Team Names: Sampurna Basu, Sagar Punhani, Kevin Wang, Matt Wenner
Product Name: Entree
Product Summary: Social recipe web app aimed at college students.

Software Summary:
We will be using a model-view-controller (MVC) approach in which the user interface is essentially accessing the data from the backend and presenting it in a user-friendly manner.

- mySQL - used for database management
- Eclipse - used as the IDE for full stack development
- Java - used for back end (server side) interaction with user information
- HTML5/CSS3/Javascript - used for creating front end GUI
- Google App Engine/Cloud Platform - Integrates Java Backend with frontend components
- Bitbucket - used for version and source control

State Diagram Summary
Due to the various functionalities incorporated into our product, we conducted each functionality's state analysis individually. See the State Diagrams folder for associated visual diagrams.

- Diagram 1: Grocery List (reference attached State Diagram)
    - State 1: Display items currently in grocery list. After completing any action associated with the grocery list, this state will be invoked and the grocery list will be refreshed to display the items currently in the grocery list. For example, once something is removed from the grocery list (i.e. State 3), the grocery list will be updated so that the removed item is no longer visible. This state is initially entered when the user navigates to their personal page.
    - State 2: Item(s) added to grocery list. This state will be entered using one of two transitions. One of these is if the user manually inputs a grocery list item using the text field in the grocery list GUI - this causes a single grocery list item to be added to the grocery list. The other is if the user clicks the "add to grocery list" button in their personal recipe feed and all the ingredients associated with the recipe are added to the grocery list - this causes multiple grocery list items to be added to the grocery list. After completing this state, the program returns to State 1.
    - State 3: Single item removed from grocery list. This state will be entered when the user clicks a "remove" button associated with a certain grocery list item. After completing this state, the program returns to State 1.

    - State 4: Clear all items from grocery list. This state will be entered when the user clicks the “Done Shopping” button. After completing this state, the program returns to State 1.
- Diagram 2: Post on Personal Page (reference attached State Diagram)
    - State 1: Post exists on personal page. This state is entered in one of two ways. The first way is when the user shares a recipe post from the main feed or another user’s personal page. This post is then called to existence on the user’s personal feed. The other way is when a user manually inputs a recipe (with associated image and text) using the “add recipe” button at the top of their personal page. This state is exited and goes to State 2 when the “view more details” button associated with the post is clicked.
    - State 2: View detailed state of post. This state is entered when the “view more details” button is clicked on the current post. During this state, the full details and contents of the ingredients and instructions associated with the recipe are revealed to the user. This state is exited when the user clicks the “view more details” button and returns to State 1.
- Diagram 3: Post on General Page (reference attached State Diagram)
    - State 1: Post exists on general page but is not added to recipe list. This state is entered in several ways. One is when the current user manually creates a new recipe on their personal page or shares another post on the general page. Another way is when a friend (someone the current user is following) either shares or manually adds a recipe to their personal page.
    - State 2: Post exists on general page but is added to recipe list. This state is entered when the current user shares the post using the “add to recipe list” button associated with the post. Once in this state, the post cannot return to State 1 but may go back and forth between this state and state 3. Once this state is entered, the “add to recipe” button is disabled.
    - State 3: View detailed state of post. This state is entered when the “view more details” button is clicked on the current post. During this state, the full details and contents of the ingredients and instructions associated with the recipe are revealed to the user. This state is exited when the user clicks the “view more details” button and returns to the undetailed state of either State 1 or State 2 (whichever state that the current state originated from).
- Diagram 4: Current User Personal Page (reference attached State Diagram)
    - State 1: Static viewing of the personal page. This state is entered when the user navigates to their personal page by clicking on their profile icon. If the page needs to be updated, state 1 will transition to state 2. If the

user wants to search their personal recipes, state 1 will transition to state 3. If the user clicks on the news feed icon, state 1 will transition to state 4.
    - State 2: Refresh state of the personal page. This state is entered in one of two ways - when a recipe is entered manually or when a new recipe is added by sharing a post on the main feed. State 2 is exited to State 1 when the information is done updating and the GUI is updated accordingly. State 2 is also entered when any aspect of the user's grocery list is updated.
    - State 3: Search state. This state is entered when the user uses the search bar to keyword search their personal recipes. While searching the recipes on your personal feed will be replaced by the recipes relevant to what the user is searching. The results of the searched will be ranked via a ranking algorithm detailed later. This state is exited to State 2 when the user clicks on their profile icon.
    - State 4: Return to main page. This state is entered when you click on the news feed icon at the top of the personal page. Entering this state results in the user navigating away from the personal page and into the main feed page.
- Diagram 5: Other User Personal Page (reference attached State Diagram)
    - State 1: Static viewing of the other user's personal page. The other user's grocery list is not visible to the current user. This state is entered when the other user's username is clicked via either a post from current user's main feed or the search results of the current user's personal page.
    - State 2: Refresh state of the other user's personal page. This state is entered when the user is scrolling through the other's user's personal recipes feed and scrolls past the first 20 posts in which case the page needs to be refreshed to display the next 20 posts. This state is also entered when the other user adds a recipe post to their personal page by either manually adding it or sharing a recipe on other user's main feed. This state is exited when the page is done updating the relevant information.
    - State 3: Return to main page. This state is entered when you click on the news feed icon at the top of the personal page. Entering this state results in the user navigating away from the other user's personal page and into the main feed page.
- Diagram 6: Main News Feed (reference attached State Diagram)
    - State 1: Static viewing of the main news feed page. This state is entered when the user logs in to the service. During this state, all GUI

