**Advanced Web Development – “Sessions and Gem Lab”**

Points: 35

Due date on blackboard.

**Instructions:**

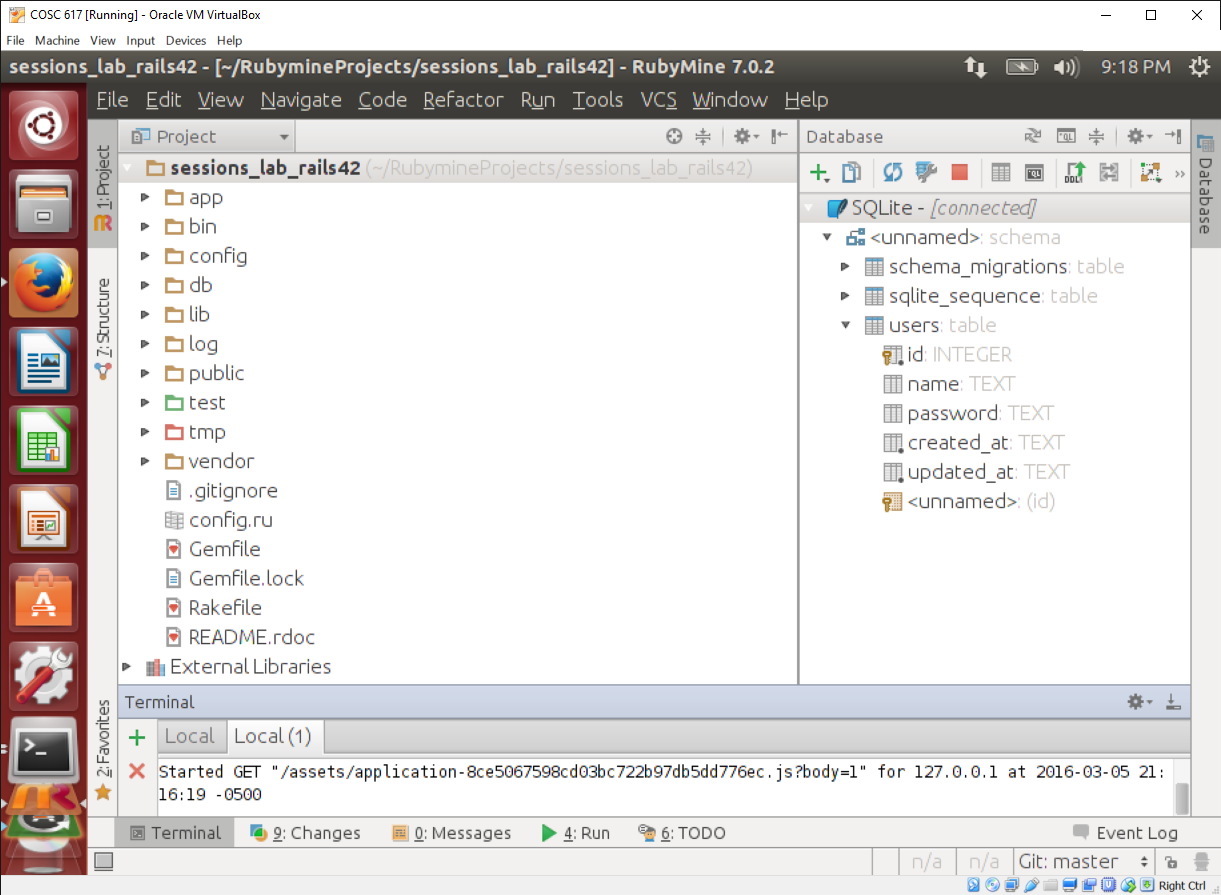
There are two aims of this lab:

1. Cement your knowledge on sessions (based on lecture).
2. Set up and use a Ruby gem.

# Sessions

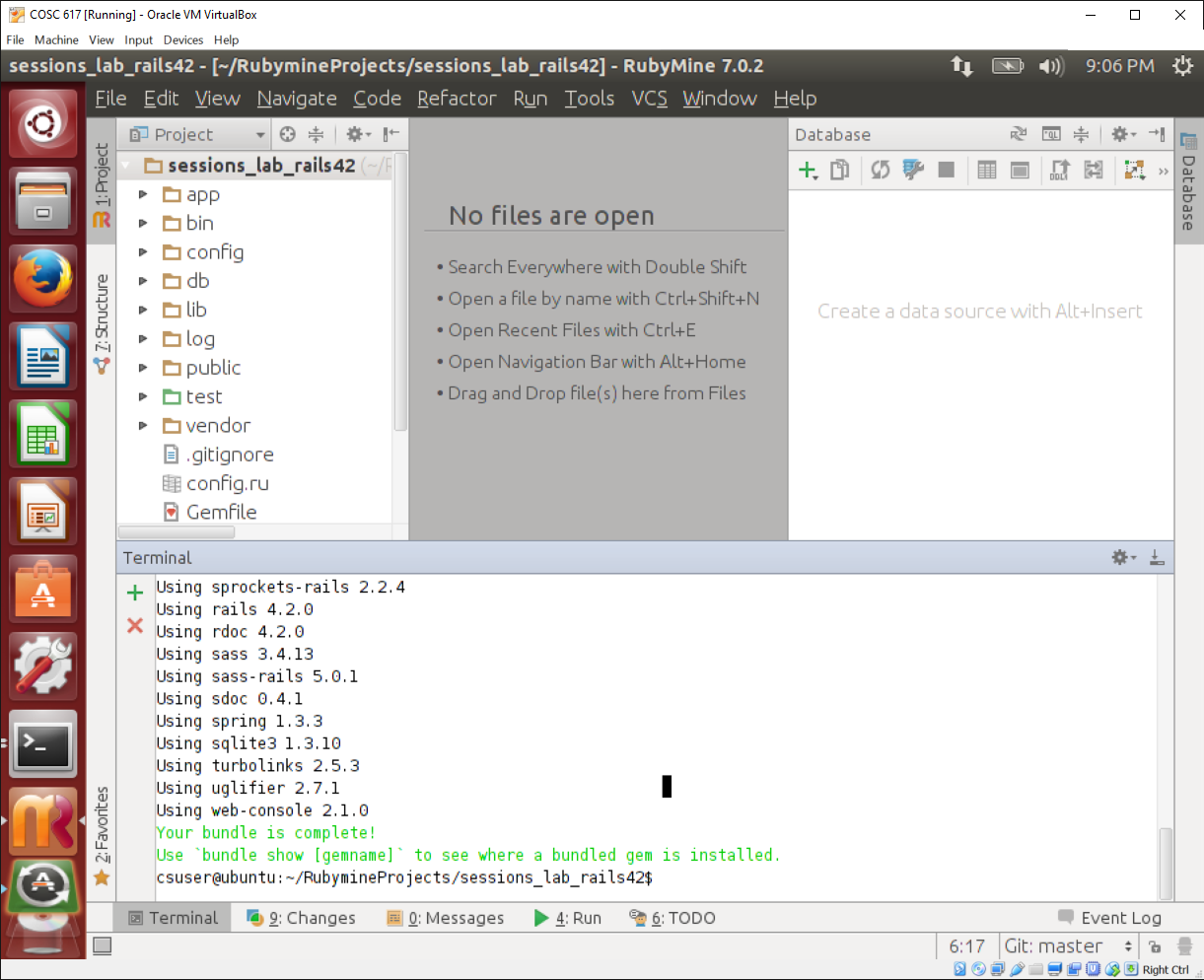
1. 5 points. Clone the code from:

