

Kitchen

Magician

SW Engineering CSC648/848

FALL 2020

Team 01

Kevin Wei (Team Lead) : kwei3@mail.sfsu.edu

Allen Sun (Backend Lead)

Paul Asu (Github Master)

Nicole Pang (Developer)

Kevin Ortiz (Developer)

Jeff Cheng (Frontend Lead)

Milestone 4

History Table

Version	Date
M1V1	10/01/2020
M1V2	10/08/2020
M2V1	11/01/2020
M2V2	11/18/2020
M3V1	11/19/2020
M3V2	11/20/2020
M4V1	12/10/2020

Table of Contents

Section 1: Product Summary	3
Section 2: Usability Testing Plan	5
Section 3: QA Test Plan	10
Section 4: Code Review	18
Section 5: Self-Check for Best Practices	27
Section 6: List of Non-Functional Requirements	28

Section 1: Product Summary

Product Name: Kitchen Magician

Priority 1 Functions:

1. Unregistered Users will be able to register for an account.
2. Unregistered Users can register for an account.
3. Unregistered Users will provide an email and password when registering for an account.
4. Unregistered Users will see links to all the features our website has to offer on every page via the navigation bar and footer.
5. Unregistered Users can search for and view all recipes, including those trending on the home page.
6. Registered users can do everything unregistered users can, except create an account, because they already have one.
7. Registered Users will be able to log in via their username and password.
8. Registered Users will be able to favorite recipes and comment on them.
9. Registered Users will be able to make their own recipes.
10. Registered users will be able to see their favorite recipes and their own recipes on their profile page.
11. Registered Users will be able to view and join food groups to become group users, doing so will give them access to that food groups forum.
12. Groups Users can do everything registered users can, with the addition that they can comment on food group forums they are a part of to initial discussion.
13. Admins can do everything group users can, but also have access to the administrative page.
14. Admins can delete any comment, recipe or group.
15. Recipe's will all have a title, multiple categories to filter them and provide detail, multiple ingredients, multiple steps, and maybe a photo.
16. Recipes that are in the top five user favorited recipes will be labeled trending, in which it will be showcased on the home page.

What is unique in our product: What makes our product unique is that we do not rely on “professional chefs” to curate our entire recipe selection. Our selection is curated not only by professionals, but also by everyday folk, who have been making their signature recipes for generations. Professional chefs do know their craft, but they don’t always get it authentic. How many times have you seen an Italian mortified at the sight of an inauthentic carbonara, or a Spaniard with a paella. Sometimes, the real professionals are people like you. Someone who has been making their family recipe the traditional way as long as they can remember. Anyone will be able to make an account and join our community to showcase what authentic recipes are to them. The most popular recipes chosen by that very community will also be on showcase on the homepage for everyone to see. For those that are even more passionate, they can join our various

food groups to have in-depth discussions about this medium that brings us all together in food. Our website is built on our community and we want to push that home. Our website is driven by you.

URL Link to Product: www.allensun623.com

Section 2: Usability Testing Plan

1) Trending Features

a) Test Objectives:

- i) Trending recipes is a feature which is key to let users know what the most popular dishes that the Admin recommends to the users. 5 different recipes will be displayed in a sliding animation where users can see what is available. We added this feature because there are times when people visit a recipe website where they are simply unsure of what recipe to follow. This allows these users to get started on their next dish.

b) Test Description:

- i) System Setup: the tests were conducted on a Windows 10 64-bit machine Build 19041.572, as it is the most up-to-date version of Windows OS at the time of testing. The browser used is Google Chrome Version 87.0.4280.88 as it is the most up-to-date version of Google Chrome at the time of testing.
- ii) Starting Point: in order to reach the trending recipes, a user will need to scroll down to the bottom of the Home page.
- iii) Intended Users: The users who will find this feature helpful will be the users who visit Kitchen Magician without any idea of what to cook. The great thing about the feature is that it allows users to get their ideas flowing on what dish they want to eat.
- iv) URL of the system to be tested: <https://www.allensun623.com/> this URL provides the link to the trending feature which is located on the Home page.

c) Usability Task Description:

- i) Visit <https://www.allensun623.com/>.
- ii) Scroll down on the Home page.
- iii) View the top 5 trending recipes.
- iv) Search a recipe keyword using the search bar.
- v) View recipe details.
- vi) Follow instructions to make the dish.

d) Questionnaire:

- i) The website made it easy to find the trending recipes.
Strongly agree Agree Neutral Disagree Strongly Disagree
- ii) I was able to get an idea of what dish to cook.
Strongly agree Agree Neutral Disagree Strongly Disagree
- iii) The recipes on the trending slide were easy to find.
Strongly agree Agree Neutral Disagree Strongly Disagree

2) Group Features

a) Test Objectives:

- i) The group feature allows Registered Users to join pre-existing groups named after the diets they cater to. A Registered User is able to join more than one group. After joining a group, the members may post to a group forum.
All group members of the same group can view and post on the corresponding group forum. We are testing this feature to see if the user is able to successfully join and leave a group or multiple groups, view and post to their designated group form.

b) Test Description:

- i) System Setup: the tests were conducted on a Windows 10 64-bit machine Build 19041.572, as it is the most up-to-date version of Windows OS at the time of testing. The browser used is Google Chrome Version 87.0.4280.88 as it is the most up-to-date version of Google Chrome at the time of testing.
- ii) Starting Point: The starting point is the group page of the Kitchen Magician website.
- iii) Intended Users: Registered users who want to connect with other registered users that share the same diet.
- iv) URL of system to be tested: <https://www.allensun623.com/groups/>.

c) Usability Task Description:

- i) View available groups.
- ii) Join a group.
- iii) View joined groups.
- iv) View group members of a joined group.
- v) Post on the group discussion.
- vi) View existing posts on the group discussion.
- vii) Leave a group.

d) Likert Questionnaire:

- i) It was easy to join a group.
Strongly agree Agree Neutral Disagree Strongly Disagree
- ii) I understood the differences between each group.
Strongly agree Agree Neutral Disagree Strongly Disagree
- iii) I was able to successfully navigate to the group discussion form on my own.
Strongly agree Agree Neutral Disagree Strongly Disagree
- iv) I was able to successfully post to the group discussion.
Strongly agree Agree Neutral Disagree Strongly Disagree

