

Receipe of the Day

By nock&null

Final Project Report

11.06.2014

MIS336-2014 Term Project
Boğaziçi Üniversitesi MIS Department

Table of Contents

1	Introduction	3
1.1	Product Description	3
1.2	Customer Information	3
1.3	Company Overview.....	3
1.3.1	Company Staff.....	3
1.3.2	Scope	4
1.3.3	Constraints	4
1.3.4	Resources	4
1.3.5	Methods.....	4
2	Work Breakdown.....	5
3	Work Effort Estimation.....	5
4.	Risk Analysis.....	6
5.	Work Schedule.....	7
6.	Project Scope.....	10
	Context Diagram	10
7.	Architectural Design	11
8.	Modular Design.....	11
9.	Data Design	12
10.	State Design	14
11.	User Role Design	15
12.	Technologies.....	16
13.	Screenshots	17
14.	Reference.....	18
15.	Coding Standards.....	18
16.	Appendix	18

1 Introduction

This is a software project which is developed by nock&null for MIS 336 Business Program Development. Analyze of the project is performed by another company. This document includes reviewing the design, making necessary changes, building and maintaining the system.

This document is organized as follows:

- Project Description
- Work breakdown
- Work Effort Estimation

1.1 Product Description

The purpose of the project was initiated in order to simplify people's daily cooking work. It aims to enable users to browse and search for recipes, save recipes they find and plan weekly cooking schedule. Additionally, the system makes users save effort for which people usually spend a lot time while thinking what and how to cook.

1.2 Customer Information

Recipe of day addresses to all internet users who can utilize this system in daily cooking. Especially housewives, workers and cooking enthusiasts who use the internet actively may benefit from project features. Furthermore, starters can utilize the project to develop their recipe range in the kitchen.

1.3 Company Overview

nock&null is a software company that adopts "Develop Responsibly" as a principle to produce cool, innovative and hipster products. The main working area is building web and mobile projects with new technologies. The company was established in 2014 and located in Istanbul.

1.3.1 Company Staff

nock&null is basically managed by four young technology enthusiasts:

Burak Sönmez: *Team Leader*

Up-to-date, night watcher, movie and series follower

Demirhan Aydın: *Software Developer*

Hardware bender, amateur drone pilot

Fatih Karadeniz: *Digital Marketing Associate*

Wanderlust, integer developer, brewery explorer, big fan of "Gannicus"

Ömer Faruk Aydın: *Software Developer*

Passionate programmer, coffee addicted, hard technology follower

1.3.2 Scope

Recipe of the day supports only Turkish content recipes. It does not cover all languages but there is no limitation for any cuisine in the world. Mobile and desktop users can reach the content easily. Only administrators are able to add new content, however users can suggest new receipts to expand the coverage of the system.

1.3.3 Constraints

Capital seems to be the biggest problem and following that time limitation can be the secondary issue.

1.3.4 Resources

Fundamental technical and operational costs are shown in the table below:

ITEM	Description	Quantity	Price/Salary (TL)	Cost monthly (TL)	Cost annually (TL)
1	Hosting details	1	200	200	2400
2	Domain	1	24	-	24
3	Salary Of Employees	4	2500	10000	120000
4	Computer Hardware(macbook air 11" details)	4	2600	-	10400
5	Internet Connection Fees	1	50	50	600
	TOTAL				133424

1.3.5 Methods

The project is built on top of Ruby programming language and Rails web framework. For database solution postgresql is used for as a persistence database. In addition to this, Redis (which is key value storage) is used for basic caching operations. All object oriented modeling operations are prepared on IBM Rational Rose. The team prefers to use agile software development. For front –end framework, Bootstrap meets the requirements.

2 Work Breakdown

1.User Interface Development	2.Database Development	3.Code Development
1.1.Determine Required Pages	2.1.Determine Entities &Tables	3.1. Preparing Local and Production Environments
1.2.Design Pages	2.2.Determine Indexes & Views	3.2.Creating MVP
1.3.User Experience Tests		

3 Work Effort Estimation

MIS336 Software project effort estimation

Project name

Recipe of The Day

Date

10.04.2014

No	Task/module name	User Interface	Database	Code complexity	Effort (days)
1	Determine require pages	2-Low	0-None	0-None	1
2	Create design template	3-Medium	0-None	0-None	2
3	Determine page content	3-Medium	0-None	0-None	2
4	Implement page templates with content	3-Medium	0-None	0-None	2
5	Apply user experience test	4-High	0-None	0-None	2
6	Design development complete	0-None	0-None	0-None	1
7	Determine entities and tables	0-None	3-Medium	0-None	1
8	Determine indexes and views	0-None	3-Medium	0-None	1
9	Database developement complete	0-None	0-None	0-None	1
10	Preparing local and production environments	0-None	0-None	2-Low	3
11	Code the system	0-None	0-None	4-High	5
12	Merge front-end with back-end	0-None	0-None	4-High	5
13	Database Connection	0-None	0-None	3-Medium	4
14	Deploy production	0-None	0-None	3-Medium	4
15	Check security vulnerabilities	0-None	0-None	4-High	5
16	Maintain environments	0-None	0-None	4-High	5
17	Code development Complete	0-None	0-None	0-None	1

Total

45

4. Risk Analysis

In order to assess the possible risks and impacts in terms of their effects, we created the risk management table which can be seen below:

- The location was assumed as Istanbul for our company, however international risks were also considered.

