# Meals for a Steal

*Final Project Report*

Andre Abedmamoore
Scotty Cardwell
Daniel Favela
Malialosa Taupule
Eduardo Martinez

Table of Contents

# 1. Project Description

## 1.1 The Problem

College students often are stretched financially -- with tuition, book, living and other personal expenses this can leave them with little money left over for food. No one should have to be forced to eat Ramen Noodles every day nor should they have to be limited to a meal variety based solely on convenience rather than taste. We want to give those students the option to be able to prepare their own homemade meals that are both delicious *and* affordable!

## 1.2 Project Purpose

Our application, Meals for a Steal, will serve as a platform where users may post recipes that are affordable. It will serve as a convenient virtual cookbook for those looking to make a delicious and affordable home cooked meal. This is advantageous over existing recipe sites because often times, other recipe sites do not take price into account.

# 2. Specifications

## 2.1 Account: Signing in, Settings, Protecting User Data

In order for the user to experience the most out of Meals for a Steal, he/she would need to create an account. On the Account page, a user can signup or login page to an already existing account. Once logged in using their password, the user could provide feedback recipe posts through leaving comments or submitting an upvote or downvote. The user would also be capable of accessing his/her created recipes via his/her profile page through clicking their username on the right side of the header bar. Upon signing up, the user will automatically be logged in. Once a user is logged in, they will automatically be redirected the homepage.

Additionally upon authentication, users will receive an encrypted access token as a cookie. This access token will allow them act on user-oriented actions, such as voting. The access token only provides them do actions based on their account privileges (ie, admin or not admin).

Users may change their password or delete their account. If they wish to update their password, the app will require the user to include their own password, and type their new password twice. If a user wished to delete their account, they will need to enter their password to confirm this action.

As opposed to storing sensitive user data in plain text, the database stores passwords and tokens as hashes using a sha1 encryption hash. Tokens and passwords have their own distinct hash salts. Additionally, new tokens are produced each time a user logs out of their account using a randomized string that is then encrypted.

## 2.2 List Ordered Recipes (Homepage & Profile Page)

Our homepage and profiles page lists a variety of recipes stored in the database ordered either by recency or popularity, depending on the user's preference. If by recency, the recipes listed will be ordered from most recent to the least recent and if by popularity, the recipes will be listed in descending order by the recipe's overall score.

In these views, a logged in user may click on either the upvote or downvote button. This will either insert a vote or update a previous vote using an HTTP request. Scores on this view are recalculated on the client when a user votes using jQuery functions. Users that are not logged in will not have the option to vote. All visitors to the site may browse user profiles and recipes regardless of their auth status.

## 2.3 Add Recipes

A user with a Meals for a Steal account has the ability to add any recipe of their choosing by simply filling out the fields found on our Add Recipe Form webpage. These fields require a title, ingredients and instructions, which is intended to contain a step-by-step process of how to complete the recipe using the listed ingredients. Images are optional. Due to the restrictive nature of the CIS server's configs, we cannot upload images to the server as originally intended. Instead, a user may use a source url on another image hosting site. However, the code that uploads files is available as a comment. We can confirm this works on a local environment. If the CIS server lifts its restrictions, we will make the necessary adjustments.

## 2.4 List Recipe Details

Any user viewing our website may view the details of any recipe stored within our database with a simple click. When a user wants to view the details of a specific recipe, he/she would just need to click the target recipe's title on the homepage or profile page. Inside the recipe details page will include its score, the recipe title, its post date, the author, a large image of the recipe, ingredients, instructions, and a comment section. If a user is logged in, they may post a comment and vote on the recipe.

## 2.5 Upvote/Downvote

If a user likes or dislikes a specific recipe, he/she would just need to click the up or down arrows located next to the recipe's title on its details page, homepage, or profiles page. This single vote contributes to the overall score of the recipe. When viewing recipes on the homepage by popularity, the recipes are listed in descending order by recipe score. In order for a user to up or downvote a recipe, he/she would need to have a Meals for a Steal account. When a user clicks on either the upvote or downvote button the specific queries to the database are either inserting a vote or updating a previous vote using an HTTP request. The logic for updating votes is handled by our API. Scores displayed on the client after voting are recalculated on the client using jQuery functions.

## 2.6 Adding Comments on Recipes

Users may communicate with other users through comments on a recipe's details page. In order for the user to leave a comment on a post, he/she would need to have a Meals for a Steal Account. When a logged-in user creates a comment, it will be verified by the handler to make sure the comment is not

empty, and will reload the page with the new comment. The handler also checks for a valid user token and will load the error page if an unauthorized user attempts to access the page.

## 2.7 View User and Recipe Website Statistics

If a user has an account, he/she could view a variety of Meals for Steal-specific statistics on his/her profile page. These statistics include number of votes and posts submitted along with the date he/she created an account. Located on the user's profile page are also all the recipe posts the user has created. A user's posts on their profile may be sorted by top voted posts. Along with viewing his/her own profile, a user can also view the profile of other users.

## 2.8 Website Maintenance: the Admin Panel & Permissions

An admin user is capable of performing simple website maintenance commands like delete post, user, and/or comment by inputting the id of the item to be deleted. When deleting an object, all of its dependencies will also be removed from the records. For example, if an admin wished to delete a user, it will also delete all the votes that user has made on other posts, their own posts, comments on their own posts, votes to their posts, and their comments on other posts.

By default, the app comes with one admin user: username, admin; password, admin. If unauthorized users of the site attempt to access any page that are not intended to be available to them (ie, non-admin attempting to access the admin panel, logged-off user attempting to post a recipe), the error page will be loaded. The error page has a message letting the user know they have encountered an error, and will have a button going back to the homepage.

# 3. User's Manual

## 3.1 Introduction

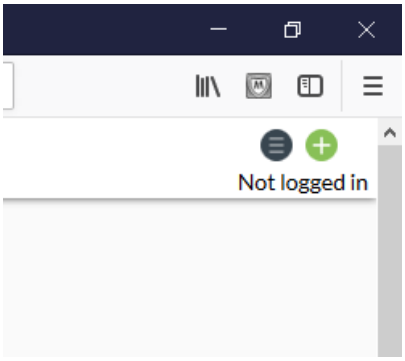
Welcome to Meals for a Steal! Our mission is to provide a virtual recipe book containing a plethora of delicious but inexpensive meals for the typical college student. Students are no longer restricted to cup of noodles and peanut butter and jelly sandwiches!

## 3.2 Getting Started

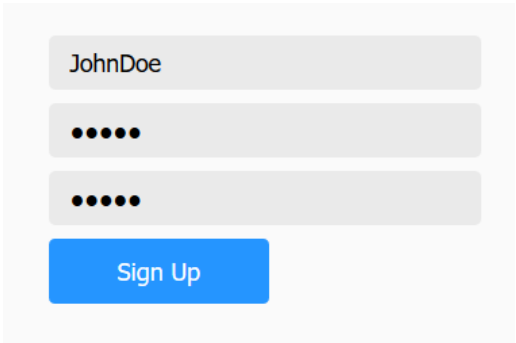
Anyone can enter the website by entering the URL:http://cis444.cs.csusm.edu/group_C/index.php. If you were to access the site through http://cis444.cs.csusm.edu/group_C/, you will automatically be redirected to the homepage. The above url will take a visitor to the homepage of the website. From the homepage, a visitor may view recent and popular uploaded recipes from registered users, votes on the recipe, date/time the recipe was added, and the username of the individual that uploaded the recipe.

### 3.2.1 Creating a New Account

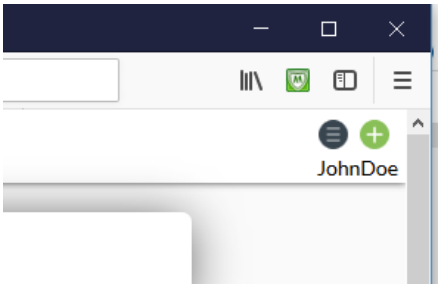
To create a user account a visitor may select the icon in the upper right hand side of the screen:




This will take the visitor to the account page where they may create a username and password to access other features of the website. To create an account the user must enter a username and password in the appropriate fields. Once “Sign up” is selected their account will be created:

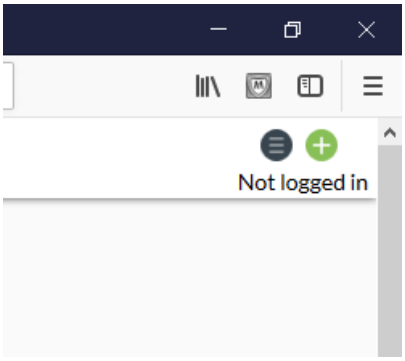



After “Sign Up” is selected the visitor is now a registered user and will be redirected to the homepage. Their username will now be visible in the upper right hand side of the page:



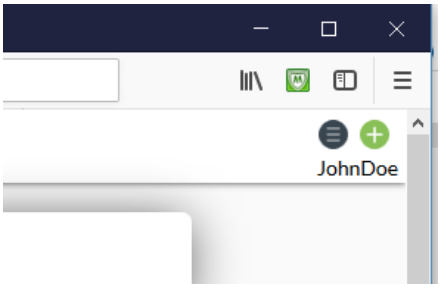

### 3.2.2 Logging Into Your Account

To log into your account select the icon in the upper right hand side of the screen:



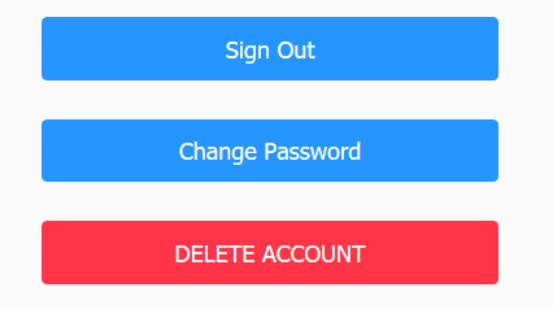

This will take the user to the account page where they may enter the username and password to access additional features of the website after “Sign In” is selected:


After “Sign In” is selected the user will be redirected to the homepage if log in was successful. The username will now be visible in the upper right hand side of the page:

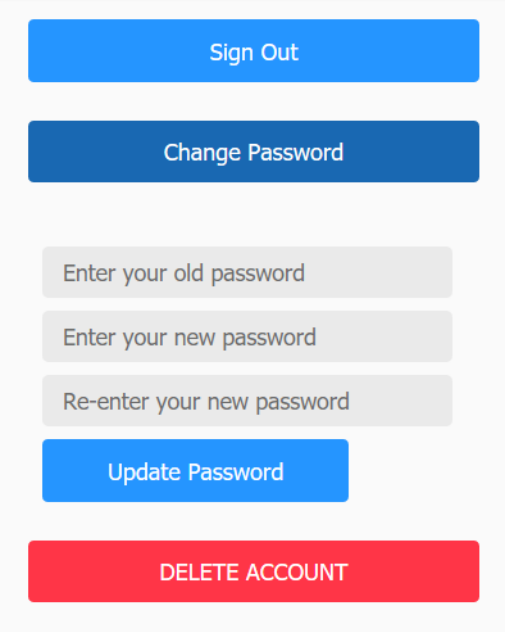



### 3.2.3 Changing Your Password

To change your user password, go to the setting page by selecting the icon in the upper right hand side of the screen. While on the setting page a user may choose to Sign Out, Change Password, or Delete Account:




When “Change Password” is selected the user will now be able to enter a new password once all fields are entered with the appropriate information:



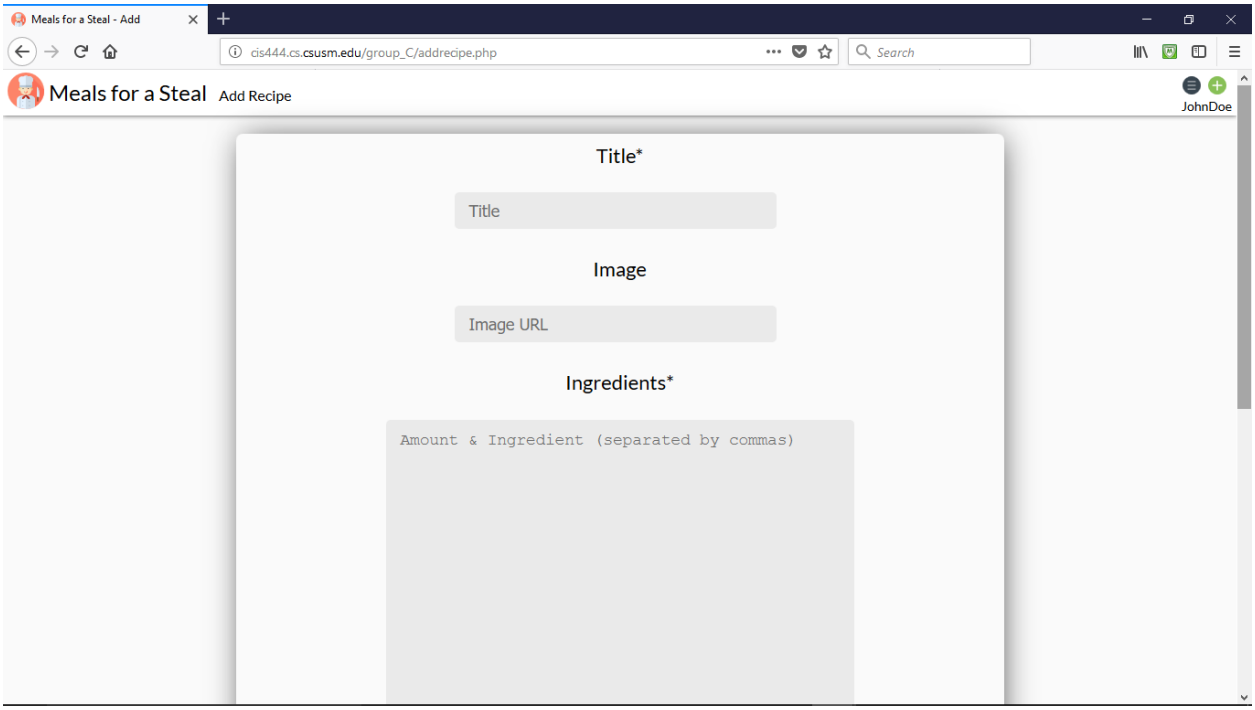

Once the user enters the new password they will be redirected to the account page where the user must log in with their username and new password.

# 3.3 Features

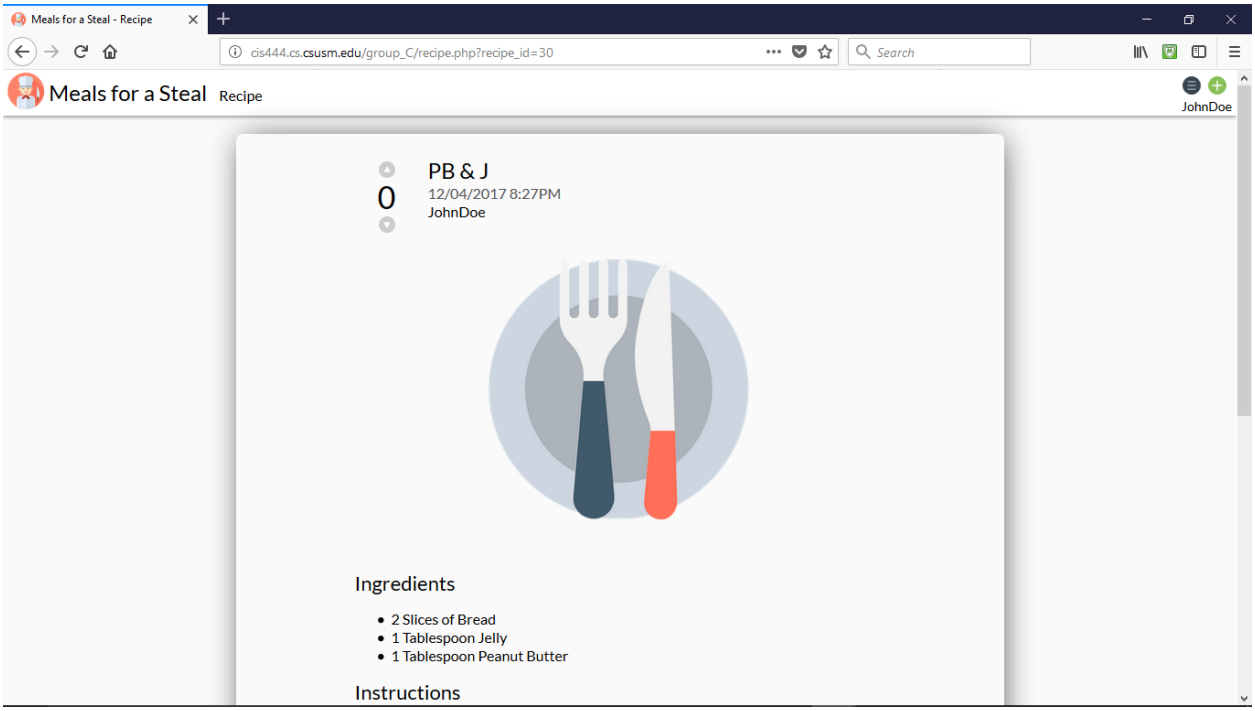
Features that are available to registered users are recipe creation, voting/commenting on a particular recipe, and deleting the current user account. These features are unavailable to visitors that are not registered with Meals for a Steal!