3) Search Function

- a) Test Objectives:
 - i) The search function allows users to interface with available recipes stored on the database using search query. Specifically, users shall be able to retrieve a set of recipes based on their search criteria, and if no criteria has been specified, the entire database of recipes will be returned.
 - b) Test Description:
 - i) System Setup: the tests were conducted on a Windows 10 64-bit machine Build 19041.572, as it is the most up-to-date version of Windows OS at the time of testing. The browser used is Google Chrome Version 87.0.4280.88 as it is the most up-to-date version of Google Chrome at the time of testing.
 - ii) Starting Point: The starting point is the group page of the Kitchen Magician website.
 - iii) Intended Users: Registered users who want to connect with other registered users that share the same diet.
 - iv) URL of System to be Tested: <https://www.allensun623.com/search/>.
 - c) Usability Task Description:
 - i) View all available recipes with no criteria specified.
 - ii) Search for recipes matching given criteria.
 - d) Likert Questionnaire:
 - i) It was easy to search for recipes.
Strongly agree Agree Neutral Disagree Strongly Disagree
 - ii) The Search/Results page was easy to understand.
Strongly agree Agree Neutral Disagree Strongly Disagree
 - iii) I could understand what the extra search filters did to my search.
Strongly agree Agree Neutral Disagree Strongly Disagree
- 4) Recipe Creation
- a) Test Objectives:
 - i) To ensure a logged-in registered user can successfully trigger the corresponding feature, submit all required input, and store relevant data in the database.
 - b) Test Description:
 - i) System Setup: the tests were conducted on a Windows 10 64-bit machine Build 19041.572, as it is the most up-to-date version of Windows OS at the time of testing. The browser used is Google Chrome Version 87.0.4280.88 as it is the most up-to-date version of Google Chrome at the time of testing.
 - ii) Starting Point: The starting points are the home page of the Kitchen Magician website, and the search box in the navigation bar on any page.

- iii) Intended Users: All users who want to conduct a search of recipes, with and without search criteria.
 - iv) URL of system to be tested: <https://www.allensun623.com/groups/>.
 - c) Usability Task Description:
 - i) Create a recipe with one photo upload.
 - ii) Verify successful recipe creation through search.
 - iii) Verify that the process of recipe creation is intuitive and easy, i.e users are aware where the feature is found and/or triggered.
 - d) Likert Questionnaire:

i)	It is clear which data are necessary, and which data are optional on the page.	Strongly agree	Agree	Neutral	Disagree	Strongly Disagree
ii)	It was intuitive to provide necessary data to create a recipe (the instructions are clear).	Strongly agree	Agree	Neutral	Disagree	Strongly Disagree
iii)	It was easy to upload a photo for a recipe.	Strongly agree	Agree	Neutral	Disagree	Strongly Disagree
iv)	Overall, it was easy to create a recipe.	Strongly agree	Agree	Neutral	Disagree	Strongly Disagree
- 5) Recipe Contents: i.e. favorite and comments
- a) Test Objectives:
 - i) To ensure a Registered User who is logged-in can successfully add a comment and favorite to a recipe. The Register User who is logged in should be able to favorite and add comment(s) to any recipes with ease.
 - b) Test Description:
 - i) System Setup: the tests were conducted on a Windows 10 64-bit machine Build 19041.572, as it is the most up-to-date version of Windows OS at the time of testing. The browser used is Google Chrome Version 87.0.4280.88 as it is the most up-to-date version of Google Chrome at the time of testing.
 - ii) Starting Point: The starting point is the recipe page of the Kitchen Magician website.
 - iii) Intended Users: Registered Users who want to favorite and add comments to a recipe that he/she likes.
 - iv) URL of system to be tested: <https://www.allensun623.com/recipe/1>
 - c) Usability Task Description:
 - i) Once they have signed in, the approval of which triggers an alternate navigation panel.
 - ii) Log in to the account.

- iii) Search recipes using the search recipe bar.
- iv) Go to the recipe page from the search recipe result page.
- v) And then favorite the recipe on the top of the page.
- vi) Then go to the bottom of the recipe page where you can add a comment.
- d) Likert Questionnaire:
 - i) The add comments/favorite page was easy to understand.
Strongly agree Agree Neutral Disagree Strongly Disagree
 - ii) It was easy to favorite a recipe.
Strongly agree Agree Neutral Disagree Strongly Disagree
 - iii) It is clear which data are necessary, and which data are optional on the page.
Strongly agree Agree Neutral Disagree Strongly Disagree
 - iv) It was easy to add comments to a recipe.
Strongly agree Agree Neutral Disagree Strongly Disagree
 - v) It was easy to upload a photo for a recipe.
Strongly agree Agree Neutral Disagree Strongly Disagree
 - vi) Overall, I was able to add comments and favorite recipes within 10 seconds.
Strongly agree Agree Neutral Disagree Strongly Disagree

Section 3: QA Test Plan

Number	Test Title	Description	Test Input	Expected Output	Pass/Fail
1	Test like name field	Test % in LIKE name field	"cake"	Get 5 results with word "cake"	Pass
2	Test keywords split	Test split words by space	"turkey thanksgiving"	Get keywords as a list ['turkey', 'thanksgiving']	Pass
3	Test keywords split	Test split words by comma	"turkey, thanksgiving"	Get keywords as a list ['turkey', 'thanksgiving']	Pass
4	Test Query	Test a query with a condition in table recipe_favorite	SELECT * FROM recipe_favorite WHERE user_id =1;	Get 7 results	Pass
5	Test Query	Test a query with a condition in table recipe	SELECT * FROM recipe WHERE user_id =1;	Get 10 results	Pass

1. Test Plan for Number 1

- a. Test Objectives:
 - i. What is being tested: search query using a single keyword
 - ii. What is not being tested: search query using multiple keywords, including a single space-separated keyword phrase
- b. Setup of HW and SW: cloud instance on GCP, MySQL running
- c. Test environment: supported desktop browsers
 - i. Human resource needs: input sample data in the search box
 - ii. Expected time to complete: within constraints set by the nonfunctional requirements, i.e. within seconds
- d. Risks, contingencies: Invalid input should return null, i.e. no results
- e. End report format requirements
 - i.

Number	Description	Test Input	Expected Output	Pass/Fail
--------	-------------	------------	-----------------	-----------

1a	Test % in LIKE name field	“shrimp”	get some results with word “shrimp”	Pass
1b	Test % in LIKE name field	“salmon”	get some results with word “salmon”	Pass
1c	Test % in LIKE name field	“crab”	get some results with word “crab”	Pass

f. Actual test sequence, test input, and test expected results

i. Test sequence:

1. Type “turkey thanksgiving”, press the Enter key (or Find a Recipe, if desired, but does not affect functionality), and, if available, query results should indicate results containing the key word specified
2. Test input: turkey thanksgiving
3. Test expected results: A number of recipes containing the keyword shrimp, if any such recipes exist in the database.

g. Final Report

Number	Description	Test Input	Expected Output	Pass/Fail
1a	Test % in LIKE name field	“shrimp”	get some results with word “shrimp”	Pass
1b	Test % in LIKE name field	“salmon”	get some results with word “salmon”	Pass
1c	Test % in LIKE name field	“crab”	get some results with word “crab”	Pass

2. Test Plan for Number 2

a. Test Objectives:

- i. What is being tested: search query using a single keyword

- ii. What is not being tested: search query using multiple keywords, including a single space-separated keyword phrase
- b. Setup of HW and SW: cloud instance on GCP, MySQL running
- c. Test environment: supported desktop browsers
 - i. Human resource needs: input sample data in the search box
 - ii. Expected time to complete: within constraints set by the nonfunctional requirements, i.e. within seconds
- d. Risks, contingencies: Invalid input should return null, i.e. no results
- e. End report format requirements
 - i.