MIS336 Software project risk management

Project name

Recipeoftheday

Date

11.06.2014

No	Risk name	Type	Probability	Effect	Impact (0-100)
1	Service Shut Down	1-Technology	1-Very Low	4-Serious	16
2	Staff Turnover	2-People	2-Low	3-Medium	24
3	Employees' Health Problems	2-People	3-Medium	2-Tolerable	24
4	Storage Crash	1-Technology	2-Low	4-Serious	32
5	Router Crash	1-Technology	3-Medium	4-Serious	48
6	General Hardware Unavailability Problems	1-Technology	2-Low	3-Medium	24
7	Hacker Attacks	1-Technology	2-Low	3-Medium	24
8	Data Lose	4-Tools	2-Low	4-Serious	32
9	Coder Estimation Mistake	2-People	3-Medium	3-Medium	36
10	Lack of Synergy Among Group Members	2-People	1-Very Low	3-Medium	12
11	Task Distribution Delay	2-People	2-Low	3-Medium	24
12	Overloading of Group Members	2-People	3-Medium	3-Medium	36
13	Organizational Meeting Delays	2-People	2-Low	2-Tolerable	16
14	Confusion Estimation	6-Estimation	2-Low	3-Medium	24
15	Product Quality Estimation	6-Estimation	2-Low	4-Serious	32
16	Design Estimation	6-Estimation	3-Medium	2-Tolerable	24
17	Project Budget Planning Mistake	6-Estimation	3-Medium	3-Medium	36
18	Project Cost Estimation	6-Estimation	3-Medium	3-Medium	36
19	Insufficient Flexibility of Interface	5-Requirements	3-Medium	3-Medium	36
20	Insufficient Functionality of Interface	5-Requirements	3-Medium	4-Serious	48
21	Customer Satisfaction	2-People	3-Medium	3-Medium	36
22	Testing Process Estimation	6-Estimation	3-Medium	2-Tolerable	24
23	Code and Database Consistency	1-Technology	3-Medium	3-Medium	36
24	Software Licencing Problem	3-Organizational	2-Low	3-Medium	24
25	Lose of Project Objectivity	2-People	3-Medium	3-Medium	36
26	Project Benefit Estimation	6-Estimation	2-Low	3-Medium	24
27	Project Members' Incapability for Tasks	2-People	2-Low	3-Medium	24
28	Compatitive Product Risk	3-Organizational	3-Medium	3-Medium	36
29	Customer Understand Failure for Requirement Change	2-People	3-Medium	3-Medium	36
31	Timing Failure in Development Process	6-Estimation	3-Medium	3-Medium	36
32	Database Security Insufficiency	1-Technology	2-Low	4-Serious	32
33	Utility Risks(gas,electricity etc.)	3-Organizational	2-Low	3-Medium	24
34	Fire	3-Organizational	2-Low	4-Serious	32
35	Earthquake	3-Organizational	2-Low	5-Catastrophic	40
36	Flooding	3-Organizational	3-Medium	3-Medium	36

TOTAL IMPACT 1060

5. Work Schedule

Project Scheduling:

Nock&null consists of four employee in the minimum basis. Although we have a team leader already, we adopt very democratic management style for decision making and for our business. We claim that we have a fine team spirit while working.

The team members worked with more than one title until we enlarge our team.

In management, the decisions were made all together

Project Manager: Burak Sönmez

Analyst: Fatih Karadeniz

Developer: Demirhan Aydın, Ömer Faruk Aydın

Tester: Ömer Faruk Aydın, Fatih Karadeniz

Trainer: Demirhan Aydın

Technical Communicator: Ömer Faruk Aydın

Deployment Team: Demirhan Aydın, Burak Sönmez

Görev Adı	Süre	Yukarıdaki Metin	Başlangıç	Bitiş	Kaynak Adları
Scope	3,5 gün	Hayır	Pzt 03.03.14	Per 06.03.14	
Determine project scope	4 sa	Hayır			Management
Secure project sponsorship	1 gün	Hayır			Management
Define preliminary resources	1 gün	Hayır			Project Manager
Secure core resources	1 gün	Hayır			Project Manager
Scope complete	0 gün	Hayır			
Analysis/Software Requirements	14 gün	Hayır	Pzt 03.03.14	Per 20.03.14	
Conduct needs analysis	5 gün	Hayır			Analyst
Draft preliminary software specifications	3 gün	Hayır			Analyst
Develop preliminary budget	2 gün	Hayır			Project Manager
Review software specifications/budget with team	4 sa	Hayır			Project Manager;Analyst
Incorporate feedback on software specifications	1 gün	Hayır			Analyst
Develop delivery timeline	1 gün	Hayır			Project Manager
Obtain approvals to proceed (concept, timeline, budget)	4 sa	Hayır			Management;Project Manager
Secure required resources	1 gün	Hayır			Project Manager
Analysis complete	0 gün	Hayır			
Design	14,5 gün	Hayır	Çar 21.05.14	Sal 10.06.14	
Review preliminary software specifications	2 gün	Hayır			Analyst
Develop functional	5 gün	Hayır			Analyst

