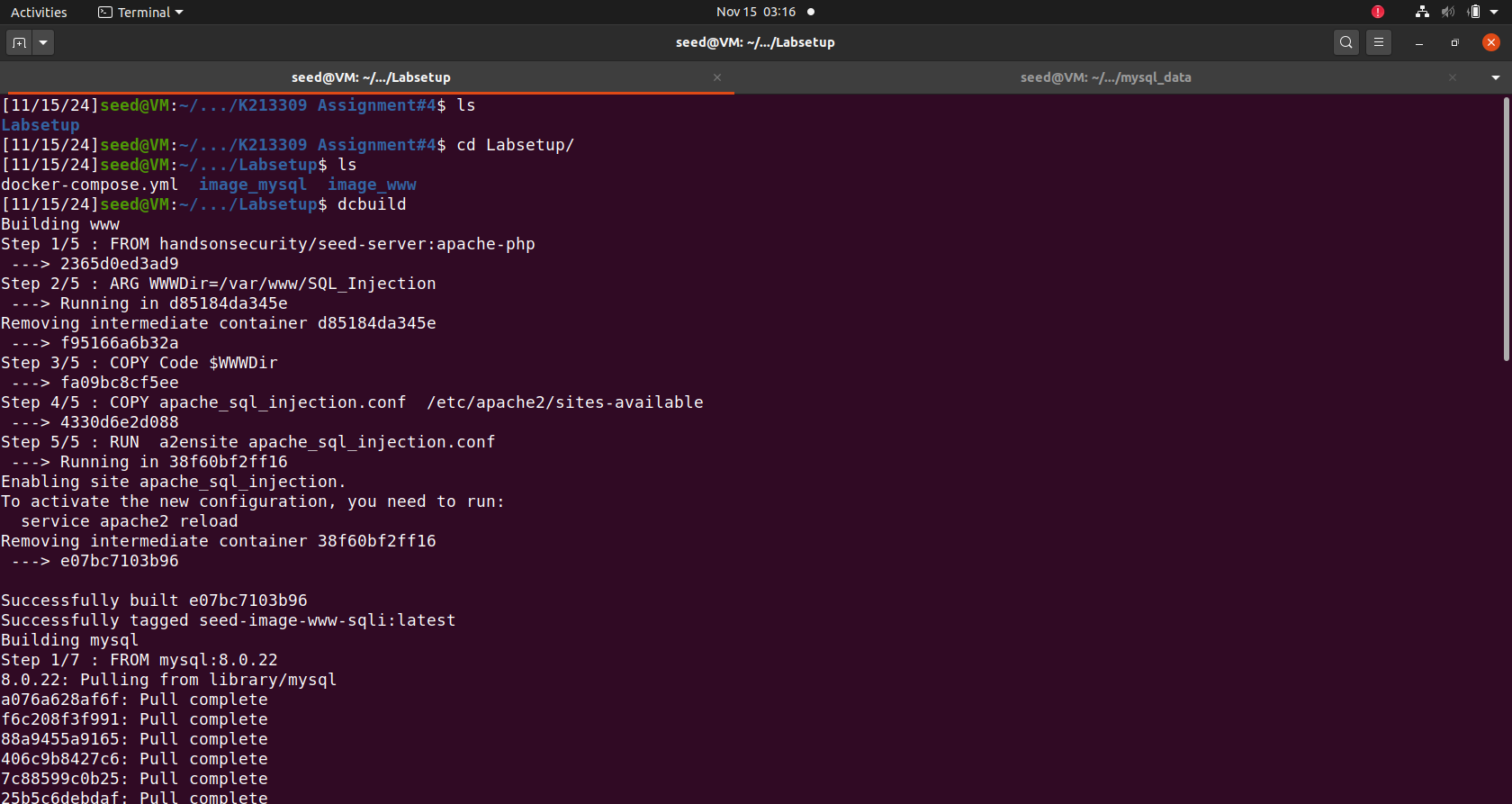
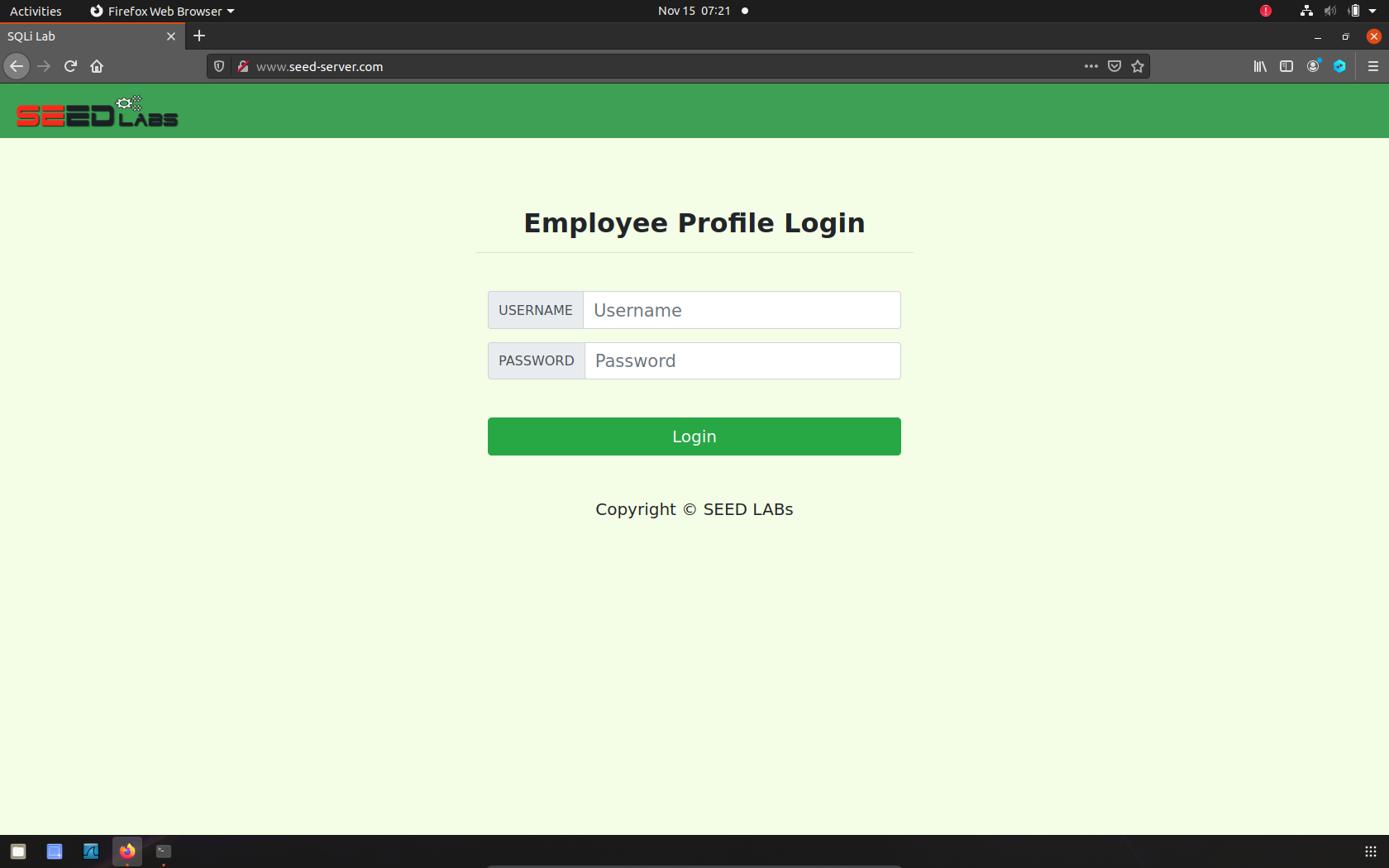
# Setup

First, we have to setup the docker container to allow access to the website ([www.seed-server.com](http://www.seed-server.com)) and then start the container.