components of the main page are displayed. This state is also entered when the user clicks on the news feed icon from their personal page.
    - State 2: Refresh state of the main news feed page. This state is entered when a new recipe is added to the feed via either the user's friends adding a recipe to their respective personal page or the user adding a recipe to their own personal page. This state is also entered when the featured menu content is updated. This state is exited back to state 1 when all the information is updated.
    - State 3: Navigate to personal page. This state is entered when the user clicks on their profile icon at the top of the main news feed page. Entering this state results in the user navigating away from the user's main news feed page and into the current user's personal page.
- Diagram 7: Featured Menu (reference attached state diagram)
    - State 1: Static viewing of the featured recipes news feed. This state is entered when the user logs in to the service. During this state, the 25 featured recipes of the day are displayed in the featured menu section. This state is also entered when the user clicks on the news feed icon from their personal page.
    - State 2: Refresh state of the featured recipes news feed. This state is entered every 24 hours when a new 25 featured recipes are selected by a selection algorithm detailed later. This state is also entered when a user clicks on the "more information" button on a post in the featured menu (see state diagram of post on general page).

Class Summary

See the State Diagrams folder for associated visual diagrams.

- Diagram 1: Main Page Hierarchy
    - Class Hierarchy: The main page class has a featured menu class and a main news feed class. The featured menu class has a list of the class featured post. The news feed class has a list of the class post.
    - Member Variables (type)
        - Main Page Class: Main News Feed (custom object), Featured Menu (custom object)
        - Featured Menu Class: List of Featured Post Objects
        - Main News Feed Class: List of /*Featured*/ Post Objects
        - GUI Components: search box text field, search box button, personal icon, company icon, main feed content pane, featured feed content pane, log out button
        - Information about posts can be viewed below.

    - Methods
        - Main Page Class: getters, getMorePosts()
        - Featured Menu Class: getters, updateFeaturedMenu(), getFriendsPosts(), /*login(), logout(), authenticate()*/
        - Main News Feed Class: getters, updateNewsFeed(),
        - Information about posts can be viewed below.
- Diagram 2: Personal Page Hierarchy
    - Class Hierarchy: The personal page class has a recipe class, grocery list class, list of the posts, list of usernames following (type string), and list of usernames followers (type string). The grocery list class has a list of class grocery item.
    - Member Variables (type)
        - Personal Page Class: Recipe Object (custom Object), Grocery List Object (custom Object)
        - Recipe Class: List of Posts from User-Recipe Database
        - Grocery List Class: List of Grocery Item
        - Grocery Item Class: String Name and String quantity (number + units)
        - GUI Components: search box text field, search box button, personal icon, company icon, profile information content pane, follow button, personal recipes content pane, grocery list content pane, log out button
        - Information about posts can be viewed below.
    - Methods
        - Personal Page Class: getters, logout(), getMorePosts(), addManualRecipe()
        - Recipe List Class: getters, updateNewsFeed()
        - Grocery List Class: getters, updateGroceryFeed(), addGroceryItem(GroceryItem), removeGroceryItem(GroceryItem), doneShopping();
        - Grocery Item Class: getters, setters, add(), remove()
- Diagram 3: Posts
    - Class Hierarchy: The abstract posts class is the parent class of the featured post class, the user post class, and following post class.
    - Member Variables (type)
        - Post Class (Abstract): timestamp (long int), name (string), picture (Image), username, instructions (list of strings), number of reheats (int), ingredients (list of custom GroceryItem Object)
        - Featured Post Class: Extends Post Class, int type

        - User Post Class: Extends Post Class, int type
        - Following Post Class: Extends from Post Class, int type
        - GUI Components: Image associated with recipe, add to recipe list button, view more information buttons
    - Methods
        - Post Class (Abstract): addToRecipeList(), showDetails(), reheat(), addToGroceryList(), addPostToMapForSearch()
        - Featured Post Class: getters, reheat(), showDetails()
        - User Post Class: getters, reheat(), showDetails()
        - Following Class: getters, reheat(), showDetails()

Database Details Summary

All of our tables are part of one database schema. We have 4 tables as detailed below.

- Table of Registered Users (see sample table below)
    - During login, if a user tries to log in, the authenticate() method will be called and will check the username and password against our database and return a boolean.
        - If the returned boolean is false, the user will be prompted to try logging in again. If the returned boolean is true, the user will be logged in
    - If the user attempts to register, a newUser() method will be called which adds the user information to our database
        - The user should then be able to log in as a registered user, described above
    - The information in this database, except for passwords and email, will also be displayed in the Personal Page of the corresponding user through our getInfo() method.