Number	Description	Test Input	Expected Output	Pass/Fail
1a	Test split words by space	"turkey thanksgiving"	get some results with the words turkey and thanksgiving	Pass
1b	Test split words by space	"vegan donut"	get some results with the words vegan and donut	Pass
1c	Test split words by space	"pancake buttermilk"	get some results with the words pancake and buttermilk	Pass

- f. Actual test sequence, test input, and test expected results
 - i. Test sequence:
 1. Type "turkey thanksgiving", press the Enter key (or Find a Recipe, if desired, but does not affect functionality), and, if available, query results should indicate results containing the key word specified
 2. Test input: turkey thanksgiving
 3. Test expected results: A number of recipes containing the keywords turkey and thanksgiving, if any such recipes exist in the database.
- g. Final Report

Number	Description	Test Input	Expected Output	Pass/Fail
--------	-------------	------------	-----------------	-----------

1a	Test split words by space	"turkey thanksgiving"	get some results with the words ""turkey thanksgiving""	Pass
1b	Test split words by space	"vegan donut"	get some results with the words ""turkey thanksgiving""	Pass
1c	Test split words by space	"pancake buttermilk"	get some results with the words ""turkey thanksgiving""	Pass

3. Test Plan for Number 3

- a. Test objectives:
 - i. What is being tested: search query using two keywords separated by a comma
 - ii. What is not being tested: search query using single keyword, using two keywords separated by a space, using two keywords separated by nothing
- b. Setup of HW and SW: Cloud instance on GCP, MySQL
- c. Test environment: Google Chrome Version 87.0.4280.88
 - i. Human resource needs: Input sample data in the search box
 - ii. Expected time to complete: Within 10 seconds.
- d. Risks, contingencies:
 - i. Invalid input should return null, i.e. no results
- e. End report format requirements
 - i.

Number	Description	Test Input	Expected Output	Pass/Fail
1a	Test split words by comma	"chicken, pie"	Recipe results with the words	Pass

			chicken and pie	
1b	Test split words by comma	“vegan, donut”	Recipe results with the words vegan and donut	Pass
1c	Test split words by comma	“pancake, buttermilk”	Recipe results with the words vegan and donut.	Pass

f. Actual test sequence, test input, and test expected results

i. Test sequence:

1. Type “chicken, pie”, press the Enter key (or Find a Recipe, if desired, but does not affect functionality), and, if available, query results should indicate results containing the key word specified
2. Test input: chicken, pie
3. Test expected results: A number of recipes containing the keywords chicken or pie, if any such recipes exist in the database.

g. Final Report

i.

Number	Description	Test Input	Expected Output	Pass/Fail
1a	Test split words by comma	"chicken, pie"	Recipe results with the words chicken and pie	Pass
1b	Test split words by comma	“vegan, donut”	Recipe results with the words vegan and donut	Pass
1c	Test split words by comma	“pancake, buttermilk”	Recipe results with the words vegan and donut.	Pass

1. Test Plan for Number 4

a. Test Objectives:

- i. What is being tested: Testing a query with a condition in table recipe_favorite
- ii. What is not being tested: Testing a query with two conditions in table recipe_favorite, testing a query with more than two conditions in table recipe_favorite
- b. HW and SW Setup: Cloud Instance on Google Cloud Platform, MySQL
- c. Test Environment: Windows 10 version 1909
 - i. Human resource needs: Input MySQL code into terminal
 - ii. Expected time to complete: Within 10 seconds.
- d. Risks, Contingencies: Invalid input should return null or nothing
- e. End Report Format Requirements
 - i.

Number	Description	Test Input	Expected Output	Pass/Fail
1a	Test a query with a condition in table recipe_favorite	“SELECT * FROM recipe_favorite WHERE user_id =1;”	Get 4 results	Pass
1b	Test a query with a condition in table recipe_favorite	“SELECT * FROM recipe_favorite WHERE user_id =2;”	Get 1 result	Pass
1c	Test a query with a condition in table recipe_favorite	“SELECT * FROM recipe_favorite WHERE user_id =6;”	Get 3 results	Pass

- f. Actual Test Sequence
 - i. Test Sequence
 1. Type “SELECT * FROM recipe_favorite WHERE user_id =1;” in the terminal and press the Enter key. if available, the query results should return results with the user_id that is equal to 1
 2. Test input: “SELECT * FROM recipe_favorite WHERE user_id =1;”
 3. Test expected results: A list of recipes where their user_id is equal to 1
- g. Final Report
 - i.

Number	Description	Test Input	Expected Output	Pass/Fail
--------	-------------	------------	-----------------	-----------

1a	Test a query with a condition in table recipe_favorite	“SELECT * FROM recipe_favorite WHERE user_id =1;”	Get 4 results	Pass
1b	Test a query with a condition in table recipe_favorite	“SELECT * FROM recipe_favorite WHERE user_id =2;”	Get 1 result	Pass
1c	Test a query with a condition in table recipe_favorite	“SELECT * FROM recipe_favorite WHERE user_id =6;”	Get 3 results	Pass

Test plan for Number 5

- a. Test Objectives:
 - i. What is being tested: A test query with a condition in a table recipe that can return 10 results.
 - ii. What is not being tested: Testing a query with letters.
- b. HW and SW setup: Cloud instance on GCP, MySQL
- c. Test environment: macOS Mojave version 10.14.6
 - i. Human resource needs: Input MySQL code into terminal
 - ii. Expected time to complete: Within 10 seconds
- d. Risk, Contingencies: Any input that does not have a registered user_id shall output “no result”
- e. End Report Format Requirements:
 - i.

Number	Description	Test Input	Expected Output	Pass/Fail
1a	Test a query with a condition in table recipe	“SELECT * FROM recipe WHERE user_id =1;”	The query should output 10 results	pass
1b	Test a query with a condition in table	“SELECT * FROM recipe WHERE user_id	The query should output 0	pass

	recipe	=2;”		
1c	Test a query with a condition in table recipe	“SELECT * FROM recipe WHERE user_id =8;”	The query should output 1	Pass

f. Actual Test Sequence:

i. Test Sequence

1. Input SELECT * FROM recipe WHERE user_id =1; in terminal and press ENTER. This will give the amount of recipes that the user_id =1 has created. The information on this table will include: name of recipe, quantity_serve, created, uopated, id, user_id.
2. Input SELECT * FROM recipe WHERE user_id =2; in terminal and press ENTER will give a 0 result output. A 0 result output means the user has not created any recipes


g. Final Report:


i.