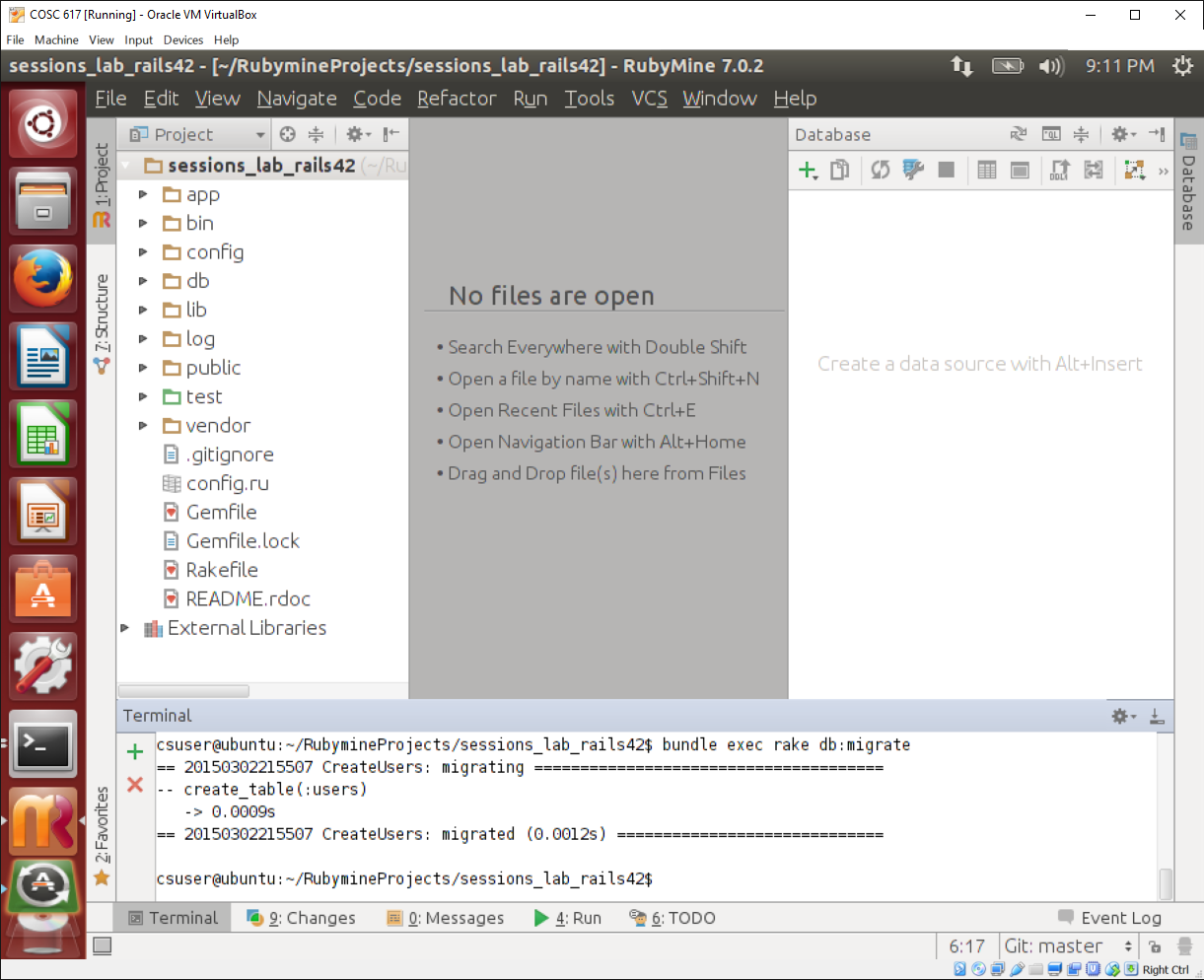
<https://github.com/siddharthkaza/sessions_lab_rails42.git>

