* **Checklist**
* **Install and Setup**
* Build basic rails app



* Connect to database



* Setup migration for 4 base tables



* Setup migration for 3 relational tables



* Test relational tables and base tables for data



* Setup controllers (Recipes, Ingredients, Users, Cuisines)



* Setup routes



* Open website to test rails server can start.



* **Setup CRUD for**
* Recipes



* Cuisines



* Ingredients



* Users



* Test all CRUD is working for these with sample data.



* Error checking and validation



* **Test Basic Functionality and website flow**
* Draw up website flow diagram



* Test all basic functionality works



* Setup ‘pages’ route to manage basic pages.



* Setup Home page, FAQ, Blog, About, Login, Intro



* Test all routing works to different pages



* Error checking and validation



* **Research API for recipes & how others did it.**
* Yummly – signup to developer site



* Look through requirements and pre-reqs



* Check no issues with local OSX or Heroku or AWS



* Find gems, applicable code and setup and test



* Need to Install controllers for the API?



* Setup API within the app



* Test API can produce data



* Error checking and validation



* **Setting up authentication for users**
* Set this up after Joel’s lesson on Monday on authentication



* **Adding twitter bootstrap**
* **Reseach and look at adding twitter bootstrap**



* **Setup Validation**
* Spend time testing validation and error checking



* **Styling**
* Style pages



* Organize logos



* Home page video - Powtoons



* **Upload to Heroku**

Uploaded to Heroku



**Administration**

Update files to github



Create Readme file in Markdown for documentation



Tidy up documentation



Develop presentation for Friday



* **Saturday 22nd March**
* **\*\* Create and configure a new Rails App**
* *Terminal -Bash*

