¼ Cup Team 16

Regression and Incremental Testing Log

Classification of Components

Module Description

Data Validation

- Description: A class that will be used to check the input from a user to make sure that, that input is valid and won't cause errors in other portions of the code.
- Inputs:
  - Takes in the shopping list or shopping list item that the user wants to add as a shopping list or shopping list item respectively.
- Outputs:
  - Returns a string saying valid if the input string was valid, or an error message to be displayed if it is not valid.

Shopping List Repository

- Description: A class that abstracts the Firebase datastore from the rest of the app. It provides methods to retrieve and update lists in Firebase.
- Inputs:
  - Receives shopping lists from Firebase
  - Receives new lists and list items from activities and fragments
- Outputs: LiveData that is synchronized with Firebase

Shopping List ViewModel

- Description: Holds data used by views that display shopping lists. It outlives activities and fragments to allow their data and state to be quickly restored after a configuration change.
- Inputs:
  - Receives data from user and shopping list repositories
  - UI controllers send id of selected list
- Outputs: LiveData transformed to meet the needs of the UI controllers (Fragments and Activities)

Shopping List UI

- Description: Displays the appropriate information about shopping lists.
- Inputs:
    - Takes data in from the module and displays the data to the screen
- Outputs: Sends changes to the data or new data to the view model

Recipe List

- Description: Displays a list of recipes that the user queries by a keyword
- Inputs:
    - User keyword(s) for query (i.e Apple Pie)
- Outputs:
    - Recipes and information from the Edamam recipe search database

Welcome Message

- Description: Displays a welcome message with a sign-in button
- Inputs: none
- Outputs: none

Category Comparator

- Description: Users have personal sorting orders based on categories. These orders can be applied to lists to have certain items further up or down the list based on that item’s category.
- Inputs:
    - Two items from a shopping list and a category sort order
- Outputs:
    - Integer showing which item should be sorted higher or lower.