Number	Description	Test Input	Expected Output	Pass/fail
1a	Test a query with a condition in table recipe	“SELECT * FROM recipe WHERE user_id =1;”	The query should output 10 results	Pass
1b	Test a query with a condition in table recipe	“SELECT * FROM recipe WHERE user_id =2;”	The query should output 0	Pass
1c	Test a query with a condition in table recipe	“SELECT * FROM recipe WHERE user_id =8;”	The query should output 1	Pass

Section 4: Code Review

Internal Code Review Request:

 Allen Sun
Thu 12/10/2020 1:33 PM
To: Kevin Sun Wei; Kevin Carlos Ortiz; Jeff C Cheng; Nicole Leilani Pang; Paul Lane Asu

 base.html
9 KB

Hi team,

Could you help with code review? The codes "base.html" is mainly for header and footer.

Code link below (attached is the same code):

https://github.com/CSC-648-SFSU/csc-648-848-04-jose-fall-2020-01/blob/dev_horizontal/application/KitchenMagician/kitchen_magician/home/templates/base.html

Thanks,
Allen

Original Code:

```

1  {% load static %}
2
3  <!DOCTYPE html>
4  <html lang="en">
5      <head>
6          <meta charset="UTF-8" />
7          <meta name="viewport" content="width=device-width, initial-scale=1.0" />
8          <link href="https://fonts.googleapis.com/css2?family=Lato:wght@300&display=swap" rel="stylesheet"/>
9          <link rel="icon" href="{% static 'home/images/icon.png' %}" />
10         <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css" integrity="sha384-TX8t27EcR36/INH7mQxvncDaySuTKz4rEkgTXeMed4M8jlfIDPvg6uqKT2xXr2" crossorigin="anonymous" />
11         <link rel="stylesheet" type="text/css" href="{% static 'home/css/base.css' %}" />
12         {% block css %}{% endblock css %}
13
14         {% if title %}
15         <title>Kitchen Magician - {{ title }}</title>
16         {% else %}
17         <title>Kitchen Magician</title>
18         {% endif %}
19     </head>
20
21     <body>
22         <header id="header">
23             <div class="container">
24                 <a href="{% url 'home' %}" class="KitchenMagician">
25                     
26                 </a>
27                 <ul id="nav-link">
28                     <li class="nav-link nav-link-underline"><a href="{% url 'about' %}" id="nav-about">About</a></li>
29                     <li class="nav-link nav-link-underline"><a href="{% url 'groups' %}" id="nav-groups">Groups</a></li>
30                     <li id="search">
31                         <form action="{% url 'search' %}" method="POST" id="search-form">
32                             <!-- Authentication Token -->
33                             {% csrf_token %}
34                             <div id="search-box" class="search-box-block">
35                                 <input id="search-text-id" class="search-text" type="text" name="keywords" placeholder="Search for a recipe">
36                                 <button type="submit" class="search-btn fas fa-search"></button>
37                             </div>
38                         </form>
39                     </li>
40                     <div id="nav-link-user">
41                         {% if user.is_authenticated %}
42                         <li id="account" class="nav-item btn-group">
43                             <button type="button" class="btn btn-secondary dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
44                                 Hi, {{ user.get_username }}
45                             </button>
46                             <div class="dropdown-menu dropdown-menu-right">
47                                 <a class="dropdown-item" href="{% url 'create_recipe' %}">Create a recipe</a>
48                                 <a class="dropdown-item" href="{% url 'user_profile' username=user.username %}">Profile</a>
49                                 <!-- <a class="dropdown-item" href="#">Groups</a> -->
50                                 <!-- <a class="dropdown-item" href="#">Recipes</a> -->
51                                 <!-- <a class="dropdown-item" href="#">Favorites</a> -->
52                                 <!-- redirect to home page after logout -->
53                                 <a class="dropdown-item" href="{% url 'logout' %}?next=/">Log out</a>
54                             </div>
55                         </li>
56                         <!-- <a class="dropdown-item" href="{% url 'logout' %}?next=/">Log out</a> -->
57                     {% else %}
58                     <li class="nav-link nav-link-underline"><a href="{% url 'login' %}" id="nav-login">Log in</a></li>
59                     <li class="nav-link nav-link-underline"><a href="{% url 'signup' %}" id="nav-signup">Sign up</a></li>
60                     {% endif %}
61                 </ul>
62             </div>
63         </header>
64
65         {% block content %}{% endblock content %}
66
67     </body>
68
69     <!-- <a href="#">Footer content -->
70
71     <footer id="main-footer">
72         <div id="footer-container">
73             <div class="container">
74                 <div id="footer-logo-content">
75                     <div id="footer-logo">
76                         
77                     </div>
78                     <div id="footer-learn-more" class="footer-content-block">
79                         <p class="footer-content-title">Learn More</p>
80                         <div class="footer-content-block-line"></div>
81                         <ul class="ul-footer-content-block">
82                             <li class="li-footer-content-item">
83                                 <a class="footer-content-text" href="{% url 'groups' %}">Groups</a>
84                             </li>
85                             <li class="li-footer-content-item">
86                                 <a class="footer-content-text" href="{% url 'search' %}">Recipes</a>
87                             </li>
88                         </ul>
89                     </div>
90                     <div id="footer-connect" class="footer-content-block">
91                         <p class="footer-content-title">Connect</p>
92                         <div class="footer-content-block-line"></div>
93                         <ul class="ul-footer-content-block">
94                             <li class="li-footer-content-item">
95                                 <a class="footer-content-text" href="{% url 'contact_us' %}">Contact us</a>
96                             </li>
97                             <li class="li-footer-content-item" id="li-footer-content-medias">
98                                 <p class="footer-content-text">Follow us</p>
99                                 <div id="footer-content-medias">
100                                     <a href="#" class="social-icon">