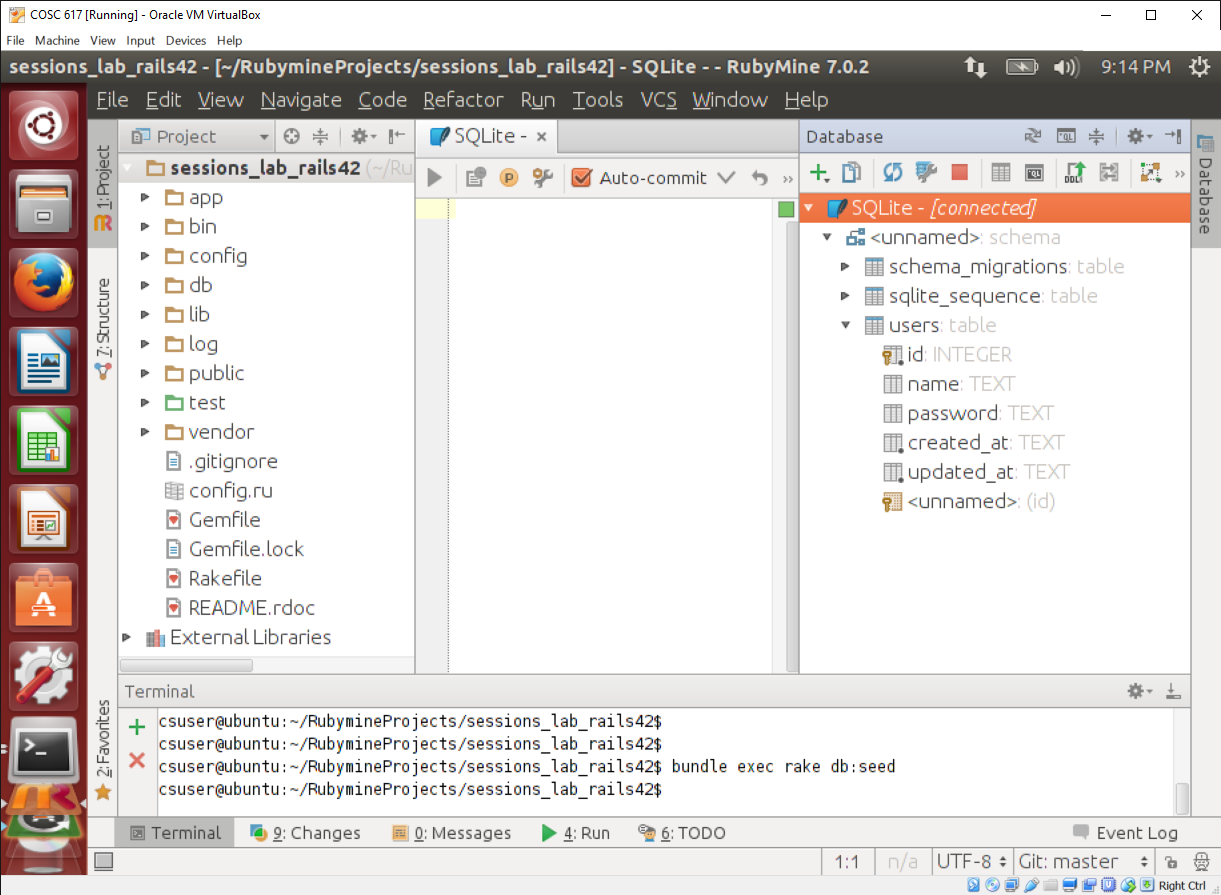


This app is a simple application that authenticates users using a username and password stored in the database. The aim is to edit the application to use a session to store the ‘id’ of the user logged in. When the ‘id’ is stored in the session in the browser, the user is considered ‘logged in.’ On logout, the ‘id’ is to be removed.

1. Run ‘bundle install’, ‘rake db:migrate’, and ‘rake db:seed’ to get the application running.







1. Navigate to localhost:3000/login. You should see a login form. Interact with it. The form is generated the ‘sessions’ controller ‘new’ action.



1. Study the sessions controller. It has three actions. What does each action do?

**The three actions for the sessions controller are new, create, and destroy. New will open the login form with the two fields empty for the user to fill out when someone opens the page. Create will check if the entered information is in the database. If both fields are correct, the program will open/navigate to the admin\_url. Otherwise, the user will be returned to the login\_url. Destroy will set the session key “user\_id” to nil which effectively logs out the user.**

