Recipe App RecipeBook - Share your original recipes

**Jana Micaela Gablan: c3gablan**

**Jennifer Leung: c4leungj**

**Romal Peccia: c4peccia**

**Dongshi Zhao: c4zhaodo**

**Language and web framework:** Nodejs + Express

**Pages**:

* Home page
* User registration page
* Page for each user profile (created upon each user account creation)
* Page for each recipe (created upon each recipe creation)

**UI elements:**

* The main user interface elements are the register, log in/out buttons; a navigation column, a search field and the body column.
* Users will need to use their mouse, or tap their screen to navigate through the site
* The app’s screen display is different on desktop than viewing on a phone
* The navigation column lists out the recipe categories linked. Each category when clicked shows the output recipes on the body column.
* The search field is for users to be able to search for a recipe containing a keyword that the user provides.
* The body column is where the recipes show up on the home page. It is also where user profile page is designated on the view port. At the bottom of this body column is for stats and navigational components such as pagination links, recipe ratings and number of comments, etc..

**Modules:**

Authentication module:  This module would contain a series of functions that checks if a user has tried logging in with a valid ID with its matching password.

* + attemptSignIn function: Check to see if given username matches given password
  + Allow third-party authentication

Page creation module:  This module will handle creating the 2 main different pages. In addition, it will also handle other common page creation functions.

Interactions: Feedback module, Rating module, Search module, Admin module, Authentication module, Account creation module.

* + createProfile function: create a profile page for a new user, or load for an existing user
  + createRecipe function: create a new recipe page for a user
  + loadRecipe function: Loads an existing recipe onto the page
  + loadLogin function: Load a page for the user to input log-in details
  + loadNewAccount function: Loads the page for the user to create a new account
  + loadMain function: Load the home page
  + \*Side bar categories: Filters that allow users to quickly filter recipes based on rtags

Feedback module:  Every page will have a comments section.

* + storeComment function: Storing the comment in the database
  + storeVote function: Updating the number of up-votes or down-votes.

Rating module: Every recipe page will have a one to five star rating.

* + storeRating function: Storing the rating in the database
  + updateRating function: Updating the average rating for the recipe
  + \*highestRated function: Find the highest rated recipe based on a given time period

Search module: Every time a user searches for something in the app, the search module will look for the phrase typed into the search bar and pulls out any pages matching it.  
Interactions: Recommendation module.

* + searchUser function: Searches user database based on user input
  + searchRecipe function: Searches recipe database based on user input
  + \*filterRecipes function: Filter recipes based on tags
  + \*rankResults function: Sort results based on given criteria

Recommendation module: This module will use the tag database to find other recipes to recommend to the user based on their recipe history.   
Interactions: Search module.

* + findSimilar function: Return a link to recipe based on recently visited tags

Admin module:  This module gives admin accounts additional page options such as deleting comments, recipes, and accounts.

* + deleteComment function: The ability to delete a comment
  + deleteRecipe function: The ability to delete a comment
  + banAccount function: The ability to ban an account
  + unbanAccount function: The ability to unban an account

Account creation module: This module creates a new account for new users and handles email verifications.

* + checkUniqueID function: Check if given ID is unique, if not print error
  + \*checkUniqueEmail function: Check if given email is unique, if not print error
  + \*sendEmail function: Send the user the verification email with the link
  + setCredentials function: Link and store account username and password

\*User group module: This module allows users to create groups to share private messages and recipes.  
Interactions: Messaging module.

* + createGroup function: Creates a new group with the current user as host
  + addMember function: Adds given user to a group
  + sendGroupMessage function: uses the sendMessage function from messaging module to send messages to everyone in the group.

\*Messaging module:  This module allows user to send messages to other users.  
Interactions: User group module.

* + sendMessage function: Sends a message from one user to another
  + messageUpdates function: If the user has followed another user, the user will receive automatic messages about new updates from the user they are following.
  + displayMessage function: Allows user to view messages sent or recieved

**\* indicate additional features that will be implemented if we extra time left to work on after completing all basic features.**

**Database Design**

Each module will need different databases in order to store their relevant information. Below is a list of each database and their keys.

**Users(**uID, username, firstname, lastname, admin)

This contains the basic information about a user’s account. The “admin” field can be set to “banned”, “admin”, or “user” by the admin control functions. Users with the “banned” status will not be able to make comments or recipes, and users with the “admin” status will be able to use the admin panel.