```

```

101         <i class="fab fa-facebook-f"></i>
102     </a>
103     <a href="#" class="social-icon">
104         <i class="fab fa-twitter"></i>
105     </a>
106     <a href="#" class="social-icon">
107         <i class="fab fa-google"></i>
108     </a>
109     <a href="#" class="social-icon">
110         <i class="fab fa-linkedin-in"></i>
111     </a>
112 </div>
113 </li>
114 <!-- <li class="li-footer-content-item" id="li-footer-content-signup">
115     <p>Join Us</p>
116     <a href="{% url 'signup' %}" id="footer-signup">Sign Up</a>
117 </li> -->
118 </ul>
119 </div>
120 </div>
121 <div id="footer-break-line"></div>
122 <ul id="footer-policy">
123     <li class="footer-policy-item">
124         <a href="{% url 'about' %}">About KitchenMagician</a>
125     </li>
126     <li class="footer-policy-item footer-policy-item-line"></li>
127     <li class="footer-policy-item">
128         <a href="{% url 'term of use' %}">Terms of Service</a>
129     </li>
130     <li class="footer-policy-item footer-policy-item-line"></li>
131     <li class="footer-policy-item">
132         <a href="{% url 'privacy policy' %}">Privacy Statement</a>
133     </li>
134 </ul>
135 <p>©copy; 2020 Kitchen Magician Inc.</p>
136 </div>
137 </div>
138 </footer>
139 <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-Dfxd22htPH0ls5S5SnCTpuj/zy4C+06pmoFVy3BMVbNe+IbbVYUew+0rCkxRkfj" crossorigin="anonymous"></script>
140 <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKA1Zp3H66LZ28H1PoYlFhbGU+6B2p6G7nlu735Sk7lN" crossorigin="anonymous">
141 <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js" integrity="sha384-w1Q4orYjBQndcko6MimVbzY0tqp4pbW4LZ7lr30WkzQv/r/aMKhXdBNNb5092v7s" crossorigin="anonymous">
142 <script src="https://kit.fontawesome.com/a076d05399.js"></script>
143 {% block script %}{% endblock script %}
144 </body>
145
146 </html>

```

- Code Review Response



Jeff C Cheng

Thu 12/10/2020 1:50 PM

To: Allen Sun



Overall, the code is highly readable.

Some comments about what the code achieves might have been preferable, i.e.

```
{% if title %}
```

```
...
```

```
{% else %}
```

```
...
```

A similar concern is found in the nav-link-user div, i.e. `Hi, {{ user.get_username }}`.

More specifically, while it is clear to coders who know Django that the former is an optional title, passed as an argument through view, and the latter is default message the user sees, some additional relevant documentation could be of use for understanding and segregating the code.

Independent of readability, spacing seems to be different through the document, sometimes no empty lines in between, sometimes one line, sometimes two lines.

Hope this helps!



Jeff C Cheng

Thu 12/10/2020 3:49 PM

To: Allen Sun



base-commented.html

10 KB

Hi Allen,

I have added some comments to the base.html, renamed as base-commented.html, for your reference.

Overall, your code offers high readability, with consistent layout throughout.

However, as a reader, some coding choices have chosen without explanations being offered.

It might be good to add them to aid understanding.

See comments in base-commented.html.

Sincerely,

Cheng, Jeff



Nicole Leilani Pang

Thu 12/10/2020 2:14 PM

To: Allen Sun



Hello Allen,

I think formatting the document (Shift + Alt + F) would make the code more readable and the document format more consistent. Adding more comments on certain parts of the document will also help other team members understand what the more vital parts of code are doing.

Best,
Nicole Pang

...



Nicole Leilani Pang

Thu 12/10/2020 3:41 PM

To: Allen Sun



base.html

5 KB

Hey Allen,

I forgot to include the file with my comments, so I have attached it to this email.
Thanks again.

Best,
Nicole Pang

...



Kevin Carlos Ortiz

Thu 12/10/2020 3:36 PM

To: Allen Sun



base.html

9 KB

Good afternoon,

I just finished up the code review on your "base.html" file. I added my comments towards the bottom.

I did my best to give you the most honest opinion. I hope my comments help.

Best,



Paul Lane Asu

Thu 12/10/2020 3:55 PM

To: Allen Sun

hello Allen,

I review your base.html code and below are comments.

1. on lines 33, 50 - 53, line 57, and line 114, I think you should remove all the unused lines of code you commented on. this is just to make your code more readable.
2. on lines 10, and 134 - 141, you can break \ or use multiple lines of codes instead of having just one long line,
3. other than that, everything including the formatting looks good to me.

thank you
Paul Asu.

...

Code Review with comments:

```

1  <!-- overall the content is highly readable with proper indentation - commented by Jeff -->
2  <!-- however, some modifications might be helpful to further readability - commented by Jeff -->
3
4  {% load static %}
5
6  <!DOCTYPE html>
7  <html lang="en">
8  <head>
9      <meta charset="UTF-8" />
10     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
11     <link href="https://fonts.googleapis.com/css2?family=Lato:wght@300&display=swap" rel="stylesheet"/>
12     <link rel="icon" href="{% static 'home/images/icon.png' %}" />
13     <!-- you can break \ or use multiple lines of codes instead of having just one long line - commented by Paul -->
14     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css" integrity="sha384-TX8t27EcRE3e/1hU7zmQxvncDySuIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" crossorigin="anonymous" />
15     <link rel="stylesheet" type="text/css" href="{% static 'home/css/base.css' %}" />
16
17     <!-- why do you have an empty block content? maybe some explanation is appropriate - commented by Jeff -->
18     {% block css %}{% endblock css %}
19
20     <!-- while the functionality of the if-else is self-explanatory, where the parameter from which it is passed should probably be noted, i.e. from def base in views.py - commented by Jeff -->
21     {% if title %}
22     <title>Kitchen Magician - {{ title }}</title>
23     {% else %}
24     <title>Kitchen Magician</title>
25     {% endif %}
26 </head>
27
28 <body>
29     <header id="header">
30         <div class="container">
31             <a href="{% url 'home' %}" class="KitchenMagician">
32                 
33             </a>
34             <ul id="nav-link">
35                 <li class="nav-link nav-link-underlined"><a href="{% url 'about' %}" id="nav-about">About</a></li>
36                 <li class="nav-link nav-link-underlined"><a href="{% url 'groups' %}" id="nav-groups">Groups</a></li>
37                 <li id="search">
38                     <form action="{% url 'search' %}" method="POST" id="search-form">
39                         <!-- Authentication Token -->
40                         {% csrf_token %}
41                         <div id="search-box" class="search-box-block">
42                             <input id="search-text-id" class="search-text" type="text" name="keywords" placeholder="Search for a recipe">
43                             <button type="submit" class="search-btn fas fa-search"></button>
44                         </div>
45                     </form>
46                 </li>
47                 <div id="nav-link-user">
48                     <!-- this is an important section, which has the defaults for users when they log in, and perhaps some relevant comments are appropriate - commented by Jeff -->
49                     <!-- include brief explanation - commented by Nicole -->
50