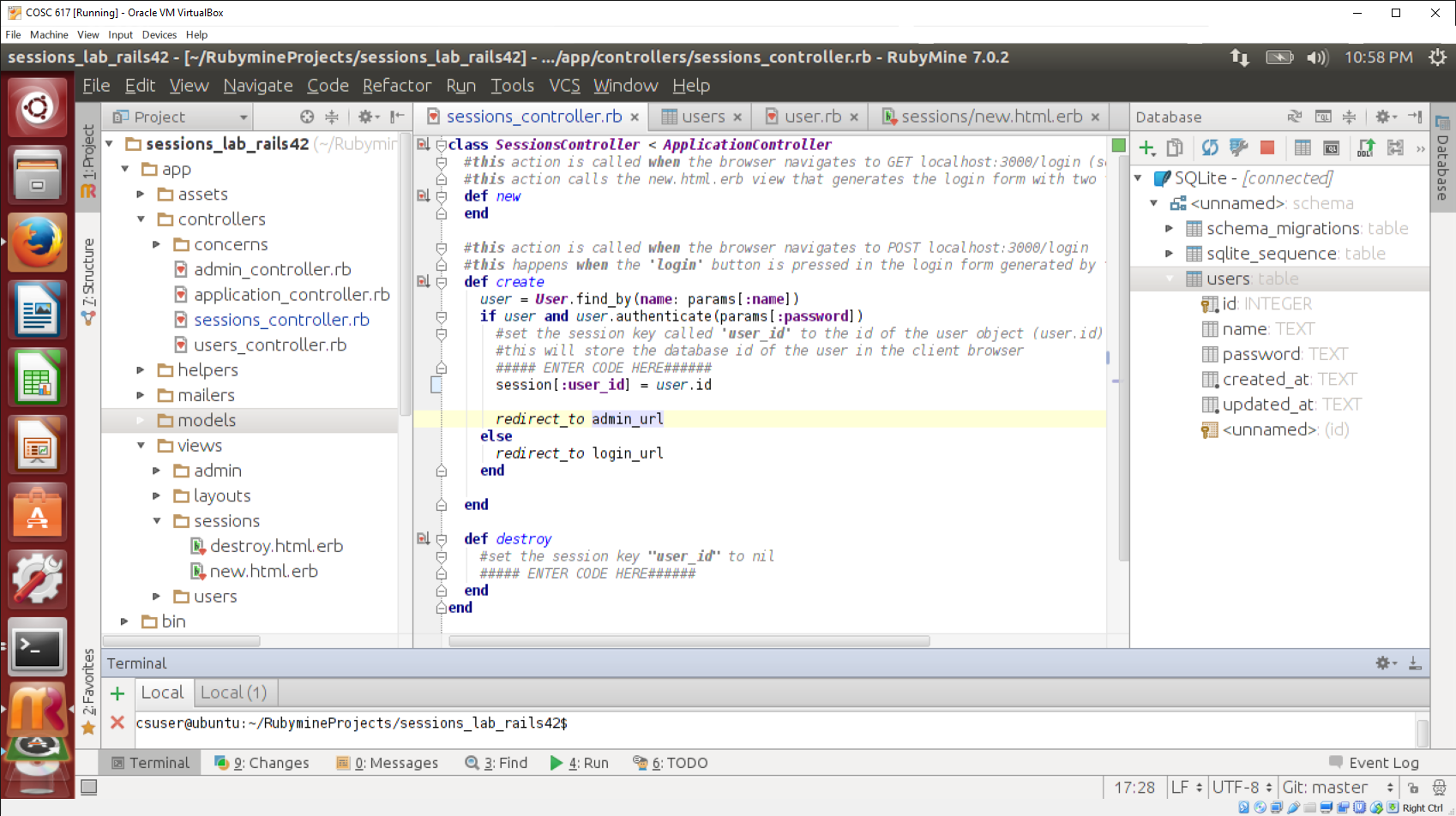
1. Study the ‘sessions#new’ view. What is the view creating? What do you think params[:name] does?

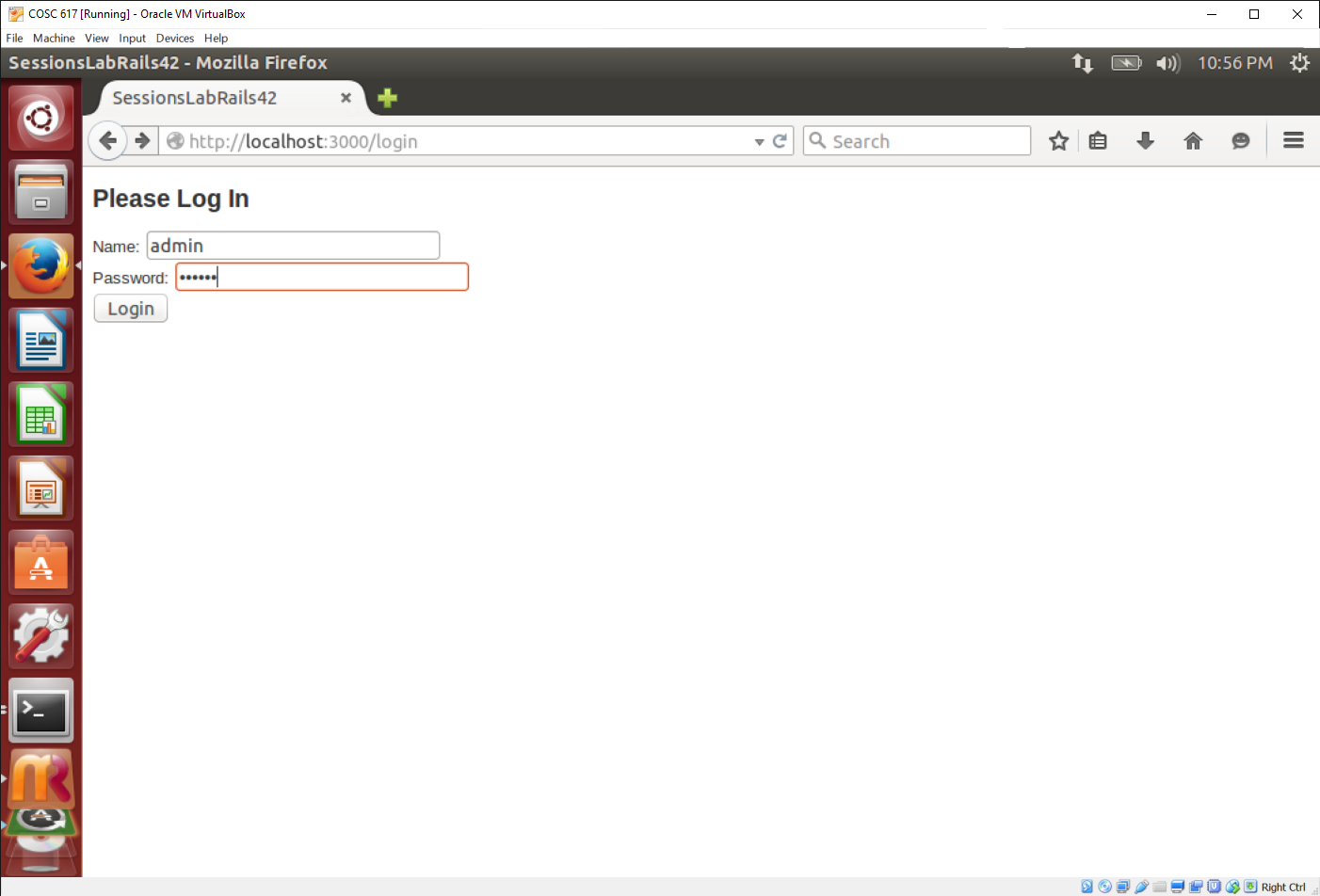
**The ‘sessions#new’ view is creating the login “Name” text field and “Password” password field along with a “Login” button. The “params[:name]” stores the value the user typed into the “Name” text field.**

