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 (currently being simulated by stub recipe data)

Welcome Message

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

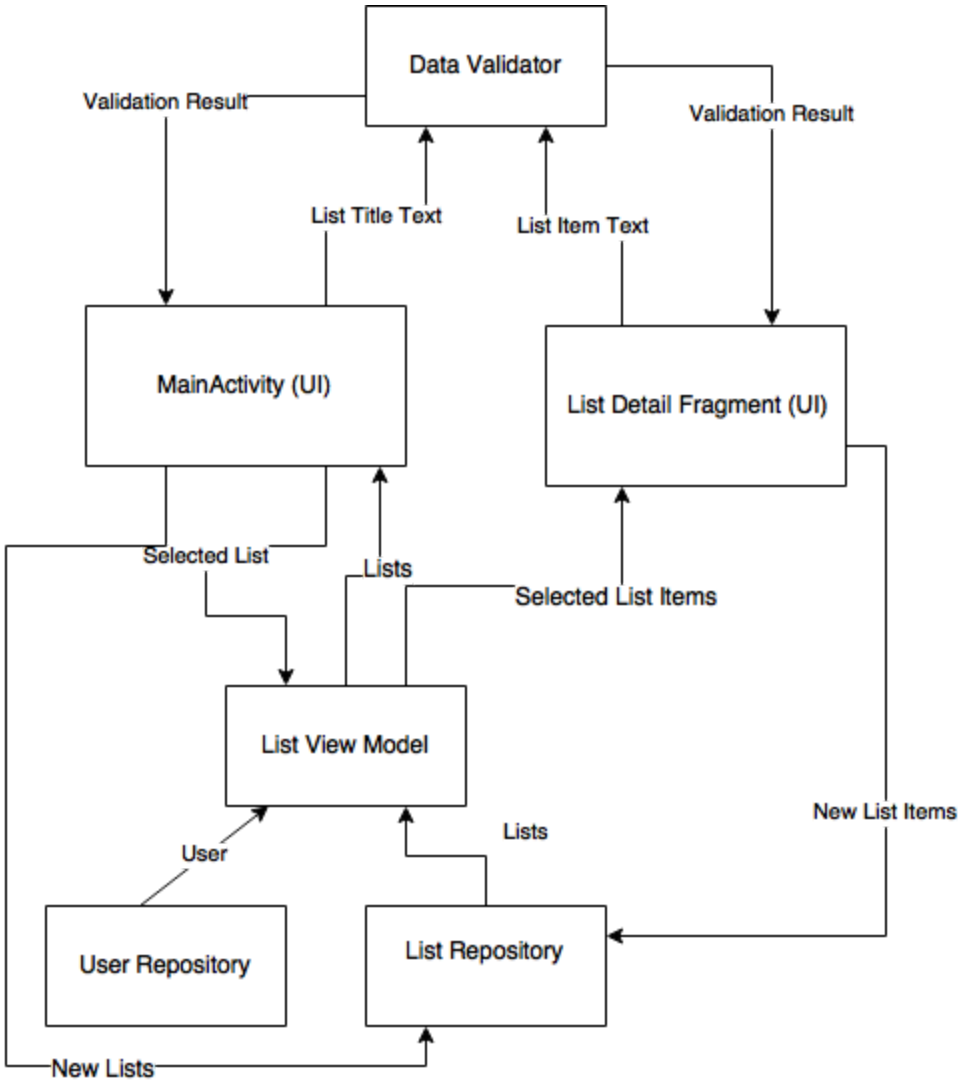
Category Sorting

- 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:
    - List of shopping items needing to be sorted.
    - List of categories used for ordering.
- Outputs:
    - List of ordered items based on the used sort order.