## 3.3.1 Creating a Recipe

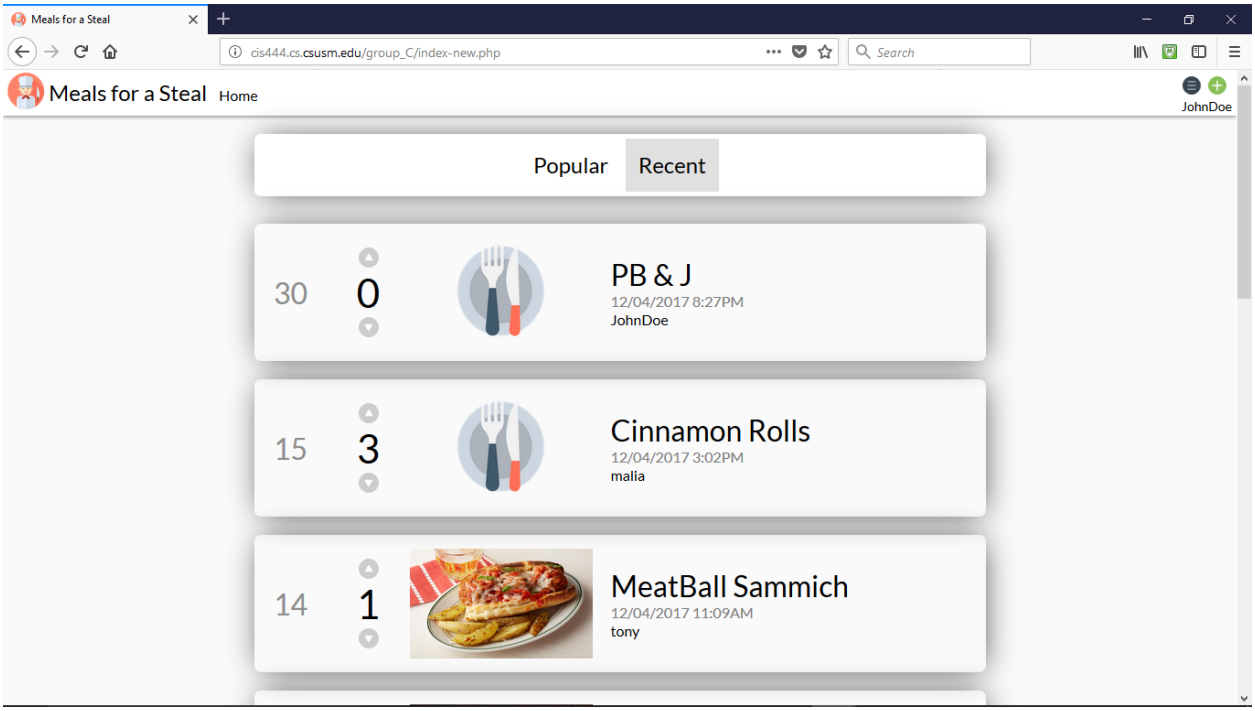
To create a new recipe a user needs to select the plus icon in the upper right hand side of the screen. This will send the user to the Add Recipe page:




On this page the user will have Title, Ingredients, and Instructions fields with * to indicate that the fields require information. Image is an optional field where the user may upload an image if they desire. At the bottom of the Add Recipe page the user can add the recipe or clear all fields if needed. Once the user adds the recipe they will be redirected to the recipe page where the new recipe can be viewed:

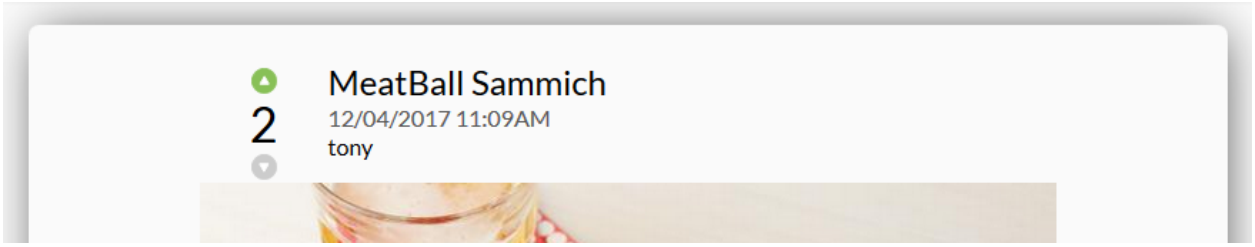


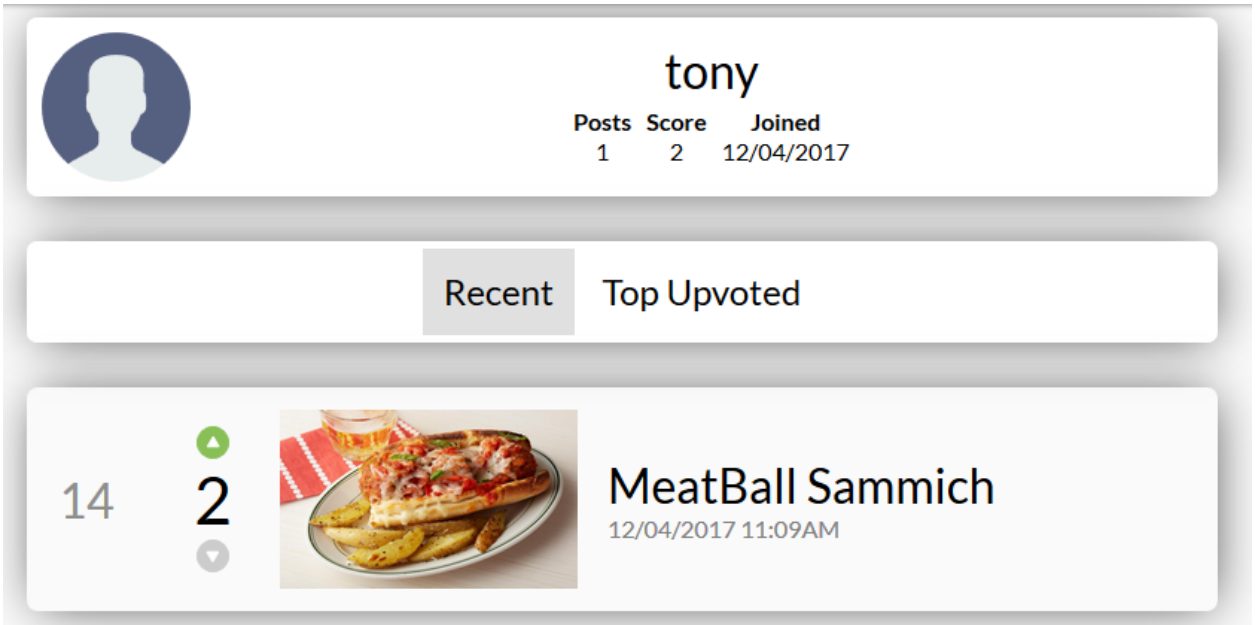

From this page the user can return to the homepage by selecting the Meals for a Steal! icon in the upper left hand side. To view the most recent uploaded recipes select Recent on the homepage:



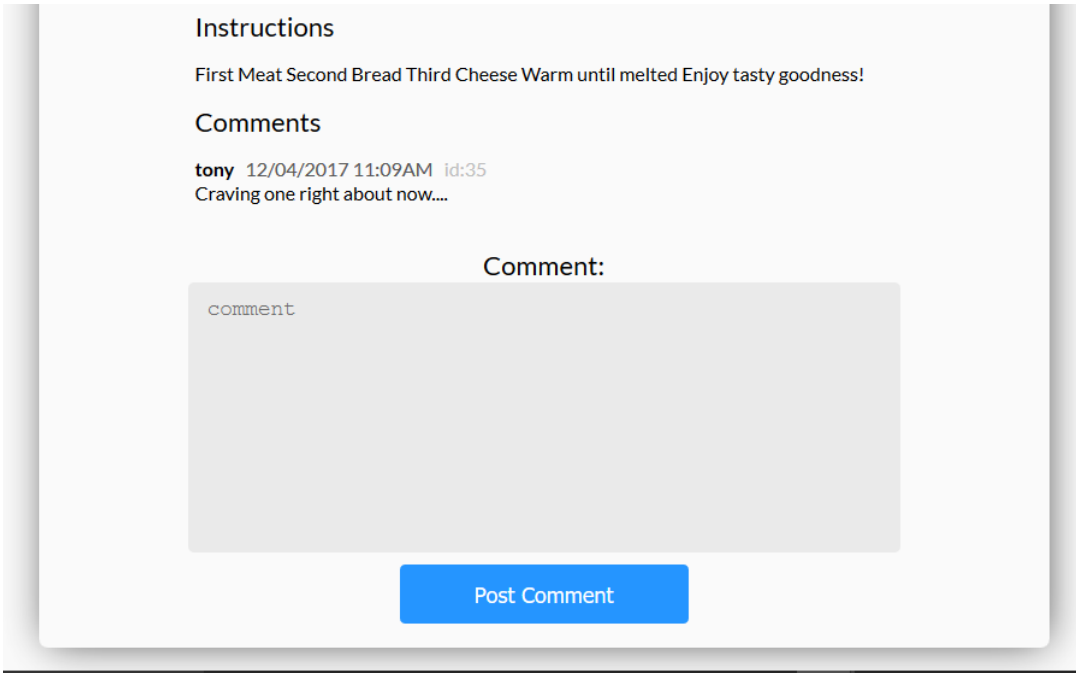

## 3.3.2 Leaving Feedback for a Specific Recipe

If a user wishes to vote on a recipe, this can be done by simply selecting the up or down arrow from the homepage, recipe, or profile pages:




If a user wishes to comment on a recipe that may have room for improvement or that they wish to comment that the recipe was delicious, this can be done on the recipe page under its instructions for any selected recipe:

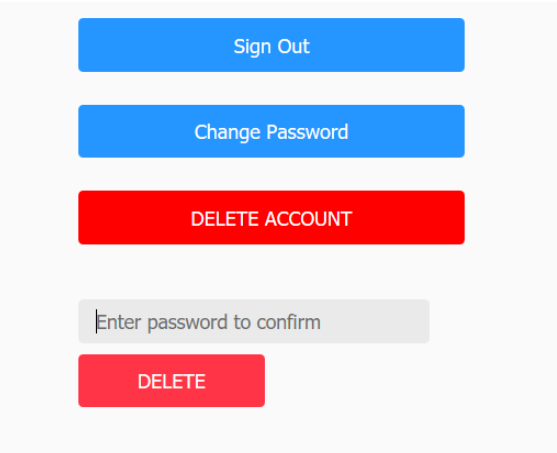



After a user adds a comment in the Comment field, select “Post Comment” to post. The user will then be redirected to the recipe page where the new comment can be viewed.

