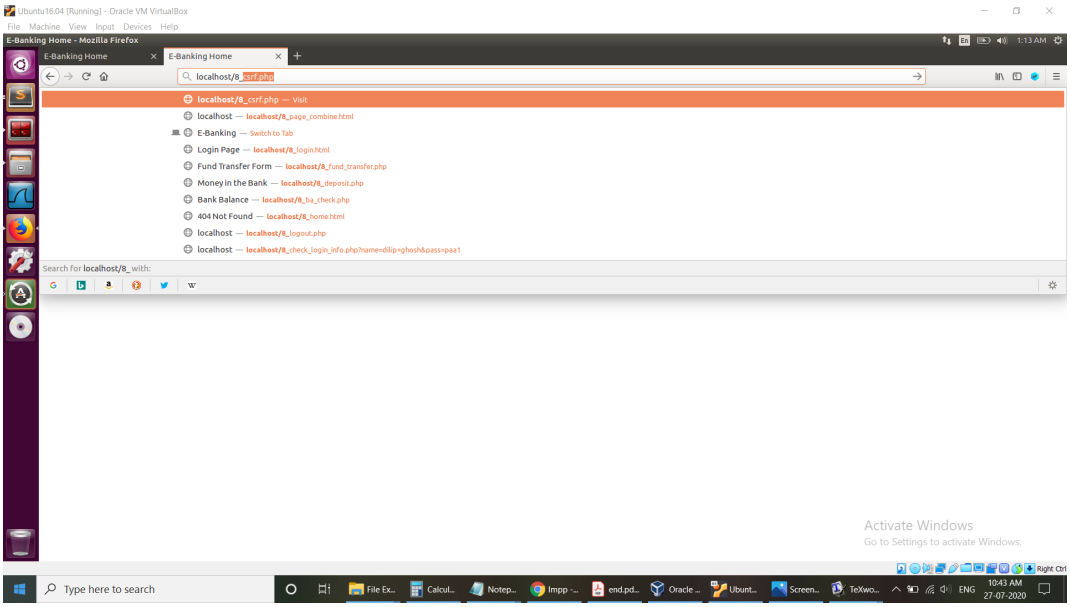

Figure 19: CSRF Demo Part 3

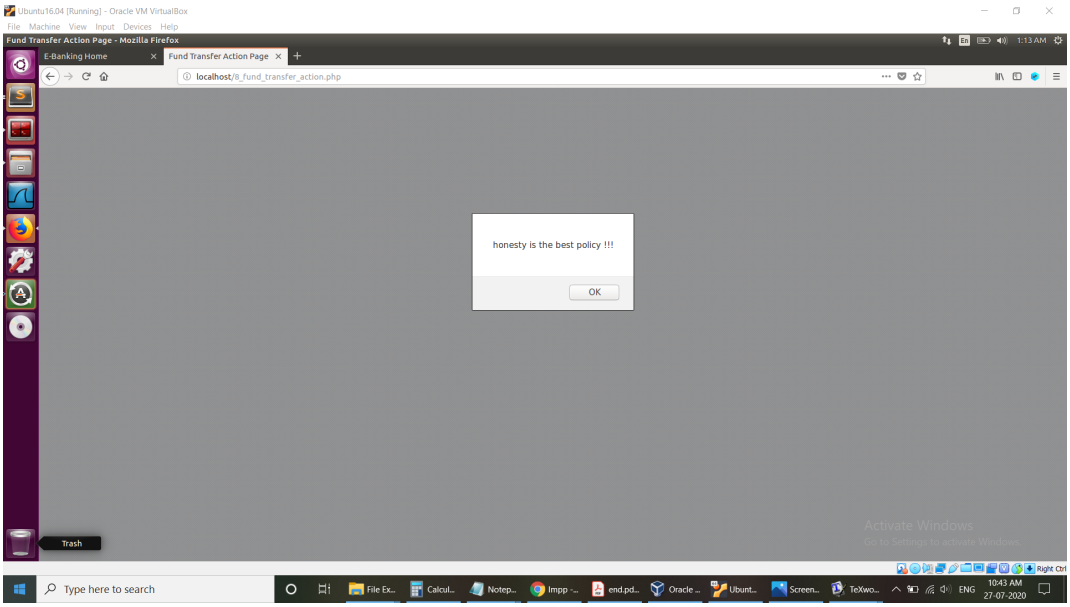

Figure 20: CSRF Demo Part 4

SQL Injection

SQL injection vilnerability comes, because of non data and code seperability. Here our banking application becomes vulnerable to SQL injection attack, as it takes user data without proper sanitaztion and mix it into code. 8_login.html page takes data from user and sends to the 8_check_login_info.php page. So we investigate and found vulnerability that in our banking application in page viz. 8_check_login_info.php one could enter without password or even without propername. This observation is shown in figure figure 21 and figure 22.