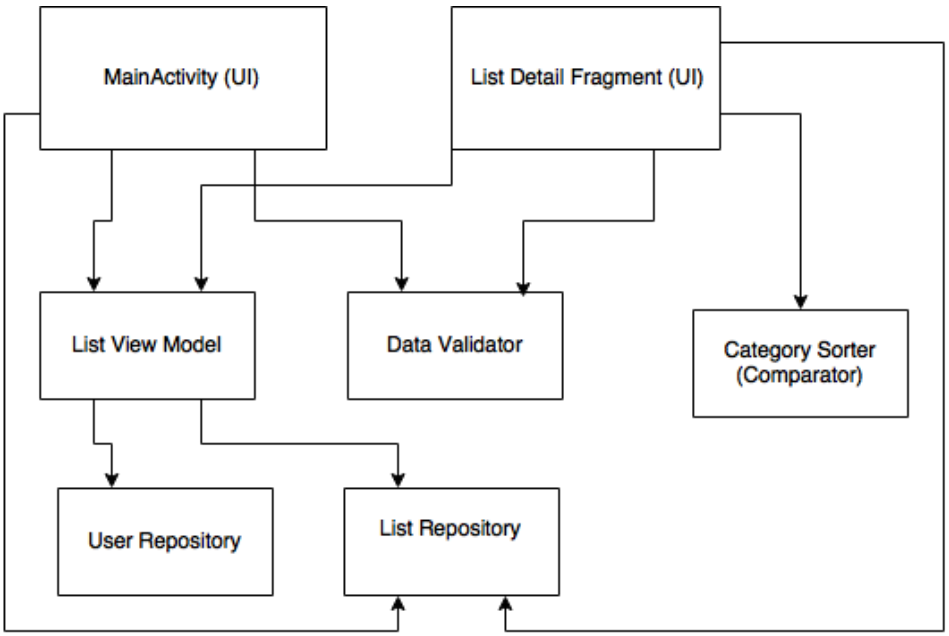
Incremental testing technique

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.

Module Diagram



Dependency Diagram



Incremental and Regression Testing

Automation

Whole App

- Using the Android Exerciser Monkey, we performed a stress test of the app, in which a series of 500 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. 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 sorted list of items are correctly sorted. 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:	Shopping List Repository

Incremental Testing:

Defect #	Description	Severity	How to correct
1	A seemingly infinite amount of items should be able to be added to a users list without the app crashing or suffering heavily in speed.	3	Remove application dependencies that stop the user from adding a certain amount of items.
2	All of the items in a list should be allowed to be deleted. The Shopping List Repository module should be able to handle having no items in the database.	1	Remove all items from any given list using the Shopping List Repository module. Unit tests to delete all of the items from a given list, under a given user, should allow for no items to be in the database. Shopping List ViewModel module should be able to handle no data in the list

			of Shopping List Items.
3	After switching the UI to the SortOrderFragment or the RecipesFragment, switching back to a ShoppingList does not display the items of that list in the ListView. Instead, the items of the previous fragment are persistent in the list view.	3	Assure that the Shopping List Repository module has complete separation from the Recipe List module. The fragments and view models for both should have independent variables and division in code segments

Regression Testing:

Defect #	Description	Severity	How to correct
1	Using the Shopping List UI module, add 100 items to the list. Items should carry down through the Shopping List Repository module and then the Shopping List ViewModel module should receive the changes, which then in turn passes the changes back to the Shopping List UI module. (Or alternatively add through function inputs into the View Model module/junit)	3	Assure that the Shopping List UI module has the capability to exponentially grow in displayed items. Add scroll functionality to the Shopping List UI. Assure that firebase has the capability to handle the necessary amount of item additions. Verify that the overhead of information passing through the view module does not overwhelm the stack.
2	When adding the Shopping List UI module to the Shopping List Repository module	1	Delete all items from any given list. The Shopping List UI module should show that there are no items in the list. This was fixed by

	the UI should be able to adequately reflect having no items in the database. If all items are swiped away (deleted) then both the Shopping List UI module and the Shopping List Repository module should be able to handle the null data values		adding appropriate handlers for null data in the UI modules front end.
3	When the Recipe List module fragment is loaded into the MainActivity page the data should overwrite and consume the data from the Shopping List UI module. If a recipe is populated in the UI then a Shopping List is populated from the Shopping List UI module the data should be replaced.	2	Assure that there is separation between the two modules. Create handlers that verify that the information has been changed. To do this we grab the drawer and wipe any left over data using != null checks and then populate the new data.

Module:	Category Sorting

Incremental Testing:

Defect #	Description	Severity	How to correct
1	The sorting algorithm had mismatched indexes when first written, causing out of bounds exceptions when running, crashing the app.	1	Fix the mismatched indexes in the algorithm to represent the correct indexed arrays.
2	When given more items than categories to sort with, the sorting algorithm is supposed to append the extra items to the end of the final list. However, it appended the index of each item, and not the actual item.	3	Fix the code typo and make sure to add from the content at the index, not the actual index.

