SI 543- The Pantry Planner

Lavanya Kumar, Huaying Song, Daniel DiNicola, Sikang Li

- **What is it about?**

The Pantry Planner provides users with cooking options based on ingredients available to them. Users are welcome to view existing recipes and add their own favorite recipes. System functions include: meal planning based on existing ingredients, meal recommendations and reviews from other users, system-generated recipes from a variety of world cuisines, a user profile that displays each user's activity on the site, favorite recipes, and photos.

- Who are the usergroups?

Our users include anybody who is interested in food, whether they just like trying new recipes, are looking for assistance in the kitchen, or like to share their recipes with others. From college students and single professionals to working families and foodies, this application focuses on men and women in their 20s and 30s but will benefit a wide range of people. Whether you're a kitchen novice or a culinary expert, The Pantry Planner is the right app for your appetite.

- **Why is it important?**

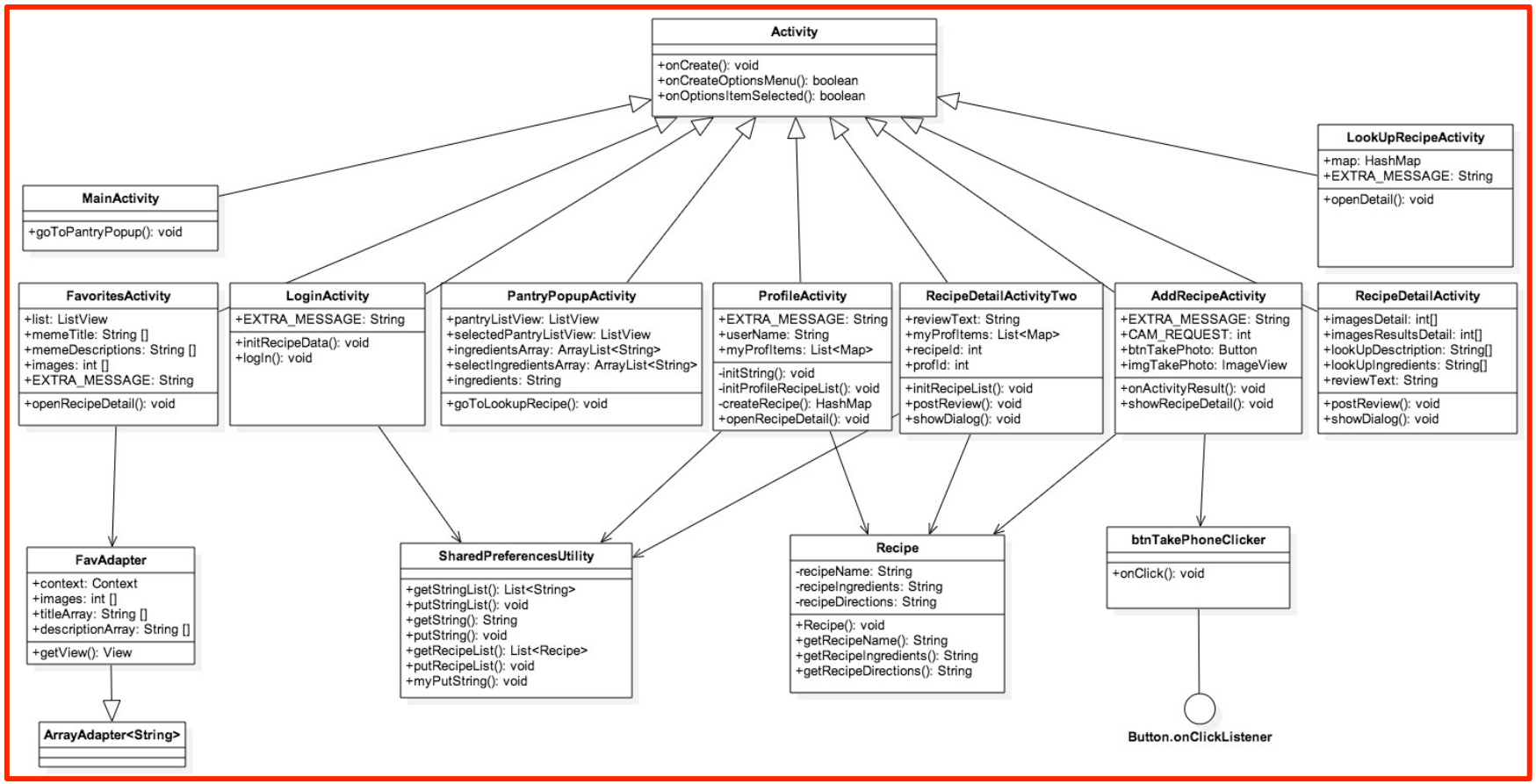
This app can help plan that special meal or save a trip to the store by cooking with ingredients available, build user's comfort around the kitchen while saving money and reducing food waste (cooking with what you have on hand), and facilitate business for restaurants and specialty stores who advertise or offer coupons. The Pantry Planner also provides a potential venue for aspiring chefs who want to start their own restaurants or build name recognition.