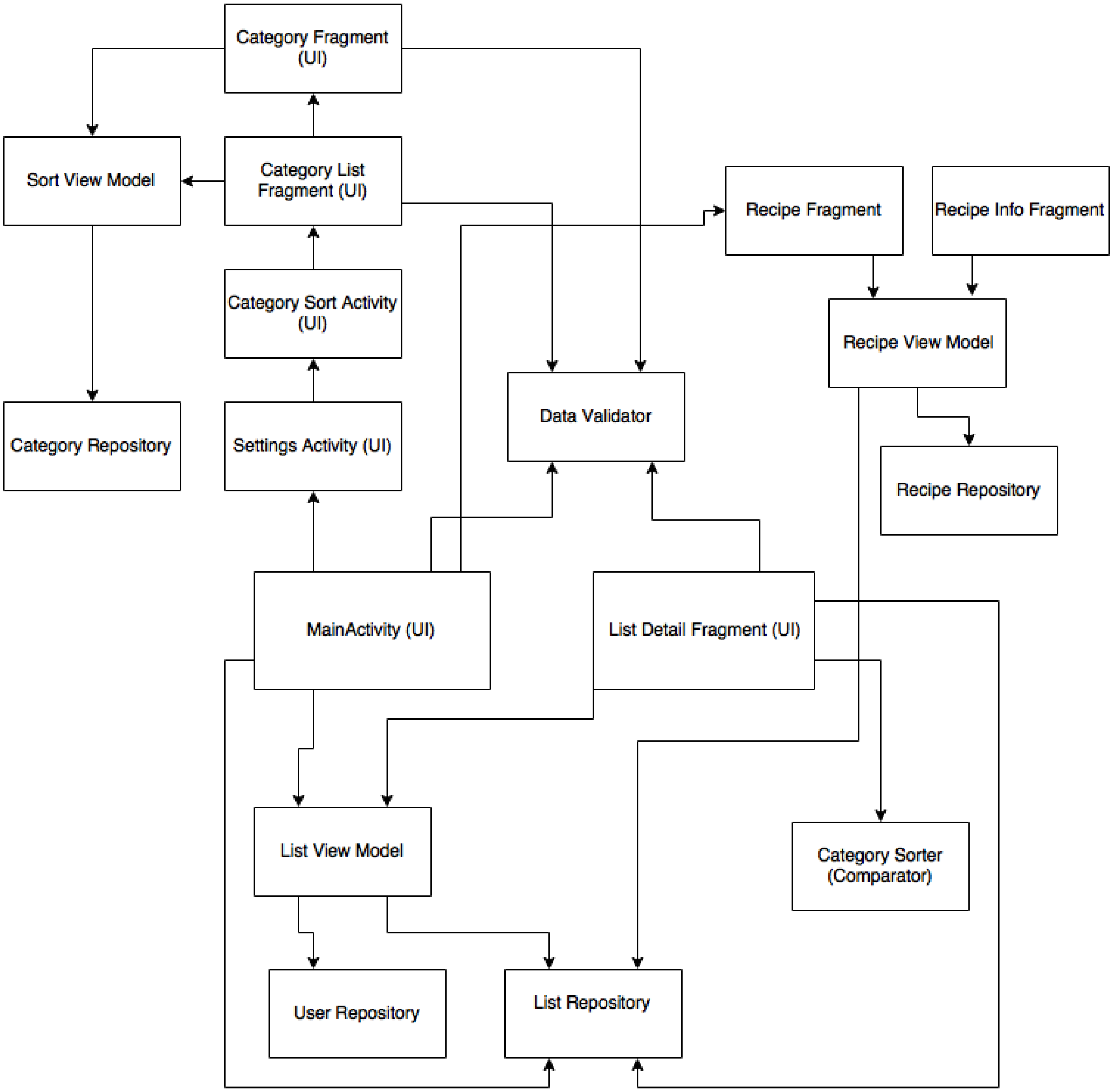
Category Order ViewModel

- Description: Holds data used by views that displays sort orders and the categories held within.
- Inputs:
    - Receives data from the category repository.
- Outputs:
    - Live data for the UI controllers and fragments to display.