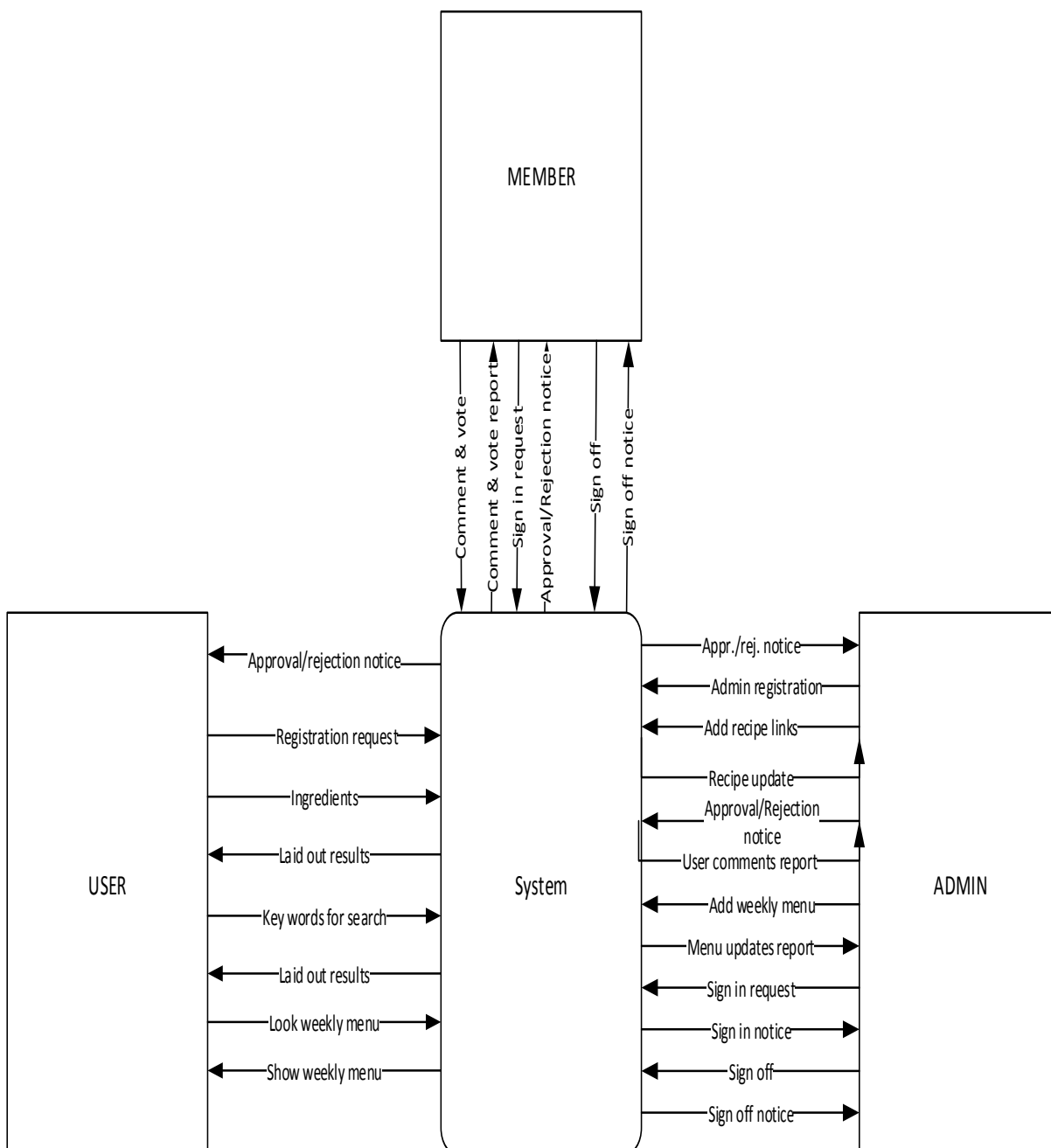
specifications					
Develop prototype based on functional specifications	4 gün	Hayır			Analyst
Review functional specifications	2 gün	Hayır			Management
Incorporate feedback into functional specifications	1 gün	Hayır			Management
Obtain approval to proceed	4 sa	Hayır			Management;Project Manager
Design complete	0 gün	Hayır			
Development	21,75 gün	Hayır	Pzt 12.05.14	Sal 10.06.14	
Review functional specifications	1 gün	Hayır			Developer
Identify modular/tiered design parameters	1 gün	Hayır			Developer
Assign development staff	1 gün	Hayır			Developer
Develop code	15 gün	Hayır			Developer
Developer testing (primary debugging)	15 gün	Hayır			Developer
Development complete	0 gün	Hayır			
Testing	16 gün	Hayır	Sal 20.05.14	Sal 10.06.14	
Develop unit test plans using product specifications	4 gün	Hayır			Testers
Develop integration test plans using product specifications	4 gün	Hayır			Testers
Unit Testing	14 gün	Hayır	Sal 20.05.14	Cum 06.06.14	
Review modular code	5 gün	Hayır			Testers
Test component modules to product specifications	2 gün	Hayır			Testers
Identify anomalies to product specifications	3 gün	Hayır			Testers
Modify code	3 gün	Hayır			Testers
Re-test modified code	2 gün	Hayır			Testers
Unit testing complete	0 gün	Hayır			
Integration Testing	9 gün	Hayır	Sal 27.05.14	Cum 06.06.14	
Test module integration	5 gün	Hayır			Testers
Identify anomalies to specifications	2 gün	Hayır			Testers
Modify code	3 gün	Hayır			Testers
Re-test modified code	2 gün	Hayır			Testers
Integration testing complete	0 gün	Hayır			
Training	45,75 gün	Hayır	Çar 12.03.14	Çar 14.05.14	
Develop training specifications for end users	3 gün	Hayır			Trainers
Develop training specifications for helpdesk	3 gün	Hayır			Trainers

support staff					
Identify training delivery methodology (computer based training, classroom, etc.)	2 gün	Hayır			Trainers
Develop training materials	3 hf	Hayır			Trainers
Conduct training usability study	4 gün	Hayır			Trainers
Finalize training materials	3 gün	Hayır			Trainers
Develop training delivery mechanism	2 gün	Hayır			Trainers
Training materials complete	0 gün	Hayır			
Documentation	15 gün	Hayır	Çar 21.05.14	Sal 10.06.14	
Develop Help specification	1 gün	Hayır			Technical Communicators
Develop Help system	3 hf	Hayır			Technical Communicators
Review Help documentation	3 gün	Hayır			Technical Communicators
Incorporate Help documentation feedback	2 gün	Hayır			Technical Communicators
Develop user manuals specifications	2 gün	Hayır			Technical Communicators
Develop user manuals	3 hf	Hayır			Technical Communicators
Review all user documentation	2 gün	Hayır			Technical Communicators
Incorporate user documentation feedback	2 gün	Hayır			Technical Communicators
Documentation complete	0 gün	Hayır			
Pilot	30 gün	Hayır	Sal 29.04.14	Pzt 09.06.14	
Identify test group	1 gün	Hayır			Project Manager
Develop software delivery mechanism	1 gün	Hayır			
Install/deploy software	1 gün	Hayır			Deployment Team
Obtain user feedback	1 hf	Hayır			Deployment Team
Evaluate testing information	1 gün	Hayır			Deployment Team
Pilot complete	0 gün	Hayır			
Deployment	5 gün	Hayır	Çar 04.06.14	Sal 10.06.14	
Determine final deployment strategy	1 gün	Hayır			Deployment Team
Develop deployment methodology	1 gün	Hayır			Deployment Team
Secure deployment resources	1 gün	Hayır			Deployment Team
Train support staff	1 gün	Hayır			Deployment Team
Deploy software	1 gün	Hayır			Deployment Team
Deployment complete	0 gün	Hayır			
Post Implementation Review	3 gün	Hayır	Cmt 07.06.14	Sal 10.06.14	

Document lessons learned	1 gün	Hayır			Project Manager
Distribute to team members	1 gün	Hayır			Project Manager
Create software maintenance team	1 gün	Hayır			Project Manager
Post implementation review complete	0 gün	Hayır			
Software development template complete	0 gün	Hayır			

