Abinet Kenore
CS390P
Fall 2018

This is a continuation of HW1 and in this homework I was asked to implement a many to many relationship between the students, and sections in order to get a full functionality of the registration banner by either creating a enrollment table which joins sections and students or or by creating a simple joint table between students and section (SectionsStudents). For me the easy was creating a joint table us the rails g model SectionsStudents section:references student:references --force-plural. Adding a validation to the project. Up on reading this paper you grasp a solid understanding of how many to many relational table works in a database system. When I did this project the versions of ruby and rails was ruby -v ==➔ ruby 2.5.1p57 (2018-03-29), and rails -v Rails 5.2.1. when you are running this application you may need to check the versions you have and you might be prompted to update your versions and in most cases "Bundle install."

```ruby
#.../app/models/student.rb
class Student < ApplicationRecord
    has_and_belongs_to_many :sections
    validates :name, presence: true
    validates :name, uniqueness: true
end
#...app/models/sections_students.rb
class SectionsStudents < ApplicationRecord
    belongs_to :section
    belongs_to :student
  end
  #...app/models/sections.rb
  class Section < ApplicationRecord
    has_and_belongs_to_many :students
    belongs_to :course
  end
  #...app/models/courses.rb
  class Course < ApplicationRecord
    has_many :sections
    validates_presence_of :name, :number,
     :dept, :crHr
    validates_numericality_of :crHr
end
```

The Many to Many Relationship and Adding a validation In the models of student, section and course

Adding a search bar and autocompletion was a part of the project in order to make a search more effiecent and fast. And the following screenshot's will show how to add a code to the controllers, views, and routes of the application. Before that knowing where to add the methods is a key. For Example if you are adding a search bar to the students

1. Go to …/app/controllers/students_controller.rb =====> create a new def search and then add search function/code

```ruby
#.../app/controllers/students_controller.rb
autocomplete :student, :name, full_search: true

def search
  @students = Student.where("name like ?",
    "%#{params[:query]}%")
  render :index
end

# GET /students
```

2. Under the … app/views/students/index.html.erb add the following code

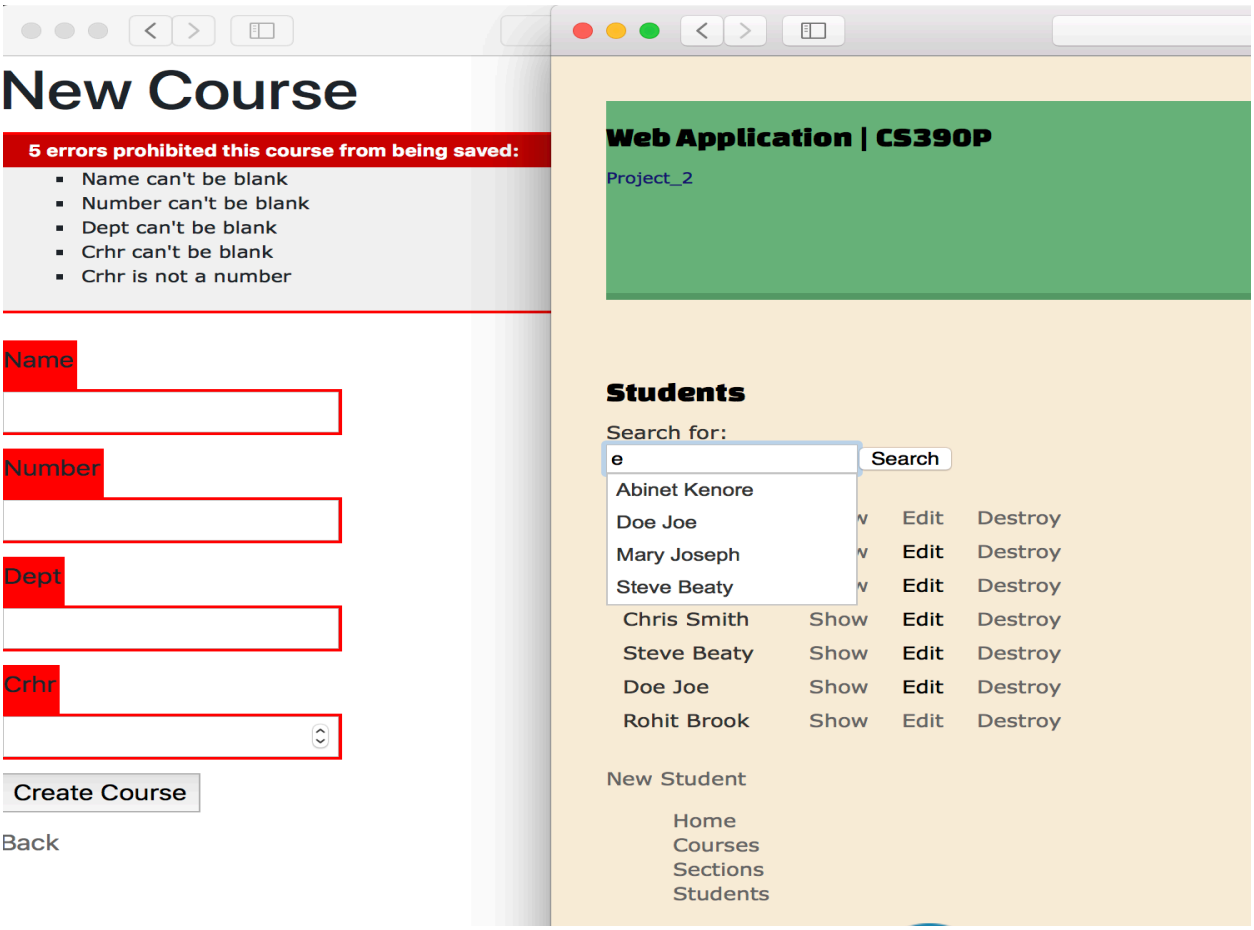
```
  </thead>
<!-- Autocomplete
.../app/views/students/index.html.erb-->

<%= form_tag(search_students_url, method: "get") do %>
<%= label_tag(:search, "Search for:") %>
<%= text_field_tag :search, params[:search], data:
{ autocomplete: autocomplete_student_name_students_path } %>
  <%= submit_tag("Search") %>
<% end %>
```