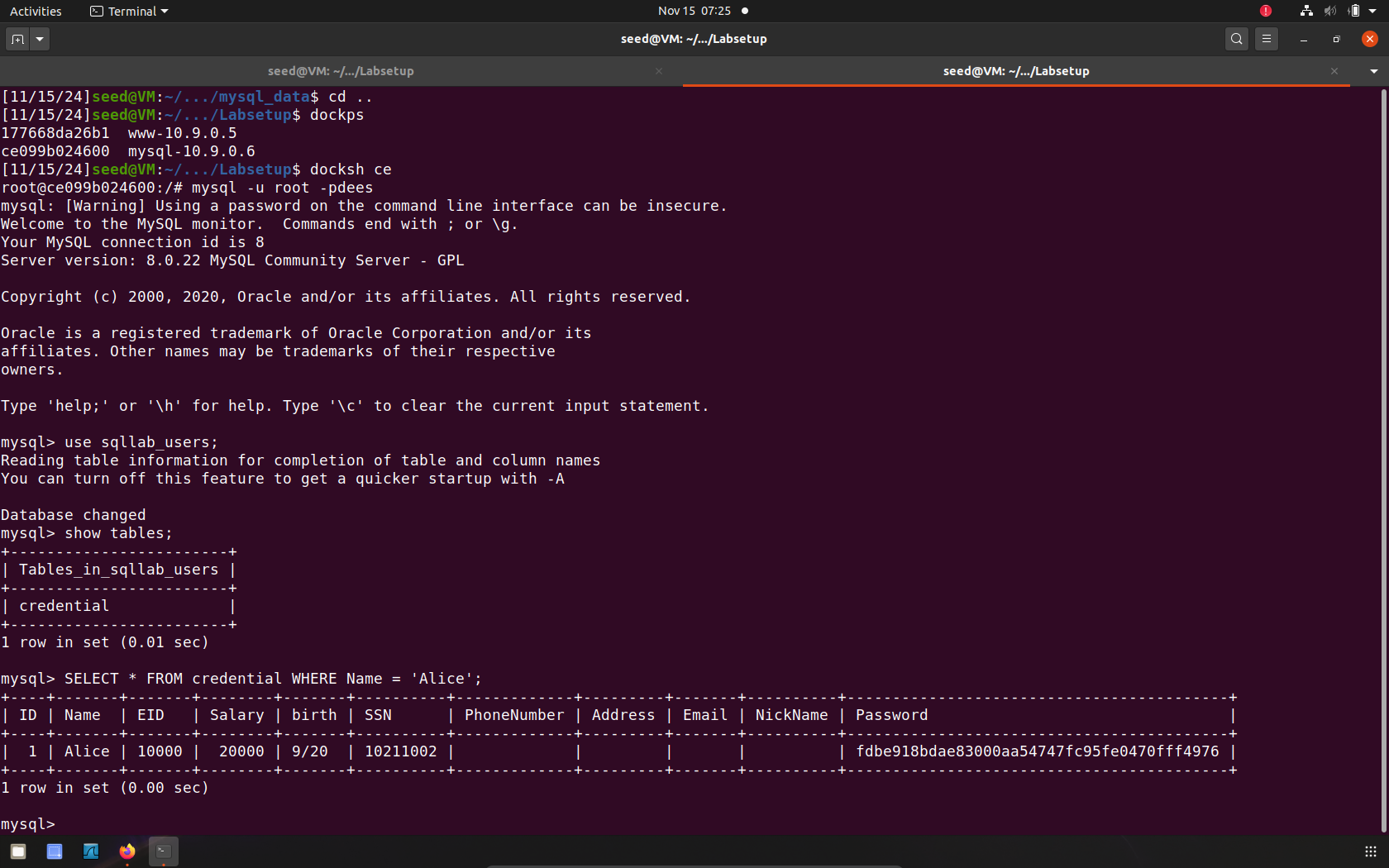


This allows us to access the website.



# Task 1: Get Familiar with SQL Statements

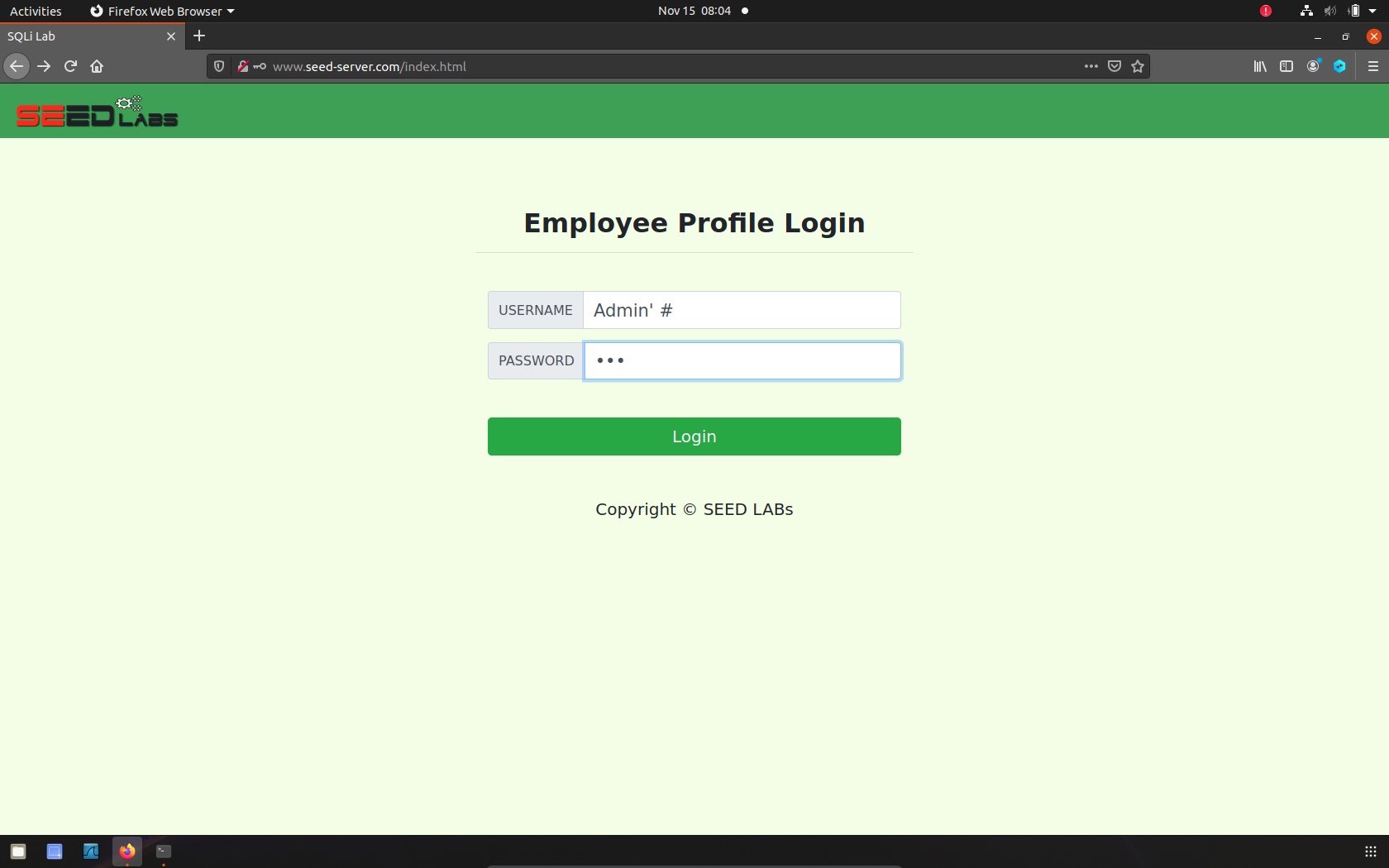
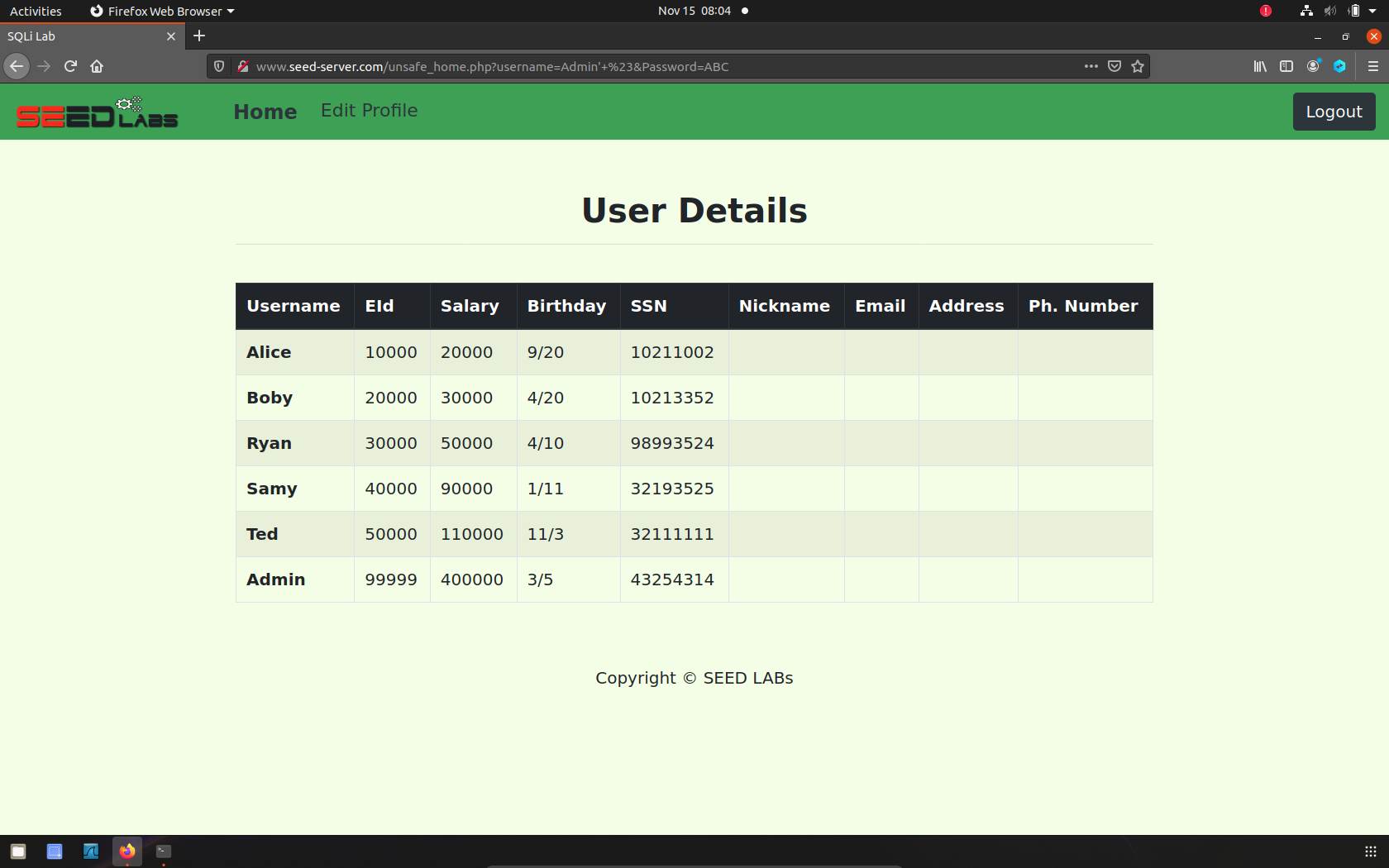
In this task, we need to just activate the mysql server in the mysql container. We then get the details of Alice.