### 3.3.3 Deleting Your Account

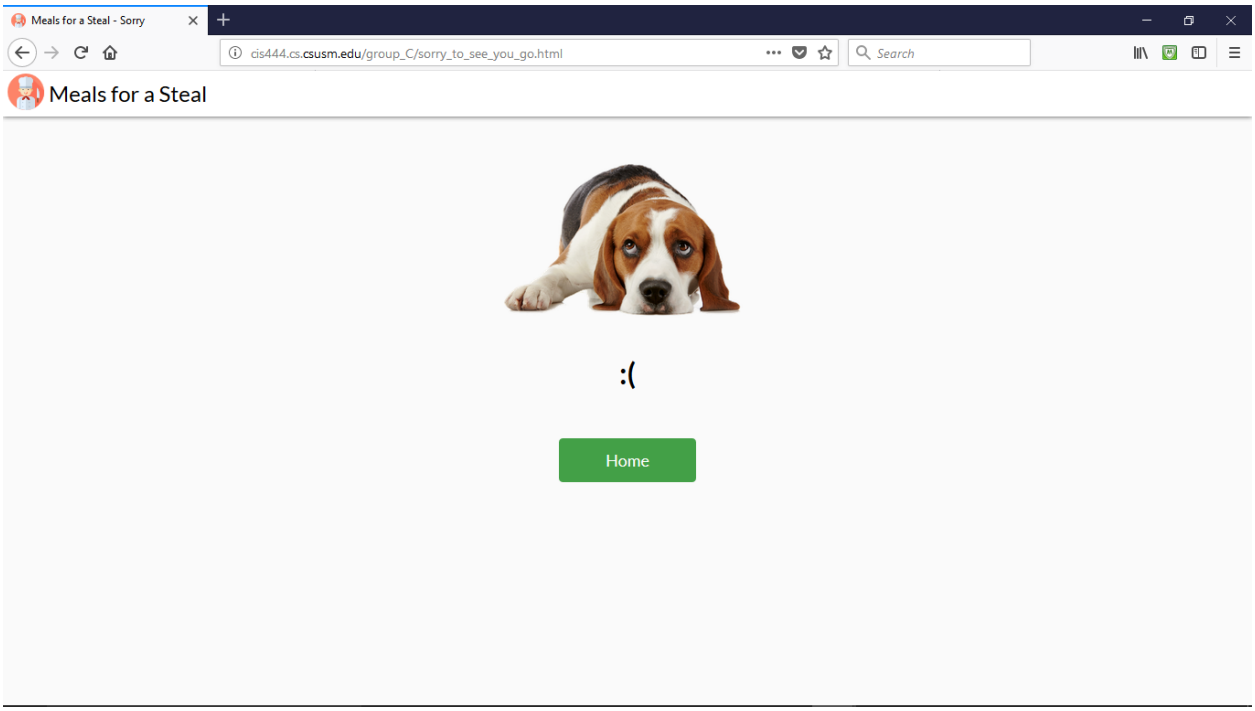
To delete your account go to the setting page by selecting the icon in the upper right hand side of the screen. While on the setting page a user may choose to Sign Out, Change Password, or Delete Account.

When “Delete Account” is selected the user will need to input their current password to continue with the operation:

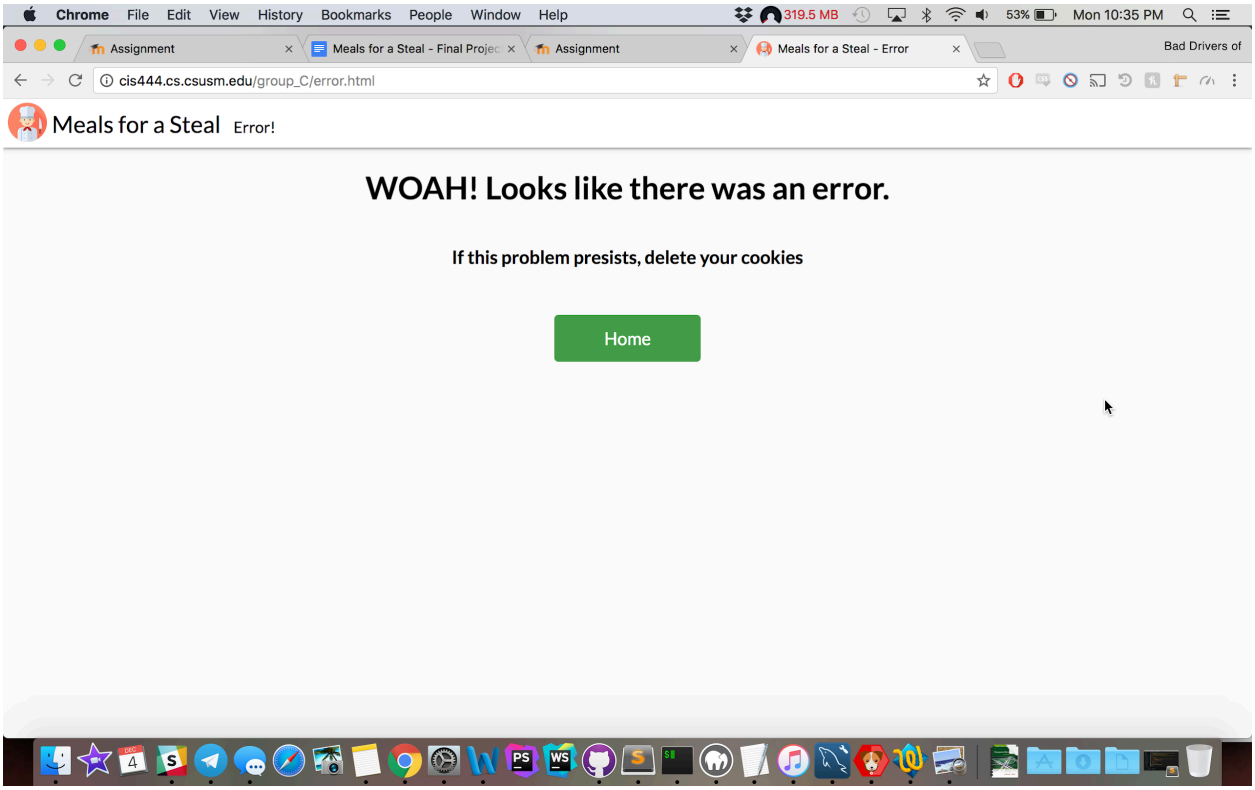



After the user selects “Delete” the account removed. All recipes, comments, and votes that the user made during their time with us will be removed.