3. Under …. /config/routes.rb
   Here you need to make sure adding a key word search to call the method and autocompletion as shown in figure below

```ruby
#...config/routes.rb
resources :students do
  get :autocomplete_student_name, on: :collection
collection do
  get 'search'
end
end
```

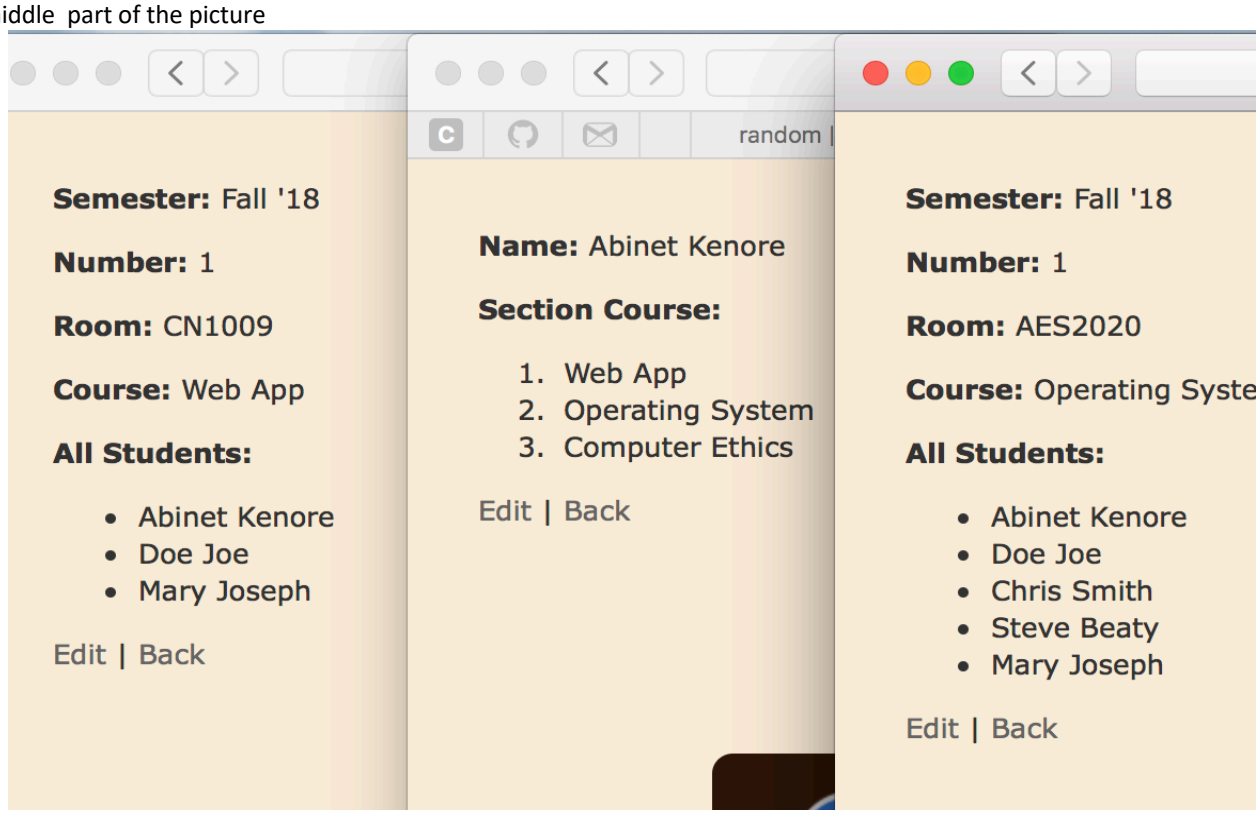
You can do the same thing for sections or courses.