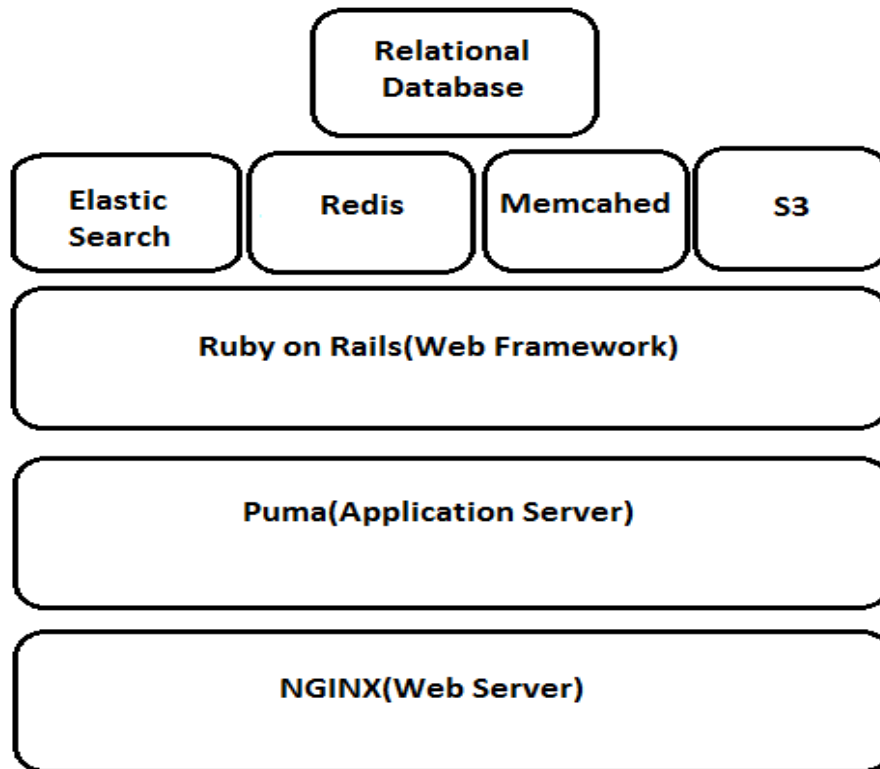
6. Project Scope

Context Diagram



7.Architectural Design

The architecture of the system can be seen in the figure below:



8.Modular Design

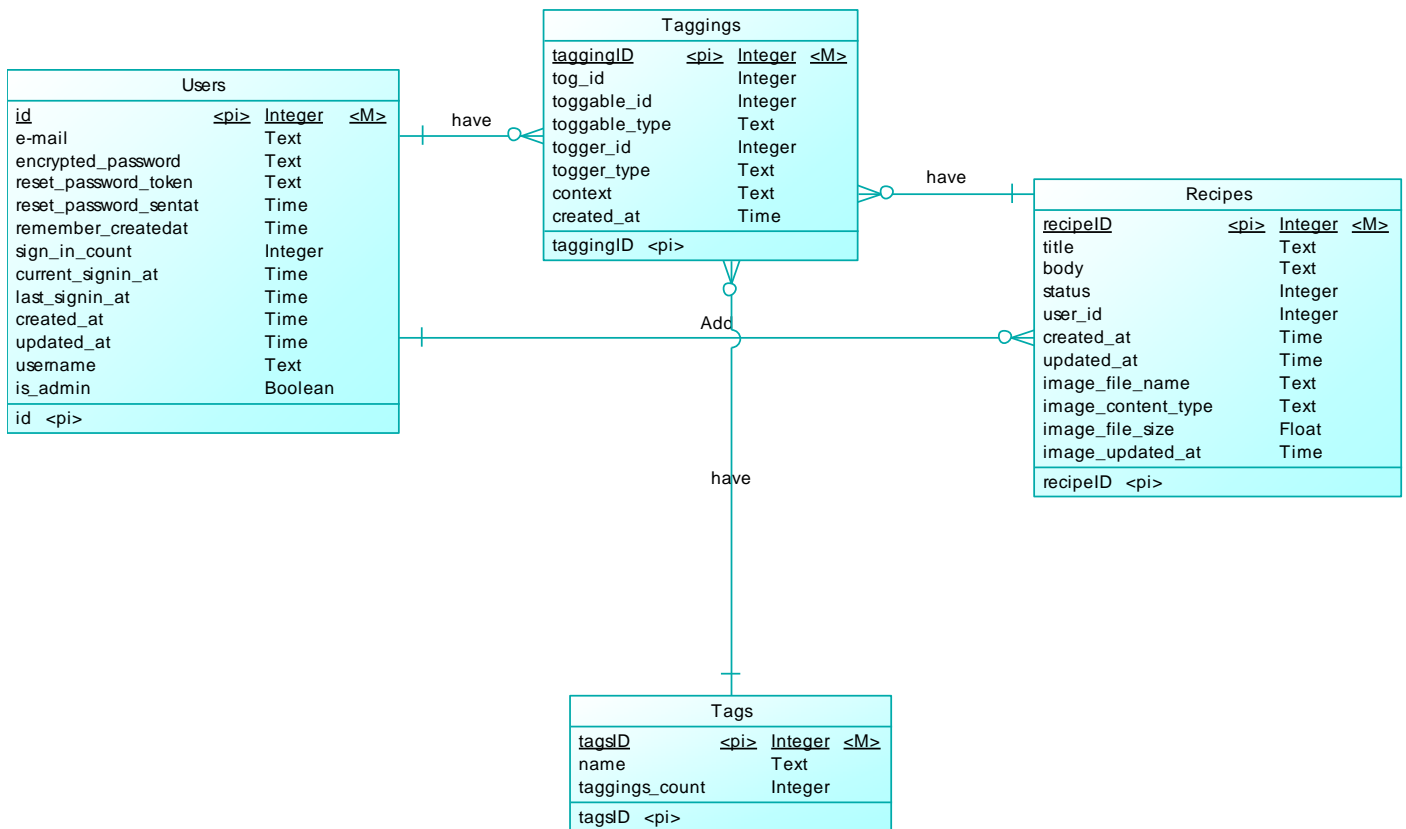
Model-view-controller (MVC) is a pattern used to isolate business logic from the user interface. Using MVC, the Model represents the information (the data) of the application and the business rules used to manipulate the data, the View corresponds to elements of the user interface such as text, checkbox items, and so forth, and the Controller manages details involving the communication between the model and view. The controller handles user actions such as keystrokes and mouse movements and pipes them into the model or view as required.(1)

Don't Repeat Yourself: DRY is a principle of software development which states that "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system." By not writing the same information over and over again, our code is more maintainable, more extensible, and less buggy.(2)

Convention Over Configuration: Rails has opinions about the best way to do many things in a web application, and defaults to this set of conventions, rather than require that you specify every minutiae through endless configuration files.(3)

9. Data Design

ER DIAGRAMS



Data Tables

- Recipes

d6c28d4k3attj5=> \d recipes

Table "public.recipes"		
Column	Type	Modifiers
id	integer	not null default nextval('recipes_id_seq'::regclass)
title	character varying(255)	
body	text	
status	integer	
user_id	integer	
created_at	timestamp without time zone	
updated_at	timestamp without time zone	
image_file_name	character varying(255)	
image_content_type	character varying(255)	
image_file_size	integer	
image_updated_at	timestamp without time zone	

Indexes:

- "recipes_pkey" PRIMARY KEY, btree (id)
- "index_recipes_on_title" btree (title)
- "index_recipes_on_user_id" btree (user_id)