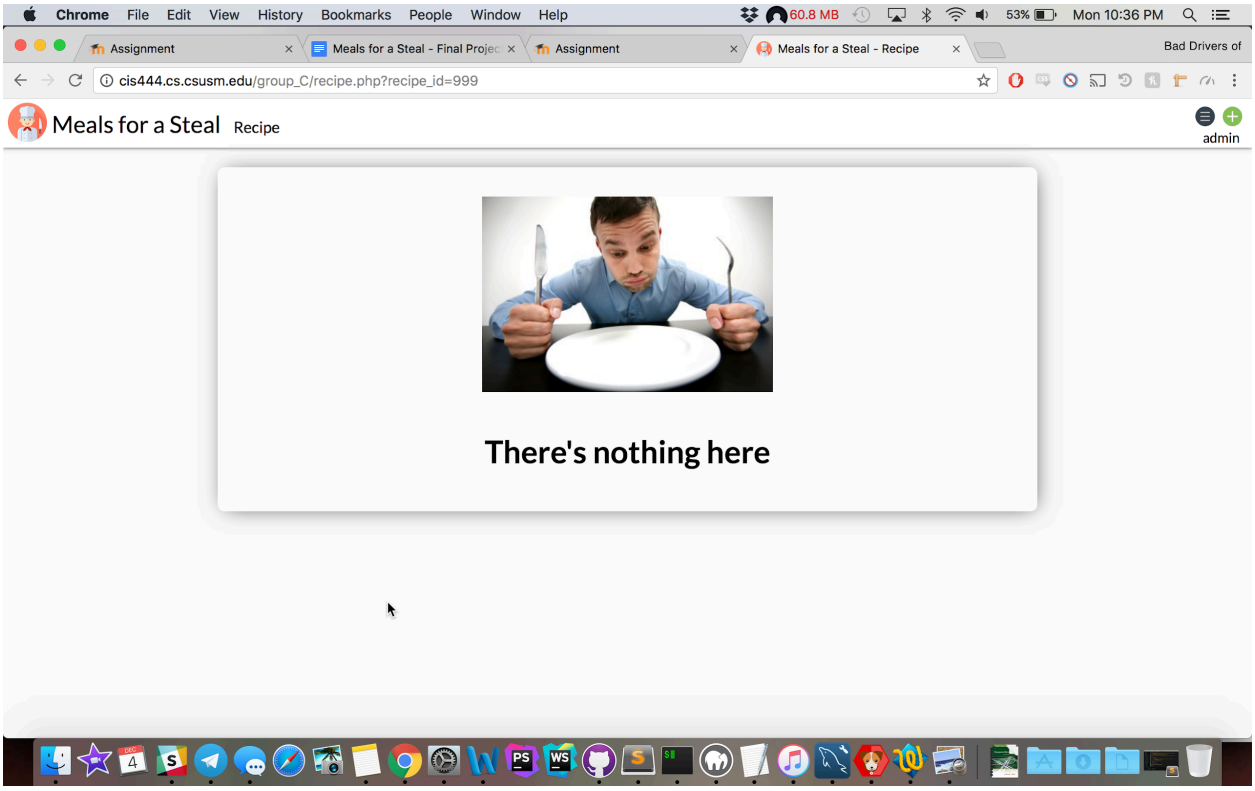
The following page will be displayed because we here at Meals for a Steal! are always sad to see a user go and we hope that they may wish to register with us again at a later time:

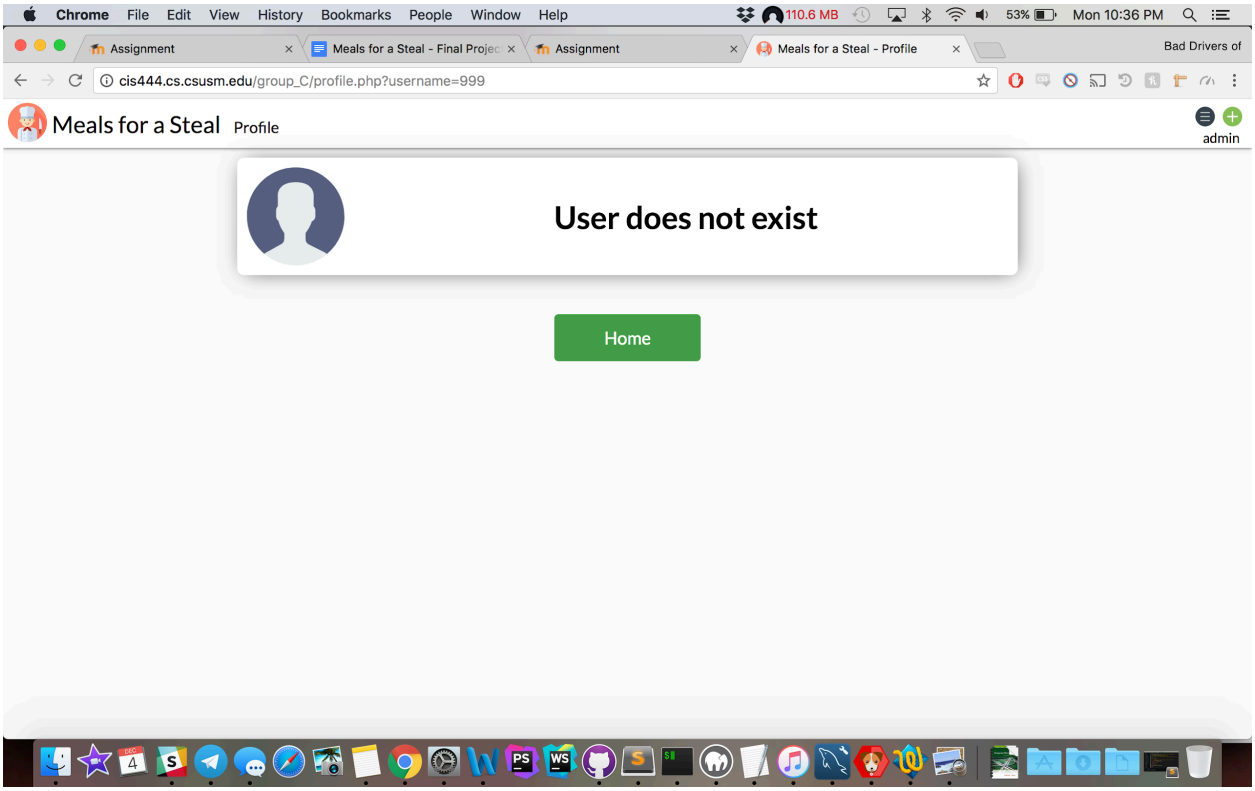



Error pages may be encountered if a user attempts to perform an action they are not allowed to. It will appear as this:

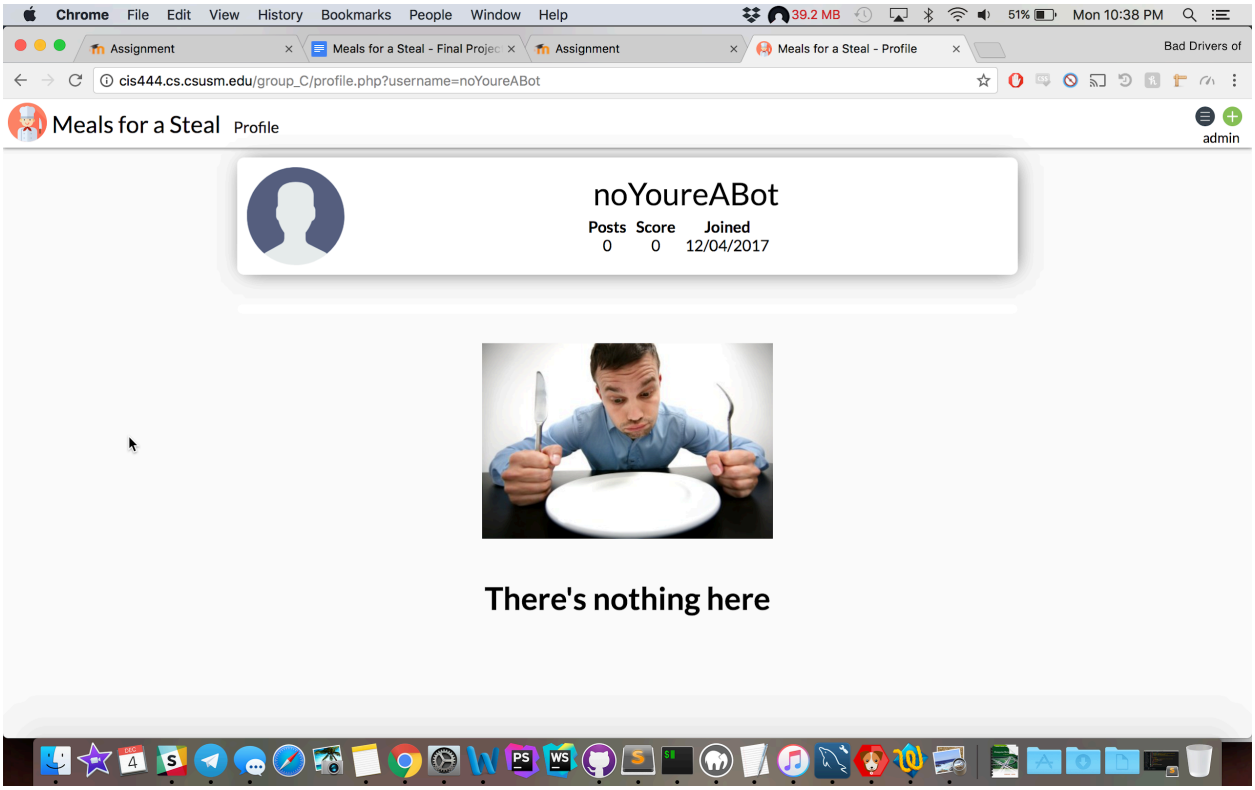



Additionally, if a user tries to load data that doesn't exist, such as a recipe or profile, they will see the following:






It is also possible that a user my stumble on a profile with no recipes. In that case, this will be displayed:




The site is also mobile compatible. The following is a sample of various screenshots throughout the application as they appear on an iPhone 6 using iOS 11.

