**Student Authentication Using Devise**

Joshua Cervantes

[Before Installing](#BeforeInstalling)

[Installing Devise](#Installing)

[Next](#Next)

[Model](#Model)

[Server](#Server)

[Configurations](#Configurations)

[Views](#Views)

[Devise Views](#DeviseV)

[Example](#Example)

[Models](#Models)

[Controller](#Controller)

[Redirection](#Redirection)

[View Link](#ViewL)

**Before Installing:**

To start, we need to make sure a few things are done before installing devise to a project

* Set up a development branch
* Stop the server if it is running
* Delete the data from a previous database using:

**rails db:truncate\_all**

* Then run a custom migration for a model for your own app:

**rails generate migration [Enter your model info]**

**Installing Devise:**

* To add devise to your project, you’ll need to add the gem to your project. Go to your gem file and add this line anywhere:

**# User authentication https://github.com/heartcombo/devise (This is a link to a GitHub repository that will help you understand devise)**

**gem "devise"**

* Then go ahead and run “**bundle install**” within your rail’s app.

**Next,**

* Run a generator for the model by running this command:

**rails generate devise:install**

* Once that is installed, go to config/environments/development.rb, and make sure the default URL options for the Devise mailer is configured to your local machine and port from docker. Add this line:

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

* When that is done, you’ll be able to add Devise to any of your models using the generator that was installed.

**Model**

* To generate and create a model that is configured with default Devise modules, run this command:

**rails generate devise [Your name for a model]**

-This command will also configure your routes file to point to the Devise controller.

-Then run **rails db:migrate** after checking any other configuration options you may want to add.

**Server**

* Restart your app and server in order for everything to update with Devise and configurations
* You can run the server again to see what changed:

**rails server -b 0.0.0.0**

**Configurations:**

**Views**

To configure views, you have two options:

1. General views: **rails generate devise:views**
2. Scoped views: **rails generate devise:views [Model name]**

The second option will allow you to customize a model’s views and add other required fields based on your implementation if your web app.

**Devise views**

Make sure your new views include “form” as “f”. For example if your previous view/forms file had “form.label”, make sure your devise/view file has form as f, so it’ll look like this “f.label”

**Example:**

<div>

   <%= form.label :first\_name, style: "display: block" %>

   <%= form.text\_field :first\_name %>

 </div>

Becomes:

<div>

   <%= f.label :first\_name, style: "display: block" %>

   <%= f.text\_field :first\_name %>

 </div>

**Models**

Make sure you update any parameters you may have within your model and devise parameters:

A screenshot of a computer program

Description automatically generated

In this example above, we all parameters in blue that we want to authenticate with devise.

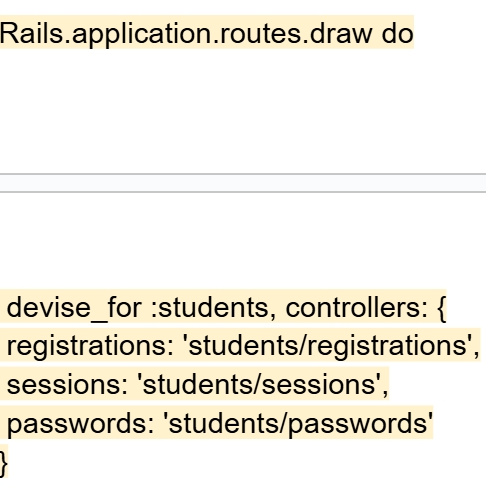
**Controller**

Generate a controller with devise with this command:

**rails generate devise:controllers [Name of your model]**

Next,

Configure and update your routes within config/routes.rb to find a models devise form. For example, this route will be updated with a student model that finds registrations forms:



Doing this will allow you to add authentication to your model page!

**Redirection:**

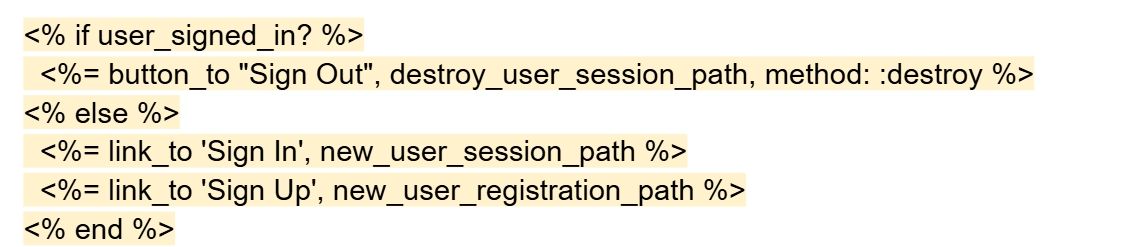
Within app/controllers/application\_controller.rb, add a method to redirect to a model’s main page after authentication. This example below will redirect to student’s profile page after authenticating:

A text on a white background

Description automatically generated

**View Links**

Using the student example and attributes from above. If you want to add a sign up, sign out or sign in link to your web application, you’re going to want to add this within your views:



Then add this to app/views/layouts/application.html.erb to make sure it works within your application:

A close-up of text

Description automatically generated

This will help you when you want to create a dropdown menu.