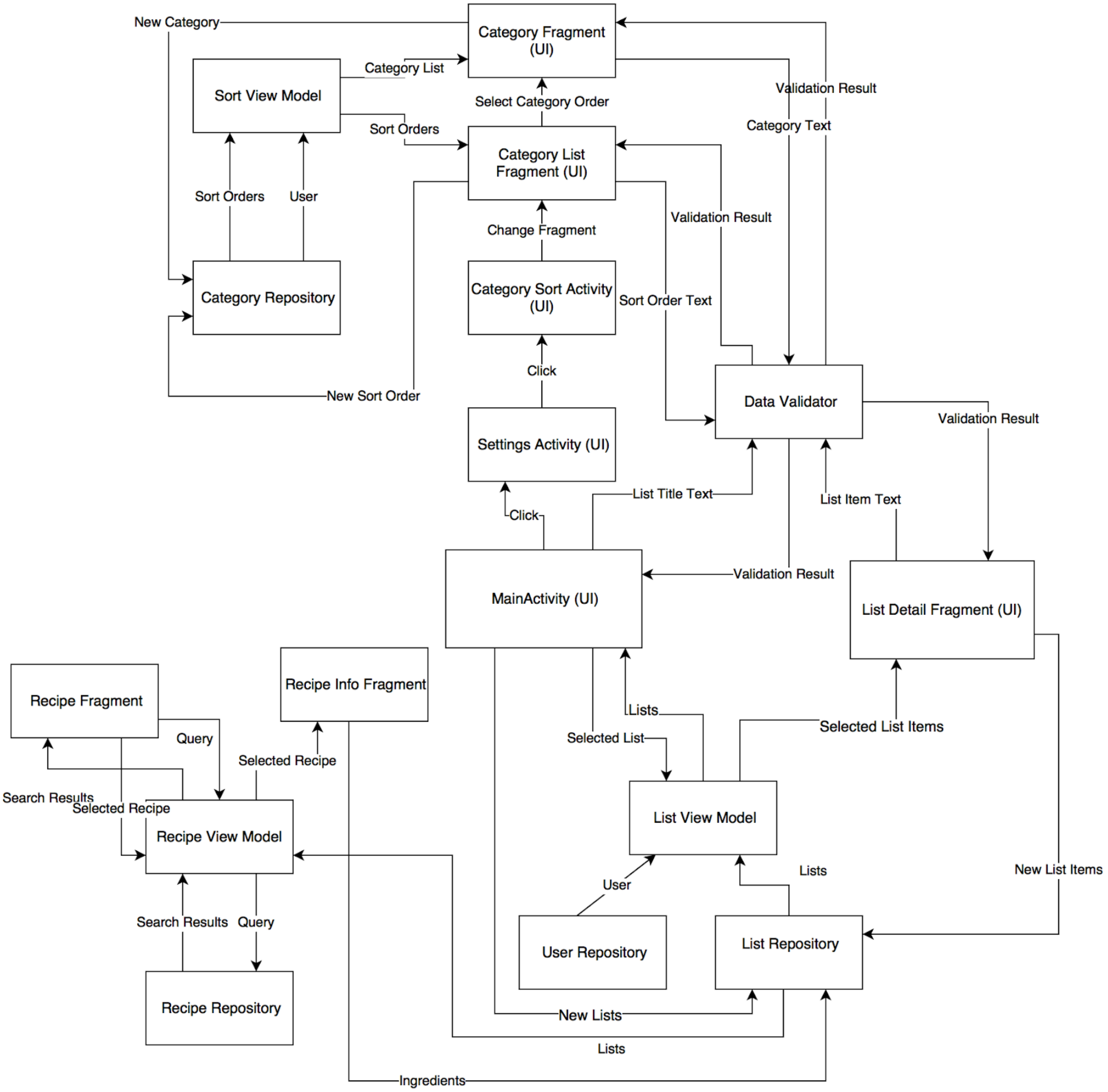
Incremental testing technique

The testing approach used is bottom-up testing. We created the submodules that we needed and tested them using dummy values (drivers) acting as the main modules.

Dependency Diagram



Module Diagram



Incremental and Regression Testing

Automation

Whole App

- Using the Android Exerciser Monkey, we performed stress tests on the app, in which a series of 500-1000 pseudo-random inputs and events were sent to the app. These tests are used to detect crashes and ANR messages caused by edge cases in a specific series of inputs.
- We used the Firebase Test Lab for Android to perform robo testing on the app. The test lab runs the app on a variety of physical and virtual devices running various versions of Android, in order to test the app's performance on different API levels.

Data Validation

- Using JUnit testing, we created a set of automated tests to make sure that the object's names are valid input strings and we tested that the category sort names are valid as well. The tests were used to unit test the class and then regression and incrementally test adding items and data validation.

Category Sorting

- Using JUnit testing, we created a set of automated tests to make sure that the category comparator correctly sorts a list of items. The tests were used to unit test the sorting function, as well as for regression and incremental testing of future uses of the category sorting feature.

Shopping List Repository

- Using JUnit testing in order to test creating new lists automatically. The tests were used to unit test the shopping lists, and then also for regression testing the lists and to aid in finding bugs during incremental testing.

Defect Logs

Module:	Category Sorting

Incremental Testing:

Defect #	Description	Severity	How to correct
1	Allows the addition of empty category sort order names, or names with just spaces.	3	Add check in the validation class for creating category sort orders.
2	Allows the addition of categories inside of sort orders with just spaces.	3	Add check in the validation class for creating categories.
3	When adding a ViewModel to the category sort activity, it was incorrectly initialized in the createFragment method instead of onCreate, causing the ViewModel to not be initialized when the activity resumed and leading to crashes.	1	Moved the ViewModel initialization to the Activity's onCreate method.

Regression Testing:

Defect #	Description	Severity	How to correct
1	When upgrading the RecyclerView adapters to ListAdapters, the new	2	This happened because DiffUtil compares the submitted list with the current list using strict

	adapters did not always update the list ordering when a new list was submitted.		equality. When they refer to the same object, it does not make further comparisons, so when only the order of a list is changed, it doesn’t notice the ordering change. A workaround was implemented that submits a new list instance to the adapter when the ordering changes.

Module:	Data Validation

Incremental Testing:

Defect #	Description	Severity	How to correct
1	When trying to test for a string with only spaces the wrong trim statement was written in the if statement. It would not allow anything but a string with all whitespaces to be entered.	1	Switched to check if the trim was 0 not > than 0.
2	When returning a string for the category sort being a category of just spaces, the wrong error was printed.	3	Fix the print statement so that it was a little bit more descriptive for the user.

Regression Testing:

Defect #	Description	Severity	How to correct
1	When testing the validation with other elements, it was noticed that at one point the wrong boolean was checked in an if statement and thus none of the items were being added correctly	1	The boolean was switched to be checking for if the string was valid instead of empty and this fixed the error.

Module:	Shopping List UI

Incremental Testing:

Defect #	Description	Severity	How to correct
1	Adding category sort functionality to the Shopping list ViewModel caused the app to crash when the user attempted to log out.	1	Checks for null pointers were added to the shopping list ViewModel and repository.

