TO RUN PROJECT:

**rail s**

starts server, go to http://0.0.0.0:3000/

Press ctrl+c to shut down the server

**rails new appName --database=postgresql**

installs a new folder with rails using postgres

**rails new appName**

installs a new folder with rails using sql light

**rails db:create**

creates the database, should be done after creating the project, though may not matter if you do it before/after creating your models

**Hirb.enable**

use this once you go to the rails console: rails c

**rails g model User first\_name:string age:integer….**

this creates/generates a model name User with a first\_name column as a string, and age as an integer. Model names needs to be capitalized and singular

**rake db:migrate** to migrate your models

**rails c**

opens rails console

**rails g controller ControllerName index new create findOne**

creates a new controller, needs to be capital and plural, anything after the ControllerName and separated by a space will be written as an empty method for that controller upon creation.

<input type="hidden" name="authenticity\_token" value="<%= form\_authenticity\_token %>">

USEFUL GEMS

put in your Gemfile, bundle install when your done. For more info on bootstrap, see

**gem 'faker'**# allows you to fill the DB with random data for names, titles, content, etc. github can be seen at: https://github.com/stympy/faker#fakergameofthrones

**gem 'hirb'**# type: Hirb.enable in the rails console to better see query results. In the event that you get a query result too large, and it shows “(END)”, just type “q” to exit, you can also scroll up and down the results with the arrow keys

**gem 'rails-footnotes', '>= 4.0.0', '<5'**# allows you to see footnotes in the web browser with useful info, after pasting in gemfile, type: rails g rails\_footnotes:install

**gem 'activerecord-session\_store'** # this allows you to use the sessions instead of the browser's cookies to store data. After bundle install type: "rake db:sessions:create" followed by "rake db:migrate". Then go to the directory inside this project /config/initializers/session\_store.rb, and change the ":cookie\_store" to ":active\_record\_store", after restarting your server, the data will now be stored in sessions instead of cookies

**gem 'bootstrap-sass', '3.2.0.2'**

**gem 'sass-rails', '~> 4.0.3'**

**gem 'pg'** # Gem for Rails to interface with PostgreSQL

# Use ActiveModel has\_secure\_password

**gem 'bcrypt', '~> 3.1.7'**

**group :test, :development do**

**gem 'rspec-rails'** # allows testing with rspec, after installing with "bundle install" type: rails g rspec:install

**gem 'database\_cleaner'** # a gem that cleans/clears the test databse for every rspec test, very important for testing

**end**

**group :test do**

**gem 'capybara'** # a web-based automation framework that simulates how users interact with your application. Basically, it is used in your tests to act like a user. Things like visit, fill\_in, click\_link, click\_button, etc. is capybara at work

**end**

PUT THIS IN RAILS\_HELPER.RB, over the “RSpec.configure do |config|” it already has in place

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* START OF COPY & PASTE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RSpec.configure do |config|

config.fixture\_path = "#{::Rails.root}/spec/fixtures"

config.use\_transactional\_fixtures = false # this needs to be false

config.infer\_spec\_type\_from\_file\_location!

config.filter\_rails\_from\_backtrace!

config.include Capybara::DSL

# The below section is for the database cleaner, which cleans/clears the test database after each test

config.before(:suite) do

DatabaseCleaner.clean\_with(:truncation)

DatabaseCleaner.strategy = :truncation

end

config.before(:each) do

DatabaseCleaner.start

end

config.after(:each) do

DatabaseCleaner.clean

end

end

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF COPY & PASTE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

VALIDATIONS EXAMPLES BELOW

user.errors.full\_messages

use in rails console to see errors/validation messages

ex:

*class* User < *ActiveRecord::Base*

EMAIL\_REGEX = /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]+)\z/i

validates :first\_name, :last\_name, presence: true, length: { in: 2..20 } ( could also be length: {is: 2})

validates :email, presence: true, uniqueness: { case\_sensitive: false }, format: { with: EMAIL\_REGEX }

end

Before\_ callbacks that are useful (examples below)

1. before\_validation
2. before\_validation\_on\_create
3. after\_validation
4. after\_validation\_on\_create
5. before\_save
6. before\_create
7. after\_create
8. after\_save

basic validations:

* **:length**- validates the length of an attribute's value
  + :minimum, :maximum, :in, :is
* **:numericality -**validates whether an attribute is a numeric value
  + **:** :odd, :even, and many others
* **:presence -**validates that the specified attributes are not empty
* **:uniqueness** - validates whether the value is unique in the corresponding database table. NOTE: always create a unique index in the database too.
* **:confirmation** - use this when you have two text fields that should receive exactly the same content; assumes the second field name has "\_confirmation" appended
* **validates\_associated** - use this when your model has associations with other models and they also need to be validated
* **:acceptance** - validate whether a checkbox was checked when a form was submitted (usually for 'terms and conditions')
* also, get familiar with **:message** and **:on**

Examples of model callbacks:

Class User < ActiveRecord::Base (This call back is run anytime a destroy is run on a User)

has\_many :posts

validates :first\_name, :last\_name, :email, presence: true

before\_destroy :delete\_posts

def delete\_posts

User.find(self.id).posts.destroy\_all

end

end