# Task 2: SQL Injection Attack on SELECT Statement

## Task 2.1: SQL Injection Attack from webpage.

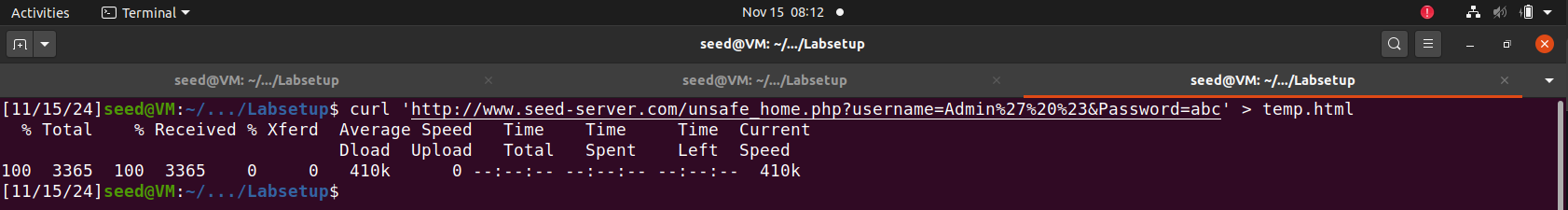
Since we need to login through the website and we only know the username, we will need to use the vulnerabilities to our advantage. We will input the username as "Admin' #" and the password can be anything as it will be turned into a comment.

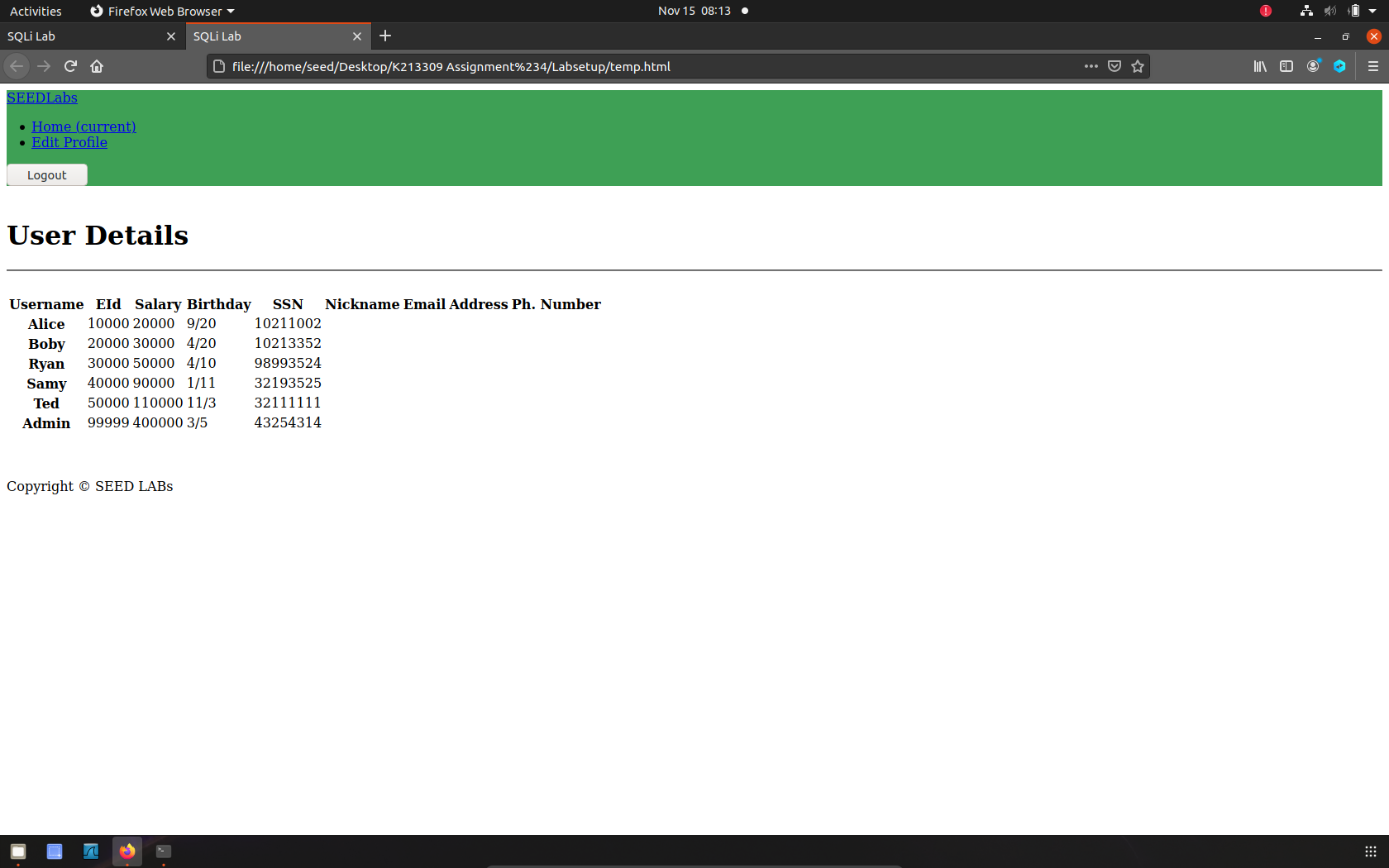
## Task 2.2: SQL Injection Attack from command line.

In this part, we will repeat the above procedure only on command line now. Using the example in the manual, we can format our command as such:

curl ‘http://www.seed-server.com/unsafe\_home.php?username=Admin%27%20%23&Password=abc’