AT&T Wi-Fi 10:54 PM 66%

cis444.cs.csusm.edu

Meals for a Steal

Profile

admin

# admin

| Posts | Score | Joined |
|---|---|---|
| 3 | 2 | 11/29/2017 |

Recent | Top Upvoted

2

honey garlic salmon

12/04/2017 7:57AM

AT&T Wi-Fi 10:54 PM 66%

cis444.cs.csusm.edu

2

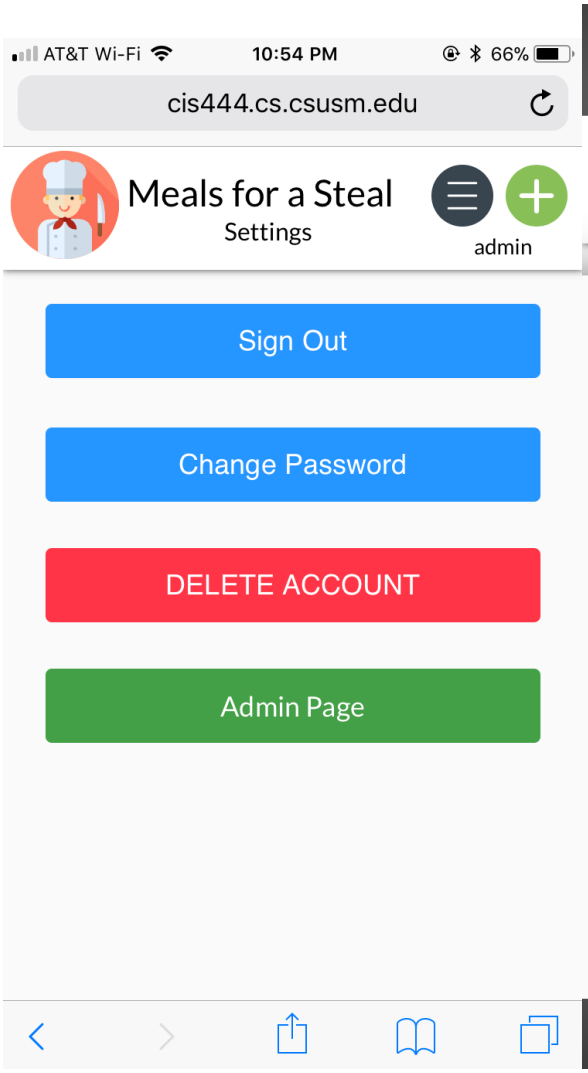
honey garlic salmon

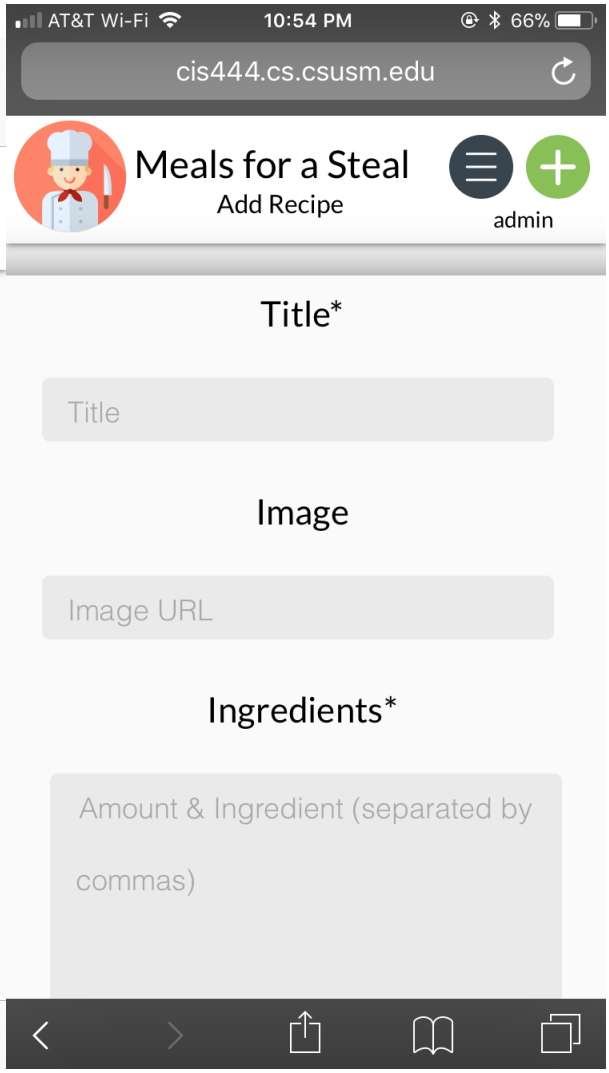
12/04/2017 7:57AM

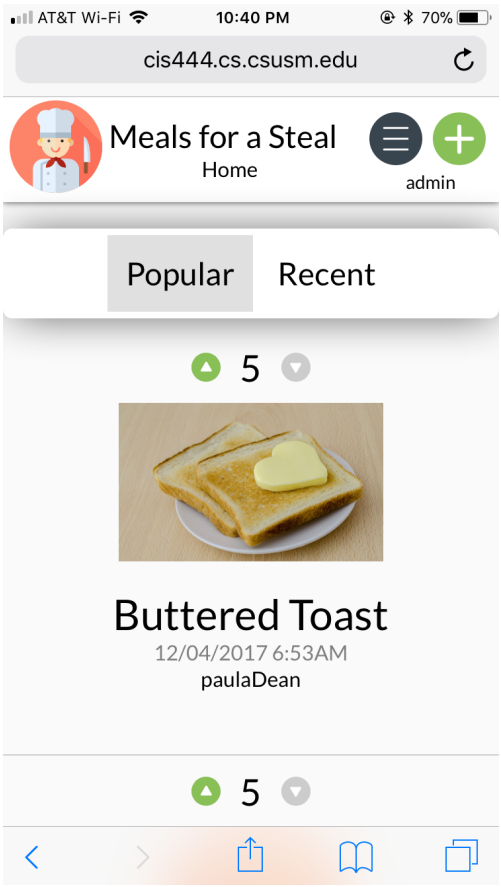
-2

More generic food

12/04/2017 6:51AM


AT&T Wi-Fi
10:54 PM
66%
cis444.cs.csusm.edu
Meals for a Steal
Settings
admin
Sign Out
Change Password
DELETE ACCOUNT
Admin Page

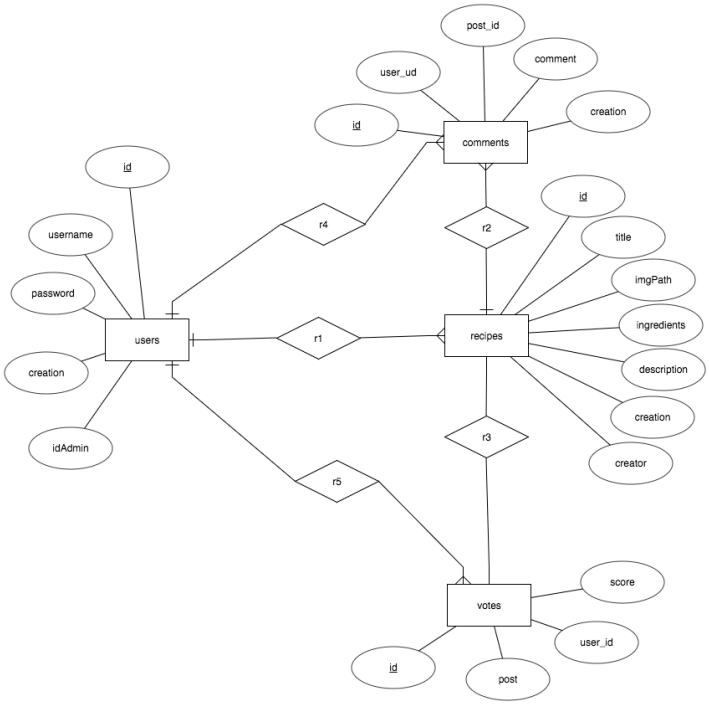

AT&T Wi-Fi
10:54 PM
66%
cis444.cs.csusm.edu
Meals for a Steal
Add Recipe
admin
Title*
Title
Image
Image URL
Ingredients*
Amount & Ingredient (separated by commas)



