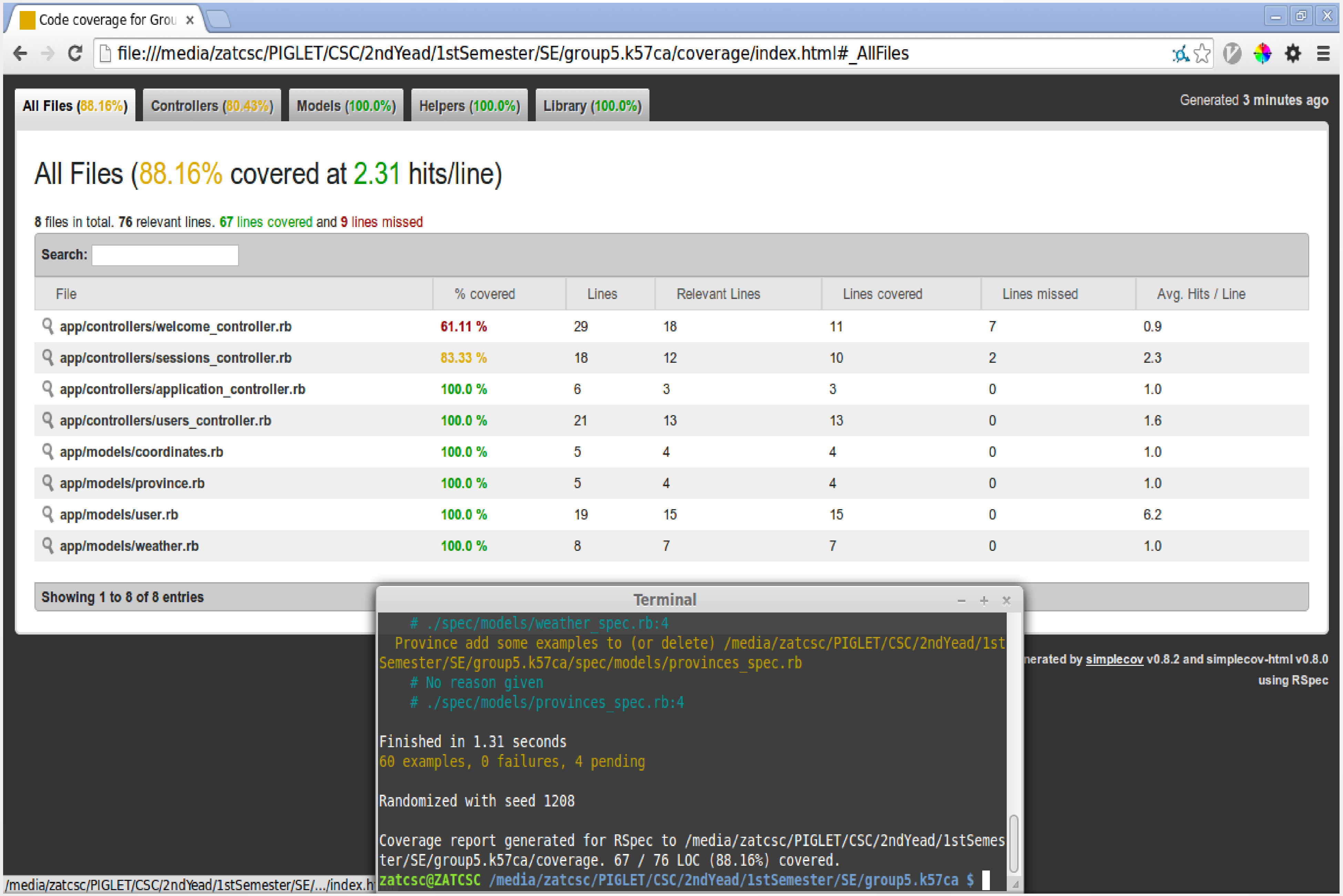
Test Report

- **Test framework:**
- **RSpec** is testing tool for the Ruby programming language. Born under the banner of Behaviour-Driven Development, it is designed to make Test-Driven Development a productive and enjoyable experience

- **Test Driver :**
- **Capybara** helps you test web applications by simulating how a real user would interact with your app. (simulating browser event).
- **Mocks:**
- **FactoryGirl :** create mocks for database (model)



file:///media/zatcsc/PIGLET/CSC/2ndYead/1stSemester/SE/group5.k57ca/coverage/index.html#_AllFiles

All Files (88.16%) | Controllers (80.43%) | Models (100.0%) | Helpers (100.0%) | Library (100.0%)