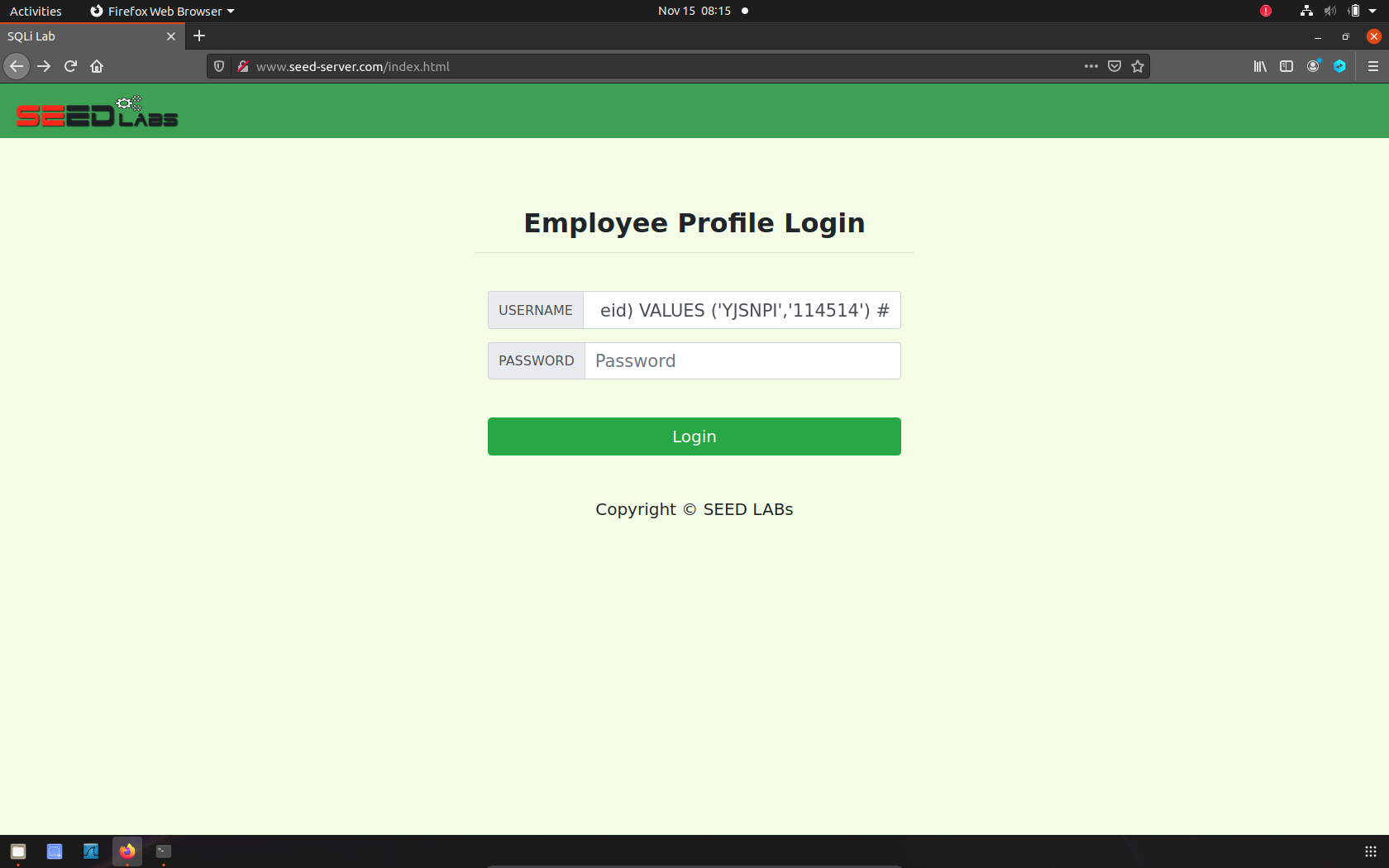


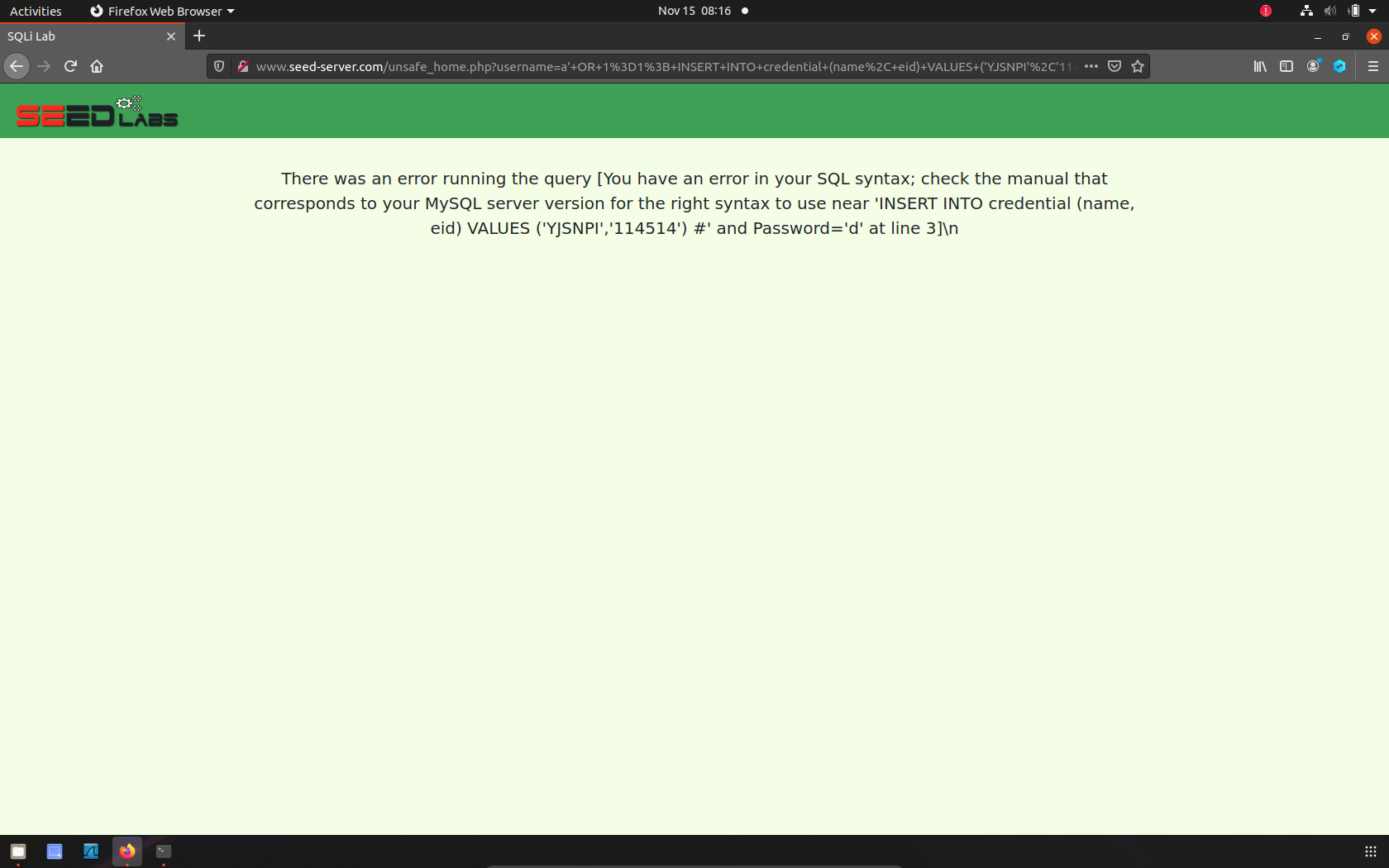
This puts all the html code into a temp file. When we open that temp file we see this:



## Task 2.3: Append a new SQL statement.

To inject another sql statement, we will need to format our text in a specific way. If we put the username as: a' OR 1=1; INSERT INTO credential (name, eid) VALUES ('YJSNPI','114514') # , we will be able to run another command.



However, when we login into the website, we are found with this error. This is because the library used to implement this has countermeasure to stop multiple sql statements at once.

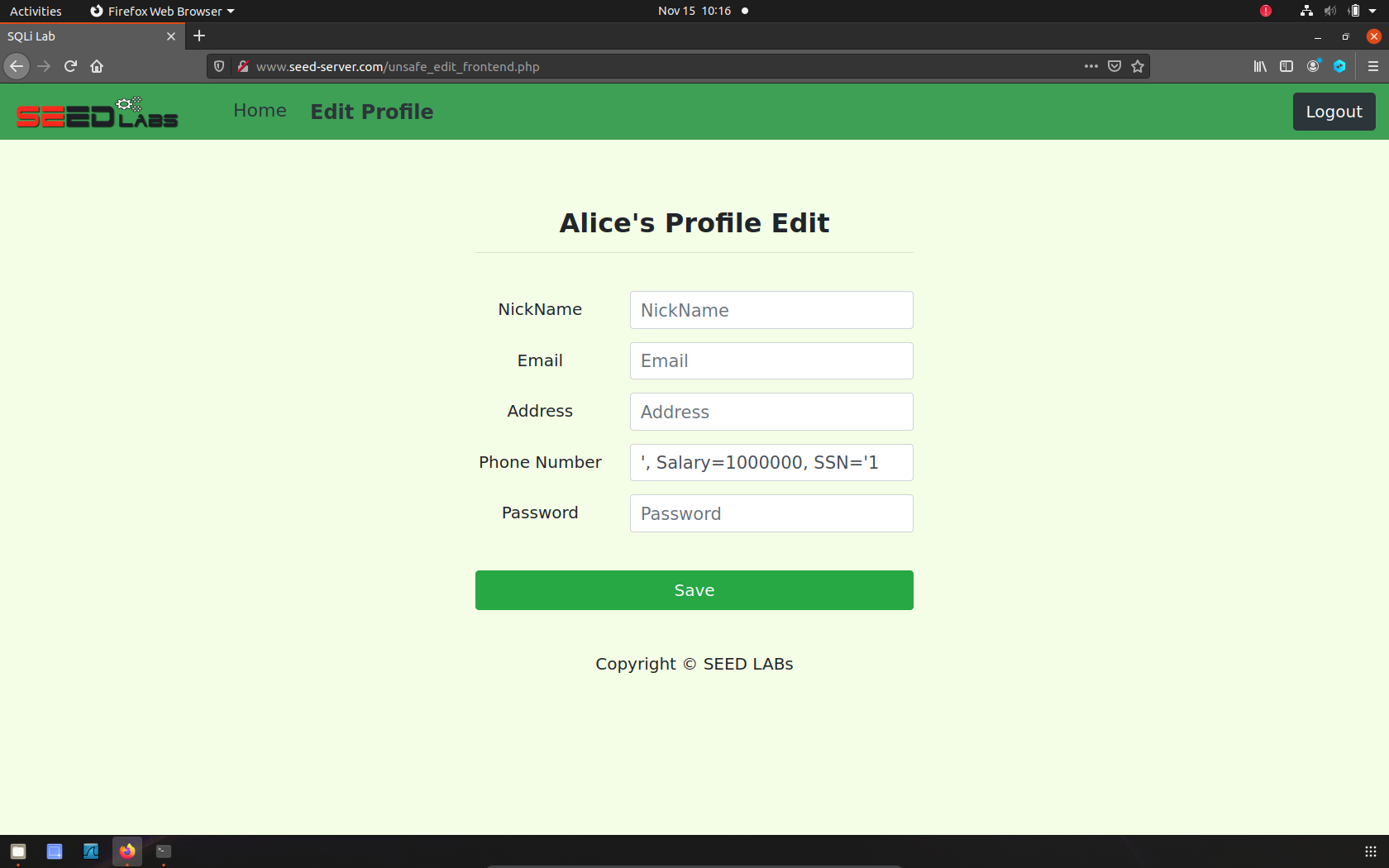
# Task 3: SQL Injection Attack on UPDATE Statement

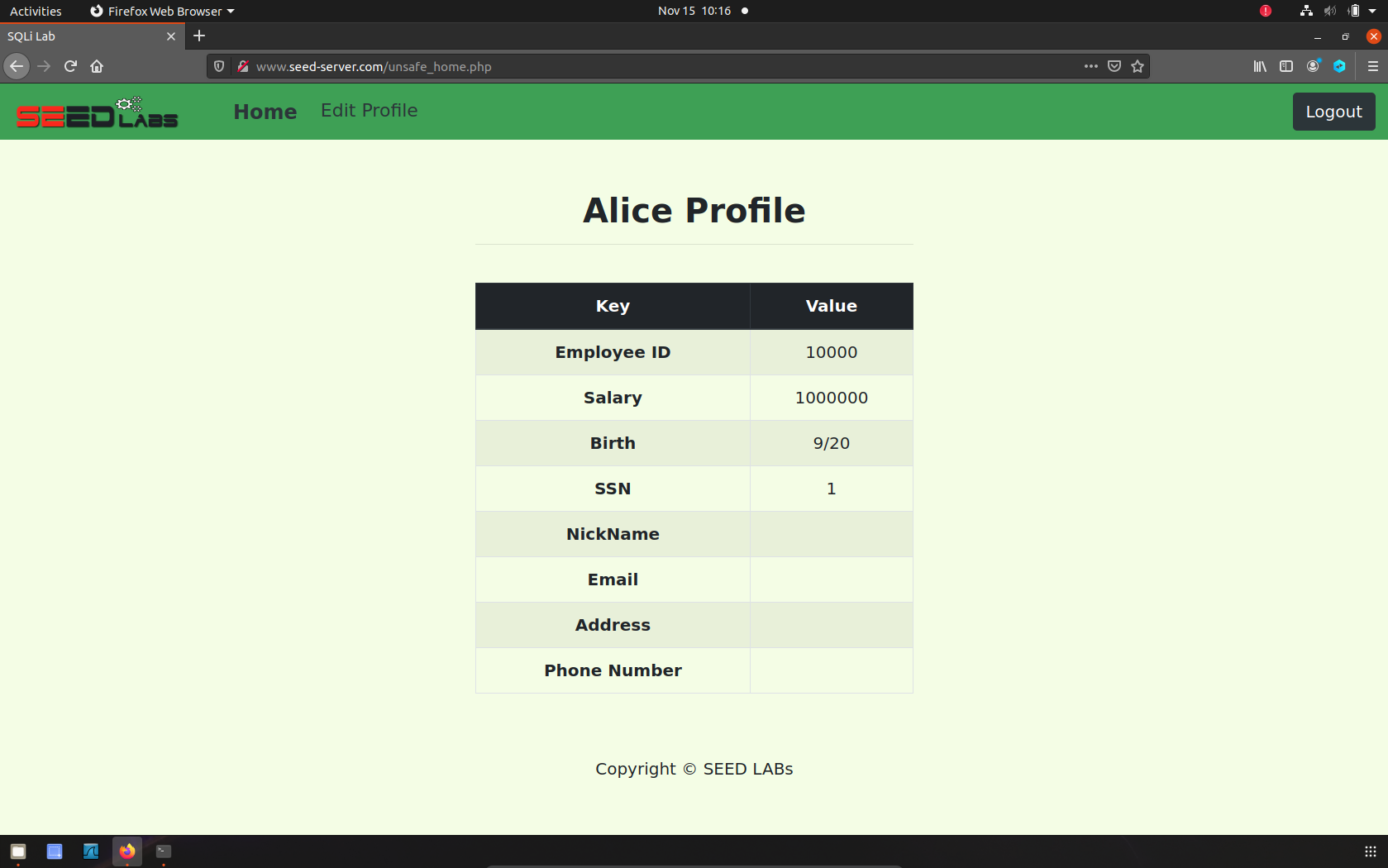
## Task 3.1: Modify your own salary.