- Users

```
d6c28d4k3attj5=> \d users
```

Table "public.users"		
Column	Type	Modifiers
id	integer	not null default nextval('users_id_seq'::regclass)
email	character varying(255)	not null default ''::character varying
encrypted_password	character varying(255)	not null default ''::character varying
reset_password_token	character varying(255)	
reset_password_sent_at	timestamp without time zone	
remember_created_at	timestamp without time zone	
sign_in_count	integer	not null default 0
current_sign_in_at	timestamp without time zone	
last_sign_in_at	timestamp without time zone	
current_sign_in_ip	character varying(255)	
last_sign_in_ip	character varying(255)	
created_at	timestamp without time zone	
updated_at	timestamp without time zone	
username	character varying(255)	not null default ''::character varying
is_admin	boolean	default false

Indexes:

- "users_pkey" PRIMARY KEY, btree (id)
- "index_users_on_email" UNIQUE, btree (email)
- "index_users_on_reset_password_token" UNIQUE, btree (reset_password_token)
- "index_users_on_username" UNIQUE, btree (username)

- Admin Users

```
d6c28d4k3attj5=> \d admin_users
```

Table "public.admin_users"		
Column	Type	Modifiers
id	integer	not null default nextval('admin_users_id_seq'::regclass)
email	character varying(255)	not null default ''::character varying
encrypted_password	character varying(255)	not null default ''::character varying
reset_password_token	character varying(255)	
reset_password_sent_at	timestamp without time zone	
remember_created_at	timestamp without time zone	
sign_in_count	integer	not null default 0
current_sign_in_at	timestamp without time zone	
last_sign_in_at	timestamp without time zone	
current_sign_in_ip	character varying(255)	
last_sign_in_ip	character varying(255)	
created_at	timestamp without time zone	
updated_at	timestamp without time zone	

Indexes:

- "admin_users_pkey" PRIMARY KEY, btree (id)
- "index_admin_users_on_email" UNIQUE, btree (email)
- "index_admin_users_on_reset_password_token" UNIQUE, btree (reset_password_token)

- Tags

```
d6c28d4k3attj5=> \d tags
```

Column	Type	Modifiers
id	integer	not null default nextval('tags_id_seq'::regclass)
name	character varying(255)	
taggings_count	integer	default 0

Indexes:

- "tags_pkey" PRIMARY KEY, btree (id)
- "index_tags_on_name" UNIQUE, btree (name)

- Taggings

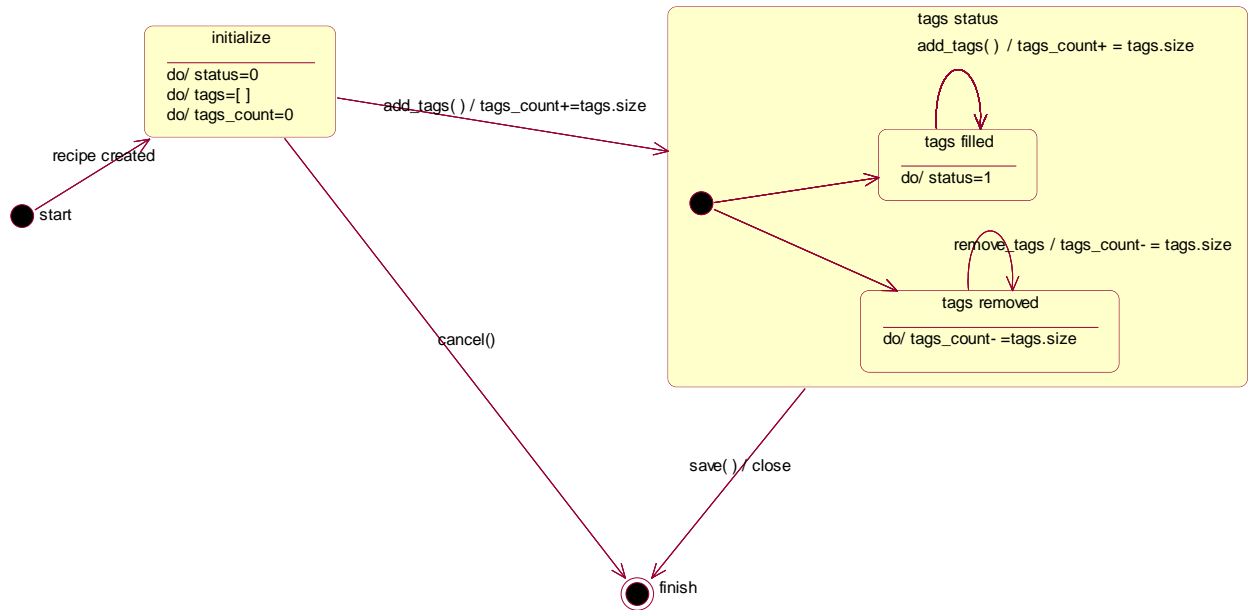
```
d6c28d4k3attj5=> \d taggings
```

Column	Type	Modifiers
id	integer	not null default nextval('taggings_id_seq'::regclass)
tag_id	integer	
taggable_id	integer	
taggable_type	character varying(255)	
tager_id	integer	
tager_type	character varying(255)	
context	character varying(128)	
created_at	timestamp without time zone	

Indexes:

- "taggings_pkey" PRIMARY KEY, btree (id)
- "taggings_idx" UNIQUE, btree (tag_id, taggable_id, taggable_type, context, tager_id, tager_type)

10. State Design



11. User Role Design

User role	Description
Admin	Manager of the website
Member	Active user of the website
Visitor	Passive user of the website

Visitors:

- Can search foods by name or ingredient
- Can comment on recipes if they join in Disqus

Members:

- Can search foods by name or ingredient
- Can comment on recipes if they join in Disqus
- Can add recipes as much as he/she intends
- Can add ingredient tags while adding recipes

Admins:

- Can activate membership applications
- Can inspect comments and recipes before publishing
- Are able to ban a user when needed

12. Technologies

Ruby: We have used Ruby as programming language because of the easiness to both for reading and writing codes.

Ruby on Rails: It's optimized for programmer happiness and sustainable productivity.

Postgresql: It is a powerful, open source object-relational database system.

Github: Used for version control system for more interactive development process.

Heroku: Heroku is very useful for web projects. It provides deployments in minutes, and the tools we need on the project.

Redis: It is an advanced key-value store. It is used for the cases that database is not suitable for our needs by giving another layer to keep data.

Elasticsearch: It provides a distributed, multitenant-capable full text search engine.

Mamcached: Our websites cache system is Mamcached. It's distributed memory object caching system provides us efficiency.

Amazon s3 Service: Used for the asset storage.

Puma: It is faster than the alternatives like thin, webrick, and unicorn. That was the reason why we preferred Puma.

Nginx: System's web server.

