Sprint #2 Backlog

Previous sprint tasks that need to be redone

- **Web application must be secure and protect confidentiality of a user's ImHungry data**
    - Register isn't required, but it's good we have it
    - We want all the pages to only be accessible by SSL so each url so have https
    - The user can't just use the website if they enter the url, they need to sign in
        - They can't just type in the url and it'll work-- they have to sign in
        - There should be tests for this! (minus some points lol)
- **Set the radius of the restaurants search (Cindy)**
    - Turn meters into miles
    - Create an error message for when you type in a small radius and there are no results. Instead of showing an empty section under the restaurants column, tell them there's an error.
- **View results of prior searches by clicking on a quick access list that shows prior search terms. (Backend- Cindy, Frontend- Alex & Lazlo)**
    - Show this at the bottom of the results page NOT the search page
    - Show them all on one line, and it needs horizontal scroll
    - Make this attractive
    - Make mini collage to display on top of each term
        - Cache photos in SQL database tied to query
        - Pull photos from database that match query (if they exist)
            - If not, query from Google Images API
        - Generate collages in same fashion with smaller box size
            - For all recent searches, generate a mini-collage
            - Scale the size of all mini-collages to be smaller
    - When you click anywhere on the box then it takes you to its individual page
- **Allow for pagination of results returned by the search**
    - Make this separate for recipes and restaurants
    - Always split it into a max of 5 pages
    - Current page number is bolded
    - Create a previous and next button
    - If the page is in the "middle", it should display the number in the middle
        - Look at google's search page for reference
- **User interfaces must look modern and be attractive (Kayla)**
    - Create a different text/sentence for the sign in page
        - Just "sign in" and I'm Hungry at the top
    - Rainbow-fy everything to make it more colorful
        - Use a gradient rainbow background

New tasks to do

- **Maintain information beyond just a single session**
    - Satiated by requirement 1 and 4 (database will provide persistence for 1 and 4)
        - Requirement 1 was “web application must be secure and protect confidentiality of a user’s I’m Hungry data”
        - Requirement 4 was “view results of prior searches by clicking on a quick access list that shows prior search terms”
    - Add a table for each of the lists
    - User id foreign key, boolean isRestaurant, String name)
    - Querying and insertion to lists in DatabaseModel.
    - On Redirection to ManageList, call DB model and re-sort based on the orders of the object. Do NOT just redirect with ResonseModel like as is done now.
    - Update ManageListView to not use ResponseModel, but rather the manually sorted and added lists from RedirectionController.
    - On “Add to list” , call DB model insertion
    - On “delete from list” call DB model removal
    - On Move list, call DB model removal and insertion.
- **View results of prior searches by clicking on a quick access list that shows prior search terms (Cindy- backend, Kayla- frontend)**
    - Modify our prior searches functionality based on Sprint 1 Feedback
        - Feedback notes: https://docs.google.com/document/d/1I2CuQhordy_wiB1h9S52qoAtIUwOXwmkdzS2w3W0AmM/edit
- **Keep track of a grocery list for selected recipes**
    - Create class for grocery list for selected recipes
    - Add querying of groceries added for a specific user
    - Add insertion of new grocery.
    - Create table in MySQL database with the following structure
        - Int object id
        - Selected Item String not null
        - user id foreign key to User(ID)
        - Int order.
    - Update RedirectionController.java to have the functionality for adding to grocery list. It will call the update function from DB model.
    - Next to each ingredient in DetailedRecipeView, have an “add to groceries” button
    - Modify Ingredient to have a boolean `isGroceryList` item
    - In RedirectionController, when the user redirects to manage the grocery list, get all the ingredients using DB model, and add an attribute for List<IngredientModel> ingredients
- **Reorder any of the three predetermined lists (Alex & Kayla)**

    - Frontend
        - Try a JS library that can easily implement reordering
            - http://sortablejs.github.io/Sortable/
    - Backend
        - Add an “order” int not null to each of the items in the list
        - On sort change - send the position of the new object. E.g. List item B is now at position 3.
        - In DatabaseModel, add a function to query the list item and all of their orders.
        - Then adjust the orders based on the position of the changed object (if an object is moved lower, all objects between its original position and the new one move up, as an example). This can be done in ListItemModel or a separate class.
        - Create function to Update the orders on the new DB with the new orders
        - In ListItemModel, add a member var for storing sorted order and modify `compareTO` to sort items assigned to a list by position as well, if both are in the same list.
- **Refactor the Code 4 Times**
    - Pull out List Management into its own Servlet from RedirectionController
        - Gets rid of RedirectionController “God Class”
    - Move out all `action` strings to an enum with member String for comparison.
        - Gets rid of “Magic Strings”
    - Set up proxy object for AllRecipes parsing. (e.g. same limit, radius, term should be cached as a JSON or similar using GSON serializable). Then on request, we check for a Term-limit-radius trio, then deserialize the JSON rather than Hit the API
        - Saves Money, Network bandwidth on our end
    - Set up proxy objects for Yelp Parsing. Yelp API response can be directly saved as a serialized JSON, limited only to the fields we need for restaurants. Leads to greater abstraction, abstract the URL call to its own class, where it can replay JSON as well.
        - Saves money, network bandwidth, makes our design lead to less redundant API calls, especially with the new “past searches” feature.
- **Set up Travis CI**
    - Set up Running of Cucumber Tests
    - Set up running for JUnit
- **Update Code based on Feedback from Sprint 1 Demo**
    - Feedback on “UI Must be Modern and Attractive”
    - Feedback on “Add radius to searches”
    - Feedback on “Security of I’m Hungry Data”