¼ Cup ?

Test Plan

Identification and classification	**Test Case 001A - Valid Email** **Severity: 1 (Critical)**
Instructions	At the account creation or login page enter a valid email address: **Email: <example>@<example>.<com>**
Expected result	If email address is in a valid format (shown above), the passwords validity will be checked before account creation/login takes place.

Identification and classification	**Test Case 001B - Password Strength** **Severity: 1(Critical)**
Instructions	At the account creation page, enter a valid password: Password must contain: ● 8+ characters ● At least 1 alpha character (a-z,A-Z) ● At least 1 number (0-9)
Expected result	If password meets above requirements, user will be able to create a new account if all other fields are correct. User will be taken to home page of the application.

Identification and classification	**Test Case 0001C - Login Functionality** **Severity: 1 (Critical)**
Instructions	Enter a valid username and password into the login page.
Expected result	If the username and password are valid and are an existing account in our database, the user should be redirected to our

	apps home screen.

Identification and classification	**Test Case 0002 - Google Login Functionality** **Severity: 1(Critical)**
Instructions	Click the button on the login page of the application to sign in with your google account. Android devices takes care of the google sign in process.
Expected result	If login is successful, google account information will be used to add the user to the app's user database and user will be brought to home page of application.

Identification and classification	**Test Case 0003 - Create a shopping list** **Severity: 1 (Critical)**
Instructions	● At the list creation page, user selects "create a new shopping list" ● User types a name into the shopping list "name" field ● User submits the new shopping list.
Expected result	1. Submit button will be disabled until the following name requirements is met: ○ Name is between 1 and 100 characters ○ Name contains at least 1 character that is not a whitespace (space, tab, crlf, etc.) 2. If name is not unique: ○ Warning will be displayed upon submit: "Shopping list name must be unique" ○ User must rename list to continue. 3. If name of shopping list is unique: ○ Shopping list will be added to the database and will be added to the list view of the UI

Identification and classification	**Test Case 0004 - Item is added to shopping list** **Severity: 1 (Critical)**

Instructions	● User selects option to add an item to their shopping list ● User is prompted with a text box to enter name of the item they would like to add (all characters except <crlf> are acceptable in the text field) ● User selects the “add item” button when they are done entering item name and category.
Expected result	4. Submit button will be greyed out until the following name requirements met: ○ Name is between 1 and 100 characters ○ Name contains at least 1 character that is not a whitespace (space, tab, crlf, etc.) 5. If constraints are met with item name and item category: ○ Item should be added to the user’s grocery list in the database. ○ User UI should be updated with the new item in the appropriate category ○ Other Users who share the grocery list with this user should be able to see the new list item new item in their UI

Identification and classification	**Test Case 0005 - Assign categories** **Severity: 2 (Important)**
Instructions	● When a user selects an item on the list and presses the “categorize” button ● User is prompted with a radio option to select which category they would like to assign to the item ● User may select a new category and press “save” or “cancel”
Expected result	1.) The item’s current category is pre-selected (if already categorized) 2.) When the user selects a new category and presses save, the item’s category is updated in the database 3.) If the user presses cancel ○ The dialog closes and the item’s category is not updated 4.) After a item has been assigned to a category when the users view their shopping list the item will appear under that category.

Identification and classification	**Test Case 0006 - Create Category Sort Orders** **Severity: 2 (Important)**
Instructions	● User selects “create new category sort order” ● User adds a name to the category sort order ● User set order of all possible categories (drag and drop) ● User submits the category sort order when finished.
Expected result	1.) If the name of new sort order is not unique: ○ User receives an error upon submit explaining that two sort order cannot have the same name. ○ User must change the name before he/she can submit again. 2.) If the name of new sort order is unique: ○ User is redirected to the list of category sort orders ○ Database will be updated with new sort order for user ○ User will be able to see new sort order in the list and will be able to add it to a shopping list