Bootstrap: It is powerful mobile first front-end framework for faster and easier web development, we preferred bootstrap as CSS framework.

Jquery: Thanks to Jquery we write less code, do more.

Sublime Text: It's interface, extraordinary features and performance helps us to develop faster and more efficiently.

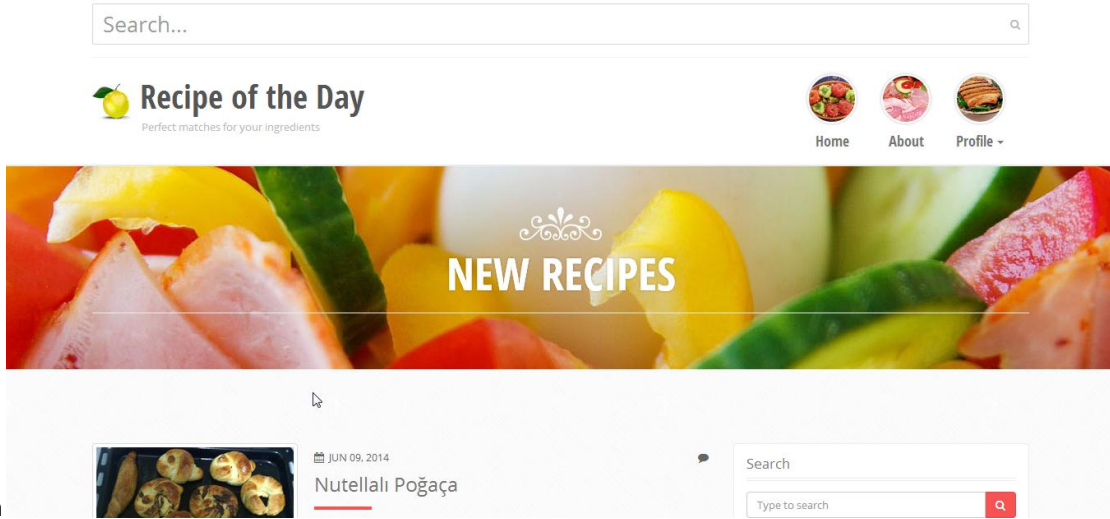
Mac OSX: More easy to develop with ruby in Mac.

Newrelic: As a performance tracking tool Newrelic provides us very detailed performance reports such as average response time.

Disqus: Used as comment management tool, it makes to manage comments very easier.

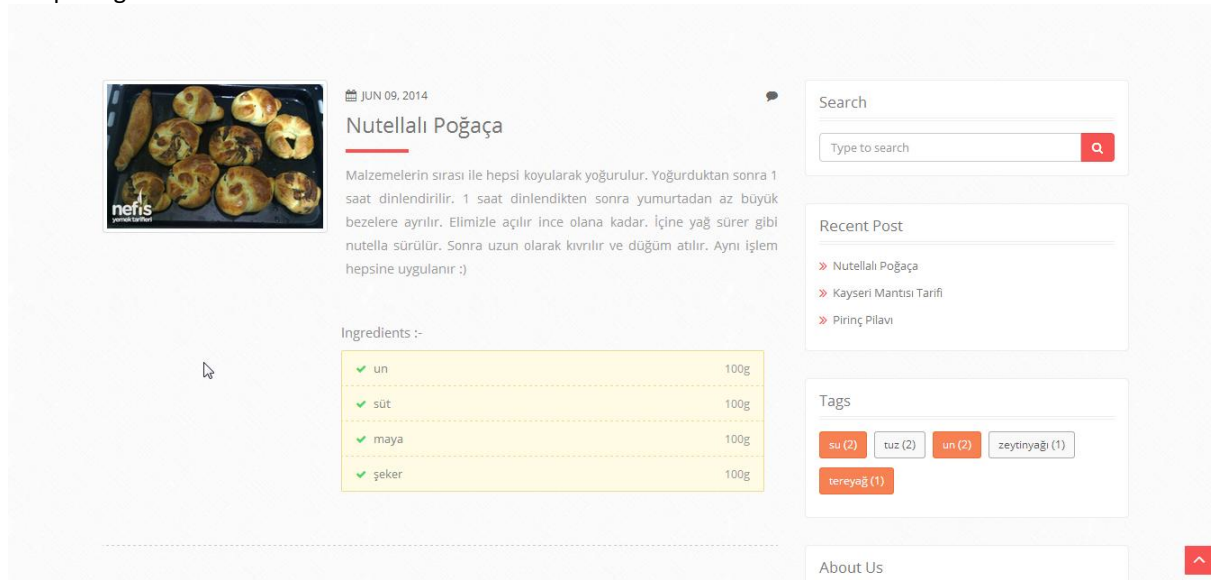
Sentry: We use it as exception tracking tool, by notifying developers constantly it provides us to ability of having control on the system always.