We will modify our salary from the website. First, we enter the username: Alice’ # to login into Alice’s account. Then we navigate to the edit profile tab. Then in the phone number tab we will enter the following statement:

', Salary=1000000, SSN='1

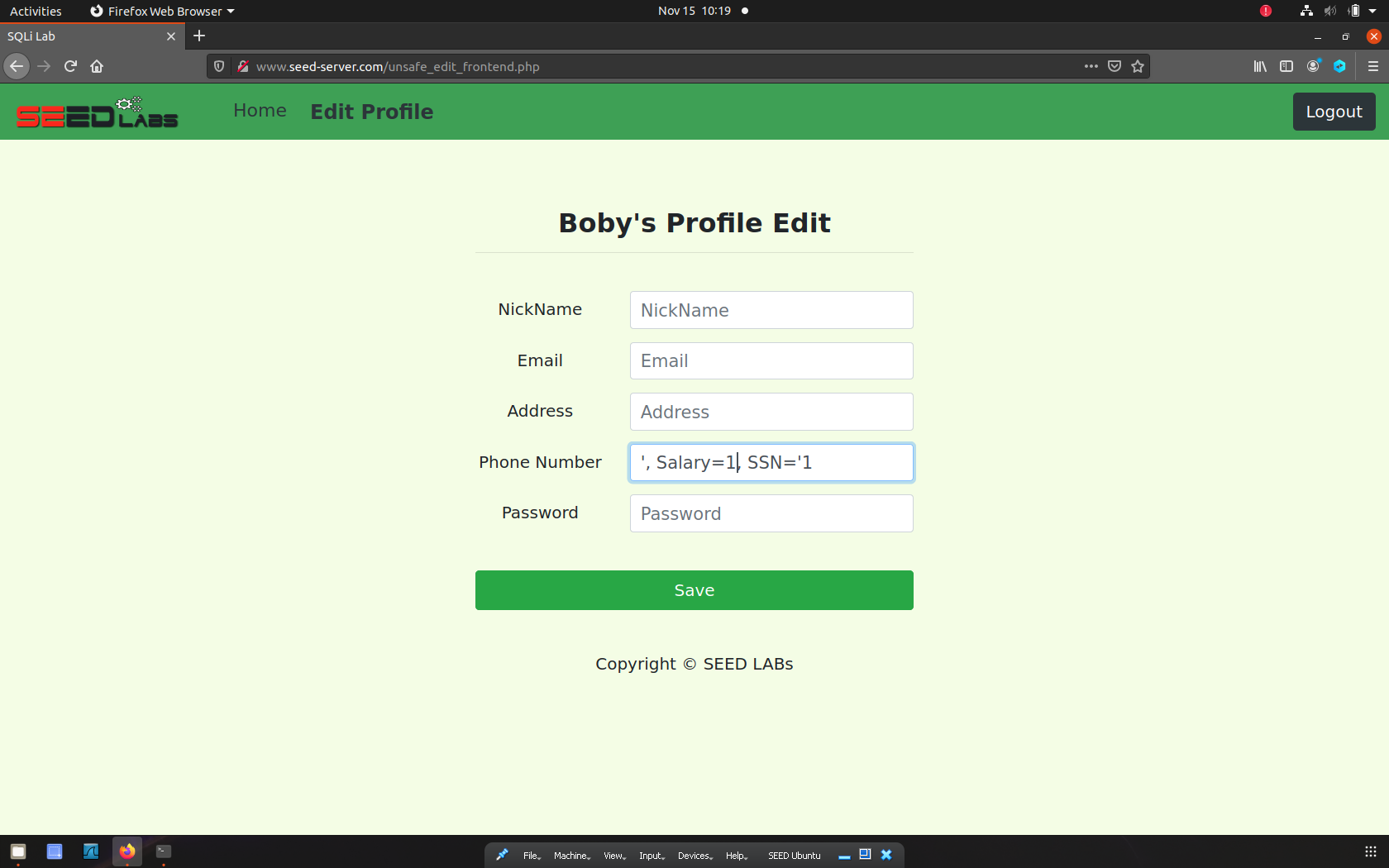
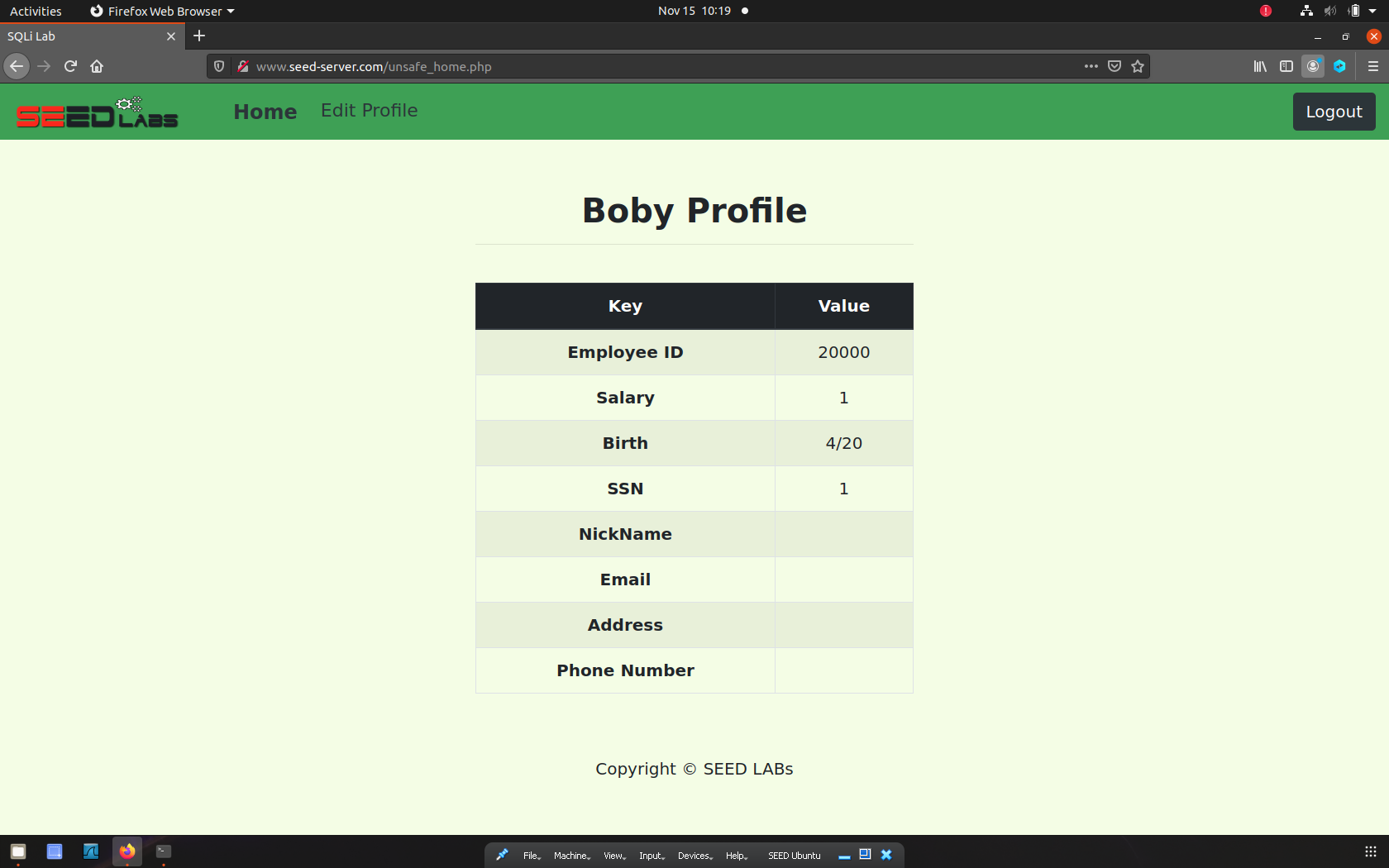
And save it.



As you can see, Alice’s salary is now increased.