Identification and classification	**Test Case 0007 - Assign Sort Order to Grocery List** **Severity: 3 (Work-around)**
Instructions	● User selects list settings for a list ● User selects which category sort order they would like to apply to the list
Expected result	1.) If a malicious user attempts to assign a sort order to list that user does not own: ○ Backend server will respond with a “Permission Denied” error and no changes will be made 2.) If a malicious user attempts to assign a sort order to a shopping list that does not exists: ○ Backend server will respond with an error message stating “Permission Denied” 3.) If assign request is successful: ○ Backend is updated to associate the sort order with the chosen shopping list (many to one relationship)

	○ Shopping list is displayed to the user in the order of categories that the chosen sort order specifies

Identification and classification	**Test Case 0008 - Auto Categorization of Items** **Severity: 2 (Important)**
Instructions	● User adds an item to the list by choosing an autocomplete suggestion
Expected result	1.) Basic Functionality - built-in autocomplete ○ The item is added with a pre-assigned category 2.) If autocomplete item is user-generated ○ Item category is the category most recently assigned to the item by the user 3.) A Changed Item ○ If the user changes the category for an item, that item’s autocomplete entry is updated with the updated category

Identification and classification	**Test Case 0009 - Sharing list** **Severity: 2 (Important)**
Instructions	● User presses list settings button ● User chooses “add user” option ● User enters the email address of the person they would like to share with and presses “share”
Expected result	1.) Basic functionality ○ The list is shared between the two users ○ Both users can access the shared list 2.) If email doesn’t exist ○ User gets an error message saying: “User email does not exist” 3.) Shared with non-email address ○ User gets an error message saying: “Invalid email address”

Identification and	**Test Case 0010 - Browse and Search Recipes**

classification	**Severity: 2 (Important)**
Instructions	● User navigates to the searching recipe page ● User types in the recipe they wish to search for ● User presses the ‘search’ button to send their search
Expected result	1.) Basic Functionality ○ The user can look through existing recipes. ○ The user can search for a specific existing recipe. 2.) Searching a non existent recipe ○ If the recipe doesn’t exist then there should be a display for no recipes matching the search. 3.) Searching a recipe with a space ○ Having a space in search should not create any errors in searching

Identification and classification	**Test Case 0011 - Adding ingredients to a list from a recipe** **Severity: 2 (Important)**
Instructions	● From recipe details view, press add to list button ● Select which list ot add the ingredients to
Expected result	1.) Basic functionality ○ Ingredients from the selected recipe are successfully added to the selected list 2.) Adding recipe to a list the user does not own ○ Results in a permission failure error ○ UI design makes this impossible to attempt from within the app 3.) Adding recipe to a list that does not exist ○ User gets an error message saying, “This list does not exist” ○ (This would not be displayed as an option in a correctly-functioning app)

Identification and classification	**Test Case 0012 - Creating user recipes** **Severity: 2 (Important)**
Instructions	● From “My Recipe’s” view, press add button ● Enter recipe title and ingredients ● Press save button
Expected result	1.) Basic Functionality (saved in user recipes) ○ Recipe is added to the “My Recipe’s” view with

	correct title and ingredients

Identification and classification	**Test Case 0013 - Deleting shopping lists** **Severity: 1 (Critical)**
Instructions	● From a shopping list, press the list settings button ● Press the delete list button ● Press the “DELETE” button
Expected result	1.) Basic Functionality ○ Shopping list is removed as well as all of its placed categories, ingredients, and recipes. 2.) Delete list a user does not own ○ User gets an error message saying, “You do not own this list” 3.) Delete a list that does not exist ○ User gets an error message saying, “This list does not exist”

Identification and classification	**Test Case 0014 - Checking off items from a list** **Severity: 1 (Critical)**
Instructions	● From the list view, swipe item to the side to check off
Expected result	1.) Basic Functionality ○ The swiped item is checked off from the list. Recipes view is updated accordingly. 2.) Undo Functionality ○ After the item is swiped off the list, a snackbar with an “UNDO” button appears, indicating that the item was checked off the list ○ If the user presses the “UNDO” button, the snackbar disappears and the item is added back to the list.