13. Screenshots

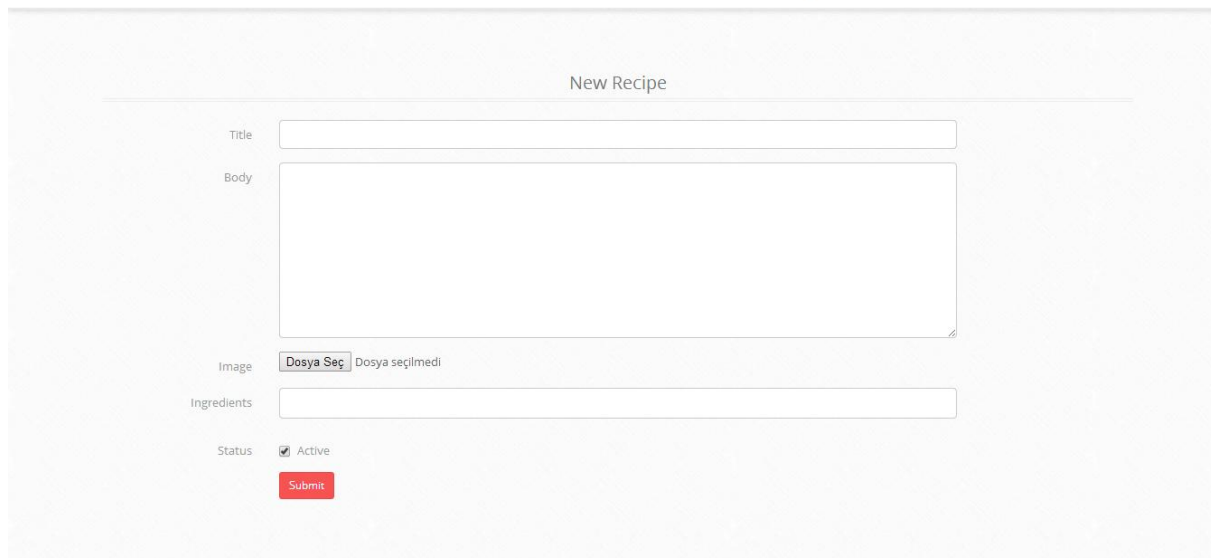


Main screen

Recipe Page



Add Recipe



Ingredient Suggestions

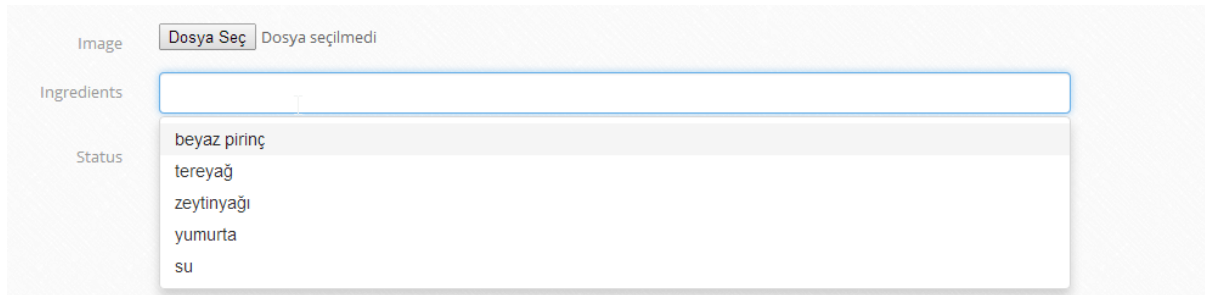


Image Dosya seçilmedi

Ingredients

Status

- beyaz pirinç
- tereyağ
- zeytinyağı
- yumurta
- su

14. Reference

- 1) <http://en.wikipedia.org/wiki/Model-view-controller>
- 2) http://guides.rubyonrails.org/getting_started.html
- 3) http://guides.rubyonrails.org/getting_started.html

15.Coding Standards

- Rails style guide <https://github.com/bbatsov/rails-style-guide>
- Google Javascript Style Guide <https://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml>
- W3C HTML&CSS Standards <http://www.w3.org/standards/webdesign/htmlcss>
- W3C Meta Formats <http://www.w3.org/standards/webarch/metaformats>

16.Appendix

Code Sample:

Recipe.rb

```
class Recipe < ActiveRecord::Base
  belongs_to :user
```

```
  acts_as_taggable
```

```
  default_scope { where(status: 1) }
```

```
  scope :recent, -> { order('id DESC').limit(5) }
```

```
  scope :popular_tags, -> { tag_counts_on(:tags).limit(5).order('taggings_count DESC') }
```

```
  has_attached_file :image, styles: { medium: "300x300>", thumb: "100x100>" }, default_url:
  "blog/blog2.jpg"
```

```
  # validates_attachment :image,
```

```

# :content_type => { :content_type => "image/jpg", :content_type => "image/jpeg", :content_type =>
"image/png" },
# :size => { :in => 0..10000.kilobytes }
validates_attachment_file_name :image, :matches => [/png\Z/, /jpe?g\Z/]
validates :title, :body, :status, presence: true

```

```

def short_desc
  self.body[0..500]
end
end

```

User.rb

```

class User < ActiveRecord::Base
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable

  has_many :recipes
end

```

Recipes_controller.rb

```

class RecipesController < ApplicationController
  before_action :authenticate_user!, only: [:new, :create]
  before_filter :assign_recipe, only: [:show]
  before_action :find_all_tag_list, only: [:new, :edit]
  def index
    @recipes = Recipe
    @recipes = search_recipes(params[:q]) if params[:q]
    @recipes = @recipes.order('updated_at DESC').paginate page: params[:page], per_page: 5
  end

  def new
    @recipe = Recipe.new
  end

  def edit
    @recipe = current_user.recipes.find params[:id]
  end

  def update
    @recipe = current_user.recipes.find(params[:id])

    if @recipe.update(recipe_params)
      redirect_to @recipe
    else
      render 'edit'
    end
  end

  def create

```

```

@recipe = current_user.recipes.new(recipe_params)
if @recipe.save
  redirect_to @recipe, notice: 'Recipe was successfully created.'
else
  render :new
end
end

def show

end

private

def assign_recipe
  @recipe = Recipe.find params[:id]
end

def recipe_params
  params.require(:recipe).permit(:title, :body, :status, :image, :tag_list)
end

def find_all_tag_list
  @all_tag_list = Recipe.tag_counts_on(:tags).collect(&:name).uniq
end

def search_recipes q
  Recipe.select('distinct recipes.*').joins("LEFT JOIN taggings on recipes.id =
taggings.taggable_id").joins("LEFT JOIN tags on tags.id = taggings.tag_id").where('title ilike ? OR body ilike ?
OR tags.name IN (?)', "%#{q}%", "%#{q}%", q.split(','))
end
end

```

Errors_controller.rb

```

class ErrorsController < ApplicationController
  layout :errors
  def error_404
    @not_found_path = params[:not_found]
  end

  def error_500
  end
end

```

index.html.erb

```

<!-- Banner Start -->
<div class="banner padd">
  <div class="container">
    <!-- Image -->
    
    <!-- Heading -->

```

```

<h2 class="white">New Recipes</h2>
<hr>
<div class="clearfix"></div>
</div>
</div>
<!-- Banner End -->

<!-- Inner Content -->
<div class="inner-page padd">
  <!-- Blog Start -->
  <div class="blog">
    <div class="container">
      <div class="row">
        <div class="col-md-8">
          <!-- The new post done by user's all in the post block -->
          <div class="blog-post">
            <% @recipes.each do |recipe| %>
              <%= render recipe %>
            <% end %>
            <% if @recipes.empty? %>
              <h1>There is no suitable recipe for your query. Please try another one.</h1>
            <% end %>
          <!-- Pagination -->
          <%= will_paginate @recipes %>
          <!-- Pagination end-->
        </div>
      </div> <!--/ Main blog column end -->
      <%= render 'right_column' %>
    </div><!--/ Row end -->
  </div>
</div>
<!-- Blog End -->
</div><!-- / Wrapper End -->