- **What are the competitors?**

1. Direct Competitors are existing meal planning apps. Some of the examples are: Food Planner, an app that only has system-generated recipes and doesn't facilitate interactions between users. AllRecipes, another app that allows users to rate uploaded meals as well as leave comments on recipes. However, users do not have the ability to retrieve meals based on the ingredients they have on hand.
2. Indirect Competitors includes local restaurants who offer takeout /delivery, cooking blogs and culinary books, and social media that can share cooking experiences.

- Why is your idea better? Based on research about existing food planner apps, our app, "The Pantry Planner," combines both system-level suggestions (menus given by system) and user activities, which will greatly encourage collaboration across the virtual community.

The Pantry Planner – UML Diagram
As of 09 December 2014



Pantry Planner Final Report:
Altering the Profile Page based on User Identity
Huaying Song

The Pantry Planner allows users to access their own profiles from the action bar menu item “My Profile”, or to view other users’ profiles by tapping on their avatars (See Figure 1). To present different profile pages (including different user avatars, usernames, user descriptions and user posts) based on whether the current user is accessing his or her own profile or another user’s profile, I took the following steps (See Figure 2).

First, in LoginActivity, the EditText with the id “editTextName” is found and typecasted and assigned to the variable “userName” based on the android EditText class. The value of “userName” is passed and stored in a string variable “displayName”. When calling the *putString()* method of the SharedPreferecesUtility class, the string value of “displayName” is stored in SharedPreferences with a key “loginName”. Next, if the user is accessing his or her own profile from the action bar menu, while starting a new intent to the ProfileActivity, the *getString()* method will be called to grab the string value with the key “loginName”, and passed it as an extra message along with the intent to ProfileActivity with a key “id”. Inside ProfileActivity, the extra message is mapped to a string variable “userName” with the key “id” via the Bundle class. An if condition is used here to decide whether to show the current user’s profile page or changing views on the profile page into another user’s information. The logic is: if the value of “userName” does not match the value of “loginName” stored in SharedPreferences, which means the user is not accessing his or her own profile, the code inside the if condition will be executed to set the “My Posts” list as invisible, and set the text of TextView “post_view”as “His/Her Posts:”, and change the text of TextView “profileAbout”, and change the image resource of the ImageView “avatar” to another user’s image inside the drawable. So in this case, it doesn’t fall into the if condition and the code won’t be executed. The current user’s profile page will be pulled up as it was written originally in the ProfileActivity’s xml and java files.

However, if the user is tapping on another user’s avatar and username, the text value of the username’s TextView will be passed as an extra message while starting the intent to the Profile Page with the key “id”. This time it falls into the if condition and the code inside the if condition will be executed to change the views in ProfileActivity.