Create the new rails app

* + rails new justlike\_app –d postgresql
  + cd into the directory (cd justlike\_app)
  + open in sublime text (subl .)
* *Sublime text*
* Added the required gems (root >> gemfile)
  + **#additional debugging gems**
  + gem 'pry-rails'
  + gem 'pry-debugger'
  + gem 'pry-stack\_explorer'
  + gem 'pry'
  + gem 'better\_errors'
  + gem 'protected\_attributes'
  + gem 'annotate'
* save file
* *Terminal -Bash*
* Bundle the gems
  + bundle
  + bundle completed.
* *Sublime text*
* Modify database.yml file ( Config >> database.yml )
  + development:
  + adapter: postgresql
  + encoding: unicode
  + database: justlike\_app\_development
  + pool: 5
  + host: localhost
  + username: markpereira
  + password:
* save file
* *Terminal -Bash*
* Create Database
* Rake db:create
* *Terminal - PSQL*
* Check if database was created
  + Psql
  + \l
  + look for: justlike\_app\_development
* *Terminal-Bash*
* *Ctrl + T opens multiple tabs*
* Setup your 4 tabs in terminal
  + Bash
  + Rails Console
  + PSQL
  + Bash to run Rails server
* **\*\* Setting up the migration scripts to import tables and update schema.**
* *Terminal - Bash*
* Generate migration for users
  + rails generate migration create\_users
* *Sublime text*
* Modify db > migrate > migrate file create\_users
  + t.string :name
  + t.string :address\_1
  + t.string :address\_2
  + t.string :city
  + t.string :postcode
  + t.string :country
  + t.string :email
  + t.text :image
  + t.timestamps
* *Terminal – Bash*
* Migrate the *users* script into the database
  + Rake db:migrate
* *Sublime text*
* Check db > migrate > schema
  + File should be updated with the create table ‘users’
* *Terminal – Bash*
* Generate migration for recipes
  + Rails generate migration create\_recipes
* *Sublime Text*
* Modify db > migrate > migrate file create\_recipes
  + t.string :name
  + t.string :description
  + t.integer :cooktime
  + t.integer :servingsize
  + t.text :instruction
  + t.text :image
  + t.integer :user\_id
  + t.integer :cuisine\_id
  + t.timestamps
* *Terminal – Bash*
* Migrate the *recipes* script into the database
* Rake db:migrate
* *Sublime text*
* Check db > migrate > schema
* File should be updated with the create table ‘recipes’
* *Terminal - Bash*
* Generate migration for ingredients
* rails generate migration create\_ingredients
* *Sublime text*
* Modify db > migrate > migrate file create\_ingredients
  + t.string :name
  + t.string :description
  + t.string :measurement
  + t.string :cost
  + t.text :image
  + t.timestamps
* *Terminal – Bash*
* Migrate the *ingredients* script into the database
* Rake db:migrate
* *Sublime text*
* Check db > migrate > schema
* File should be updated with the create table ‘ingredients’
* *Terminal - Bash*
* Generate migration for cuisines
* rails generate migration create\_cuisines
* *Sublime text*
* Modify db > migrate > migrate file create\_cuisines
  + t.string :name
  + t.string :description
  + t.timestamps
* *Terminal – Bash*
* Migrate the *cuisines* script into the database
* Rake db:migrate
* *Sublime text*
* Check db > migrate > schema
* File should be updated with the create table ‘cuisines’
* *Terminal – PSQL*
* Check psql > login to the justlike\_app\_development database to check
* \d users
* \d recipes
* \d ingredients
* \d cuisines
* *Sublime text*
* Setup models
  + user.rb
    - class User < ActiveRecord::Base
      * attr\_accessible :name, :address\_1, :address\_2, :city, :postcode, :country, :email, :image
      * end
  + recipe.rb
    - class Recipe < ActiveRecord::Base
      * attr\_accessible :name, :description, :cooktime, :servingsize, :instruction, :image, :user\_id, :cuisine\_id
      * end
  + ingredient.rb
    - class Ingredient < ActiveRecord::Base
    - attr\_accessible :name, :description, :measurement, :cost, :image
    - end
  + cuisine.rb
    - class Cuisine < ActiveRecord::Base
    - attr\_accessible :name, :description
    - end
* Setup seeds.rb file with seed data
  + db > migrate >seeds.rb
* Too much data to populate in the documentation. (basic structure)
  + User.destroy\_all
  + Recipe.destroy\_all
  + Ingredient.destroy\_all
  + Cuisine.destroy\_all
  + u1 = User.create(add your :key => ‘value’,)
  + r1 = Recipe.create(add your :key => ‘value’,)
  + i1 = Ingredient.create(add your :key => ‘value’,)
  + c1 = Cuisine.create(add your :key => ‘value’,)
* *Terminal – Bash*
* Import the seeds.rb data (seed data) into the database
* rake db:seed
* >seed file not working with cuisines.



* + All others imported (recipes, users, ingredients)
* >>Working now – changes name in cuisines table from ‘type’ to ‘name’
* rolled back migrate “rake db:rollback”
* changed table name from “type” to “name” and ran “db:migrate again
* **Issue resolved**
* Re-ran seeds.rb “rake db:seed” and all populated ok.
* **\*\* Setup of the relational tables**
* \*\*\*Setup the recipes\_users relational table
* *Terminal – Bash*
* Generate the migration file
* rails generate migration create\_recipes\_users
* *Sublime text*
* Modify db > migrate > migrate file create\_recipes\_users
  + create\_table :recipes\_users, :id => false do |t|
  + t.integer :recipe\_id
  + t.integer :user\_id
* *Terminal – Bash*
* Migrate the *recipes\_users* script into the database
* Rake db:migrate
* *Sublime text*
* Check db > migrate > schema
* File should be updated with the create table ‘recipes\_users’
* \*\*\*Setup the ingredients\_recipes relational table
* *Terminal – Bash*
* Generate the migration file
* rails generate migration create\_ingredients\_recipes
* *Sublime text*
* Modify db > migrate > migrate file create\_ingredients\_recipes
* *Terminal – Bash*
* Migrate the *ingredients\_recipes* script into the database
* Rake db:migrate
* *Sublime text*
* Check db > migrate > schema
* File should be updated with the create table ‘ingredients\_recipes’
* \*\*\*Setup the cuisines\_recipes relational table
* *Terminal – Bash*
* Generate the migration file
* rails generate migration create\_cuisines\_recipes
* *Sublime text*
* Modify db > migrate > migrate file create\_cuisines\_recipes
* *Terminal – Bash*
* Migrate the *cuisines\_recipes* script into the database
* Rake db:migrate
* *Sublime text*
* Check db > migrate > schema
* File should be updated with the create table ‘cuisines\_recipes’
* **\*\* Adding the relationships in the model**
* *Sublime Text*