```

_recipe.html.erb

```

<div class="entry">
  <!-- Post Images -->
  <div class="blog-img pull-left">
    <%= image_tag recipe.image, class: "img-responsive img-thumbnail" %>
  </div>
  <!-- Meta for this block -->
  <div class="meta">
    <i class="fa fa-calendar"></i>&nbsp;<%= recipe.created_at.strftime('%b %d, %Y')%>
    <span class="pull-right"><i class="fa fa-comment"></i> <a href="<%=
recipe_path(recipe)%>#disqus_thread"></a></span>
  </div>
  <!-- Heading of the post -->
  <h3><a href="<%= recipe_path(recipe)%>"><%= recipe.title %></a></h3>
  <hr /><!-- Horizontal line -->
  <!-- Paragraph -->
  <p><%= recipe.short_desc %></p>

```

```
<div class="clearfix"></div>
</div>
```

```
<script type="text/javascript">
/* ** Just replace ExampleShortname with your shortname ** */
var DisqusShortname = 'recipeoftheday'; // required: replace example with your forum shortname
/* ** Don't make any change for below lines ** */
(function () {
var s = document.createElement('script'); s.async = true;
s.type = 'text/javascript';
s.src = 'http://' + DisqusShortname + '.disqus.com/count.js';
(document.getElementsByTagName('HEAD')[0] ||
document.getElementsByTagName('BODY')[0]).appendChild(s);
})();
</script>
```

Route Files:

Prefix	Verb	URI Pattern	Controller#Action
new_admin_user_session	GET	/admin/login(.:format)	active_admin/devise/sessions#new
admin_user_session	POST	/admin/login(.:format)	active_admin/devise/sessions#create
destroy_admin_user_session	DELETE GET	/admin/logout(.:format)	active_admin/devise/sessions#destroy
admin_user_password	POST	/admin/password(.:format)	active_admin/devise/passwords#create
new_admin_user_password	GET	/admin/password/new(.:format)	active_admin/devise/passwords#new
edit_admin_user_password	GET	/admin/password/edit(.:format)	active_admin/devise/passwords#edit
	PATCH	/admin/password(.:format)	active_admin/devise/passwords#update
	PUT	/admin/password(.:format)	active_admin/devise/passwords#update
admin_root	GET	/admin(.:format)	admin/dashboard#index
batch_action_admin_admin_users	POST	/admin/admin_users/batch_action(.:format)	admin/admin_users#batch_action
admin_admin_users	GET	/admin/admin_users(.:format)	admin/admin_users#index
	POST	/admin/admin_users(.:format)	admin/admin_users#create
new_admin_admin_user	GET	/admin/admin_users/new(.:format)	admin/admin_users#new
edit_admin_admin_user	GET	/admin/admin_users/:id/edit(.:format)	admin/admin_users#edit
admin_admin_user	GET	/admin/admin_users/:id(.:format)	admin/admin_users#show
	PATCH	/admin/admin_users/:id(.:format)	admin/admin_users#update
	PUT	/admin/admin_users/:id(.:format)	admin/admin_users#update
	DELETE	/admin/admin_users/:id(.:format)	admin/admin_users#destroy
admin_dashboard	GET	/admin/dashboard(.:format)	admin/dashboard#index
batch_action_admin_recipes	POST	/admin/recipes/batch_action(.:format)	admin/recipes#batch_action

admin_recipes	GET	/admin/recipes(.:format)	admin/recipes#index
	POST	/admin/recipes(.:format)	admin/recipes#create
new_admin_recipe	GET	/admin/recipes/new(.:format)	admin/recipes#new
edit_admin_recipe	GET	/admin/recipes/:id/edit(.:format)	admin/recipes#edit
admin_recipe	GET	/admin/recipes/:id(.:format)	admin/recipes#show
	PATCH	/admin/recipes/:id(.:format)	admin/recipes#update
	PUT	/admin/recipes/:id(.:format)	admin/recipes#update
	DELETE	/admin/recipes/:id(.:format)	admin/recipes#destroy
batch_action_admin_users	POST	/admin/users/batch_action(.:format)	admin/users#batch_action
admin_users	GET	/admin/users(.:format)	admin/users#index
	POST	/admin/users(.:format)	admin/users#create
new_admin_user	GET	/admin/users/new(.:format)	admin/users#new
edit_admin_user	GET	/admin/users/:id/edit(.:format)	admin/users#edit
admin_user	GET	/admin/users/:id(.:format)	admin/users#show
	PATCH	/admin/users/:id(.:format)	admin/users#update
	PUT	/admin/users/:id(.:format)	admin/users#update
	DELETE	/admin/users/:id(.:format)	admin/users#destroy
new_user_session	GET	/users/sign_in(.:format)	devise/sessions#new
user_session	POST	/users/sign_in(.:format)	devise/sessions#create
destroy_user_session	DELETE	/users/sign_out(.:format)	devise/sessions#destroy
user_password	POST	/users/password(.:format)	devise/passwords#create
new_user_password	GET	/users/password/new(.:format)	devise/passwords#new
edit_user_password	GET	/users/password/edit(.:format)	devise/passwords#edit
	PATCH	/users/password(.:format)	devise/passwords#update
	PUT	/users/password(.:format)	devise/passwords#update
cancel_user_registration	GET	/users/cancel(.:format)	devise/registrations#cancel
user_registration	POST	/users(.:format)	devise/registrations#create
new_user_registration	GET	/users/sign_up(.:format)	devise/registrations#new
edit_user_registration	GET	/users/edit(.:format)	devise/registrations#edit
	PATCH	/users(.:format)	devise/registrations#update
	PUT	/users(.:format)	devise/registrations#update
	DELETE	/users(.:format)	devise/registrations#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
root	GET	/	welcome#index
about	GET	/about(.:format)	welcome#about

File/Folder	What is it good for
-------------	---------------------

app/	This is the folder that contains model, view and the controller.
app/model	Contains all models for the project.
app/view	Includes all HTML files inside a folder. These files are named after an corresponding action inside the controller. Additionally, these files are grouped into folders with the controller name.
app/contr oller	Holds the logic for the Rails application.
db/	All database related files go into this folder.
db/migrat e	Has all the migrations that inside in the project.
config/	As the names suggest, it has all the necessary configuration files of your Rails application. Localizations, routes and all basic files can be found here.
public/	All the static files (files that do not change dynamically) CSS or JavaScript files are inside this folder.
public/ima ges	All static Images
public/java scripts	Rails will automatically look inside these to find the proper js files.
public/styl esheets	Rails will automatically look inside this to find the proper css files.
script/	Holds scripts for Rails that provide a variety of tasks. These scripts link to another file where the "real" files are.
log/	Log files from the application.
test/	All files for testing the application.
doc/	Documentation for the current Rails application.
lib/	Extended modules for the application.
vendor/	3rd-party plug ins will be found in this folder.
tmp/	Temporary files
README	Basic guide for others on how to setup the application, what specialities it has or what to be careful about.
Rakefile	Handles all the Rake tasks inside the application.