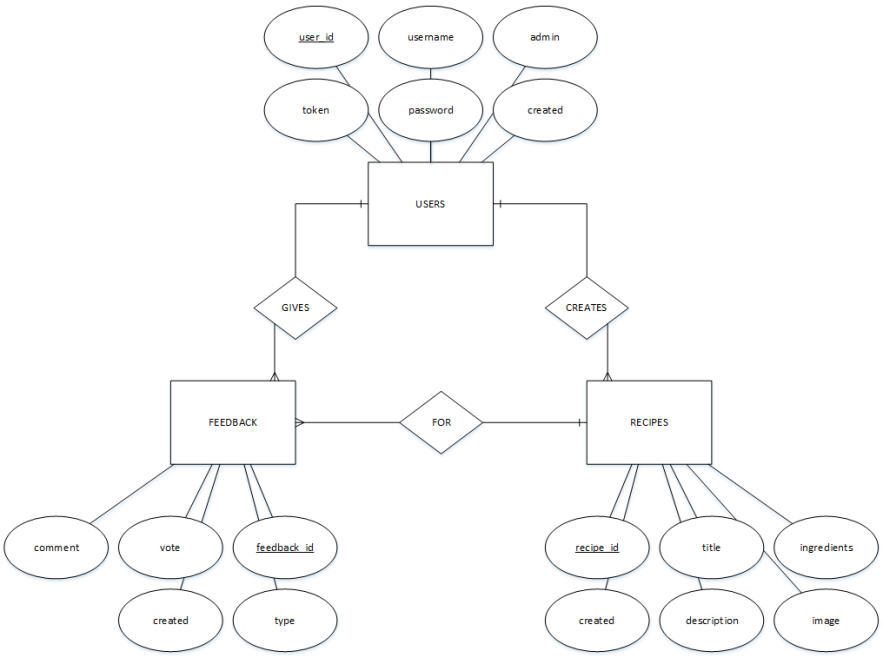

# 4. Database Design & Implementation

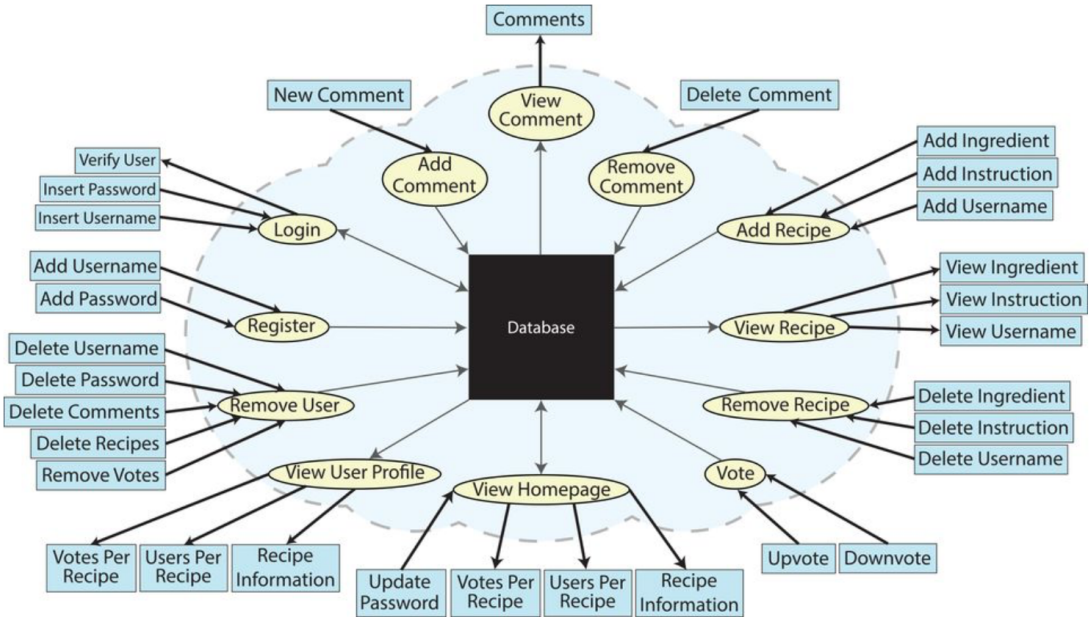
## 4.1 Entity Relationship Diagram (ERD)

## 4.1.1 Initial ERD




## 4.1.2 Final Revised ERD






# 4.2 Database Implementation Details

## 4.2.1 SQL Table implementation code (in database: 'group_c')

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";
```

Table structure for table `feedback`:

```
CREATE TABLE `feedback` (
  `feedback_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `recipe_id` int(11) NOT NULL,
  `comment` varchar(255),
  `vote` int(11),
  `created` int(11) NOT NULL,
  `type` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Indexes for table `feedback`:

```
ALTER TABLE `feedback`
  ADD PRIMARY KEY (`feedback_id`),
  ADD KEY `recipeRM` (`recipe_id`),
  ADD KEY `UserRecipeRM` (`user_id`);
```

AUTO_INCREMENT for table `feedback`:

```
ALTER TABLE `feedback`
  MODIFY `feedback_id` int(11) NOT NULL AUTO_INCREMENT;
```

Constraints for table `feedback`:

```
ALTER TABLE `feedback`
  ADD CONSTRAINT `UserRecipeRM` FOREIGN KEY (`user_id`) REFERENCES `users`
(`user_id`)  ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `recipeRM` FOREIGN KEY (`recipe_id`) REFERENCES `recipes`
(`recipe_id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Table structure for table `recipes`:

```
CREATE TABLE `recipes` (
  `recipe_id` int(11) NOT NULL,
  `title` varchar(255) NOT NULL,
  `ingredients` varchar(255) NOT NULL,
  `description` varchar(255) NOT NULL,
  `created` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `image` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Indexes for table `recipes`:

```
ALTER TABLE `recipes`
  ADD PRIMARY KEY (`recipe_id`),
  ADD KEY `userRM` (`user_id`);
```

AUTO_INCREMENT for table `recipes`:

```
ALTER TABLE `recipes`
  MODIFY `recipe_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
```

Constraints for table `recipes`:

```
ALTER TABLE `recipes`
  ADD CONSTRAINT `userRM` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`)
ON DELETE CASCADE ON UPDATE CASCADE;
```

Table structure for table `users`:

```
CREATE TABLE `users` (
  `user_id` int(11) NOT NULL,
  `token` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `username` varchar(255) NOT NULL,
  `created` int(11) NOT NULL,
  `admin` tinyint(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Indexes for table `users`:

```
ALTER TABLE `users`
  ADD PRIMARY KEY (`user_id`);
```

AUTO_INCREMENT for table `users`:

```
ALTER TABLE `users`
  MODIFY `user_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
```

## 4.2.2 Queries

Queries used throughout:

Get user id:

```
SELECT user_id
 FROM users
WHERE username ="'.$_COOKIE['username'].'";
```

Used to check if user is logged in:

```
SELECT *
FROM users WHERE
username = '".$_COOKIE['username']."'
AND token = '".$_COOKIE['token'].'";
```

Queries used in Homepage:

Get Posts Info (Popular posts desc.):

```
SELECT sum(f.vote), r.title, r.user_id, r.created, r.image, r.recipe_id FROM recipes r inner join
feedback f on r.recipe_id = f.recipe_id group by r.title, r.user_id, r.created, r.image, r.recipe_id
order by sum(f.vote) DESC LIMIT 15;
```

Get Posts info(Recent posts desc.):

```
SELECT sum(f.vote), r.title, r.user_id, r.created, r.image, r.recipe_id FROM recipes r inner join
feedback f on r.recipe_id = f.recipe_id group by r.title, r.user_id, r.created, r.image, r.recipe_id
order by r.created DESC LIMIT 15;
```

Queries used in Account page:

get token for username and password (for sign in):

```
SELECT token
FROM users
WHERE username = '".$_POST['username']."'
AND password = '".$_POST['password'].'";
```

check if username exist (for sign up):

```
SELECT username
FROM users
WHERE username = '".$_POST['username'].'";
```

sign up user:

```
INSERT INTO users
```

(token, password, username, created, admin)
VALUES ('".$encryptedToken."', '".$_POST['password1']."', '".$_POST['username']."',
'".time()."', 1);

Queries used in Settings page:

Change password:
UPDATE users
SET password = '".$_POST['newPassword1']."'
WHERE username = '".$_COOKIE['username']."'
AND token = '".$_COOKIE['token']."'
AND password = '".$_POST['oldPassword']."';

Delete Account:
DELETE FROM users
WHERE username = '".$_COOKIE['username']."'
AND token = '".$_COOKIE['token']."'
AND password = '".$_POST['dPassword']."';

Queries used in Profile page:

Used to check if user exists:
SELECT user_id, created FROM users WHERE username='" . $profile_username . "';

Number of post:
SELECT count(*) from recipes WHERE recipes.user_id = " . $profile_id . ";

Post Score:
SELECT sum(f.vote) from recipes r inner join feedback f on r.recipe_id = f.recipe_id WHERE
r.user_id = ".$profile_id."  group by r.user_id;

Get Posts Info (Popular posts desc.):
SELECT sum(f.vote), r.title, r.user_id, r.created, r.image, r.recipe_id from recipes r inner join
feedback f on r.recipe_id = f.recipe_id WHERE r.user_id = ".$profile_id." group by r.title,
r.user_id, r.created, r.image, r.recipe_id order by sum(f.vote) DESC;

Get Posts info(Recent posts desc.):
SELECT sum(f.vote), r.title, r.user_id, r.created, r.image, r.recipe_id from recipes r inner join
feedback f on r.recipe_id = f.recipe_id WHERE r.user_id = ".$profile_id." group by r.title,
r.user_id, r.created, r.image, r.recipe_id order by sum(f.vote) DESC;

Queries used in Add Recipe page:

Add recipe:
INSERT INTO recipes(title, ingredients, description, created, user_id, image)
VALUES('" . $_POST['title'] . "', '" . $_POST['ingredients'] . "', '"
. $_POST['instructions'] . "', '" . $date_created . "', '"
. $rows[0] . "', '" . $target_dir . "');

Queries used in Recipe page:

Add comment:
INSERT INTO feedback

(user_id, recipe_id, comment, created, type)
VALUES('.$user_id.', '.$_REQUEST['recipe_id'].', "'.$_POST['comment'].'", '.time().', “c”);

Get recipe info:
SELECT *
FROM recipes
WHERE recipe_id = "' .$_GET['recipe_id']. ‘”;

Get number of votes:
SELECT SUM(vote)
FROM feedback
WHERE type = "v"
AND recipe_id = "' .$_GET['recipe_id']. ‘”;

Get username for recipe:
SELECT username
FROM users, recipes
WHERE users.user_id = recipes.user_id
AND recipe_id = "' .$_GET['recipe_id']. ‘”;

Get comments for recipe:
SELECT feedback_id, user_id, comment, created
FROM feedback
WHERE recipe_id = '.$_GET['recipe_id'].'
AND type = “c”;

Get username from comment:
SELECT username FROM users where user_id = ".$row['user_id'].";

Queries used in Admin page:

Get admin value:
SELECT admin
FROM users WHERE token = "'.$_COOKIE['token'].'" AND admin = 0;

Delete post:
DELETE
FROM recipes WHERE recipe_id = ".$id.";

Delete User:
DELETE FROM users WHERE user_id = ".$id.”;

Delete Comment:
DELETE from feedback WHERE feedback_id=".$id.”;

Get user id from username:
SELECT user_id from users WHERE username='".$r_id.”’;

Queries used for votes:

vote status:
SELECT vote FROM feedback where user_id = ".$user_id." and recipe_id = ".$row['recipe_id']."
and type = ‘v’;