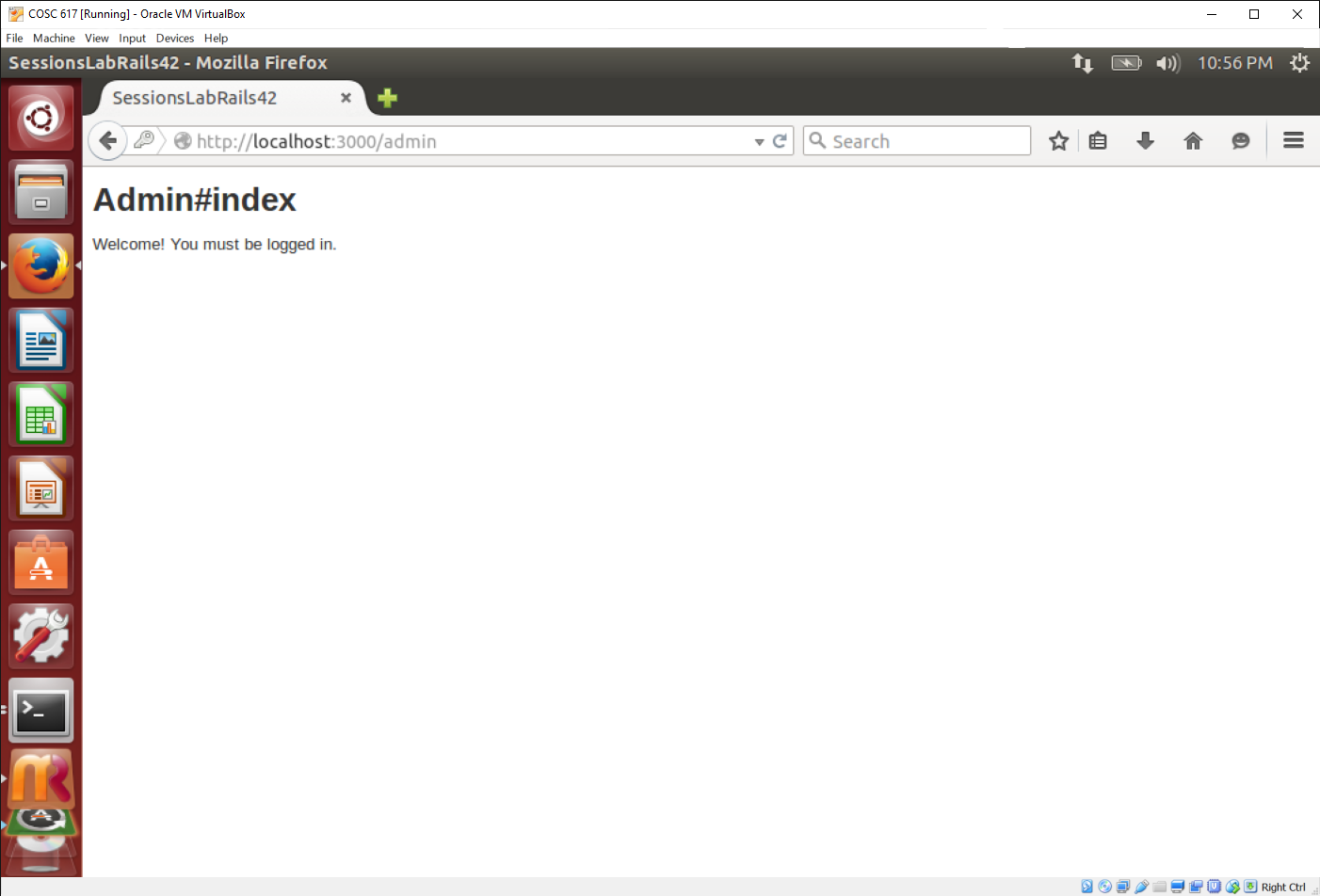
1. Notice that the ‘create’ action uses a method in the User model called ‘authenticate’. Study that method.

**The authenticate method is called on a user and checks the password passed in to the method against that users password from the database (if any).**