Identification and classification	**Test Case 0015 - Delete items from a list** **Severity: 1 (Critical)**
Instructions	● From the list view, tap on item ● Press delete button on toolbar

Expected result	1.) Basic Functionality ○ Item is deleted from the list, recipe progress is not updated 2.) Delete Items from a list a user does not own ○ User gets an error message saying, “You do not own this list” ○ Should never be possible to do in-app 3.) Delete Items from a list that does not exist ○ User gets an error message saying, “This list does not exist”

Identification and classification	**Test Case 0016 - Creating custom categories** **Severity: 3 (Work-around)**
Instructions	● User will select a option from the settings menu to add categories. ● The user will then be able to enter the name of a category. ● The app will verify that it is not a pre existing category. ● If the category is added then it will appear in the list of categories in the category section of the app settings ● If the category was not added then the user will be notified and returned to the add category page.
Expected result	1.) The user enters a string between 1 and 100 characters and a value that doesn’t already exist for the category name and then they select the add button. They are then notified that their item was added to the list. 2.) If the category name exceeds 100 characters then either the user will be notified when they select the add category button, and the value will not be added, or the add category button will be disabled. 3.) The add category button will be disabled until the user has entered in at least one character. 4.) If the category already exists then the user will be notified of such and be allowed to enter a new category name.

Identification and classification	**Test Case 0017 - Autocompleting suggestions to add items to lists** **Severity: 3 (Work-around)**
Instructions	● The user selects a list to add to or creates one. ● The user selects to add a new item to that list.

	● When the user begins entering a item name the box will attempt to auto populate the potential item the user is trying to add.
Expected result	1.) Basic Functionality: ○ When the user selects to add a new item to a list they will be prompted with a box that will allow them to enter in the item name. ○ Upon entering part of the name (potentially 3-4 characters) the box will begin auto populating the rest of the item name. ○ If the user enters a character that doesn’t match the characters in the auto populate name the name will be populated with a new item that could potentially match the users entry. 2.) If the user enters an item that doesn’t exist in the auto population bank then they will be able to add the item to the list anyway.

Identification and classification	**Test Case 0018 - Seeing when an item was last purchased** **Severity: 3 (Work-around)**
Instructions	● When the user is viewing a list they can check off items as they buy them. ● Select the purchased button on any list to view items that have been purchased and when ● Click the button again to indicate that the item was put back or that more is needed.
Expected result	1.) Basic Functionality: ○ User selects a list they want to view. ○ User chooses “view completed” option from menu ○ The list of purchased items is displayed with their corresponding purchase dates 2.) If the user leaves the list and returns the items will continue to be indicated as purchased or not when they return. 3.) Other users who are shared on this list will see the changes that have been made by all users.

Identification and classification	**Test Case 0019 - Revoking sharing rights** **Severity: 1 (Critical)**

Instructions	● For any given list the user will be able to view all of the users who have been given access to that list. ● There will be a remove button that will allow the user to remove other users from the viewing functionality of the list. ● The user will select the remove button to remove any given user from the shared list.
Expected result	1.) Basic Functionality ○ The user will select a person from the list of shared people on the shopping list and click a remove button ○ That user will be removed from the list view and will no longer be able to view that list. 2.) The user will be told that they cannot be removed from the list if they attempt to do so. 3.) The user will be unable to remove a person from a given shopping list if they are not the creator of the list. This will either involve hiding the button options or populating a message if the user attempts to do this. 4.) If the user being removed from the sharing rights of the list is on the list when they are removed then they will still continue to be able to view and access the list until they return to a previous page and attempt to re enter a list.