Generated **3 minutes ago**

All Files (88.16% covered at 2.31 hits/line)

8 files in total. **76** relevant lines. **67** lines covered and **9** lines missed

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/controllers/welcome_controller.rb	61.11 %	29	18	11	7	0.9
app/controllers/sessions_controller.rb	83.33 %	18	12	10	2	2.3
app/controllers/application_controller.rb	100.0 %	6	3	3	0	1.0
app/controllers/users_controller.rb	100.0 %	21	13	13	0	1.6
app/models/coordinates.rb	100.0 %	5	4	4	0	1.0
app/models/province.rb	100.0 %	5	4	4	0	1.0
app/models/user.rb	100.0 %	19	15	15	0	6.2
app/models/weather.rb	100.0 %	8	7	7	0	1.0

Showing 1 to 8 of 8 entries

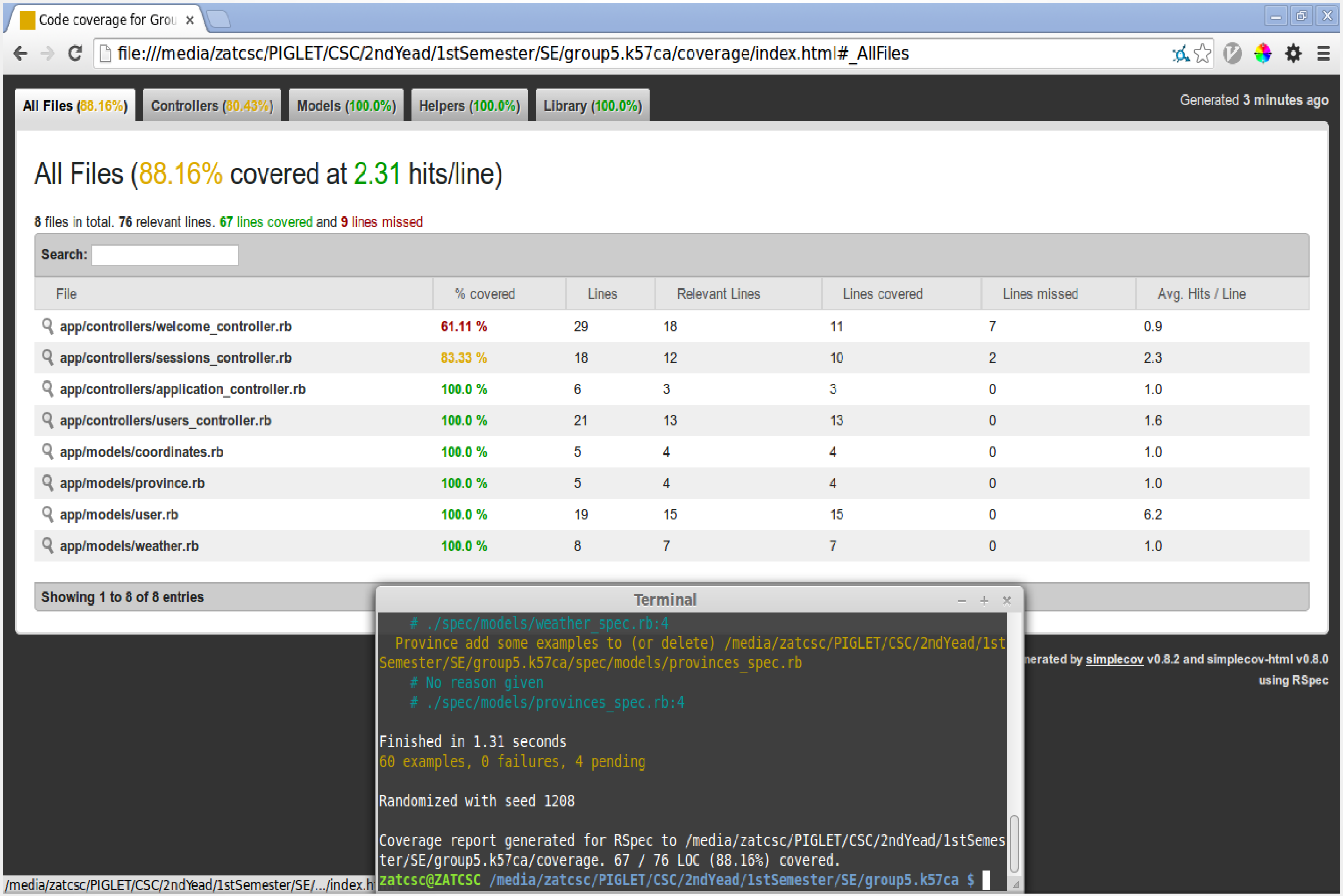
nerated by simplecov v0.8.2 and simplecov-html v0.8.0 using RSpec

```
    # ./spec/models/weather_spec.rb:4
  Province add some examples to (or delete) /media/zatcsc/PIGLET/CSC/2ndYead/1st
Semester/SE/group5.k57ca/spec/models/provinces_spec.rb
    # No reason given
    # ./spec/models/provinces_spec.rb:4

Finished in 1.31 seconds
60 examples, 0 failures, 4 pending

Randomized with seed 1208

Coverage report generated for RSpec to /media/zatcsc/PIGLET/CSC/2ndYead/1stSemes
ter/SE/group5.k57ca/coverage. 67 / 76 LOC (88.16%) covered.
zatcsc@ZATCSC /media/zatcsc/PIGLET/CSC/2ndYead/1stSemester/SE/group5.k57ca $
```

