**Goal – add owner, relationships, validation and testing to restaurant app**

*Aside: Difference between find() and find\_by\_id()*

.find([:id]) will raise an error if the record is not found

.find\_by\_id([:id]) will return NIL if record does not exist (can recover)

*Aside: Links for destroy/delete actions must have “method: :delete” specified in order for complete deletion.*

<% link\_to “Logout”, destroy\_owner\_session\_path, method: :delete %>

This logout link was added to the view/layouts/application.html.erb file in our last session

**Owner/Restaurant schema**

One owner can have many restaurants but a restaurant can have only one owner.

Owners Restaurants

owner\_id restaurant\_id

owner\_id (foreign key)

If restaurants could have more than one owner, we would need to add a join table between them

Owners Owner\_Restaurants Restaurants

owner\_id owner\_id restaurant\_id

restaurant\_id

Foreign keys need to be indexed – instead of searching through the entire table each time a find() is executed, the index will be used to increase efficiency of the search.

**To Add owner\_id to Restaurant Table**

% rails generate migration AddOwnerToRestaurant owner:reference

(option % add\_reference :restaurants, :owner, index:true, foreign\_key:true)

% rake db:migrate

**To Add relationship (has\_many and belongs\_to) between Owner and Restaurant**

Owner.rb

Add at top has\_many :restaurants dependent: :destroy

dependent: :destroy instructs rails to delete all restaurants when an owner is deleted

Restaurant.rb

Add at top belongs\_to :owner

**To make connection between Owner and Restaurant in controller**

Restaurant controller – change the following methods

new @restaurant = current\_owner.restaurants.build

create @restaurant = current\_owner.restaurants.build

if @restaurant.save …….

edit @restaurant = current\_owner.restaurants.find(params[:id])

update @restaurant = current\_owner.restaurants.find(params[:id])

destroy @restaurant = current\_owner.restaurants.find(params[:id])

**To restrict access to functions from the view** (whether or not links are visible)

show.html.erb <% if current\_owner && @restaurant.owner\_id == current\_owner.id %>

<%= link\_to 'Edit', edit\_restaurant\_path(@restaurant) %> |

<% end %>

<%= link\_to 'Back', restaurants\_path %>

index.html.erb <td><%= link\_to "Show", restaurant %></td>

<% if current\_owner && restaurant.owner\_id == current\_owner.id %>

<td><%= link\_to "Edit", edit\_restaurant\_path(restaurant) %></td>

<td><%= link\_to "Delete", restaurant, method: :delete, data: {confirm:

'Are you sure?'} %></td>

<% end %>

**To add Login function**

Application.html.erb <% if current\_owner %>

(views/layouts) <%=link\_to "Logout", destroy\_owner\_session\_path, method: :delete %>

<% else %>

<%= link\_to "Login", new\_owner\_session\_path %>

<% end %>

**Validations for Restaurant and Owner**

Restaurant.rb

Add to top validates\_presence\_of :name, :description, :phone, :owner\_id

validates\_uniqueness\_of :name

(can add – check google stackoverflow for examples)

email address validation

phone number validation

*Aside: Different places to validate data include*

Database field definition ex: owner\_id: not\_null

Model/Object validation ex: validates\_presence\_of

Controller strong parameters

View input verification (javascript)

View validation is not secure! Use it to make it easier for the user to enter data.

Data validation should be done in the model or controller to keep information secure.

Anyone can access the view.

**Add Name to Owner**

Two methods to add the field

% rails generate migration AddNameToOwner name:string

create your own migration file

class AddNameToOwners < ActiveRecord::Migration

def change

add\_column :owners, :name, :string

end

end

To add the Name entry field – override the Devise sign-in view

Create a directory app/view/devise/registrations

Add a file new.html.erb

(copied from Devise github file)

Add the field in the new.html.erb file in the registrations directory

<div class="field">

<%= f.label :name %><br />

<%= f.text\_field :name, autofocus: true %>

</div>

The sign-in form works but the name value is not being saved since the Devise controller uses strong parameters (review github devise read.me file – strong parameters section)

Add the following code to the application controller file

before\_action :configure\_permitted\_parameters, if: :devise\_controller?

protected

def configure\_permitted\_parameters

devise\_parameter\_sanitizer.for(:sign\_up) << :name

end

What happens to the records already stored in the Owners file – Name values are Null

With this migration there are two options when rolling the change out to production.

1. Log everyone out, put change in place and include functionality to have everyone add a name before logging back in.
2. As part of the migration, could set the name to a value. For example, name could equal the fist part of the email address – u.email.split(‘@’)[0]

**Testing – Rspec (please review github doc – not sure about all these steps – too quick)**

Review rspec-rails and factory\_girl on GitHub

Gemfile Add both to test and development groups in the gemfile (bottom)

% bundle install

% rails generate rspec:install

This creates a spec directory with helpers spec\_helper and rails\_helper that set configuration necessary to perform the tests

(check factory\_girl setup requirements in GitHub – we never finalized the steps)

mkdir spec/models

add restaurant\_spec.rb

require ‘rails\_helper’

describe Restaurant do

it ‘should validate presence of name’ do

restaurant = Restaurant.new(

description: ‘true’

phone: ‘111-111-1111’

weburl: ‘http://test.com’

owner\_id: 1

name: nil

)

expect (restaurant).not\_to\_be\_valid

restaurant = ‘valid’

expect (restaurant).to\_be\_valid

end

end

run rspec spec/models/restaurant\_spec.rb

could also use ‘let’ to check values without creating instance variables

again – review readme docs on github