Pantry Planner Final Report:
Altering the Profile Page based on User Identity

Huaying Song

(Screen Captures and Code)

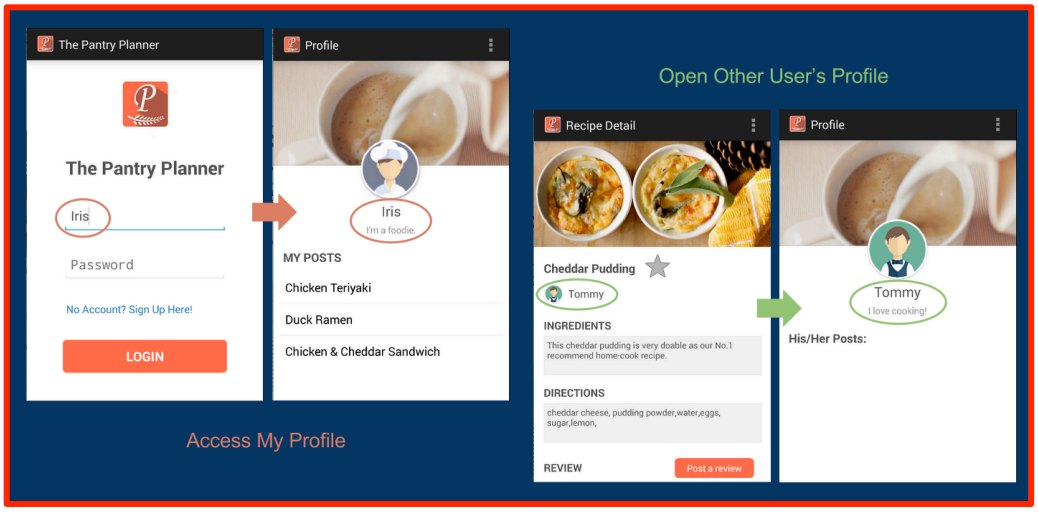


Figure 1. Access My Profile (Left) and Open Another User's Profile (Right)

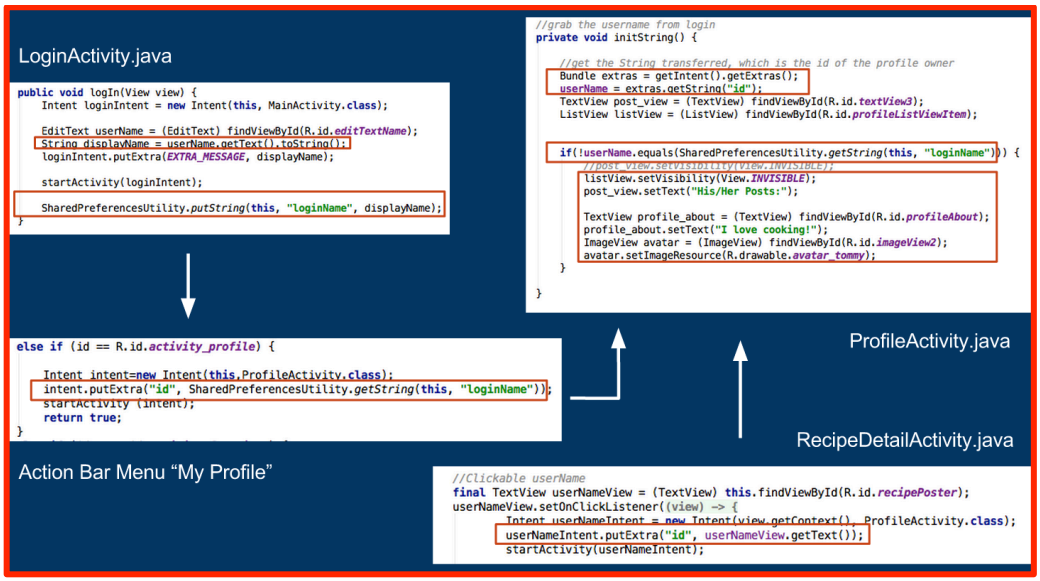


Figure 2. Source Code for Altering the Profile Page

Pantry Planner Final Report:
Pantry Popup Functionality – Moving Strings from One List to Another
Dan DiNicola