cuisine.rb

* + has\_many :recipes
* ingredient.rb
  + has\_and\_belongs\_to\_many :recipes
* recipe.rb
  + belongs\_to :cuisine
  + has\_and\_belongs\_to\_many :users
  + has\_and\_belongs\_to\_many :ingredients
* user.rb
  + has\_and\_belongs\_to\_many :recipes
  + would I ever need to use has\_many\_through (for cuisines?) <<



* **\*\*Test relationships through rails console.**
* *Terminal – Rails Console*
* \*\*\*Test recipes\_users relational table

u = User.first

* u.recipes >> data ok
* r = Recipe.first
* r.users >> data ok
* \*\*\*Test cuisines\_recipes relational table
* r = Recipe.first
* r.cuisine >> data ok
* c = Cuisine.first
* c.recipes >> data ok
* \*\*\*Test ingredients\_recipes relational table
* i = Ingredient.first
* i.recipes >> data ok
* r = Recipe.first
* r.ingredients >> data ok
* **\*\*Create the routes**
* *Sublime Text*
* Config > routes.rb
  + JustlikeApp::Application.routes.draw do
  + resources :users, :recipes, :ingredients, :cuisines
  + end
* *Terminal – Bash*
* rake routes
* Prefix Verb URI Pattern Controller#Action
* users GET /users(.:format) users#index
* POST /users(.:format) users#create
* new\_user GET /users/new(.:format) users#new
* edit\_user GET /users/:id/edit(.:format) users#edit
* user GET /users/:id(.:format) users#show
* PATCH /users/:id(.:format) users#update
* PUT /users/:id(.:format) users#update
* DELETE /users/:id(.:format) users#destroy
* recipes GET /recipes(.:format) recipes#index
* POST /recipes(.:format) recipes#create
* new\_recipe GET /recipes/new(.:format) recipes#new
* edit\_recipe GET /recipes/:id/edit(.:format) recipes#edit
* recipe GET /recipes/:id(.:format) recipes#show
* PATCH /recipes/:id(.:format) recipes#update
* PUT /recipes/:id(.:format) recipes#update
* DELETE /recipes/:id(.:format) recipes#destroy
* ingredients GET /ingredients(.:format) ingredients#index
* POST /ingredients(.:format) ingredients#create
* new\_ingredient GET /ingredients/new(.:format) ingredients#new
* edit\_ingredient GET /ingredients/:id/edit(.:format) ingredients#edit
* ingredient GET /ingredients/:id(.:format) ingredients#show
* PATCH /ingredients/:id(.:format) ingredients#update
* PUT /ingredients/:id(.:format) ingredients#update
* DELETE /ingredients/:id(.:format) ingredients#destroy
* cuisines GET /cuisines(.:format) cuisines#index
* POST /cuisines(.:format) cuisines#create
* new\_cuisine GET /cuisines/new(.:format) cuisines#new
* edit\_cuisine GET /cuisines/:id/edit(.:format) cuisines#edit
* cuisine GET /cuisines/:id(.:format) cuisines#show
* PATCH /cuisines/:id(.:format) cuisines#update
* PUT /cuisines/:id(.:format) cuisines#update
* DELETE /cuisines/:id(.:format) cuisines#destroy
* **\*\* Create the Controllers**
* Controllers required:
  + Users
  + Recipes
  + Ingredients
  + Cuisines
* \*\*\*Setup Controller for Users
* *Terminal – Bash*
* rails generate controller Users index create new edit show update destroy
* *Sublime Text*
* Check the following was created:
  + app > controllers > users\_controller.rb
  + app > views > users ( 7 x .rb files)
* \*\*\*Setup Controller for Recipes
* *Terminal – Bash*
* rails generate controller Recipes index create new edit show update destroy
* *Sublime Text*
* Check the following was created:
  + app > controllers > recipes\_controller.rb
  + app > views > recipes ( 7 x .rb files)
* \*\*\*Setup Controller for Ingredients
* *Terminal – Bash*
* rails generate controller Ingredients index create new edit show update destroy
* *Sublime Text*
* Check the following was created:
* app > controllers > ingredients\_controller.rb
* app > views > ingredients ( 7 x .rb files)
* \*\*\*Setup Controller for Cuisines
* *Terminal – Bash*
* rails generate controller Cuisines index create new edit show update destroy
* *Sublime Text*
* Check the following was created:
* app > controllers > cuisines\_controller.rb
* app > views > cuisines ( 7 x .rb files)
* \*\*\*Setup Pages Controller , routes, and stylesheet (custom)
* *Sublime Text*
* app > controllers > pages\_controller.rb
  + class PagesController < ApplicationController
  + def home
  + end
  + end
* app > views > pages ( 1 x .rb files)
  + [home.html.erb](http://home.html.erb) (setup basic home page for root to point to)
* config > routes.rb
  + root :to => 'pages#home'
  + resources :users, :recipes, :ingredients, :cuisines, :pages
* assets > stylesheets > pages.css.scss
* \*\* Start rails server and test I can open a page
* *Terminal – Bash*
* rails server