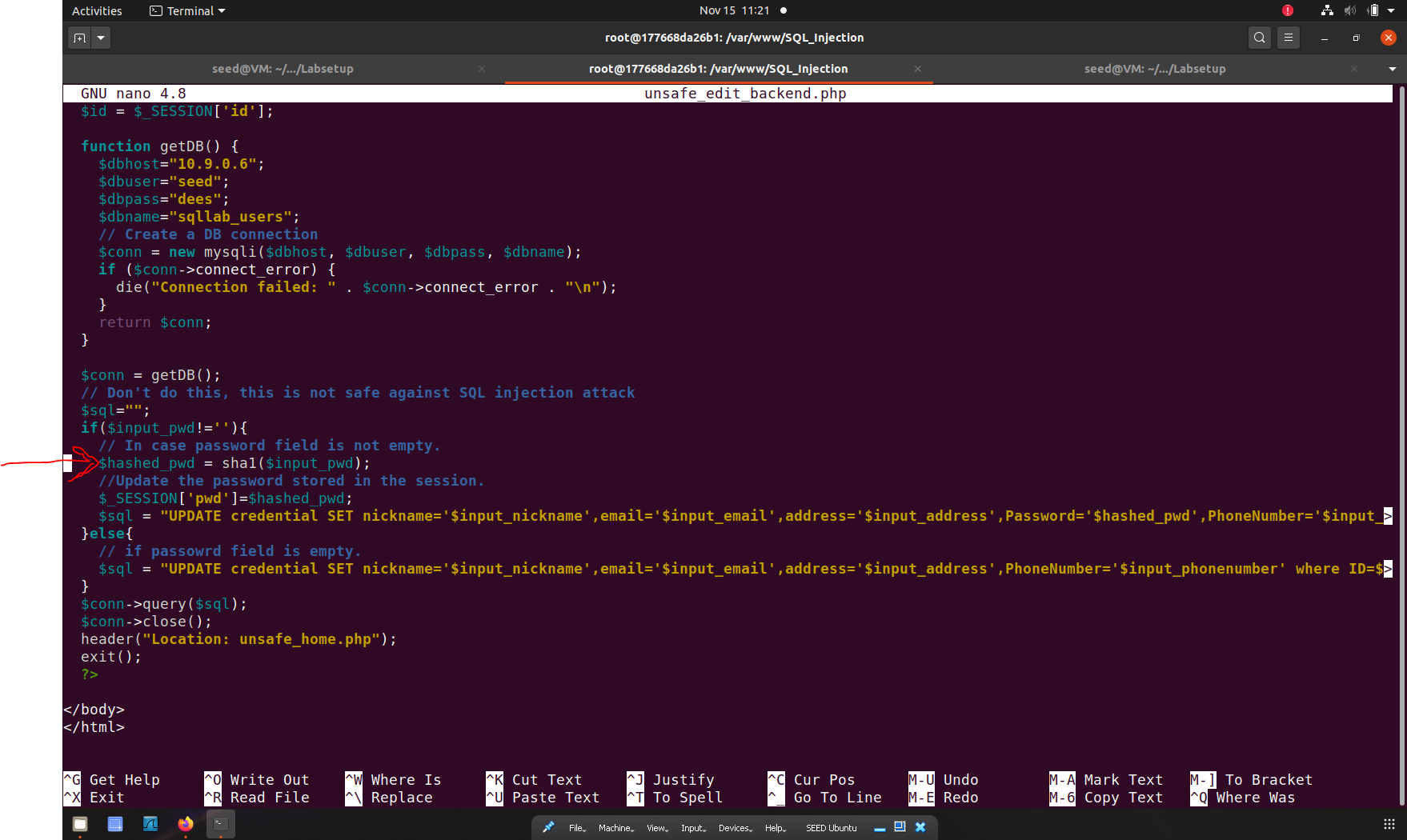
## Task 3.2: Modify other people’ salary.

Now we will log in as Boby and do the same.

## Task 3.3: Modify other people’ password.

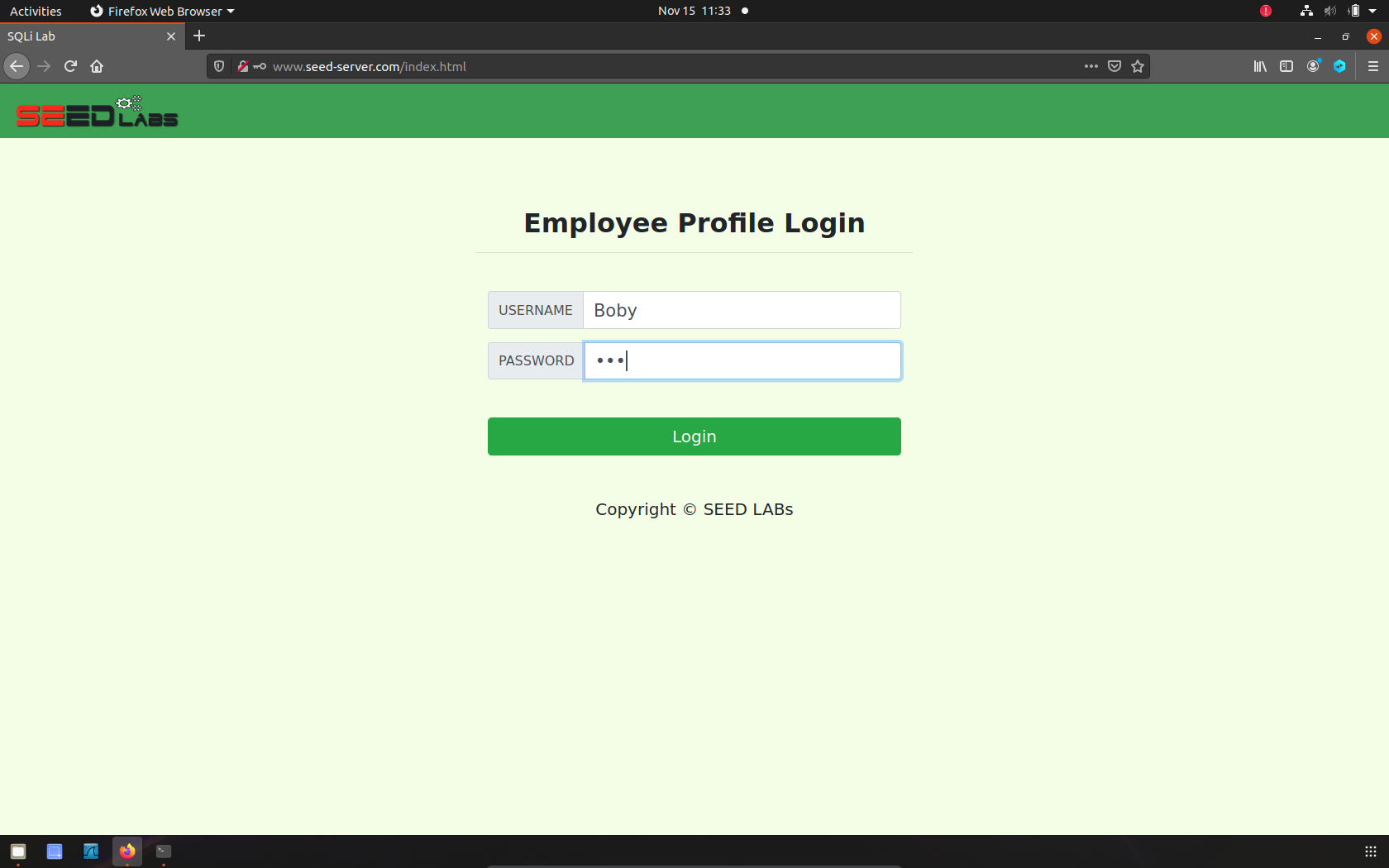
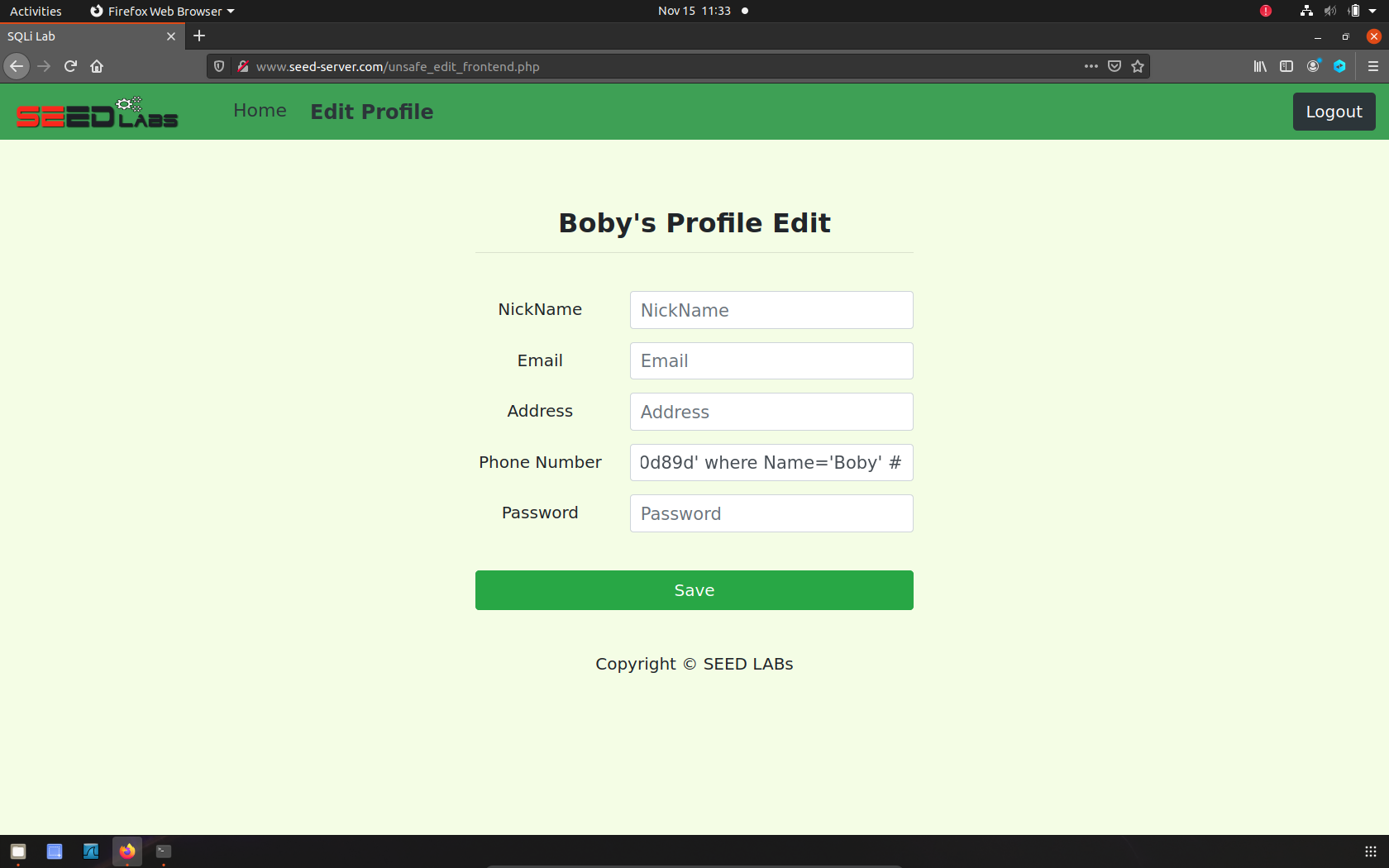
Since the password is not stored as is, but is actually hashed, we will need to find the hash value of the password we are going to set. We can see that SHA1 is being used.