```
make vote:
INSERT INTO feedback
        (user_id, recipe_id, vote, created, type)
        VALUES('.$user_id.', '.$_REQUEST['recipe_id'].', '.$vote.', '.time().', "v");

update vote:
'UPDATE feedback
SET vote = '.$vote.'
WHERE user_id = '.$user_id.'
AND recipe_id = '.$_REQUEST['recipe_id'].'z
AND type = "v";

UPDATE feedback
SET vote = 0
WHERE user_id = '.$user_id.'
 AND recipe_id = '.$_REQUEST['recipe_id'].';
```

# 5. Implementation Process

## 5.1 Resolved Difficulties

### 5.1.1 Allowing the user to associate an image with a recipe post.

As previously mentioned, the restrictive nature of the CIS server's configs meant that php was not able to handle the upload of files and media to the server. Originally, we had intended for users to upload their own images to the own server when posting photos. We had already implemented the code on our local environments only to find out that the php config file blocked this action.

Instead, we proposed that a user may use a source url on another image hosting site if they wish to submit a recipe with an image. The code that uploads files is available as readily available and can quickly be enabled once the proper permissions are changed. We can confirm this works on a local environment. If the CIS server lifts its restrictions, we will make the necessary adjustments.

### 5.1.2 MySQL Restrictions: Allowing the user to view sorted recipes by popularity or recency.

Currently, there are several restrictions put in place on the CIS's server's mysql interface. Many things are locked down and cannot be changed. Several queries had to be rewritten to accommodate the added restrictions. One example:

To load data for the homepage, one would need to grab data for recipe titles, recipe ids, recipe author usernames based on the submitter's id, and its score -- Essentially, spanning data across all three of our

users tables. To increase performance and reduce the tax of queries on the server, several queries were joined, unioned, or subqueried. For the purpose of demonstration, the query to load posts by most recent was initially written as:

```
SELECT sum(f.vote), r.title, r.user_id, r.created, r.image,
r.recipe_id FROM recipes r inner join feedback f on r.recipe_id =
f.recipe_id group by r.title order by sum(f.vote) DESC LIMIT 15;
```

Although this ran well on our local environments while using the same MySQL version on the deployment environment, MySQL threw an error because full-grouping-only was enabled. We were unable to change change this and we had to rewrite this query, as with several others. The new query is now written as:

```
SELECT sum(f.vote), r.title, r.user_id, r.created, r.image,
r.recipe_id FROM recipes r inner join feedback f on r.recipe_id =
f.recipe_id group by r.title, r.user_id, r.created, r.image,
r.recipe_id order by sum(f.vote) DESC LIMIT 15;
```

More annoyingly so, reversing some queries were also locked down when their opposite was not. For example: you could drop your own database, but you cannot create another database. So if you ran a script (which a team member almost did) that drops a database and reinstantiates a new one, the script will run up to dropping the database since creating a database was locked down on the server. If this was to happen before a large deadline, the team would be SOL until IT could create it again for us. And so, we had to scan all queries carefully to make sure that all queries were reversible and only altered table data, not database settings or setup.

# 5.2 Future Enhancements

## 5.2.1 Allow the user to modify a recipe post/comment.

Our website current has the ability for the user to add a comment or post. Super users have the ability to both add and delete posts. If allotted more time, we would like to also allow the user to have the ability to also delete and modify only the comments and posts they themselves have created. The modification would be done on the recipe details page.

## 5.2.2 Allow the user to reply directly to a comment left by another user.

We would also like to include functionality that would allow users to directly reply to other user comments using username mentions. This simple feature would help organize comments in a way that would improve the social aspect of our website.

## 5.2.3 Enhancing the way the user inputs recipe ingredients.

Currently, users are able to add ingredients in the ingredients text input field found on the Add Recipe webpage. Instead of inputting both ingredient name and amount in the ingredients text field, our plan is to allow the user to input each in separate sections. In the future, the ingredients text field would be split into

two sections: ingredient amount and ingredient name. The numerical value of of the ingredient amount would be inputted by the user and he/she would indicate the measurement of the amount (e.g. cups, ounces, grams, etc.) using a drop down menu. The user would then indicate the ingredient name by editing the ingredient name input.

### 5.2.4 Implement a recipe search bar and categories web page.

In the future, we plan to implement a search bar that would allow users to search through our recipes database for specific recipes. The user would simply need to input keywords and all related recipes would be retrieved and displayed for the user to view. Along with a search bar, we would also like to create a categories section (e.g. breakfast, lunch, dinner) which would allow the user to view specific kinds of recipes.

### 5.2.5 Implement the adding of more admins, and super-admins.

As of now, there is only one admin user: admin. As the app would scale, it would be difficult for one person to administer governance on the site. We approached the idea of an admin promoting regular users as admins for the purpose of moderating, but scrapped the plan due to governance issues amongst other admins. The alternative would be to implement super-admins, who are permanent to the site and cannot be affected by lower-admins attempting to dethrone their status.

# 6. Website Implementation Details

Meals for a Steal

Homepage: http://cis444.cs.csusm.edu/group_C/index.php

CSUSM MySQL and Server Group login:

Username: group_c
Password: RhzMCyDR6Q0su7nz

# 7. Contributions

Daniel Favela:

- Implemented javascript functionality to the mainpage to simulate loading posts.
- Implemented php functionality to the main page to display information retrieved from the db.
- Created the database view for easier implementation of queries to the main page.
- Assisted with the initial ERD concept.

Andre Abedmamoore:

- Introduced most of the technologies used in this application to several group members, including the following: jQuery, HTTP Request, Structured-CSS, git and github, phpMyAdmin, MAMP and WAMP, and user authentication as well as the importance of encrypting sensitive user data such as passwords and access tokens in a database.
- Contributed to the development, debugging, and planing of all pages created, functionally and visually. This includes the style and layout of wireframes, backend logic and data handling/validation/retrieval, client-side data handling, and ensuring the app cross-browser compatible and mobile compatible.
- Helped manage group members and delegated tasks after introducing different ways of solving problems and accomplishing action items.
- Deployed the application on the CIS server and made record of any adjustments necessary as well as accomplishing many of those adjustments.
- Checked into all of my teammates branches using git/github, comment and fixing their code and suggesting improvements. I reviewed all code merges for approval going into the master branch.
- Specific app features that I developed: Loading a user's profile, updating or inserting votes via an HTTP Request and displaying that using jQuery, user authentication using tokens, sensitive data encryption, changing account settings such as deletion and changing passwords, displaying recipes' datas based on date or total score.

Scotty Cardwell:

- Implemented javascript functionality to perform the simulation of upvote and downvote options on recipes.
- Assisted in the design of the information flow diagram as well as the initial ERD concept.
- Assisted with the implementation of the server-side recipe.php.

Malialosa Taupule:

- Assisted with the implementation of both the client and server-side of addrecipe.php
- Assisted with the implementation of the client-side of admin_page.php
- Assisted with the implementation of the server-side of recipe.php
- Performed testing on overall application: identified and reported application bugs to the appropriate team member
- Assisted with creating test queries needed by our application to perform the appropriate behavior

Eduardo Martinez:

- Assisted with implementation of client-side profile.php
- Assisted with implementation of the server-side of admin.php
- Assisted with the implementation of the server-side of settings.php
- Assisted with ERD concept