/media/zatcsc/PIGLET/CSC/2ndYead/1stSemester/SE/.../index.h

Overview

- Number of Tests: 60 examples
- Test Coverage: overal 88.16 %
- Test includes
  - Model Test
  - Request Test
  - View Test



All Files (88.16% covered at 2.31 hits/line)

8 files in total. 76 relevant lines. 67 lines covered and 9 lines missed

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/controllers/welcome_controller.rb	61.11 %	29	18	11	7	0.9
app/controllers/sessions_controller.rb	83.33 %	18	12	10	2	2.3
app/controllers/application_controller.rb	100.0 %	6	3	3	0	1.0
app/controllers/users_controller.rb	100.0 %	21	13	13	0	1.6
app/models/coordinates.rb	100.0 %	5	4	4	0	1.0
app/models/province.rb	100.0 %	5	4	4	0	1.0
app/models/user.rb	100.0 %	19	15	15	0	6.2
app/models/weather.rb	100.0 %	8	7	7	0	1.0

Showing 1 to 8 of 8 entries

nerated by simplecov v0.8.2 and simplecov-html v0.8.0
using RSpec

/media/zatcsc/PIGLET/CSC/2ndYead/1stSemester/SE/.../index.h

Terminal

```
    # ./spec/models/weather_spec.rb:4
  Province add some examples to (or delete) /media/zatcsc/PIGLET/CSC/2ndYead/1st
Semester/SE/group5.k57ca/spec/models/provinces_spec.rb
    # No reason given
    # ./spec/models/provinces_spec.rb:4

Finished in 1.31 seconds
60 examples, 0 failures, 4 pending

Randomized with seed 1208

Coverage report generated for RSpec to /media/zatcsc/PIGLET/CSC/2ndYead/1stSemes
ter/SE/group5.k57ca/coverage. 67 / 76 LOC (88.16%) covered.
zatcsc@ZATCSC /media/zatcsc/PIGLET/CSC/2ndYead/1stSemester/SE/group5.k57ca $
```

Model Test::User

- Users model Test (spec/model/user_spec.rb)
- User spec covers following aspecs of Users model
  - User model should have properties:
    - name
    - email
    - password_digest
    - password
    - password_confirmation
    - remember_token

Example Code:

```
it {should respond_to(:name)}

it {should respond_to(:email)}

it {should respond_to(:password_digest)}

it {should respond_to(:password)}

it {should respond_to(:password_confirmation)}

it {should respond_to(:remember_token)}

it {should respond_to(:authenticate)}
```

Model Test::User

- A User should have a method to authenticate if a user with given email and password do existed in database or not .
  - Method: authenticate

Example Code:

```
 describe "return value of authenticate method" do
     before {@user.save}
     let(:found_user) {User.find_by(email: @user.email)}
     describe "with valid password" do
     it {should eq found_user.authenticate(@user.password)}
     end
     describe "with valid password" do
     let(:user_for_invalid_password)
{ found_user.authenticate("invalid")  }
specify {expect(user_for_invalid_password).to be_false}

     end
 end
```

Model Test::User

- A User should not be valid when:
  - name is blank
  - name has less than 6 characters
  - email is blank
  - email is not right form
  - password is blank
- A User should be created when all above fields are valid

Example Code:

```
  describe "when email format is invalid" doode
    it "should not be valid" do
    addresses=%w[user@foo,com user_at_foo.org
example.user@foo.
    foo@bar_baz.com foo@bar+baz.com]
    addresses.each do |invalid_address|
    @user.email=invalid_address
    expect(@user).should_not be_valid
    end
    end
  End
  (find more in spec/requests)
```