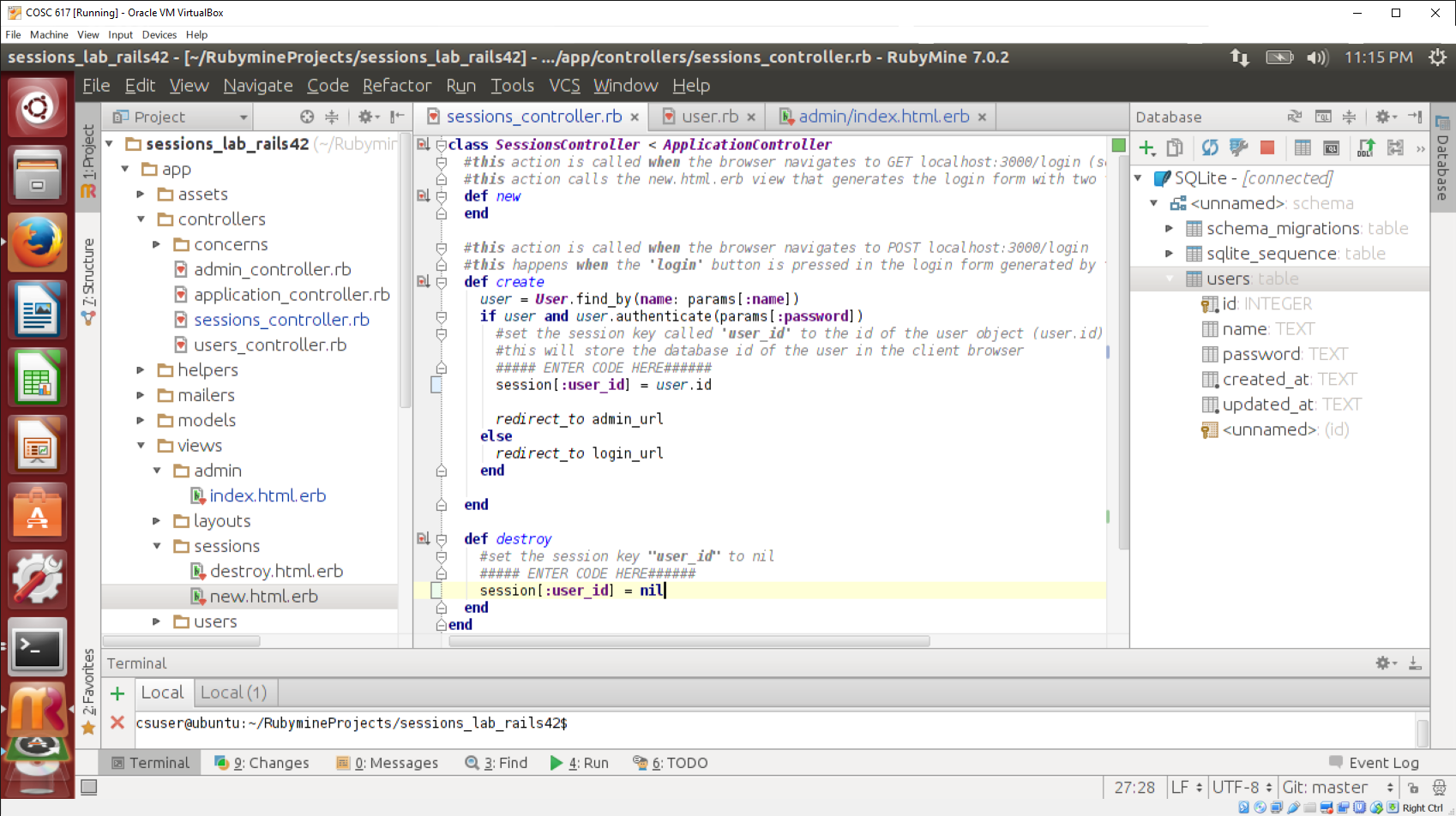
1. Now, back to the sessions#create method, study the comments and enter code to make the user login work.