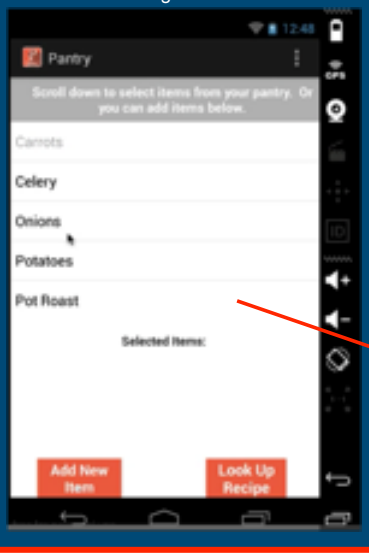
I was responsible for the Main Activity and the Pantry Popup Activity within The Pantry Planner. In order for the user to look up a recipe in The Pantry Planner, my intent was for the user to select ingredients from his / her pantry, then use those ingredients to initiate the "look up recipe" activity using our recipe database.* After several unsuccessful attempts to implement this, I decided that the best way to implement would be to remove from one list and add to a second list using the *OnItemClickListener* and listView *getItemAtPosition* methods. Both **ingredientsArray** and **selectedIngredientsArray** are array adapters of type String. I am most proud of learning how to first instantiate ingredients array, then implementing the transfer of items selected by the user from **ingredientsArray** to **selectedIngredientsArray.** In order to improve the appearance of the application and clearly show the transition from **ingredientsArray** to **selectedIngredientsArray**, I added an animation framework - view.animate().setDuration(2000).alpha(0).withEndAction(new Runnable() - from an Android Developer tutorial. Screen captures from the Genymotion emulator, along with java code associated with these points in the Pantry Popup Activity are attached on the next two pages.

*Note, we did in fact create a database, but I was not able to implement it successfully. That's definitely something we'll do should we decide to carry this application forward.

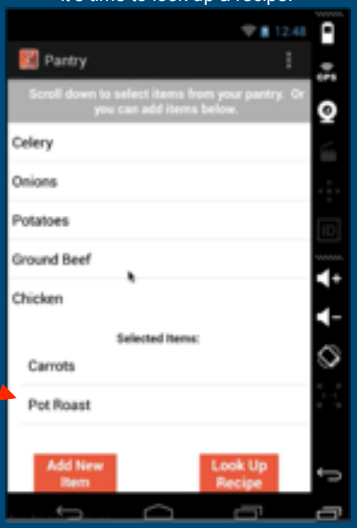
Pantry Planner Final Report:
Pantry Popup Functionality – Moving Strings from One List to Another
Dan DiNicola
(Screen Captures and Code)

Viewing and Selecting Items from User's Pantry

Scroll thru your pantry to select ingredients

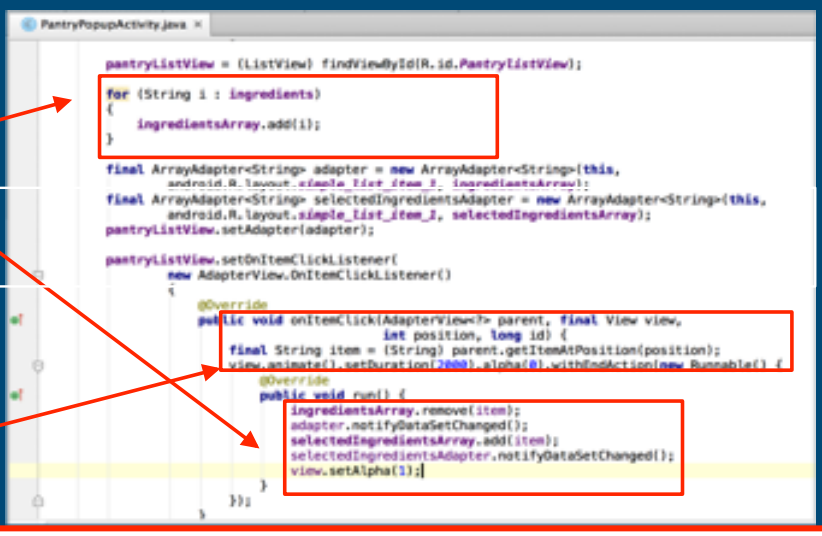


Once you've chosen your ingredients, it's time to look up a recipe!