```



```

51     {% if user.is_authenticated %}
52     <li id="account" class="nav-item btn-group">
53         <button type="button" class="btn btn-secondary dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
54             <!-- include brief explanation - commented by Nicole -->
55             Hi, {{ user.get_username }}
56         </button>
57         <!-- perhaps some comments about why some links are commented out are appropriate - commented by Jeff -->
58         <div class="dropdown-menu dropdown-menu-right">
59             <a class="dropdown-item" href="{% url 'create_recipe' %}">Create a recipe</a>
60             <a class="dropdown-item" href="{% url 'user_profile' username=user.username %}">Profile</a>
61             <!-- remove all the unused lines of code - commented by Paul -->
62             <!-- <a class="dropdown-item" href="#">Groups</a> -->
63             <!-- <a class="dropdown-item" href="#">Recipes</a> -->
64             <!-- <a class="dropdown-item" href="#">Favorites</a> -->
65             <!-- redirect to home page after logout -->
66             <a class="dropdown-item" href="{% url 'logout' %}?next=/">Log out</a>
67         </div>
68     </li>
69     <!-- <a class="dropdown-item" href="{% url 'logout' %}?next=/">Log out</a> -->
70
71     {% else %}
72     <li class="nav-link nav-link-underlined"><a href="{% url 'login' %}" id="nav-login">Log in</a></li>
73     <li class="nav-link nav-link-underlined"><a href="{% url 'signup' %}" id="nav-signup">Sign up</a></li>
74     {% endif %}
75 </div>
76 </ul>
77 </div>
78 </header>
79
80 <!-- why do you have an empty block content? maybe some explanation is appropriate - commented by Jeff -->
81 {% block content %}{% endblock content %}
82
83 <!-- the footer is very long and complex. maybe some comments and divisions are appropriate - commented by Jeff -->
84 <div id="main-footer">
85     <div id="footer-container">
86         <div class="container">
87             <div id="footer-learn-more">
88                 <div id="footer-learn-more">
89                     
90                 </div>
91                 <div id="footer-learn-more" class="footer-content-block">
92                     <p class="footer-content-title">Learn More</p>
93                     <div class="footer-content-block-line"></div>
94                     <ul class="ul-footer-content-block">
95                         <li class="li-footer-content-item">
96                             <a class="footer-content-text" href="{% url 'groups' %}">Groups</a>
97                         </li>
98                         <li class="li-footer-content-item">
99                             <a class="footer-content-text" href="{% url 'search' %}">Recipes</a>
100                         </li>

```

```

101     </ul>
102 </div>
103 <div id="footer-connect" class="footer-content-block">
104     <p class="footer-content-title">Connect</p>
105     <div class="footer-content-block-line"></div>
106     <ul class="ul-footer-content-block">
107         <li class="li-footer-content-item">
108             <a class="footer-content-text" href="{% url 'contact_us' %}">Contact us</a>
109         </li>
110         <li class="li-footer-content-item" id="li-footer-content-medias">
111             <p class="footer-content-text">Follow us</p>
112             <div id="footer-content-medias">
113                 <a href="#" class="social-icon">
114                     <i class="fab fa-facebook-f"></i>
115                 </a>
116                 <a href="#" class="social-icon">
117                     <i class="fab fa-twitter"></i>
118                 </a>
119                 <a href="#" class="social-icon">
120                     <i class="fab fa-google"></i>
121                 </a>
122                 <a href="#" class="social-icon">
123                     <i class="fab fa-linkedin-in"></i>
124                 </a>
125             </div>
126         </li>
127         <!-- remove all the unused lines of code - commented by Paul -->
128         <!-- <li class="li-footer-content-item" id="li-footer-content-signup">
129             <p>Join Us</p>
130             <a href="{% url 'signup' %}" id="footer-signup">Sign Up</a>
131         </li> -->
132     </ul>
133 </div>
134 </div>
135 <div id="footer-break-line"></div>
136 <ul id="footer-policy">
137     <li class="footer-policy-item">
138         <a href="{% url 'about' %}">About KitchenMagician</a>
139     </li>
140     <li class="footer-policy-item footer-policy-item-line"></li>
141     <li class="footer-policy-item">
142         <a href="{% url 'term_of_use' %}">Terms of Service</a>
143     </li>
144     <li class="footer-policy-item footer-policy-item-line"></li>
145     <li class="footer-policy-item">
146         <a href="{% url 'privacy_policy' %}">Privacy Statement</a>
147     </li>
148 </ul>
149 <p>©copy; 2020 Kitchen Magician Inc.</p>
150 </div>

```


Section 5: Self-Check for Best Practices

- Major assets:

The user database, the recipe database, the groups database, and the search keywords database.

- Password encryption:

Django provides an API, `django.contrib.auth`, for authentication systems. One of its model `User` was applied to this application to encrypt the password.

Users > team1

Change user

Username:

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: **algorithm: pbkdf2_sha256 iterations: 216000 salt: 4mugYM***** hash: a9NDs7*******

Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using [this form](#).

- User validation.

- Validate the unique username and email, and confirm the password when users sign up.

```
if User.objects.filter(username=username).exists(): # user already exists
    context['error'] = f'The user {username} already exists.'
elif User.objects.filter(email=email).exists(): #unique email
    context['error'] = f'The email {email} already used.'
elif password1 != password2:
    context['error'] = 'The two password fields didn't match.'
```

- Authentication check when users log in.

```
# user authentication check
user = authenticate(request, username=username, password=password)
```

Section 6: List of Non-Functional Requirements

System requirements:

1. The application must have one unified system for search, with all fields searchable. DONE
2. Search shall be able to be filtered by username. ISSUE. We removed the username from the filter.
3. Search shall be able to be filtered by recipe keywords. DONE
4. Search shall be able to be filtered by recipe categories. DONE
5. Search shall be able to be filtered by recipe by estimated total preparation time. ISSUE. We removed the username from the filter.
6. Search shall be able to be filtered by recipe ingredients. ISSUE. We removed the username from the filter.
7. The application shall provide the ability to specify multiple operands and operators for a search. DONE
8. The application registration interface must be keyboard accessible. DONE
9. The application registration interface must allow users to tab to the next entry field. DONE
10. The system should be organised in such a way that user errors are minimised. DONE
11. The application system shall have an admin page. DONE
12. The application system shall have a login page. DONE
13. The application system shall have a registration page. DONE
14. The application system shall have a general recipe page. DONE
15. The application system shall have a general community page. DONE
16. The application system shall have a user profile page. DONE
17. The application system shall have an about page. DONE
18. All pages shall have a header on the top. DONE
19. The header shall contain a logo. DONE
20. The header shall contain a search input. DONE
21. All pages shall have a footer on the bottom. DONE
22. The footer shall contain a statement of copyright. DONE
23. The system shall give users the option to print the recipes. ISSUE. Lack of time to implement it.
24. The system shall provide a means of representing external files. DONE
25. The system shall provide a means of accessing external files. DONE
26. The system shall give users the option to download the recipes. ISSUE. Lack of time to implement it.
27. The system shall give users the option to download recipes in pdf format. ISSUE. Lack of time to implement it.