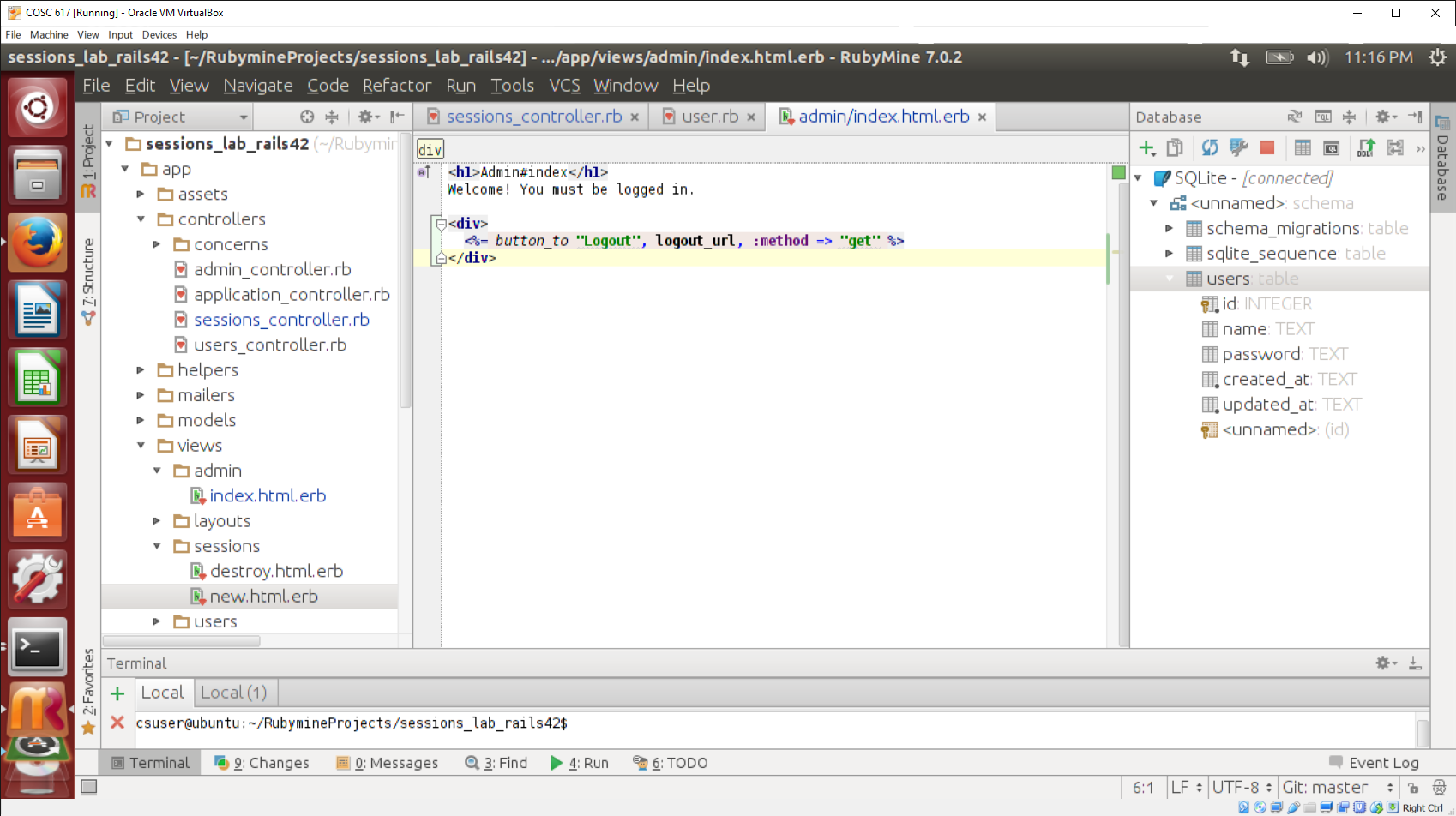




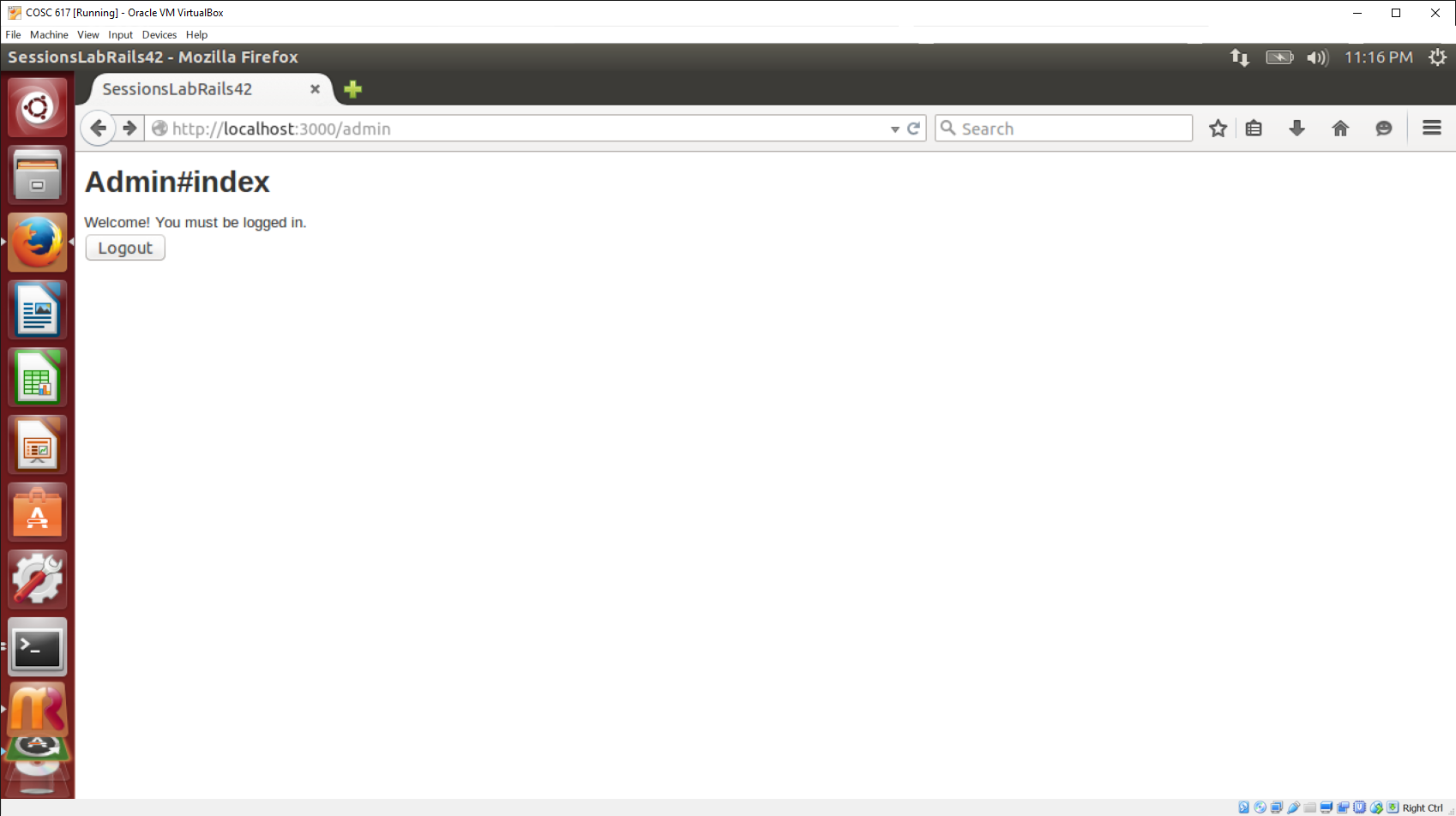


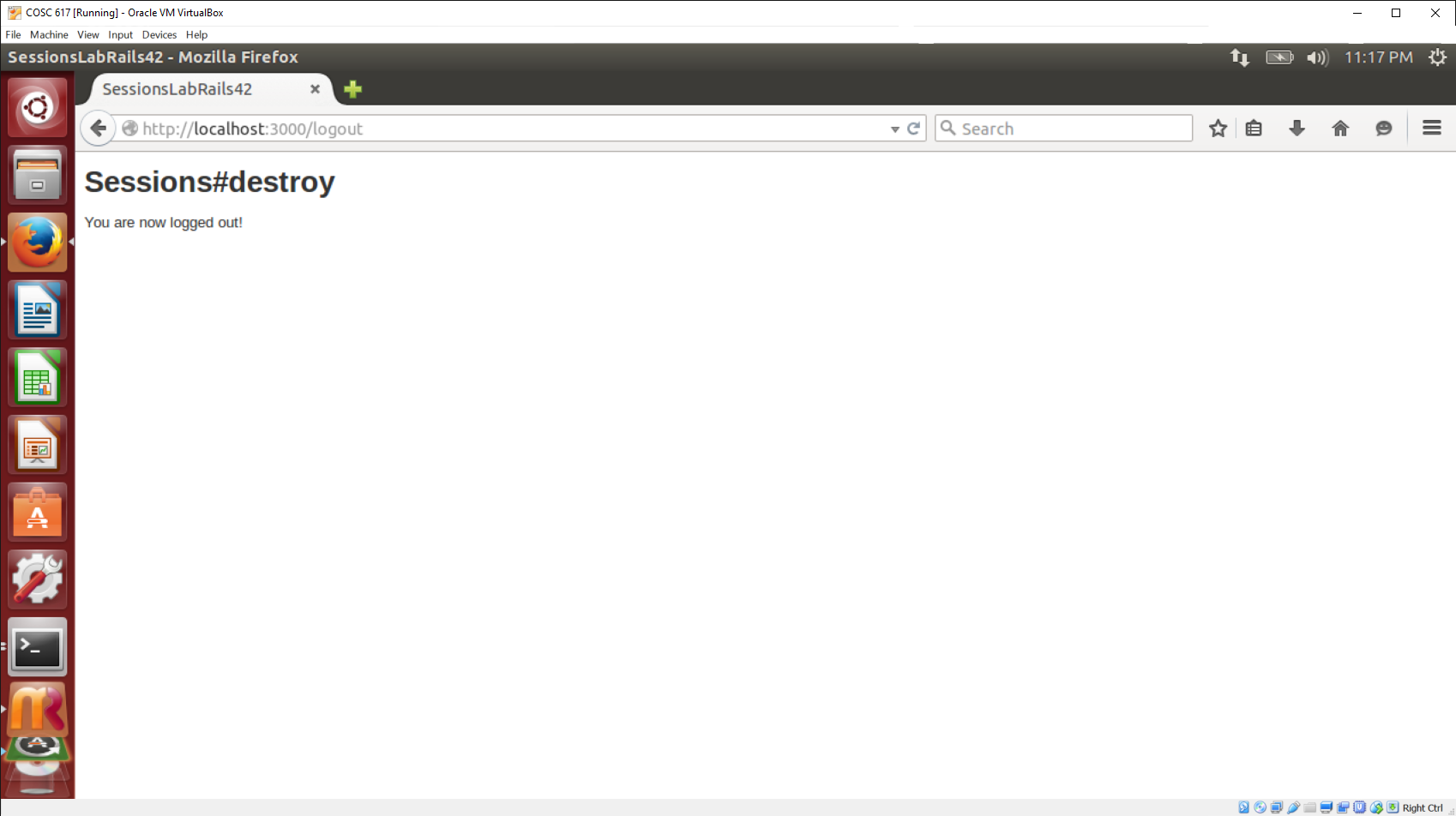
1. Complete the code in sessions#destroy





1. Navigate to ‘localhost:3000/login’ to test the code you wrote. Try the ‘logout’ action too.





1. 10 points. Modify the depot application, so the sidebar displays when the user last visited the page (date and time). This should only happen if the user visited within the last 120 days. Otherwise, it should state “Your last visited was too long ago.” Use client-side state for this.

# Gems

**What is a gem?**

Gems can be used to extend or modify functionality in Ruby applications. Commonly they’re used to distribute reusable functionality that is shared with other Rubyists for use in their applications and libraries. Some gems provide command line utilities to help automate tasks and speed up your work. (<http://guides.rubygems.org/>)

1. 10 points. Follow the tutorial for using the Devise gem below and submit a screen shot of the login screen generated.

Revised from <http://tech.pro/tutorial/1718/rails-4-authentication-using-devise> (now offline)

**Rails 4 Authentication using Devise.**

Today I'm going to give you a quick step-by-step guide on how to implement user authentication from scratch using [Devise](https://github.com/plataformatec/devise).

This entire tutorial can be completed in about 10 minutes and is very simple to follow.

Our end goal is to have users be able to register, log in and log out of the website. We'll also create a small partial view letting us know if we're logged in or out.

**Install the Devise gem.**

Open up your Gemfile and install the Devise gem.

gem 'devise'

Then in your terminal run the bundle install command to install the gem.

$ bundle install

**Run some Devise generators to set up the initial configurations.**

Run this command from your terminal:

rails generate devise:install

This generator installs the initializer that configures all of Devise's available settings.

**Generate your User model.**

Next we need to generate our User model. I'm going to name it User but you can name it whatever you like, just replace User with Whatever.

rails generate devise User

rake db:migrate