The following is a demo of validation for course and search and autocompletion for student



The left side screenshot is to show that in order to create a course you must fill the required fields with the appropriate information and the picture on the right is to show you that how it is try to fill the form automatically.

After all has been done you can see a student "Abinet Kenore" under section number 1, and Room CN1009 for a Web App in addition to the other students under the same section for the same course. Also you can see that he registered for many courses in the middle part of the picture

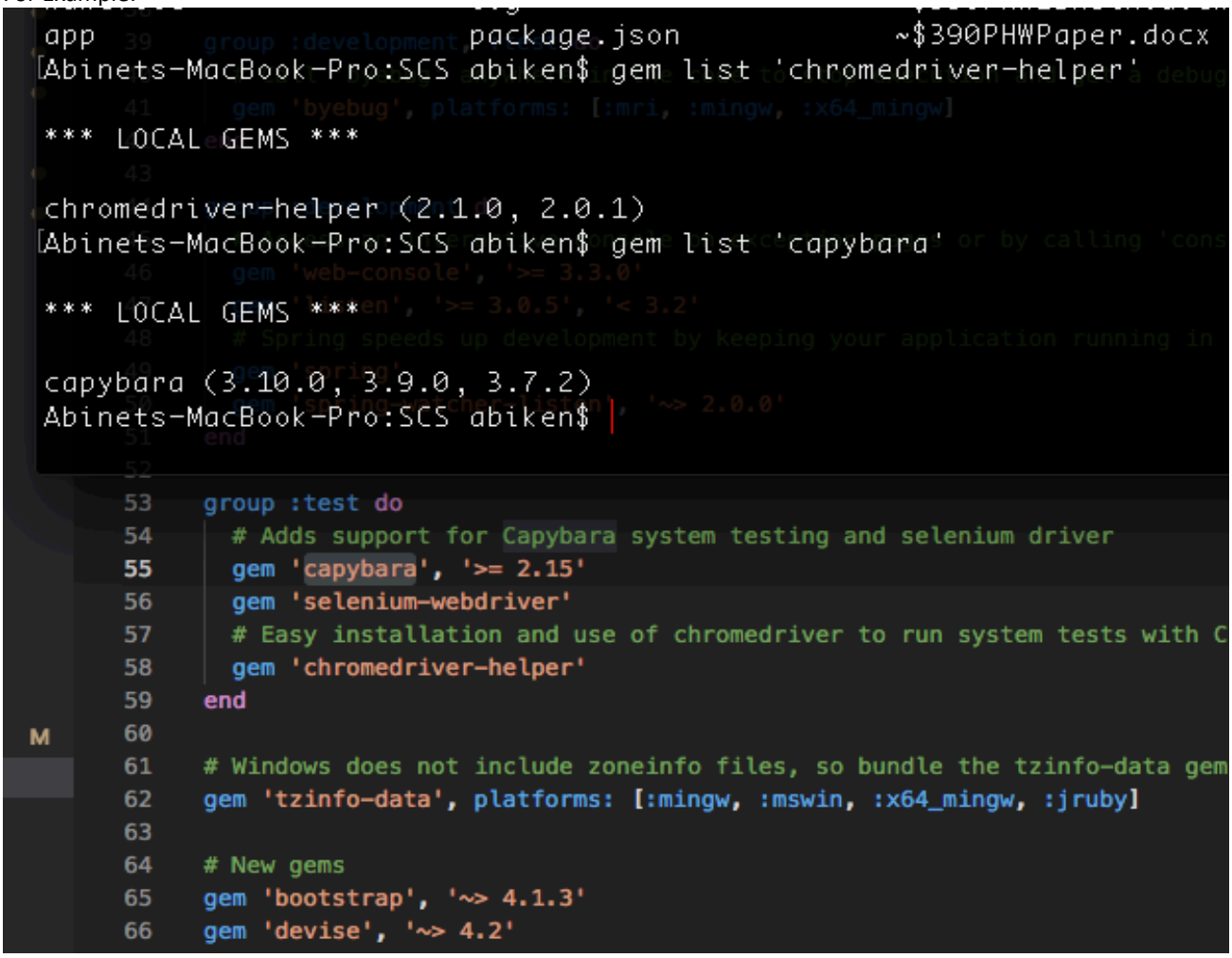


The following picture is a git difference before and after Adding a JavaScript for autocompletion in the files

```
3 HW2/SCS/app/assets/javascripts/application.js                    View
@@ -13,4 +13,7 @@
 //= require rails-ujs
 //= require activestorage
 //= require turbolinks
+//= require jquery
+//= require jquery-ui
+
 //= require_tree .

7 HW2/SCS/app/assets/javascripts/students.coffee                   View
@@ -1,3 +1,10 @@
 # Place all the behaviors and hooks related to the matching controller here.
 # All this logic will automatically be available in application.js.
 # You can use CoffeeScript in this file: http://coffeescript.org/
+#autocomplete
+
+$(document).on 'turbolinks:load', ->
+  $("input[data-autocomplete]").each ->
+    url = $(this).data('autocomplete')
+    $(this).autocomplete
+      source: url

6 HW2/SCS/app/assets/stylesheets/application.css                   View
@@ -10,6 +10,8 @@
  * files in this directory. Styles in this file should be added after the last require_* statement.
  * It is generally better to create a new file per style scope.
  *
- *= require_tree .
- *= require_self
+ *= require_tree .
+ *= require_self
+ *= require jquery-ui
+
  */
```

Writing a unit test was a part of the project which I went through a lot and ended unsuccessfully.
In order to write a test you will go and appropriate gem to the GemFile if it its not already there and then … /test folder and add tests. One thing which I would like to point you is how you can add a gem's to the Gemfile and how you will do the setip. On you terminal or command line run $ gem list 'name of the' gem and it it comes back with name of the gem you are looking for with the version number you are good get started writing a unit test.

For Example:

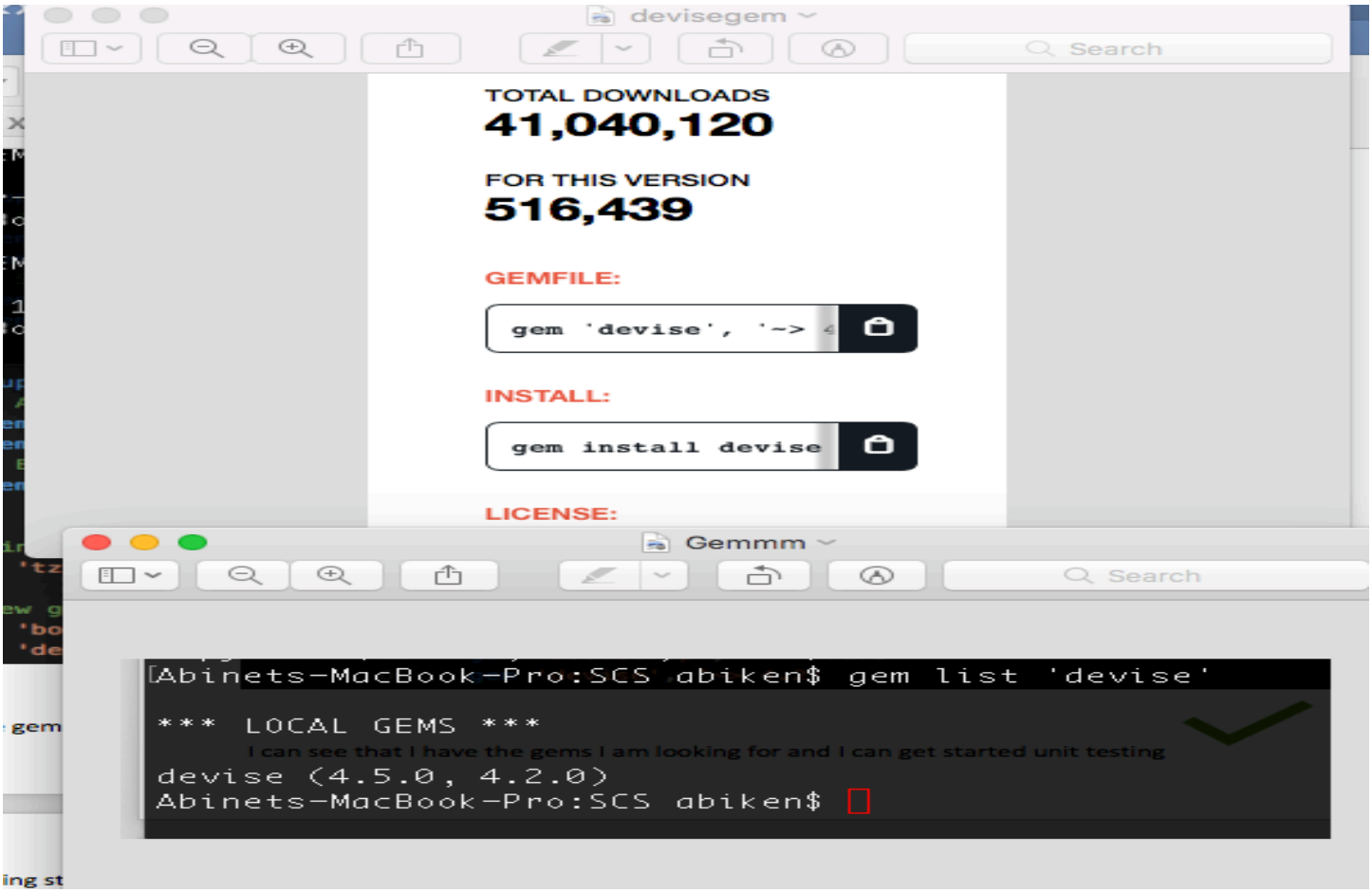


```
app                    package.json                ~$390PHWPaper.docx
Abinets-MacBook-Pro:SCS abiken$ gem list 'chromedriver-helper'

*** LOCAL GEMS ***

chromedriver-helper (2.1.0, 2.0.1)
Abinets-MacBook-Pro:SCS abiken$ gem list 'capybara'

*** LOCAL GEMS ***

capybara (3.10.0, 3.9.0, 3.7.2)
Abinets-MacBook-Pro:SCS abiken$

group :test do
  # Adds support for Capybara system testing and selenium driver
  gem 'capybara', '>= 2.15'
  gem 'selenium-webdriver'
  # Easy installation and use of chromedriver to run system tests with C
  gem 'chromedriver-helper'
end

# Windows does not include zoneinfo files, so bundle the tzinfo-data gem
gem 'tzinfo-data', platforms: [:mingw, :mswin, :x64_mingw, :jruby]

# New gems
gem 'bootstrap', '~> 4.1.3'
gem 'devise', '~> 4.2'
```

I can see that I have the gems I am looking for and I can get started unit testing

Unless follow the following steps.
Go to the https://rubygems.org/gems and search for the gem you need
For Example I can search for the gem ‘devise’
Copy gem' from the above web site and paste gem 'devise', '~> 4.5' under Gemfile
And the go to your terminal and run bundle install.
And check it if you have it by running a gem list ‘devise’