Taking from One List and Placing in Another

Sets the array "pantryListView", then passes those items selected by user to "selectedIngredientsAdapter". Includes a pause within the method in order to make the transition clearer for the user.



Pantry Planner Final Report:
Pantry Popup Functionality – Moving Strings from One List to Another
Dan DiNicola
(Screen Captures and Code: Continued)

```
for (String i : ingredients)
{
    ingredientsArray.add(i);
}

final ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, ingredientsArray);
final ArrayAdapter<String> selectedIngredientsAdapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, selectedIngredientsArray);
pantryListView.setAdapter(adapter);

pantryListView.setOnItemClickListener(
        new AdapterView.OnItemClickListener()
        {
            @Override
            public void onItemClick(AdapterView<?> parent, final View view,
                                    int position, long id) {
                final String item = (String) parent.getItemAtPosition(position);
                view.animate().setDuration(2000).alpha(0).withEndAction(new Runnable() {
                    @Override
                    public void run() {
                        ingredientsArray.remove(item);
                        adapter.notifyDataSetChanged();
                        selectedIngredientsArray.add(item);
                        selectedIngredientsAdapter.notifyDataSetChanged();
                        view.setAlpha(1);
                    }
                });
            }
        }
);
```

Pantry Planner Final Report:
Altering multiple strings and images
Connie Li

The function that I am proud of is to open the recipe from the favorites list. The pantry planner allows users to save their favorite recipes and access them in a list. Users are also able to click on each recipe in the list, and view the details on another page (recipe detail page). In the class, we talked about how to alter views in android development, where we especially focused on altering one singe text. However, in this project, my challenge is how to alter multiple strings and images at once. When users click on one recipe, it should show the name, description, ingredients and images of the particular recipe on the detail page. The way I solved this challenge is that I created a string variable and save the recipe's name, ingredients, directions and image address (which is a integer) in it. And then I passed the variable to the recipe detail page, where I split out each part. Using what I have learnt in class, I altered the strings and made them shown on each view where they should be. And for the image position integer, I used it as a pointer to refer to the image I should retrieve. The images—their addresses actually—have already been stored in the same file where I get the image address pointer.

Pantry Planner Final Report:
Altering multiple strings and images
Connie Li
(Screen Captures and Code)