**Configure your default URL option for Development.rb**

Inside of config/environments/development.rb, set the Action Mailer's default URL to localhost:

config.action\_mailer.default\_url\_options = { :host => 'localhost:3000' }

**Make sure you have a root route declared in Routes.rb**

You need to make sure that routes.rb has a default root route - if you don't have one, set it! Create the ‘home’ controller and ‘index’ view first

root to: 'home#index'

**Create a partial view to see if we're logged in or not.**

Inside of your views/layouts folder create a file named \_user\_widget.erb and copy this code in:

<% if user\_signed\_in? %>

<p>Welcome <%= current\_user.email %></p>

<%= link\_to 'Logged In [click to logout]', destroy\_user\_session\_path, :method => :delete %>

<% else %>

<p>You are not signed in.</p>

<%= link\_to 'Login', new\_user\_session\_path %>

<% end %>

And invoke it within your layout (views/layouts/application.html.erb):

<!DOCTYPE html>

<html>

<head>

<title>FacebookAuthTest</title>

<%= stylesheet\_link\_tag "application", media: "all" %>

<%= javascript\_include\_tag "application" %>

<%= csrf\_meta\_tags %>

</head>

<body>

<p class="notice"><%= notice %></p>

<p class="alert"><%= alert %></p>

<%= yield %>

<%= render 'layouts/user\_widget' %>

</body>

</html>

**Make sure you stop and restart the server otherwise you will find all sorts of nasty bugs!** It's always best to restart your local server when you update your gemfile or change anything in the environment configuration file.

With all this in place, you should be able to sign up, log in and log out from your very own Rails website.

1. 10 points. There are several sites that list collections of useful gems on the web. Some are listed under ‘rails resources’ on blackboard. There is one I like: <https://github.com/hothero/awesome-rails-gem> . The site lists several gems classified by functionality.
2. You have the stories of your project ready now, go through the list and find three gems that you can use to implement your project’s functionality. List them here along with what they do.
3. Implement one of the gems. You can use it in a rails app you write from scratch, or fit it into the addressbook, depot, or any of the applications you wrote for the previous assignments.