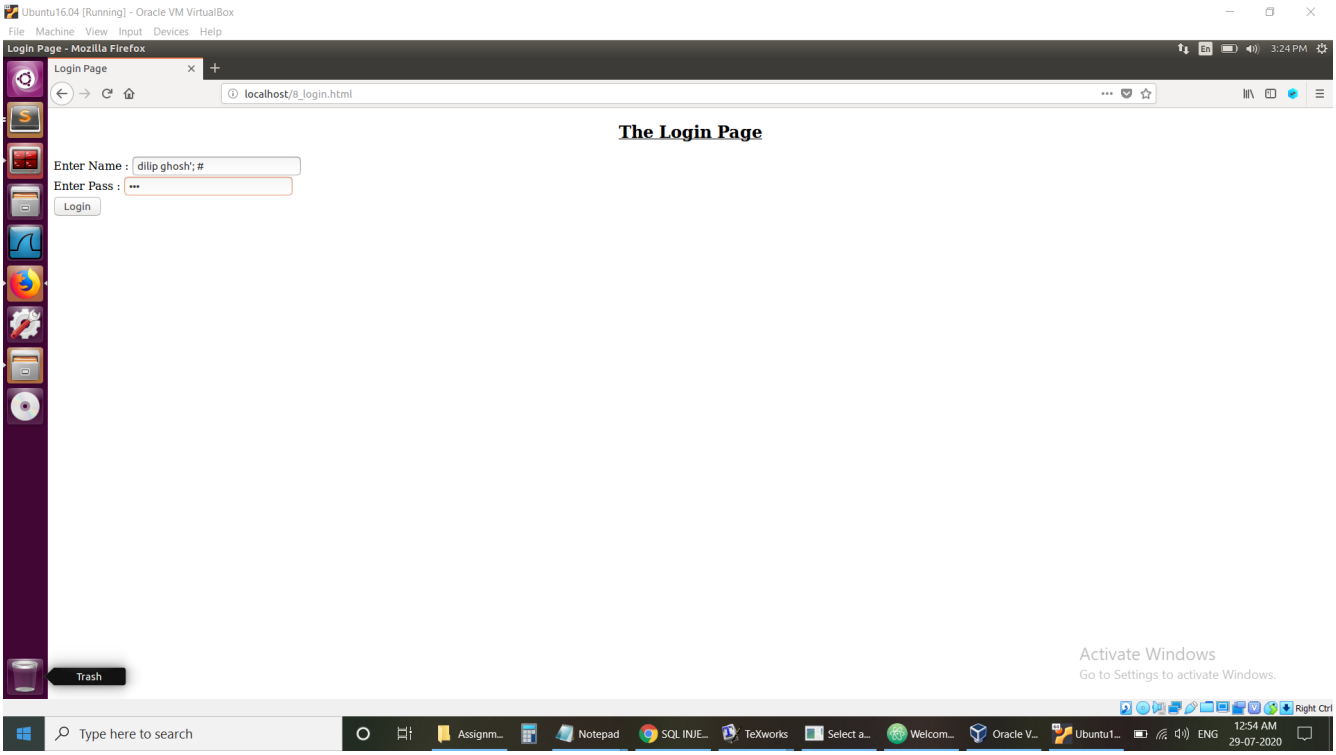

Figure 21: SQL injection 1

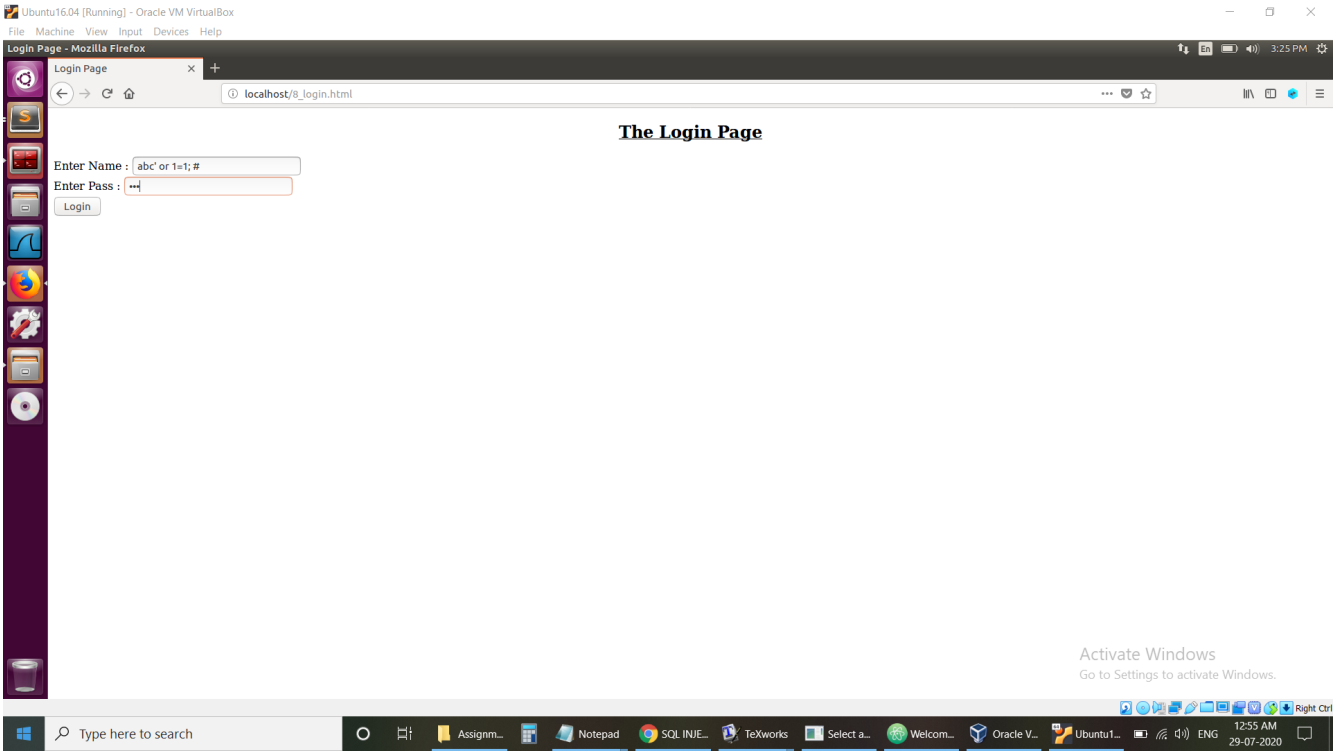

Figure 22: SQL injection 2

Reason for vulnerability : For this we first give the vulnerable block of code in 8_check_login_info.php

```
$sql = "SELECT * FROM log_cred WHERE name = '$name' AND pass = '$pass'";
    $result6= $conn->query($sql);
    $row6 = $result6->fetch_assoc();
    //echo $row6['name'];
    if($row6 > 0)
    {
        $_SESSION['user']= $row6['name'];
                header("location:8_home.php");
    }
```

If we give dilip ghosh'; # in the name field and xyz in the pass word field then without knowing the passwords of the user by barely knowing his name can login into his account. More over if any one give input abc' or 1=1; # in name field and any random string into password then even an attacker can login into bank page.
If we put this string into the proper field then we will see the query is still vaild.
SELECT * FROM log_cred WHERE name =dilip ghosh'; # AND pass =xyz.
SELECT * FROM log_cred WHERE name =abc' or 1=1; # AND pass =xyz.
Thus these caused vulnerabilty in bank page. To encounter this problem we will use sql prepared statement and update the block as follows.

Counter measure of SQL Injection

```
//block with security
    $query="SELECT * FROM log_cred WHERE name = ?";
        if($row = $conn->prepare($query))
        {

                $row->bind_param("s",$name);
                $row->execute();
                $row->bind_result($acc_no_q,$name_q,$pass_q);
                $row->fetch();
```

```
        //echo "Hi";
}
else
{
        echo "failed";
}
if($name == $name_q)
{
        if($pass == $pass_q)
        {
                $_SESSION['user']= $name_q;
                header("location:8_home.php");
        }
        else
        {
                Print '<script>alert("OMG Incorrect Password!");</script>';
                Print '<script>window.location.assign("8_login.html");</
                    script>';
        }
}
else
{
        Print '<script>alert("OMG Username not valid!");</script>';
        Print '<script>window.location.assign("8_login.html");</script>';
}
```