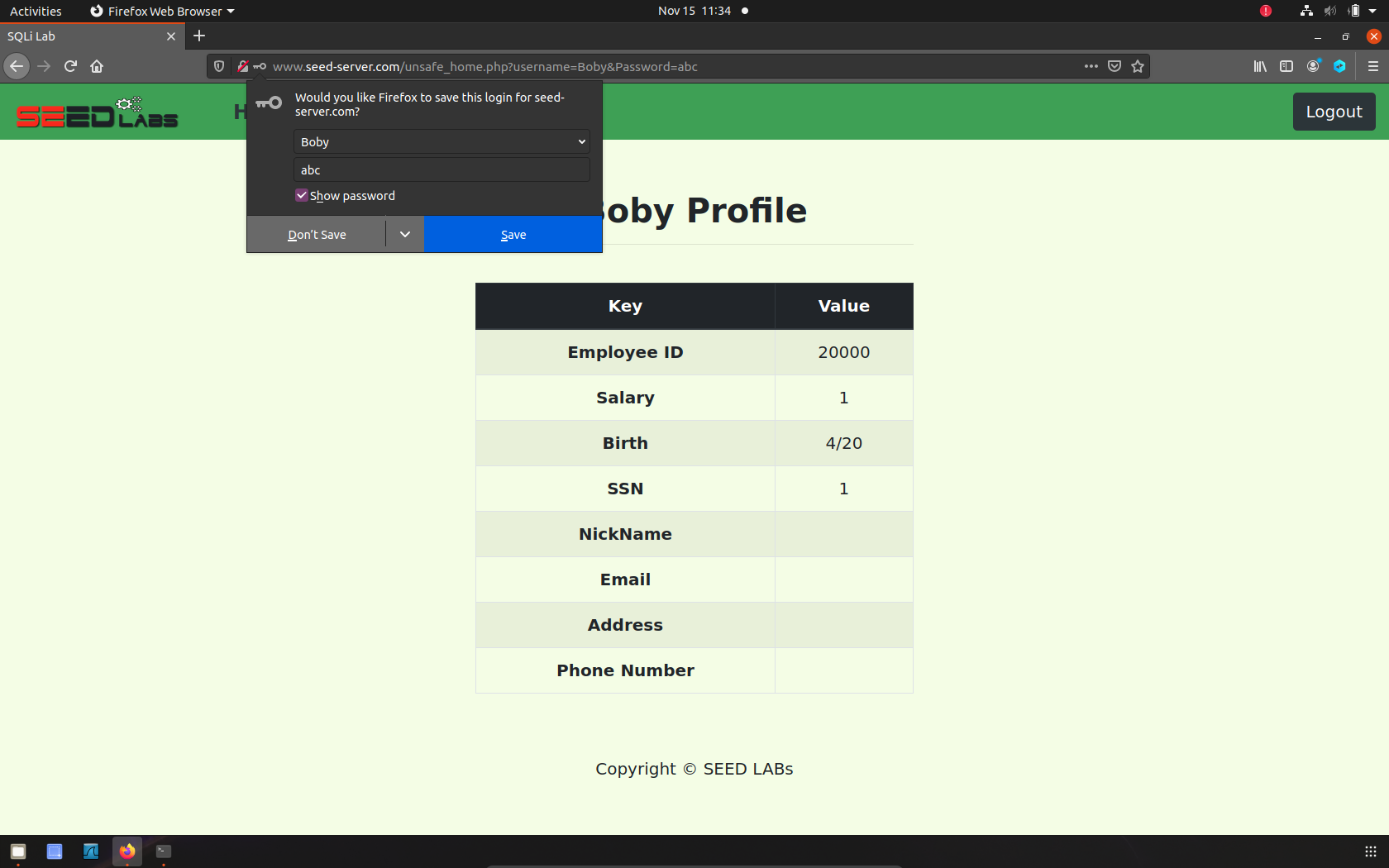
So we will use an online editor for finding the hash value. We want to set the password as abc. The hash equivalent is **a9993e364706816aba3e25717850c26c9cd0d89d.**

Now we will repeat the steps in the previous task and put this string in the box:

', password='a9993e364706816aba3e25717850c26c9cd0d89d' where Name='Boby' #

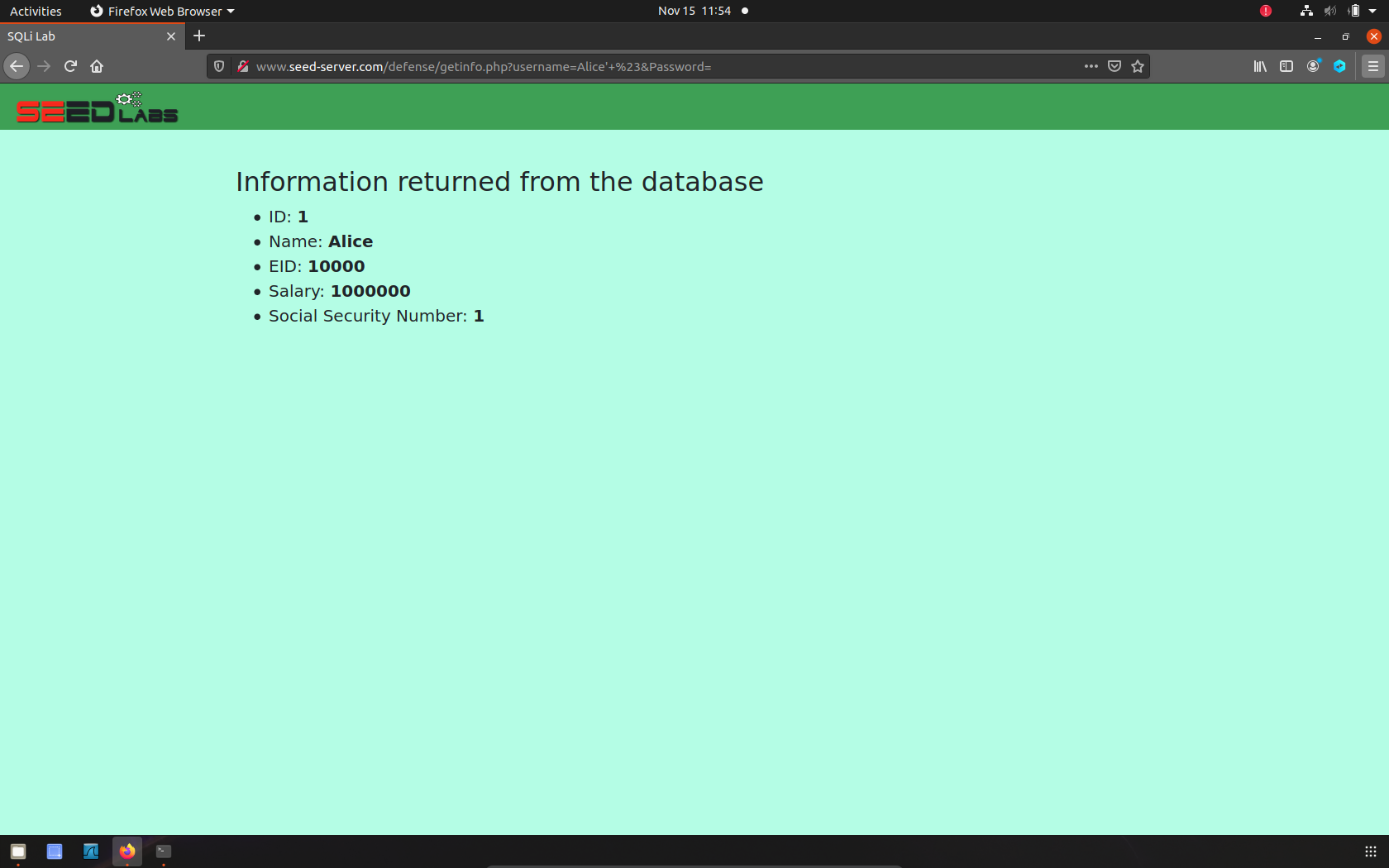
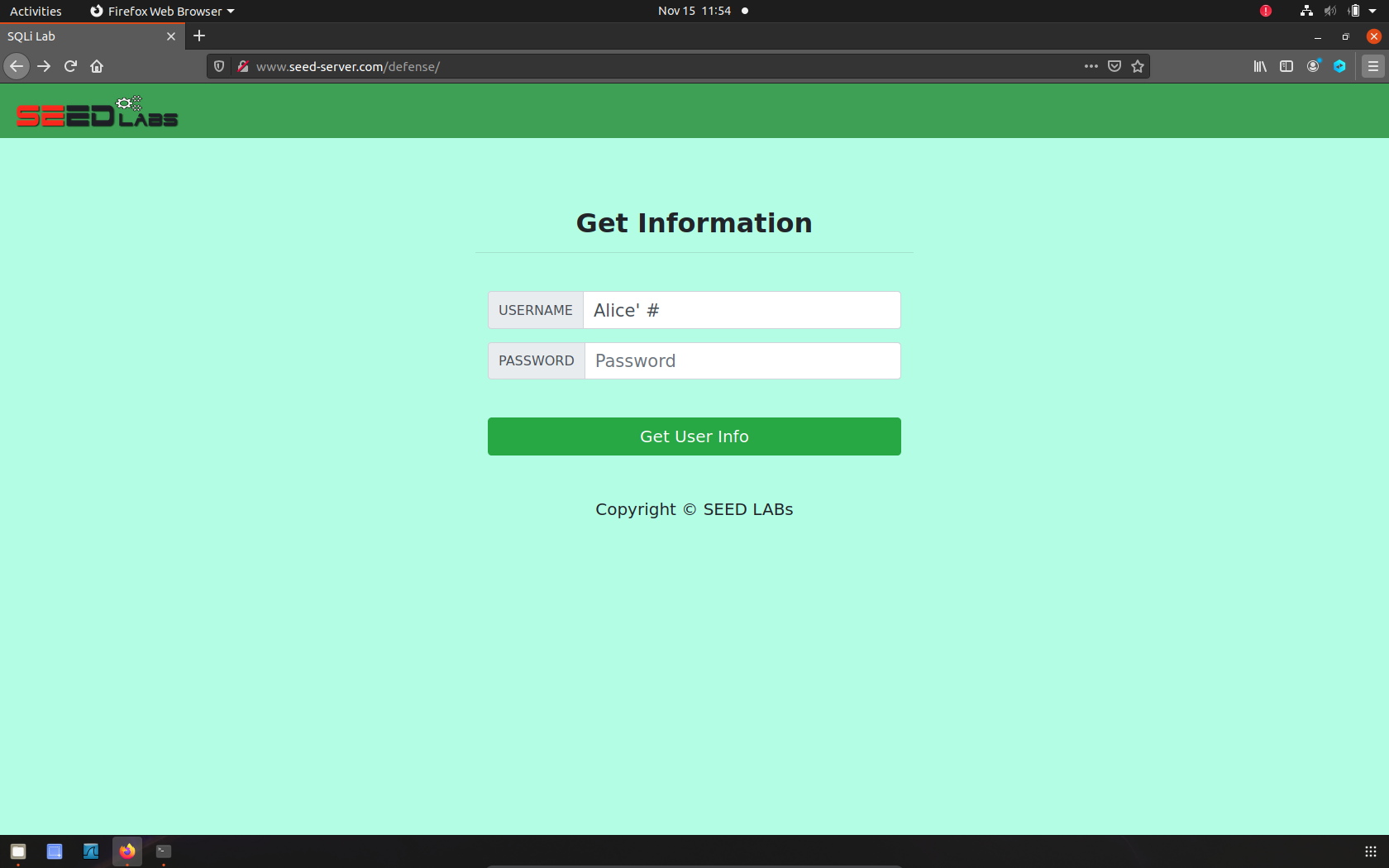


