# Capybara

https://rubydoc.info/github/teamcapybara/capybara

Capybara is a browser automation tool used for feature testing and to simulate user interactions with a web page

Capybara integrates nicely with Rspec and has a similar syntax, for my project I follow a flow of testing models first (adding data, deleting data, editing, etc.) then use capybara to; sign in, fill form details, click submit, and check for valid / invalid responses.

## Setup:

First, ensure you have Rspec set up with your Rails app, if not, add:

```
group :development, :test do
  gem "rspec-rails", "~> 6.0.0"
end
```

and the capybara gem:

```
group :test do
  gem "capybara"
  gem "selenium-webdriver" # Choose a webdriver for
                                                    # automati
end
```

to your gemfile, then run bundle install

Then, to finish setting up Rspec run:

```
rails generate:rspec install
```

```
rails generate:rspec request <model name>
```

This should set up everything you need for rspec so we can move to finishing the setup for Capybara

In the spec/rails_helper.rb file that was created after running that command, update the file with:

```
abort("The Rails environment is running in production mode!") if
require 'rspec/rails'
require 'capybara/rails'
require 'capybara/rspec'
```

Now you're ready to write Capybara tests

You should have a models/ folder with <model>_spec.rb file(s), this is where you'll write your rspec and capybara tests

Here is an example of a test I wrote that creates a user with FactoryBot, builds it into a test, then fills in various form fields with information from :user data

```
RSpec.feature "User Registration", type: :feature do
  scenario "User signs up with valid credentials" do
    user = FactoryBot.build(:user)

    visit new_user_registration_path

    fill_in "Email", with: user.email
    fill_in "Password", with: user.password
    fill_in "Password confirmation", with: user.password_confirm

    click_button "Sign up"

    expect(page).to have_current_path(root_path)
    expect(page).to have_content("Home")
  end
end
```

```
feature "User signs in", type: :feature do
    user = FactoryBot.create(:user)

   scenario "with correct credentials" do
    visit '/users/sign_in'
      fill_in "Email", with: user.email
      fill_in "Password", with: user.password
      click_button "Log in"

      expect(page).to have_current_path(root_path)
      expect(page).to have_content("Home")
  end

  scenario "With invalid credentials" do
    visit '/users/sign_in'
      fill_in "Email", with: ""
      fill_in "Password", with: ""
      click_button "Log in"

      expect(page).to have_current_path(user_session_path)
      expect(page).to have_content("Invalid Email or password")

  end
end
```

Feature tests are created with Rspec.feature, with various scenarios to test such as “User signs in” (feature) “with correct credentials” (scenario)