Regression Testing:

Defect #	Description	Severity	How to correct
1	After testing the sorting function, realized we needed to check for edge cases such as trying to sort an empty item list, sorting with an empty category sort, and both at the same time.	2	Added edge cases in the sort function to return the item list (If given an empty category sort) or an empty array (if given an empty item list).

Module:	Data Validation

Incremental Testing:

Defect #	Description	Severity	How to correct
1	After adding the data validation into the shopping list it was noticed that data validation was returning a blank string when it was needed to return “valid”	1	Change the return from “” to “valid”
2	When the data validation was added to work with the shopping list item it errored out due to the new item fragment being created.	1	Changing the location of the data check to the item fragment rather than main activity
3	Data validation toast message displaying the incorrect message and the toast going away before the user could read it.	3	Correcting the messages to be more descriptive, and making the toast message longer

Regression Testing:

Defect #	Description	Severity	How to correct
1	When adding a new item in the list page, there was an error with the item name being too long	3	Added a check to make sure that the string length was under 256 chars
2	After adding the data validation into the shopping list it was noticed that data	1	Change the return from “” to “valid”

	validation was returning a blank string when it was needed to return “valid”		

Module:	Shopping List UI

Incremental Testing:

Defect #	Description	Severity	How to correct
1	If items are added to the list they should be in sorted order of when they were created (items newly created appearing at the top of the list)	3	Change the order of acquiring the live data based on the creation date. **Fix in progress**
2	If lists are added they should be in sorted order of when they were created (lists newly created appearing at the top of the page)	3	Change the order of acquiring the live data based on the creation date. **Fix in progress**
3	Shopping List UI should reflect checked off items.	1	Address the checked off item functionality in the UI module. Add swipe to indicate item deletion. The Shopping List Repository should also reflect this item was deleted.

Regression Testing:

Defect #	Description	Severity	How to correct
1	When the Shopping List UI module is added to the Shopping List Repository module the Shopping List Repository	3	Shopping List Repository should grab the live data in

	module should acquire the live data in order by the creation date. Shopping List UI should reflect this sort order.		order from firebase. Shopping List UI module should reflect this. **Fix in progress**
2	When the Shopping List UI module is added to the Shopping List Repository module the Shopping List Repository module should acquire the live data in order by the creation date. Shopping List UI should reflect this sort order.	3	Shopping List Repository should grab the live data in order from firebase. Shopping List UI module should reflect this. **Fix in progress**

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	In Progress, Moved to Sprint 2
6	As a user, I would like to create category sort orders.	10	In Progress, Moved to Sprint 2
7	As a user, I would like to assign category sort orders to a list.	8	Planned,

			moved to Sprint 2
8	As a user, I would like items to be categorized automatically.	8	Planned for Sprint 2
9	As a user, I would like to share lists with other users.	10	Planned, Moved Sprint 2
10	As a user, I would like to be able to browse and search recipes.	7	In Progress, Moved to Sprint 2
11	As a user, I would like to add ingredients from one recipe to a shopping list.	10	In Progress, Moved to Sprint 2
12	As a user, I would like to create my own recipes.	5	Planned for Sprint 2
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	Planned, moved to Sprint 2
17	As a user, I would like to use autocomplete suggestions to add items to lists.	8	Planned for Sprint 2
18	As a user, I would like to see when an item was last purchased.	7	Planned for Sprint 2
19	As a user, I would like to revoke list sharing rights.	7	Planned for Sprint 2
20	As a user, I would like to save recipes to my personal collection.	8	Planned for Sprint 2
21	As a user, I would like to view my saved recipes.	10	Planned for Sprint 2

22	As a user, I would like to see which recipe a list item was added for.	10	Planned for Sprint 2
23	As a user, I would like to see the progress of each recipe as items are checked off.	10	Planned for Sprint 2
24	As a user, I would like to receive a notification whenever a shared list was changed.	7	Planned for Sprint 2
25	As a developer, I would like no duplicate ingredients added from recipes.	10	Planned for Sprint 2
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/Moved to Sprint 2
30	As a developer, I would like to track and review bugs.	4	Completed in Sprint 1/Moved to Sprint 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/Moved to Sprint 2
33	As a user, I would like the application to look appealing	6	Completed in Sprint 1/Moved to Sprint 2
34	As a user, I would like the application fit to any device screen.	6	Completed in Sprint 1/Moved to Sprint 2
	Total:	42	