28. The system shall give users the option to download recipes in jpeg format. ISSUE. Lack of time to implement it.
29. When the public user accesses the system, the pages displayed shall be rendered without failing to display the pages consistently across all identified browsers. DONE
30. The user interface of the system shall be compatible with Chrome 85.0.4183.121 and later. DONE
31. The user interface of the system shall be compatible with Safari Mac os X 10.5 and later. DONE
32. The user interface of the system shall be compatible with Firefox 81.0.1 and later. DONE
33. All content shall be displayed error-free, e.g., without conflicts, overlapping, or other performance errors. ON TRACK
34. Each page must have an official logo of the application. DONE
35. The application system shall return to the home page in case of error loading of the page. ISSUE. Lack of time to implement.

Performance requirements:

1. The system shall be able to handle up to 1000 concurrent users when satisfying all their requirements. DONE
2. The system shall be able to handle up to 2500 concurrent users with browsing capabilities. DONE
3. The system shall display content with the smallest delay possible, and scale to handle dozens of thousands of new users. DONE
4. When a user clicks any links in the navigation bar, the route should take less than 2 seconds. DONE
5. Time it takes for the user profile page to load after login shall be less than 2 seconds. DONE
6. 80% of searches shall return results in less than 3 seconds. DONE
7. Each image shall be loaded within 2 seconds. DONE
8. Any recipe page with video shall be loaded within 3 seconds. ISSUE. Lack of time to implement.
9. Any recipe page without video shall be loaded within 2 seconds. DONE
10. Any community page shall be loaded within 3 seconds. DONE
11. It shall be less than 2 seconds to start to play a video after a user clicks to play it. ISSUE. Lack of time to implement.

Storage requirements

1. URLs must be unique in the repository. DONE
2. The application must allow users to delete all their content (recipes, comments, custom post types), their profiles, and all associated user account information. ISSUE. Only the admin can do it.

3. All config data stored in XML. ISSUE. Lack of time to implement.
4. Data stored in a MySQL database. DONE
5. All components shall leverage existing, central dataset/databases. DONE
6. The admin user shall be about to interact with the database through a web interface. DONE

Security requirements

1. The system shall validate all user input to ensure it does not exceed the size specified for that type of input. DONE
2. The server shall authenticate every request accessing the restricted web pages. DONE
3. The system shall have security controls to protect against denial-of-service attacks. ISSUE. Lack of knowledge to implement it.
4. The system shall encrypt sensitive data entered by users. DONE
5. The system will only accept valid URL requests. DONE
6. The system will apply SOA web service security standards. ISSUE. Lack of knowledge to implement it.
7. The system shall comply with SSE-CMM capability. ISSUE. Lack of knowledge to implement it.
8. The system shall allow users the option to change password. ISSUE. Only the admin can do it. Lack of time to implement it.
9. The system shall give admin access to change users' passwords. DONE
10. Every password shall be hashed. DONE
11. Every password shall be stored in the system database. DONE
12. The password will be case sensitive. DONE
13. System should timeout when there is inactivity for 10 minutes. ISSUE. Lack of time to implement it.
14. The password shall be at least 6 characters. ON TRACK
15. The password must include at least one number. ON TRACK
16. The password must include at least one special character. ON TRACK
17. The system shall allow authenticated users to access the resources of the system. DONE
18. All authentication tokens shall be saved on a local device for comparison and need user permission to gain access. ISSUE. Lack of time to implement it.
19. The system shall ensure the contents of a message are not altered in transit. DONE

Environmental requirements

1. The system shall use an independent Python virtual environment. DONE
2. The system shall use pip3 to install packages in the virtual environment. DONE
3. The virtual environment shall include all necessary packages compatible with the operating system Ubuntu 20.04 LTS. DONE

4. The system shall have a file named requirements.txt to store all packages installed in the virtual environment. DONE
5. The server Apache shall connect to the virtual environment in the master branch. DONE
6. The server Apache shall use the correct path of the virtual environment. DONE
7. The system shall have a Django development environment up and running on local computers for testing. DONE
8. The system shall have a Django development environment up and running on local computers for development. DONE
9. The system shall use the default port, the internal IP at port 8000, for local testing. DONE
10. The system shall use the default port, the internal IP at port 8000, for local development. DONE
11. The system shall use the default port, the internal IP at port 8080, for global testing. DONE
12. The system shall use the default port, the internal IP at port 8080, for global development. DONE
13. The system shall use the port 443 for SSL certification. DONE

Legal requirements

1. Users shall agree to the privacy agreement when registering for an account. DONE
2. The application shall display a copyright notice visible to users at all time. DONE
3. The application shall display the Kitchen Magician policies for users to read and agreed to comply with in accordance with state and federal law. DONE
4. The application shall display the licensing agreements relating to intellectual property rights. DONE
5. Users shall agree to the code of conduct when registering an account. DONE
6. Personalization and data collection processes should comply with the General Data Protection Regulation (GDPR). DONE
7. The website contains links to other websites and it cannot be responsible for the privacy practises of content of other websites. DONE
8. Users shall not use any recipe from the application for commercial purposes without permission from the admin. DONE
9. Users shall comply with business rules and administrative functions as provided by admin. DONE
10. Users shall not use any video from the application for commercial purposes without permission from the admin. DONE
11. Users shall not use any application content for commercial purposes without permission from the admin. DONE

Marketing requirements

1. All commercials shall go through the approval process before posting on the application. ISSUE. Lack of time to implement it.
2. Pages shall contain sponsored commercial approved by the admin. ISSUE. Lack of time to implement it.
3. Pages shall contain sponsored advertisements approved by the admin. ISSUE. Lack of time to implement it.
4. When referencing KitchenMagician in text as a combined word, use camel case, with no space between “Kitchen” and “Magician”. DONE
5. All commercials on the application shall use corporate style guide colors. ISSUE. Lack of time to implement commercials.
6. All advertisements on the application shall use font size approved by the admin. ISSUE. Lack of time to implement advertisements.
7. All customer facing web pages will use the corporate style guide colors, fonts and corporate approved logos. DONE

Content requirements

1. All pages and files must be current and accurate. ON TRACK
2. The system shall support the appearance of a component being determined by the type of page on which the component appears. DONE
3. The system shall support the appearance of a component being determined by the location on the page on which the component appears. DONE
4. The system shall support the appearance of a component being determined by the content of the component. DONE
5. The system shall support the appearance of a component being determined by the type of the component. DONE
6. The system shall be able to accommodate any image types, and images must be publishable. DONE
7. The system shall be able to accommodate any image types. DONE
8. The application shall always display the logo at a size that is clear and legible. DONE
9. To ensure legibility of KitchenMagician branding, the system shall use the logos with appropriate sizes and not smaller than 24px. DONE
10. Images in the system shall be browsable. DONE
11. Images shall include alt text. ON TRACK
12. Image size shall not exceed 2 Mbytes. ISSUE. DONE
13. Videos shall be captioned and may also include a transcript. ISSUE. Lack of time to implement it.
14. Videos shall stop when users leave the content view. ISSUE. Lack of time to implement it.
15. Audios shall stop when users leave the content view. ISSUE. Lack of time to implement it.