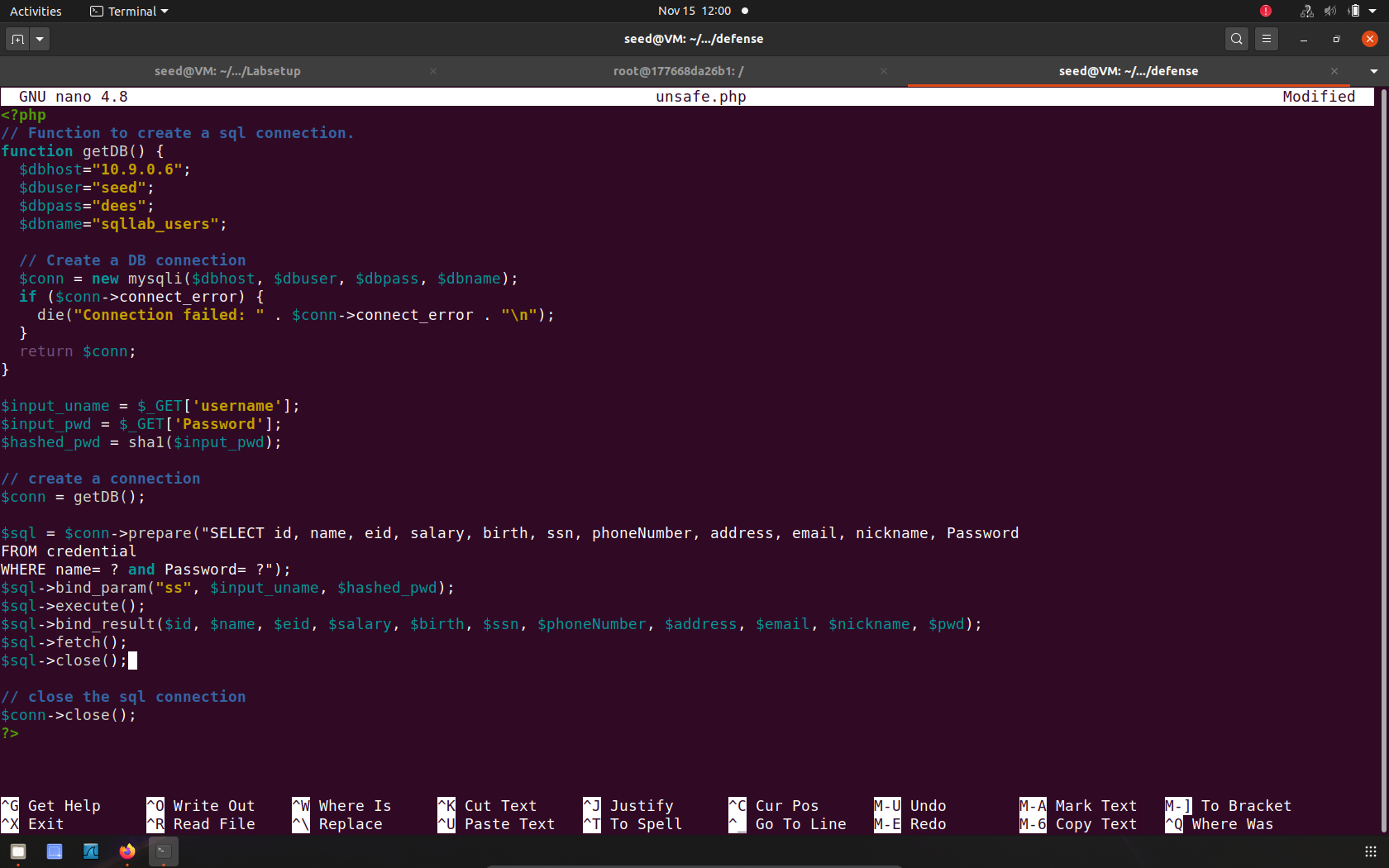
We put abc as the password and are able to login as shown below.

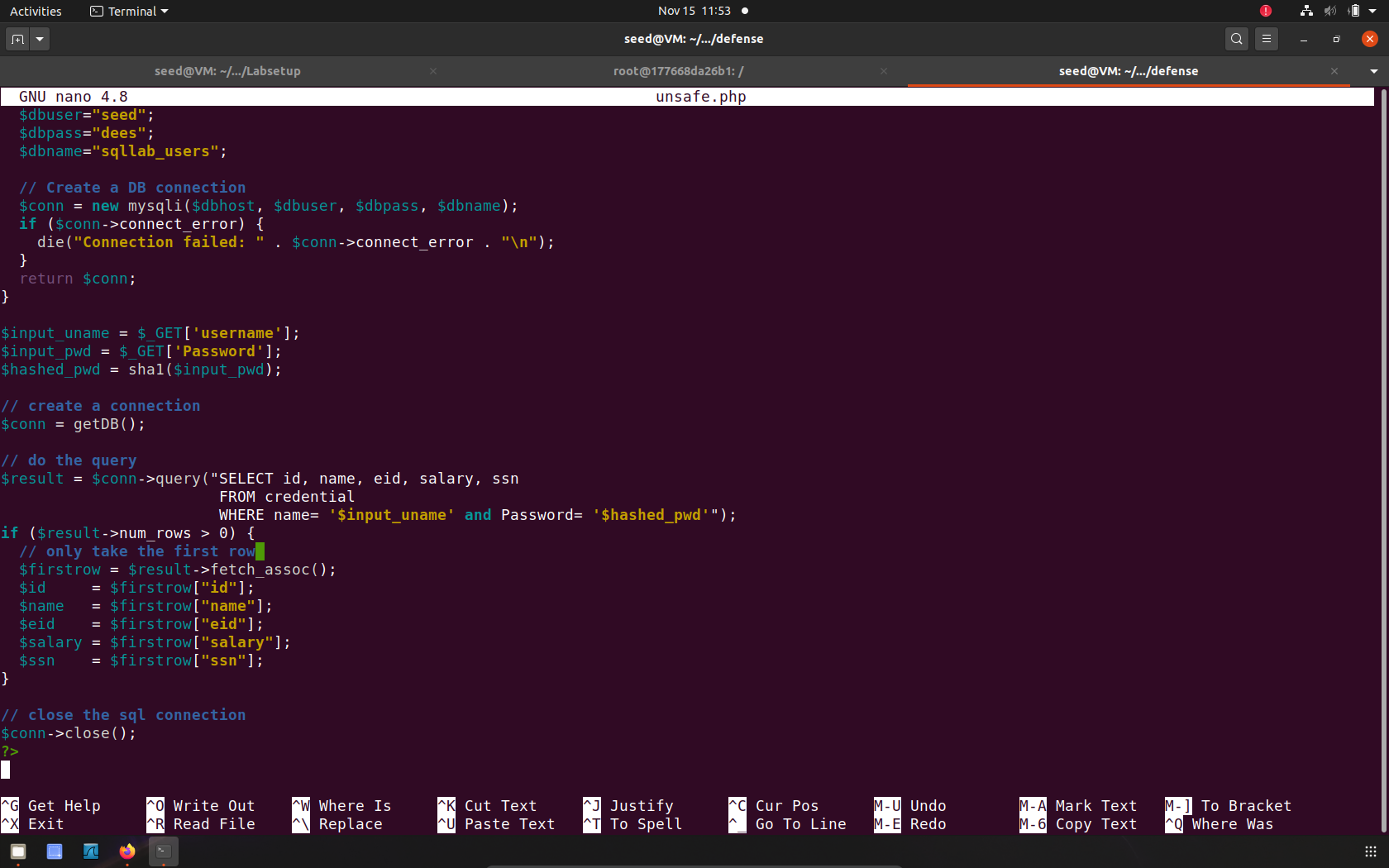


# Task 4: Countermeasure — Prepared Statement

Firstly, we can see what happens when we exploit the vulnerabilities in the web as follows:



So we the code from this:

To this

Now rebuild the container, and as you can see now that we enter Alice’ # it doesn’t give us the credentials anymore.