Model Test::User

- A User should have a method to authenticate if a user with given email and password do existed in database or not .
  - Method: authenticate

Example Code:

```
  describe "return value of authenticate method" do

      before {@user.save}

      let(:found_user) {User.find_by(email: @user.email)}

      describe "with valid password" do

      it {should eq found_user.authenticate(@user.password)}

      end

      describe "with valid password" do

      let(:user_for_invalid_password)
{ found_user.authenticate("invalid")  }

specify {expect(user_for_invalid_password).to be_false}


      end

  end
```

Model Test::Coordinates

- Coordinates is a model to save a coordinate in a map
- A Coordinates should have following properties
  - Longitude
  - Lattitude

Example Code:

```
it {should respond_to(:logitude)}

it {should respond_to(:latitude)}
```

Model Test::Coordinates

- A Coordinates is valid when its properties are in range:
  - -180 <= longitude <=180
  - -90 <= latitude <= 90

Example Code:

```
describe "when the coordinates is out of range" do
   describe "with logitude is less than -180" do
      before { @example.logitude=-200 }
      it {should_not be_valid}
   end
   describe "with logitude is greater than 180" do
      before { @example.logitude = 200}
      it {should_not be_valid}
   end
end
```

Model Test::Coordinates

- A Coordinates is valid when its properties are in range:
  - -180 <= longitude <=180
  - -90 <= latitude <= 90

Example Code:

```
describe "when the coordinates is out of range" do
    describe "with logitude is less than -180" do
        before { @example.logitude=-200 }
        it {should_not be_valid}
    end
    describe "with logitude is greater than 180" do
        before { @example.logitude = 200}
        it {should_not be_valid}
    end
end
```

Model Test::Weather

- A Weather is used to store weather infomation in a place in a specific day of week
- A Weather model "must" have following properties
  - Day
  - Curent date
  - Min_temperature
  - Max_temperature

Example Code:

```
it {should respond_to(:day)}

it {should respond_to(:current_date)}

it {should respond_to(:min_temperature)}

it {should respond_to(:max_temperature)}
```

Model Test::Weather

- Test for the presense and validation of Weather model's properties.

Example Code:

```
  describe "when day is valid" do
       before {@example.day = "Mon"}
       it {should be_valid}
  end
  describe "when min_temperature is not valid"
do
       before {@example.min_temperature = -50}
       it {should_not be_valid}
  end
```

ViewTest::Sign in

- When visit Sign In Page, it should have:
  - Right title “Sign In”
  - Right header “Sign in”
  - Have email box and label
  - Have password box and label
  - Have submit b

- **Example code:**

```
before {visit signin_path}

it {should have_content("Sign in")}

it {should have_title("Sign in")}        ...
```

ViewTest::Sign up

- Test view for Sign up and some other page such as : contact page, about page is similar with “sign in” page. So that we just mention the idea to avoid the duplication. For more details, please see “spec/request” in project folder.

TestRequest::Sign in

- When user type in the correct email and password →Broswer should redirect to Home page with links to profile page, sign out

Example Code:

```
    describe "with valid information" do
    let(:user) { FactoryGirl.create(:user) }
    before do
      fill_in "Email", :with => user.email
      fill_in "Password", :with => user.password
      click_button "Sign in"
    end
it {should have_link("Profile", href: user_path(user))}
it {should have_link("Sign out", href: signout_path)}
it {should_not have_link("Sign in",href: signin_path)}
    end
```

TestRequest::Sign in

- When user type in incorrect email and password → Broswer should not redirect to other page and show error messsage

Example Code:

```
describe "with invalid information" do
  before { click_button "Sign in" }

  it { should have_title('Sign in') }
  it { should have_selector('div.error') }

  describe "after visiting another page" do
    before { click_link "About" }
    it { should_not have_selector('div.error_explanation') }
  end
end
```

TestRequest::Sign up

- When user type in with valid information. Server should create a new account and redirect user to profile page
- And if with invalid information. Server should not create a new account and the number of user account doesn't change

Example Code:

```
describe "with invalid information" do

it "should not create a user" do

expect{click_button submit}.not_to change(User,:count)


end

end

describe "with valid information" do

it "should create a user" do

fill_in "Name", with: "Nguyen Thac Thong"

fill_in "Email", with: "nguyenthacthong1@gmail.com"

fill_in "Password", with: "1234567"

fill_in "Password confirmation", with: "1234567"

expect{click_button submit}.to change(User,:count).by(1)

end

end
```