<< Sunday 2.45am. Finished >>

<< Site working and able to load localhost:3000 successfully. >>

**Sunday 23rd March**

**\*\*Setup Recipe controllers with CRUD**

app > controllers > recipes\_controller.rb

* + class RecipesController < ApplicationController
  + def index
  + @recipes = Recipe.all
  + end
  + def create
  + recipe = Recipe.create params[:recipe]
  + redirect\_to recipe
  + end
  + def new
  + @recipe = Recipe.new
  + end
  + def edit
  + @recipe = Recipe.find params[:id]
  + end
  + def show
  + @recipe = Recipe.find params[:id]
  + end
  + def update
  + recipe = Recipe.find params [:id]
  + recipe.update\_attributes params[:recipe]
  + redirect\_to recipe
  + end
  + def destroy
  + recipe = Recipe.find params[:id]
  + recipe.destroy
  + redirect\_to recipes\_path
  + end
  + end
* Setup views form. (Recipes)
* app > views >recipes
  + Setup New Form: \_form.html.erb
    - <%= form\_for @recipe do |f| %>
    - <%= f.label :name %>
    - <%= f.text\_field :name %>
    - <%= f.label :description %>
    - <%= f.text\_field :description %>
    - <%= f.label :cooktime, "Cooking time" %>
    - <%= f.number\_field :cooktime %>
    - <%= f.label :servingsize, "Serving size" %>
    - <%= f.number\_field :servingsize %>
    - <%= f.label :instruction %>
    - <%= f.text\_area :instruction %>
    - <%= f.label :image %>
    - <%= f.url\_field :image %>
    - <% Ingredient.order(:name).each do |ingredient| %>
    - <span><%= ingredient.name %></span>
    - <%= check\_box\_tag 'recipe[ingredient\_ids][]', ingredient.id, ingredient.in?(@recipe.ingredients) %>
    - <% end %>
    - <%= f.submit %>
    - <% end %>
* save file
* Edit file: index.html.erb
* app > views > recipes > index.html.erb
  + modify file.
    - <ol>
    - <% @recipes.each do |recipe| %>
    - <li>
    - <ul>
    - <li><%= link\_to(recipe.name, recipe) %></li>
    - <li><%= recipe.course %></li>
    - <li><%= recipe.cooktime %></li>
    - <li><%= recipe.servingsize %></li>
    - <li><%= recipe.instruction %></li>
    - <li><%= image\_tag(recipe.image) %></li>
    - <li></li>
    - </ul>
    - </li>
    - <% end %>
    - </ol>
* Test localhost/recipes
* Images and info shows up.
* Edit file: show.html.erb
* app > views > recipes > show.html.erb
  + modify file.
    - <p><%= @recipe.name %></p>
    - <p><%= link\_to('Edit', edit\_recipe\_path(@recipe)) %> <%= link\_to('Delete', @recipe, :method => :delete) %></p>
    - <p>Description: <%= @recipe.description %></p>
    - <p>Serves: <%= @recipe.servingsize %></p>
    - <p><%= @recipe.instruction %></p>
    - <p><%= image\_tag @recipe.image, :class => 'recipe' %></p>
    - <% @recipe.ingredients.each do |ingredient| %>
    - <p><%= ingredient.name %></p>
    - <p><%= image\_tag ingredient.image, :class => 'ingredient' %></p>
    - <% end %>
    - <div class="edit\_delete\_recipe">
    - <%= link\_to 'Edit Recipe', edit\_recipe\_path(@recipe) %>
    - |
    - <%= link\_to 'Delete Recipe', @recipe, :method => :delete %>
    - </div>
* Edit file: edit.html.erb
* app > views > recipes > edit.html.erb
  + modify file.
    - <h1>Edit Recipe</h1>
    - <%= render :partial => 'form' %>
* Edit file: new.html.erb
* app > views > recipes > new.html.erb
  + modify file.
    - <h1>New Recipe</h1>
    - <%= render :partial => 'form' %>
* Test Add recipe >> ok