16. All audios shall display with user-controlled captioning. ISSUE. Lack of time to implement it.
17. All videos shall display with user-controlled captioning. ISSUE. Lack of time to implement it.
18. The application shall ensure audio and video accessible. ISSUE. Lack of time to implement it.
19. The date format for recipes shall be as follows: Month Day, Year. DONE
20. The date format for comments shall be as follows: Month Day, Year. DONE

Privacy requirements

1. The system shall protect user information in accordance to the federal government's Privacy Act. DONE
2. The system shall comply with California Consumer Privacy Act. DONE
3. The system shall comply with California Online Privacy Protection Act 2003 (CalOPPA). DONE
4. Users shall agree to the privacy agreement when registering for an account. DONE
5. Users shall agree to the code of conduct when registering for an account. DONE
6. The company is not responsible for images, videos, or any resources posted by users without permission of the copyright holder. DONE
7. The application collects anonymous data from every visitor of the Website to monitor traffic and fix bugs. DONE
8. The application collects log information including a user's Internet Protocol address, browser type, browser language, the date and time of a user's query and one or more cookies that may uniquely identify a user's browser or downloaded software. DONE
9. The application must support authorized users to be able to access, update or delete any comments, receipts, or users. DONE
10. The application will collect usernames only if users agree to comply with the company's policy. DONE
11. The application will collect users' emails only if users agree to comply with the company's policy. DONE
12. The application will collect users' addresses only if users agree to comply with the company's policy. DONE
13. The application will collect users' birthdays only if users agree to comply with the company's policy. DONE
14. The application will collect information on users' search history, favorited, and viewed recipes. DONE
15. The application shall not share personal information with any third parties. DONE
16. The application uses all of the information users provide voluntarily in order to make users' visits on the application possible. DONE

17. The application collects information only to add customized elements to the application or to plan its content more appropriately, based on user interests. DONE
18. The application will allow users to delete their own accounts via emailing the requests. ISSUE. Lack of time to implement it.

Testing requirements

1. The application shall be tested during deployment to verify that it functions as intended and that all requirements are met. DONE
2. The application shall be tested before deployment to verify that it functions as intended and that all requirements are met. DONE
3. The application shall be tested after deployment to verify that it functions as intended and that all requirements are met. DONE
4. Prior to launch, the applications shall be placed in a testing environment. DONE

Coding Standards

- **File encoding**
 1. All documentation source files must be in UTF-8 encoding. ON TRACK
 2. All documentation source files shall allow special characters. DONE
- **English Syntax**
 1. The application must use proper spelling. ON TRACK
 2. The application must use proper grammar. ON TRACK
 3. The application must use proper punctuation. ON TRACK
- **Indentation**
 1. There must be a space after giving a comma between two function arguments. ON TRACK
 2. Each nested block should be properly indented and spaced. ON TRACK
 3. Proper Indentation should be there at the beginning and at the end of each block in the program. ON TRACK
 4. All braces should start from a new line and the code following the end of braces also starts from a new line. ON TRACK
- **Coding style**
 1. Avoid using a coding style that is too difficult to understand. DONE
 2. Code should be easily understandable. ON TRACK
 3. Code should be well documented. ON TRACK
 4. The code should be properly commented for understanding easily. ON TRACK
 5. Comments regarding the statements increase the understandability of the code. DONE
 6. Length of functions should not be very large. DONE
 7. Functions should be small enough to carry out small work. DONE

8. Lengthy functions should be broken into small ones for completing small tasks.
ON TRACK
9. Length of methods should not be very large. DONE
10. Methods should be small enough to carry out small work. DONE
11. Lengthy methods should be broken into small ones for completing small tasks.
ON TRACK

- **Frontend**

1. All JavaScript code should follow the Google JavaScript Style Guide. ON TRACK
2. All HTML and CSS code should follow the Google HTML/CSS Style Guide. ISSUE. Lack of time to refactor the application.
3. The headline on the home page should only take up one line of text, less than 50 characters. DONE
4. The size of the headline on the home page should be smaller than 200px. DONE
5. The application must ensure that styles and naming for interactive elements are used consistently. DONE
6. The application provides sufficient contrast between foreground and background. DONR
7. The application provides informative, unique page titles. DONE
8. The application uses headings to convey meaning and structure. DONE
9. The application makes link text meaningful. DONE
10. The application writes meaningful text alternatives for images. DONE
11. The application creates transcripts and captions for multimedia. ISSUE. Lack of time to implement it.
12. The application provides clear instructions. DONE
13. The application must keep content clear and concise. DONE
14. The application must ensure that interactive elements are easy to identify. DONE
15. The application provides clear and consistent navigation options. DONE
16. The application ensures that form elements include clearly associated labels. DONE
17. The application provides easily identifiable feedback. ISSUE. Lack of time to implement it.
18. The application uses headings and spacing to group related content. DONE
19. The application style uses rem for font-size. ISSUE. Team members use different styles.
20. The application style uses em for padding and margin. ISSUE. Team members use different styles.
21. The application style uses em or percentage for widths. ISSUE. Team members use different styles.
22. The application does not use all caps. DONE

23. The style of headlines of comments shall be bolder. DONE

24. The max length of comment is 65,535 characters. DONE

- **Backend**

1. All Python code should follow a derivation of PEP-8 style guidelines. ISSUE.
Team members use different styles.

2. Whitespace rules and other rules are relaxed (for example, it is not necessary to put two newlines between classes, though that's just fine if you do). DONE.

3. Single-line imports DONE

Do this:

```
import os
import sys
```

Do **not** do this:

```
import os, sys
```

4. Import Order: Imports should be ordered by their origin. Names should be imported in this order: DONE

- i. Python standard library
- ii. Third party packages
- iii. Other modules from the current package

5. Wildcard Imports:

- i. Do not import all the names from a package (in other words, never use `from package import *`) DONE
- ii. Import just the ones that are needed. DONE
- iii. Single-line imports apply here as well: each name from the other package should be imported on its own line. DONE

6. Naming conventions for local variables, global variables, constants and functions:

- i. Local variables and global variables should be named using lower case lettering and underscore between words (e.g. `local_data`). Constant names should be formed using capital letters only (e.g. `CONSDATA`). DONE
- ii. It is better to avoid the use of digits in variable names. DONE
- iii. The names of the function should be written using lower case lettering and underscore between words. DONE
- iv. The name of the function must describe the reason for using the function clearly and briefly. DONE
- v. The names of the function should be written using lower case lettering and underscore between words. DONE
- vi. The name of the function must describe the reason for using the function clearly and briefly. DONE
- vii. Class names should normally use the CapWords convention. DONE

7. All MySQL code should follow the MySQL Coding Guidelines. DONE