Regression Testing:

Defect #	Description	Severity	How to correct
1	When a user attempts to add an item before creating a list, the app crashes.	2	Add checks to ensure a valid list is selected before attempting to add an item.
2	After deleting a all lists, the items from the last list are still visible. Interacting with them causes a crash.	2	Add additional checks to hide list items UI when there are no lists. And do not take action of the user attempts to interact with a non-existent item.
3	Adding a feature which sorts lists by creation default caused no lists to be displayed.	1	This was caused by the way Firestore is designed. In order to issue compound queries a

			composite index must be manually created that includes both fields. Creating this index solved the issue.

Module:	Recipe List

Incremental Testing:

Defect #	Description	Severity	How to correct
1	When we added to the search field for the Recipe List Module the recipes populated but when we viewed a recipe and then went back to the recipe search, the default search populated the recipe list.	2	We updated the recycler view and deleted the if statement that was unnecessary for this point in the code. We also got rid of the hardcoded value in the query.
2	After connecting the recipe fragment to the ViewModel, when the user attempted to add ingredients to a non-existent list (if the user had not created a list yet), the app crashed.	2	An additional check was added to the recipe info fragment to ensure that the selected list exists and notify the user if it does not.
3	Searching for Recipes after the users max Edamam API calls per minute were depleted cause the UI to show a progress bar that would spin until the user changed the UI context.	2	A search timeout was implemented so the spinner would show an error message after 10 sec of not receiving the correctly parsed HTTP response.

Regression Testing:

Defect #	Description	Severity	How to correct
1	When initially querying the edamam API, the RecipeList object did not match the JSON object sent to use by the database. Therefore, gson was unable to construct Ingredient objects in the	1	Inspected the json object being sent in detail and added the missing “Hits” array to the

	RecipeList, so searching returned ‘null’ recipes.		RecipeList object.

Revised Product Backlog

Functional Requirements

ID	Functional Requirement	Hours	Status
1	As a user, I would like to be able to create an account and sign in with an email address and password.	10	Completed in Sprint 1
2	As a user, I would like to be able to sign in with my Google account.	4	Completed in Sprint 1
3	As a user, I would like to create shopping lists.	5	Completed in Sprint 1
4	As a user, I would like to add an item to a list.	4	Completed in Sprint 1
5	As a user, I would like to assign a category to an item.	4	Completed in Sprint 2
6	As a user, I would like to create category sort orders.	10	Completed in Sprint 2
7	As a user, I would like to assign category sort orders to a list.	8	Completed in Sprint 2
8	As a user, I would like items to be categorized automatically.	8	Not Completed
9	As a user, I would like to share lists with other users.	10	Not Completed
10	As a user, I would like to be able to browse and search recipes.	7	Completed in Sprint 2
11	As a user, I would like to add ingredients from one recipe to a shopping list.	10	Completed in Sprint 2
12	As a user, I would like to create my own recipes.	5	Not

			Completed
13	As a user, I would like to be able to delete shopping lists.	5	Completed in Sprint 1
14	As a user, I would like to check items off from the list.	8	Completed in Sprint 1
15	As a user, I would like to delete items from a list.	5	Completed in Sprint 1
16	As a user, I would like to create custom categories.	7	Completed in Sprint 2
17	As a user, I would like to use autocomplete suggestions to add items to lists.	8	Not Completed
18	As a user, I would like to see when an item was last purchased.	7	Not Completed
19	As a user, I would like to revoke list sharing rights.	7	Not Completed
20	As a user, I would like to save recipes to my personal collection.	8	Not Completed
21	As a user, I would like to view my saved recipes.	10	Not Completed
22	As a user, I would like to see which recipe a list item was added for.	10	Not Completed
23	As a user, I would like to see the progress of each recipe as items are checked off.	10	Completed in Sprint 2
24	As a user, I would like to receive a notification whenever a shared list was changed.	7	Not Completed
25	As a developer, I would like no duplicate ingredients added from recipes.	10	Not Completed
26	Total:	187	

Non-functional requirements

ID	Functional Requirement	Hours	Status
27	As a developer, I would like changes to lists be persisted across devices in real time.	4	Completed in Sprint 1
28	As a developer, I would like users to only have access to their own lists and lists shared with them.	6	Completed in Sprint 1
29	As a developer, I would like my application to handle errors gracefully.	6	Completed in Sprint 1 and 2
30	As a developer, I would like to track and review bugs.	4	Completed in Sprint 1 and 2
31	As a developer, I would like a database to store data.	6	Completed in Sprint 1
32	As a user, I would like to know when an error occurs.	4	Completed in Sprint 1 and 2
33	As a user, I would like the application to look appealing	6	Completed in Sprint 1 and 2
34	As a user, I would like the application fit to any device screen.	6	Completed in Sprint 1 and 2
	Total:	42	