Table of Registered Users

User ID	*Username*	*Email*	*Password (hashed)*	*Image*	*Favorite Food*
1	ttrojan	ttrojan@usc.edu	9292920	Image.png	Pizza

- Table of all Registered Recipes (see sample table below)
    - This table stores information for each recipe
    - Whenever a new recipe is created on the personal page through the makeNewRecipe() method, our database is updated with the information

of that new recipe. Name, picture, recipe, and ingredients are mandatory and the rest of the fields are initialized by themselves
- Whenever a user shares a recipe, that post's number of reheats increases for the ranking algorithm later through the addHeat() method
- A subset of these posts will be stored for each user to be displayed on their personal page

Table of Registered Recipes

Recipe ID	*Name*	*Picture*	*User*	*Instructions (List of Strings)*	*Ingredients (List of Strings)*	*Number of Reheats*	*Timestamp*
1	Pizza	pizza.png	1	["cook it", you're done"]	["cheese", "tomatoes"]	7	10:25:15

- Table of all Featured Content (see sample table below)
  - This table represents the data for all current and past featured content
  - Every 24 hours, the database pulls 25 new featured recipes from our yumly api and sets those as the current featured recipes in our featured tab through our getFeatured() method. [deprecated]
  - Every recipe is also added to the main database

Table of Featured Content

Featured ID	*Recipe ID*	*Name*	*Picture*	*Instructions (List of Strings)*	*Ingredients (List of Strings)*	*Number of Reheats*	*Timestamp*
1	1	Tacos	tacos.png	["cook it", you're done"]	["cheese", "tomatoes""lettuce"]	3	10:25:20

- Table of User-Recipe Relations (see sample table below)
  - This table represents the recipes that should appear on every user's feed.
  - Every time a user creates a new post, they add that post to their list of recipes through the addRecipe() method
  - Every time a user shares a post, they add that post to their list of recipes through the addHeat() method

- Every time this database gets updated, all of the user's followers' news feeds get updated with the new recipe through the refreshNewsFeed() method

Table of User-Recipe Relations

User ID	*List of Recipe ID's*
1	[1,2,5]

Algorithms Summary

- Search capabilities including searching users of our services and recipes (both personal and general). On the main feed page, any searches made using the search bar will keyword search the database of all recipes and return relevant results. On the personal feed page, we will be performing augmented search. If the user includes an "@" before the search terms, the database of usernames will be parsed. If not, the user's personal recipe feed will be searched.
    - For friend search, the returned search results will be ranked based on a breadth first search algorithm and the highest ranking results will signify the user with the largest number of mutual friends.

      Pseudocode:

```
for each friend v and current user u
     pred[v] = -1, d[v] = inf.
     Q = new Queue
     Q.enqueue(u), pred[u] = -2, d[u]=0
     while Q is not empty
          v = Q.front(); Q.dequeue()
          for each neighbor, w, of v:
               if pred[w] == -1 // w not found
                    Q.enqueue(w)
                    pred[w] = v, d[w] = d[v] + 1
sorted friends = sort(all friends by d[w])
display sorted friends
```

    - For the recipe search (on both the main page and personal page), the returned search results will be ranked based on the number of times added which is tracked by the number of reheats associated with that post.

      Pseudocode:

```
index all words of all recipes in a map to a set of
all recipes it links to
map<Key - word,set<Recipe>>
finalset set<Recipe>
for each key in search query
     finalSet union with map<key>
display all recipes in finalSet
```

- Update featured menu content by fetching additional 25 recipes (randomly) from given Yumly API database. This fetch will be conducted every 24 hours and will fetch all associated information of the recipe (including ingredients, instructions, title, and image).

```
Pseudocode:
on time = 0:00:00 am
create new list<Recipe>
for int i = 0; i < 25; ++i
     parse Yumly Recipe into custom post object
     list.append(post object)
     featuredDatabase.insert(post object)
updateFeaturedRecipes() [deprecated]
```

- Parsing ingredients from recipe after “add to grocery list” button clicked on a post in the personal recipe feed. After parsing, accordingly add items to the grocery list. If the item already exists in the grocery list, only update the quantity of that item.

```
Pseudocode:
get recipe from addToGroceryList()
for GroceryItem item in recipe
     if item not in groceryList
          add item to groceryList
     else
          increase quantity of existing item in
          groceryList by item’s quantity
```

Miscellaneous Major Data Structures Summary

- Friends will be stored in a social network graph.
    - We are keeping a local database represented as an adjacency list which is a linked list of linked lists that matches every user to all of their followers
    - An additional adjacency list will be needed that matches every user to all the people they are following

- Each user should be associated with their grocery list. To accomplish this, an unordered map will be used with the user id as the key and a vector of grocery items as the value. A vector has been selected due to its removal capabilities.

GUI Summary

See scanned GUI plan.