* Test Edit recipe >> ok
* Add a recipe to test delete function > ok
* **\*\*Setup Ingredients controllers with CRUD**
* app > controllers > ingredients\_controller.rb
  + class IngredientsController < ApplicationController
  + def index
  + @ingredients = Ingredients.all
  + end
  + def create
  + ingredient - Ingredient.create params[:ingredient]
  + redirect\_to ingredient
  + end
  + def new
  + @ingredient = Ingredient.new
  + end
  + def edit
  + @ingredient = Ingredient.find params[:id]
  + end
  + def show
  + @ingredient = Ingredient.find params[:id]
  + end
  + def update
  + ingredient = Ingredient.find params[:id]
  + ingredient.update\_attributes params[:ingredient]
  + redirect\_to ingredient
  + end
  + def destroy
  + ingredient = Ingredient.find params[:id]
  + ingredient.destroy
  + redirect\_to ingredients\_path
  + end
  + end
* Setup views form. (Ingredients)
* app > views > ingredients
* Setup New Form: \_form.html.erb
  + Modify file
    - <%= form\_for @ingredient do |f| %>
    - <%= f.label :name %>
    - <%= f.text\_field :name %>
    - <%= f.label :description %>
    - <%= f.text\_field :description %>
    - <%= f.label :measurement %>
    - <%= f.text\_field :measurement %>
    - <%= f.label :cost %>
    - <%= f.text\_field :cost, :step => :any %>
    - <%= f.label :image %>
    - <%= f.url\_field :image %>
    - <%= f.submit %>
    - <% end %>
* Edit file: show.htm.erb
* app > views > ingredients > show.html.erb
  + modify file.
    - <p><%= @ingredient.measurement %> <%= @ingredient.name %></p>
    - <p><%= link\_to 'Edit', edit\_ingredient\_path(@ingredient) %> <%= link\_to 'Delete', @ingredient, :method => :delete %></p>
    - <p><%= @ingredient.cost %></p>
    - <p><%= image\_tag @ingredient.image, :class => 'ingredient' %></p>
    - <div class="edit\_delete\_ingredient">
    - <%= link\_to 'Edit Ingredient', edit\_ingredient\_path(@ingredient) %>
    - |
    - <%= link\_to 'Delete Ingredient', @ingredient, :method => :delete %>
    - </div>
* Edit file: index.html.erb
* app > views > ingredients > index.html.erb
  + modify file.
    - <ul>
    - <% @ingredients.each do |ingredient| %>
    - <li><%= link\_to ingredient.name, ingredient %></li>
    - <% end %>
    - </ul>
* Edit file: edit.html.erb
* app > views > ingredients > edit.html.erb
  + modify file.
    - <h1>Edit Ingredient</h1>
    - <%= render :partial => 'form' %>
* Edit file: new.html.erb
* app > views > ingredients > new.html.erb
  + modify file.
    - <h1>New Ingredient</h1>
    - <%= render :partial => 'form' %>
* Test Add ingredient >> ok
* Test Edit ingredient >> ok
* Add an ingredient to test delete function > ok
* **\*\*Setup Cuisines controllers with CRUD**
* app > controllers > cuisines\_controller.rb
  + class CuisinesController < ApplicationController
  + def index
  + @cuisines = Cuisine.all
  + end
  + def create
  + cuisine = Cuisine.create params[:cuisine]
  + redirect\_to cuisine
  + end
  + def new
  + @cuisine = Cuisine.new
  + end
  + def edit
  + @cuisine = Cuisine.find params[:id]
  + end
  + def show
  + @cuisine = Cuisine.find params[:id]
  + end
  + def update
  + cuisine = Cuisine.find params[:id]
  + cuisine.update\_attributes params[:cuisine]
  + redirect\_to cuisine
  + end
  + def destroy
  + cuisine = Cuisine.find params[:id]
  + cuisine.destroy