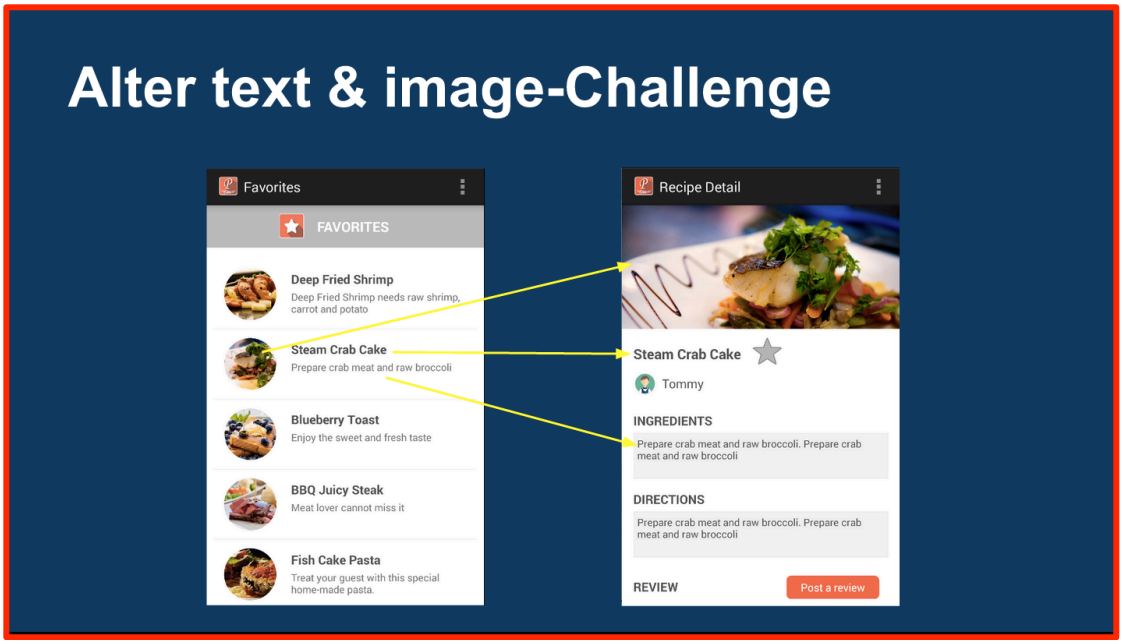


Figure 1. Open recipe from favorite list

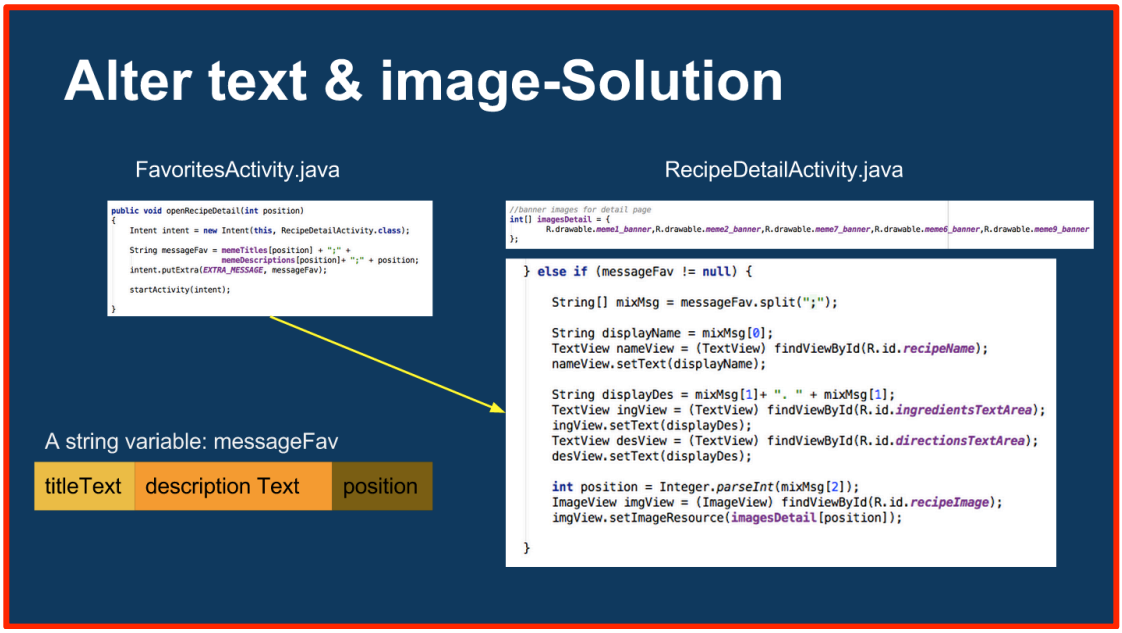
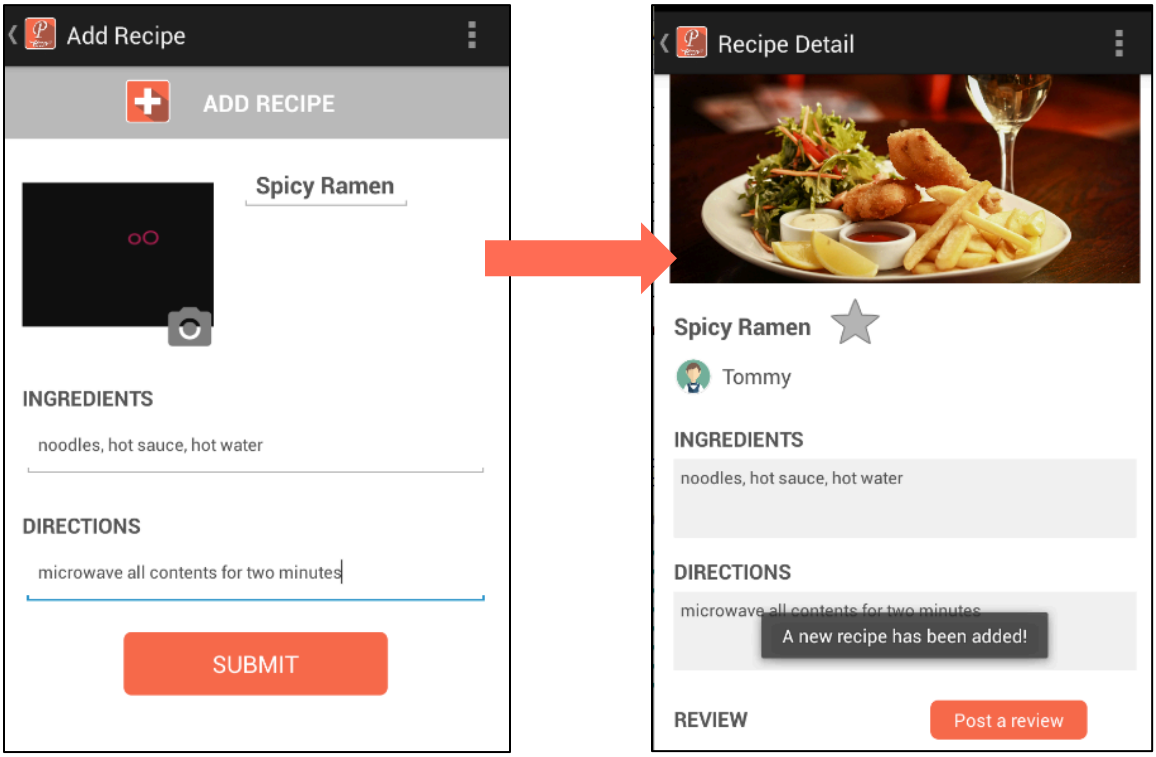
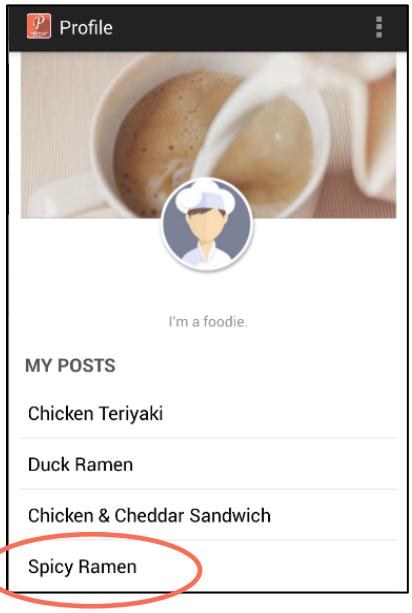


Figure 2. Code implementation

Pantry Planner Final Report
Adding a Recipe / Recipe Detail Activity Two
Lavanya Kumar

In this feature, the user can add a recipe name, its ingredients, and its directions in the “AddRecipeActivity” class, which is passed onto the “RecipeDetailActivityTwo” class. (After this recipe is added, the user can navigate back to the “MyProfile” activity to see the name of the recipe added to a list on that screen. However, I won’t be explaining this functionality.)