Identification and classification	**Test Case 0020 - Saving recipes to personal collection** **Severity: 1 (Critical)**
Instructions	● The user selects view recipes ● They navigate to any recipe via previously mentioned functionality. ● On any of the recipes they will see a button that provides them the option to save their recipe.
Expected result	1.) Basic functionality: ○ When the user is on the view for a recipe they will have the option to save that recipe to their personal collection via a save button on the recipe view. ○ When they select the save button the button will reflect that that recipe has been saved. 2.) When the user selects a recipe that they have previously already saved to their personal collection, the option to save that recipe will indicate that it has already been saved and the user will not have the option to click on it.

Identification and classification	**Test Case 0021 - Viewing saved recipes** **Severity: 2 (Important)**
Instructions	● User navigates to the saved recipe page ● If nothing is there, go to the recipes and save a recipe ● Check the saved recipes again ● If there is a recipe there go and delete all saved recipes ● Make sure that there are no saved recipes in the page
Expected result	1.) Basic Functionality ○ The user should have a way to view all of the recipes that they saved or ‘favorited’ 2.) No saved recipes ○ If the user has no saved recipes, then there should be no saved recipes on the saved recipes page

Identification and classification	**Test Case 0022 - Seeing which recipe a list item was added for** **Severity: 3 (Work-around)**
Instructions	● Add ingredients from no recipe ● Check if those show no recipe ● Add a recipe and thus adding all the ingredients to the list ● Check that the ingredients all show the recipe ● Add another recipe that has some overlapping ingredients ● Check that the overlapping ingredients show bothe recipes ● Check that the non overlapping ingredients show the correct recipe or lack of
Expected result	1.) Basic functionality ○ Each item in a list should show which recipe added it to the list (if applicable) 2.) Same Ingredients from different recipes ○ If an ingredient was added for multiple recipes it should show all the recipes that it is from 3.) Ingredients from no recipe ○ An ingredient from no recipe should not show any recipes

Identification and classification	**Test Case 0023 - Seeing the progress of each recipe as items are checked off** **Severity: 3 (Work-around)**
Instructions	● User navigates to the recipe page ● User adds the recipe to their shopping list ● User checks off multiple items from the recipe as ‘in cart’ ● User checks that the recipe progress shows these items as checked off ● User checks that items not checked off, do not show in recipe progress
Expected result	1.) Basic Functionality ○ Once an item is checked off, it should show in the recipe that, that item has been checked off 2.) Items that aren’t checked off ○ Items that aren’t checked off of the list, but are still from the recipe shouldn’t show progress towards the recipe

Identification and classification	**Test Case 0024 - Receiving a notification whenever a shared list is changed** **Severity: 2 (important)**
Instructions	● The user updates one of their own lists ● The user checks for no notification ● The user joins a shared list ● A different user updates the shared list ● The original user checks for a notification ● The original user leaves the list ● A different user on the previously shared list modifies the list ● The original user checks for a notification (shouldn’t get one)
Expected result	1.) Basic functionality ○ The user should receive a notification when a shared list is changed 2.) Should not receive notifications for no reason ○ The user should not receive a notification if any list is changed (i.e. their own private list)

Identification and classification	**Test Case 0025 - No duplicated ingredients added from recipes** **Severity: 2 (Important)**
Instructions	● The user should add an item to the list ● The user should attempt to add that item again and see that it doesn’t add twice ● The user should add a recipe with that item ● The user should check that the item wasn’t added twice ● The user should add another recipe that has an overlapping item ● The user should check that, the item wasn’t added twice
Expected result	1.) Add multiple different recipes with the same ingredient (list item) ○ When an item appears in multiple recipes the amounts should be added together and not duplicated 2.) Adding multiple identical list items with no recipe correlation ○ The list should add the quantities together and not display duplicate items 3.) Adding a list item that is the same as one or more recipe ingredient. ○ Adding an item that is in an existing recipe it should add up the quantities and not duplicate the item in the list