  + redirect\_to cuisines\_path
  + end
  + end
* Setup views form. (Cuisines)
* app > views > cuisines
* Setup New Form: \_form.html.erb
  + Modify file
    - <%= form\_for @cuisine do |f| %>
    - <%= f.label :name %>
    - <%= f.text\_field :name %>
    - <%= f.label :description %>
    - <%= f.text\_field :description %>
    - <%= f.submit %>
    - <% end %>
* Edit file: show.htm.erb
* app > views > cuisines> show.html.erb
  + modify file.
    - <p> <%= @cuisine.name %></p>
    - <p> <%= @cuisine.description %></p>
    - <p><%= link\_to 'Edit', edit\_cuisine\_path(@cuisine) %> <%= link\_to 'Delete', @cuisine, :method => :delete %></p>
    - <div class="edit\_delete\_cuisine">
    - <%= link\_to 'Edit Cuisine', edit\_cuisine\_path(@cuisine) %>
    - |
    - <%= link\_to 'Delete Cuisine', @cuisine, :method => :delete %>
    - </div>
* Edit file: index.html.erb
* app > views > cuisines > index.html.erb
  + modify file.
    - <ul>
    - <% @cuisines.each do |cuisine| %>
    - <li><%= link\_to cuisine.name, cuisine %></li>
    - <% end %>
    - </ul>
* Edit file: edit.html.erb
* app > views > ingredients > edit.html.erb
  + modify file.
    - <h1>Edit Cuisine</h1>
    - <%= render :partial => 'form' %>
* Edit file: new.html.erb
* app > views > ingredients > new.html.erb
  + modify file.
    - <h1>New Cuisine</h1>
    - <%= render :partial => 'form' %>
* Test Add Cuisine >> ok
* Test Edit Cuisine>> ok
* Add a cuisine to test delete function > ok
* **\*\*Setup Users controllers with CRUD**
* app > controllers > users\_controller.rb
* Setup views form. (Users)
  + class UsersController < ApplicationController
  + def index
  + @users = User.all
  + end
  + def create
  + user = User.create params[:user]
  + redirect\_to user
  + end
  + def new
  + @user = User.new
  + end
  + def edit
  + @user = User.find params[:id]
  + end
  + def show
  + @user = User.find params[:id]
  + end
  + def update
  + user = User.find params[:id]
  + user.update\_attributes params[:user]
  + redirect\_to user
  + end
  + def destroy
  + user = User.find params[:id]
  + user.destroy
  + redirect\_to users\_path
  + end
  + end
* app > views > users
* Setup New Form: \_form.html.erb
  + Modify file
    - <%= form\_for @user do |f| %>
    - <%= f.label :name %>
    - <%= f.text\_field :name %>
    - <%= f.label :address\_1 %>
    - <%= f.text\_field :address\_1 %>
    - <%= f.label :address\_2 %>
    - <%= f.text\_field :address\_2 %>
    - <%= f.label :city %>
    - <%= f.text\_field :city %>
    - <%= f.label :postcode %>
    - <%= f.text\_field :postcode %>
    - <%= f.label :country %>
    - <%= f.text\_field :country %>
    - <%= f.label :email %>
    - <%= f.text\_field :email %>
    - <%= f.label :image %>
    - <%= f.url\_field :image %>
    - <%= f.submit %>
    - <% end %>
* Edit file: show.html.erb
* app > views > users> show.html.erb
  + modify file.
    - <p><%= link\_to 'Edit', edit\_user\_path(@user) %> <%= link\_to 'Delete', @user, :method => :delete %></p>
    - <p> <%= @user.name %></p>
    - <p> <%= @user.address\_1 %></p>
    - <p> <%= @user.address\_2 %></p>
    - <p> <%= @user.city %></p>
    - <p> <%= @user.postcode %></p>
    - <p> <%= @user.country %></p>
    - <p> <%= @user.email %></p>
    - <p><%= image\_tag @user.image, :class => 'user' %></p>
    - <div class="edit\_delete\_user">