**Passwords**(uID, password, email)

Passwords and emails will be stored separately because they do not need to be loaded as often as the other information in the **Users** table. This information will be loaded only when the user logs in and stored when the user registers

**Recipe(**rID, uID, name, ingredients, steps, rating, time, timestamp, popularity)

When a user goes to the recipe submitting page, the information will be formatted and stored into this database, and loaded when a user searches for a recipe, or when a user’s profile is viewed.

**Rated(**rID, uID)

When a user rates a recipe, a pair will be made. Every time a user rates, this checks that they have not already rated this recipe.

**rTags**(rID, tag)

Each recipe will have tags that will allow the search and recommendation functions to find relevant recipes.

**Comments**(rID, uID, time, message, upvotes, downvotes)

Each recipe will have its own comments, and will be displayed based on most recently posted. This database will be used whenever the comments are upvoted/downvoted, a new comment is made, or a recipe is viewed.

**Reccomendations(**uID, num, tag)

Each user will have up to 5 tags that will be stored for the recommendation system to use, which will change each time the user searches something.

**Additional Feature Databases:**

The admin attribute in the **Users** database will have a new option, “unauth”. Users with this attribute will not be able to post recipes or comments until their status is changed.

For private groups, 3 databases will be added. Additonally, a “gname” attribute will be added to each recipe, and will be set to NULL if the recipe is public, or the name of a group to make it private.

**Groups(**gname, owner)

**Members(**gname, uID)

Each group will have an owner, and upon loading the private group page, all groups a user is either a member of or a part of will be displayed, with owners having the ability to add/remove members.

**Messages(**uID, gname, time, message)

Each group will have its own comments, and will be displayed based on most recently posted. This database will be used whenever a group is viewed.

**REST API design:**

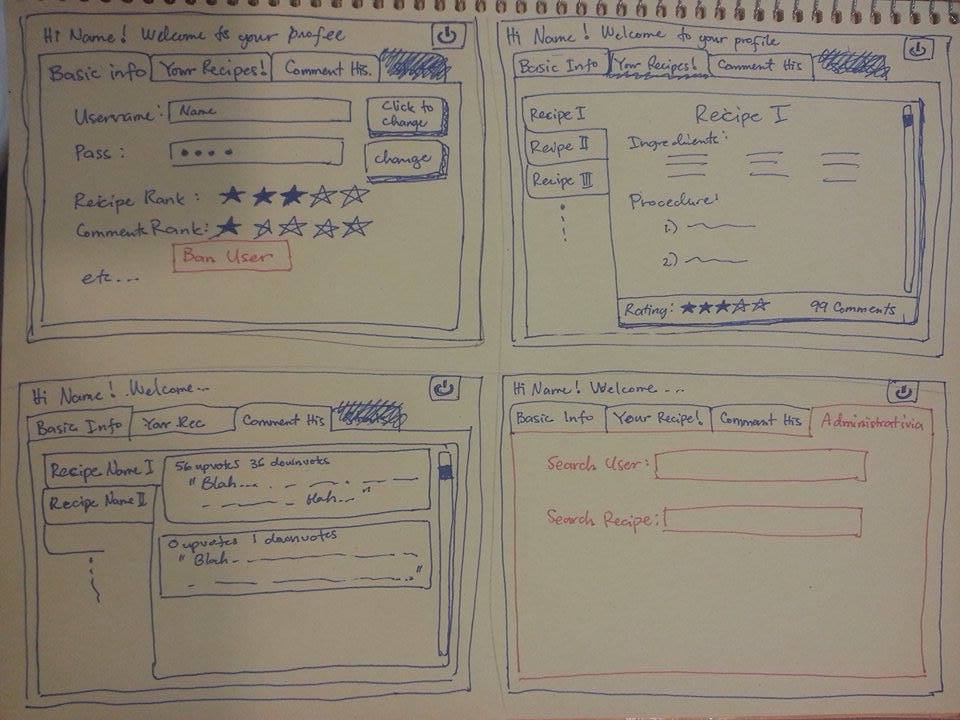
Users are only allowed to view pages such as recipe pages, other user profiles, main page and the login page.  The databases users are only limited to are recipes and other user profiles.

/profiles/[username]

This URL would allow users to view a user profile for a specific user.  The variable to pull out this resource out of the database would be ‘username’.

/recipes/[recipe ID]

This URL would allow users to view a recipe based on an ID associated to the recipe in the database

**Sketches of what the app should quite look like:**