Add Recipe Activity

```
public void showRecipeDetail(View view) {

    List<Recipe> recipes = SharedPreferencesUtility.getRecipeList(this,"recipes");

    int id = recipes.size();

    EditText recipeName = (EditText) findViewById(R.id.recipeName);
    EditText recipeIngredients = (EditText) findViewById(R.id.recipeDetailIngredients);
    EditText recipeDirections = (EditText) findViewById(R.id.profileAbout);

    recipes.add(new Recipe(recipeName.getText().toString(), recipeIngredients.getText().toString(),
    recipeDirections.getText().toString() ));
```

In the method showRecipeDetail (named after the value for the onClick attribute of the "Submit" button in the addrecipe.xml file), I specify what happens when the user presses that button. There are "EditText" classes, recipeName, recipeIngredients, and recipeDirections. They correspond with the views, "recipeName," "recipeDetailIngredients," and "profileAbout," (not very intuitive, but it matches the id of that particular view in the xml file) which we are going to typecast to type EditText so that our intent knows to take the user generated content of those classes. Then, we take the instances of these EditText Classes defined previously and append or add them to our list "recipes" with <Recipe> strings defined in our "Recipe" class.

Continuing in our AddRecipe activity, we have created a new object of type Intent and are calling it recipeIntent. Once we have instantiated it, we give it two parameters. "this," or the context or current class that we are in, and "RecipeDetailActivityTwo.class," or the destination of the intent so that the program knows the origin and destination of the user generated content. We then create a new String object called "recipeMessage" and set it equal to the last string in our list called "recipes." This id was defined in the code, "int id = recipes.size()" in the second line of code under the showRecipeDetail method. We also want to pass the information that the user entered with the recipeIntent, so we do so by passing the method "putExtra" with the key "EXTRA_MESSAGE" (string defined at the beginning of the Class) and the value "recipeMessage."

```
    SharedPreferencesUtility.putRecipeList(this,"recipes", recipes);

    //toast message for successfully adding recipe
    Toast.makeText(getApplicationContext(), "A new recipe has been added!",Toast.LENGTH_LONG).show();


    Intent recipeIntent = new Intent(this,RecipeDetailActivityTwo.class);
    String recipeMessage = String.valueOf(id);

    recipeIntent.putExtra(EXTRA_MESSAGE,recipeMessage);
    startActivity(recipeIntent);

    finish();

}
```

Recipe Detail Activity Two

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_recipe_detail);


    Intent intent = getIntent();


    initRecipeList();

    String recipeMessage = intent.getStringExtra(AddRecipeActivity.EXTRA_MESSAGE);

    TextView recipeNameText = (TextView) findViewById(R.id.recipeName);
    TextView recipeIngredientsText = (TextView) findViewById(R.id.ingredientsTextArea);
    TextView recipeDirectionsText = (TextView) findViewById(R.id.directionsTextArea);

    int recipeId = (int) Long.parseLong(recipeMessage);

    recipeNameText.setText(myProfItems.get(recipeId).getRecipeName());
    recipeIngredientsText.setText(myProfItems.get(recipeId).getRecipeIngredients());
    recipeDirectionsText.setText(myProfItems.get(recipeId).getRecipeDirections());
```

In this activity, the information that the user enters is displayed in the appropriate fields. We start by receiving the intent. To do so, we have to create an object of type Intent called “intent” that performs the method getIntent(). Next, we create a String called “recipeMessage” which contains the information that the user entered in the previous activity (taking the EXTRA_MESSAGE key from AddRecipeActivity). Similar to typecasting the Views to type EditText, we do the same in this activity by typecasting the Views to type TextView since it is just displaying the text. Each of these new objects are called “recipeNameText,” “recipeIngredientsText,” and “recipeDirectionsText.” Getting the item from our SharedPreferenceUtility list called myProfItems (which we “put” our item in during the previous activity), we use the recipeId that contains the user-generated content. Then we set “recipeNameText,” the name of our TextView instance, to the value of “recipeId.” However, this specific item in our list contains three values, the name, ingredients and directions. So we have to specify which one to return using the appropriate method (ie. getRecipeName()), defined in our Recipe Class.