    - <%= link\_to 'Edit User', edit\_user\_path(@user) %>
    - |
    - <%= link\_to 'Delete User', @user, :method => :delete %>
    - </div>
* Edit file: index.html.erb
* app > views > users > index.html.erb
  + modify file.
    - <ul>
    - <% @users.each do |user| %>
    - <li><%= link\_to user.name, user %></li>
    - <% end %>
    - </ul>
* Edit file: edit.html.erb
* app > views > users > edit.html.erb
  + modify file.
    - <h1>Edit User</h1>
    - <%= render :partial => 'form' %>
* Edit file: new.html.erb
* app > views > user > new.html.erb
  + modify file.
    - <h1>New User</h1>
    - <%= render :partial => 'form' %>
* Test Add User >> ok
* Test Edit User>> ok
* Add a user to test delete function > ok
* << Finish Sunday night 11.12.>>
* **Monday 24th March**
* \*\*\*Install yummly gem
* *Terminal – Bash*
* Gem install yummly
* >> 3 gems installed
* \*\*\* Add other required gems to gemfile
* *Sublime Text*
* root > gemfile
* Modify gem files to add Yummly, JSON, HTTPARTY gem
  + gem 'yummly'
  + gem 'json'
  + gem 'httparty'
* Save file.
* *Terminal-Bash*
* Run bundle
* Your bundle is complete!
* Use `bundle show [gemname]` to see where a bundled gem is installed.
* *Sublime Text*
* config > routes.rb
* Add routes for pages
  + JustlikeApp::Application.routes.draw do
  + root :to => 'pages#home'
  + get '/home' => 'pages#home'
  + get '/about' => 'pages#about'
  + get '/blog' => 'pages#blog'
  + get '/intro' => 'pages#intro'
  + resources :users, :recipes, :ingredients, :cuisines, :pages
  + end
* app > controllers > pages\_controller.rb
* check that the Home, About, blog, login, intro pages are defined
  + class PagesController < ApplicationController
  + def home
  + end
  + def about
  + end
  + def blog
  + end
  + def intro
  + end
  + end
* app > views > layout > application.html.erb
* Add navigation pane
  + <nav>
  + <a href="/home">Home</a>
  + |
  + <a href ="/about">about</a>
  + |
  + <a href ="/blog">blog</a>
  + |
  + <a href ="/">login</a>
  + </nav>
* app > views >pages
* add page about.html.erb
  + <h1>This is the about page</h1>
* add page blog.html.erb
  + <h1>This is the blog page</h1>
* Test home page >> ok
* Test about page >> ok
* Test blog page >> ok
* \*\*\* Setup login page
* Setup controller for sessions and login
* *Terminal - Bash*
* rails generate controller Session
* *Sublime Text*
* Update login routes
* config > routes.rb
  + JustlikeApp::Application.routes.draw do
  + root :to => 'pages#home'
  + get '/home' => 'pages#home'
  + get '/about' => 'pages#about'
  + get '/blog' => 'pages#blog'
  + get '/intro' => 'pages#intro'
  + get '/login' => 'session#new'
  + post '/login' => 'session#create'
  + delete '/login' => 'session#destroy'
  + resources :users, :recipes, :ingredients, :cuisines, :pages
  + end
* set up new, create and destroy definitions in the session controller
* app > controllers > session\_controller
  + class SessionController < ApplicationController
  + def new
  + end
  + def create
  + end
  + def destroy
  + end
  + end
* set up new.html.erb
* apps > views > sessions > new.html.erb
* modify
  + <%= form\_tag '/login' do %>
  + <%= label\_tag :name %>
  + <%= text\_field\_tag :name %>
  + <%= label\_tag :password %>
  + <%= password\_field\_tag :password %>
  + <%= submit\_tag %>
  + <% end %>