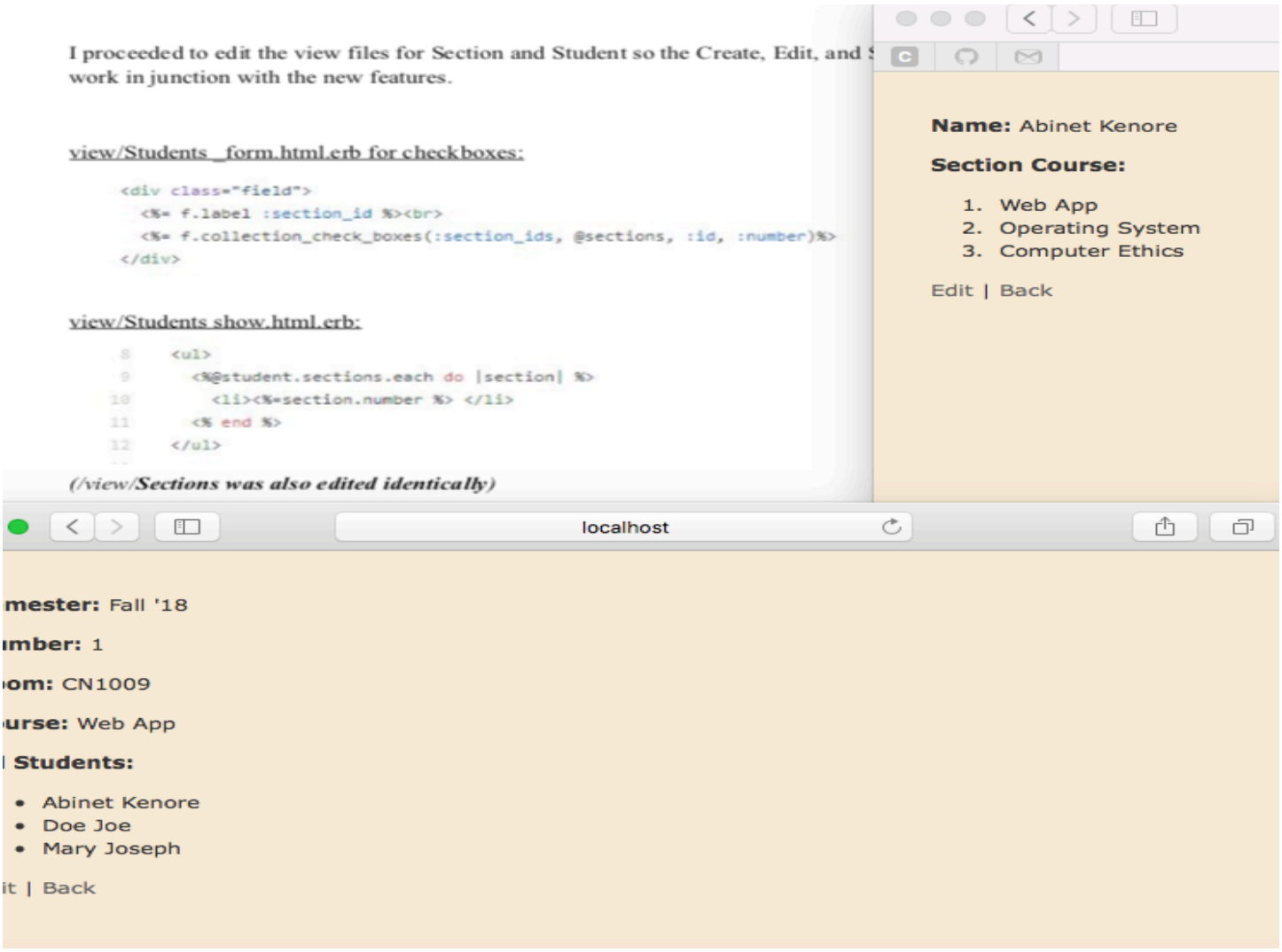
Note That I liked to install a gem devise not to write a unit test but to show you how you can add any gem to the gem file.
Also if you are wondering to know why I installed gem wait for my next paper and then you will get the importance of devise gem.



The top part is what a specific gem looks like on the website and bottom of the picture show that I successful installed a devise gem.. For further reading you can visit: https://guides.rubygems.org/command-reference/#gem-install.

While I was working on this project, I got stuck in almost everywhere from understanding the relationships between students, sections, and courses to creating a joint table, installing/adding new gems to the GemFile, creating a dropdown or checkbox, auto completion method and more. In order tackle these and other problems as primary resource I used the class presentation and power points presented by a professor <””> in the class., asking classmates, and googling things using key words and watching YouTube video’s..

Up on a successful completion of this HW I learned many to many relation ships
How to add a search method and autocompletion
How to implement a validations to the appropriate files.
How to edit , add, show info for the user.
How to install a new gem’s and more.



Bibliography

Formal:

1. Dr. Beaty's PowerPoint slides, 03ActiveRecord, 08Controllers.pptx, 06RoutesSearch.pptx
2. Guides.rubyonrails.org. (2018). *Active Record Associations — Ruby on Rails Guides*.

   [online]

3. https://blog.teamtreehouse.com/static-pages-ruby-rails#setup

4. https://www.youtube.com/watch?v=Nf_Si8_szmM

5. https://github.com/rails/rails/issues/22584
6. https://rubygems.org/gems/devise/versions/4.2.0
7. Guides.rubyonrails.org. (2018). *Testing Rails Applications — Ruby on Rails Guides*.
8. [online] Available at: https://guides.rubyonrails.org/testing.html [Accessed 29 Oct. 2018].
9.