API Yummly

* **Performing GET requests to search for recipes**
* The base url for the Search Recipes GET is http://api.yummly.com/v1/api/recipes?\_app\_id=app-id&\_app\_key=app-key&your \_search\_parameters
* **2- Attribution Requirements**
* Whenever you display information obtained from the Yummly API to your users, you must give attribution to Yummly and the source of the recipe. The objects returned by the Get Recipe and Search Recipes API calls contain an attribution subobject which consists of the following fields:
* **html**: If your application uses HTML to display data to the users, you must include the contents of the html field verbatim in a visible place near the data. You can ignore the other fields. You may not modify the HTML snippet.
* **url, text, logo**: If your application is native (desktop or mobile) or otherwise doesn’t use HTML, you must display the text, the logo, and a link to the URL in a visible place near the data. Clicking this should open a browser on the url. You may not modify the URL, the link text, or the logo image.
* **source**: The recipe objects returned by the Get Recipe and Search Recipes API calls contain sourceattribution subobject which consists of the following fields – sourceRecipeUrl,sourceSiteUrl, sourceDisplayName. You must display the sourceDisplayName, and a link to either the sourceRecipeUrl and/or sourceSiteUrl in a visible place near the data. Clicking this should open a browser on the url. You may not modify the sourceRecipeUrl, sourceSiteUrl, or the sourceDisplayName.
* q: This is the search phrase. Use space to separate words (url-encoded as either + or %20).
* To search for “Onion Soup” recipes append &q=onion+soup or &q=onion%20soup   
  For example: http://api.yummly.com/v1/api/recipes?\_app\_id=YOUR\_ID&\_app\_key=YOUR\_APP\_KEY&q=onion+soup
* maxResult, start : The maxResult and start parameters allow pagination and # of results control. By default 6 recipes are returned by the search API.
* For example, if you want 10 recipes per page and want to see the second page of results, you would append &maxResult=10&start=10. Start is set to 10 versus because the numbering for results starts at 0 (versus 1).   
  For example: http://api.yummly.com/v1/api/recipes?\_app\_id=YOUR\_ID&\_app\_key=YOUR\_APP\_KEY&q=onion+soup  
  &maxResult=10&start=10

Look at other recipe projects on github.

Twitter bootstrap

* Check with Joel
  + Database relational tables all working.
  + If I wanted to add tables later (to coincide with extra data coming through from Yummly) I assume it’s as easy as setting up another migration with “add columns” then run a db migrate?
  + I have seen somewhere that they had a separate controller for the api. Why would you do this and what are the benefits / catches with this
  + Check Task list, see I haven’t left anything major out.