

# System Design

Team: Lions

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>CRC Cards - Sprint 1</b>	<b>2</b>
<b>System Interaction with Environment</b>	<b>6</b>
<b>Architecture of the System</b>	<b>7</b>

# CRC Cards - Sprint 1

Class name:	<b>CreateRecipeActivity</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- Acts as the View for the activity that handles creating a recipe</li> <li>- Detects user interaction with screen and passes data to the Presenter called CreateRecipePresent</li> </ul>
Collaborators:	<ul style="list-style-type: none"> <li>- CreateRecipeContract</li> <li>- CreateRecipePresent</li> </ul>

Class name:	<b>CreateRecipeContract (Interface)</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- Coordinates the CreateRecipeActivity class and CreateRecipePresent class</li> </ul>
Collaborators:	<ul style="list-style-type: none"> <li>- CreateRecipeActivity</li> <li>- CreateRecipePresent</li> </ul>

Class name:	<b>CreateRecipePresent</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- Acts as the presenter for the activity that handles creating a recipe</li> <li>- Takes in data from the associated view class and deals with the data appropriately.</li> </ul>
Collaborators:	<ul style="list-style-type: none"> <li>- CreateRecipeActivity</li> <li>- CreateRecipeContract</li> </ul>

Class name:	<b>CookBookContract (Interface)</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- Coordinates the CookBookActivity class and CookBookPresent class</li> </ul>
Collaborators:	<ul style="list-style-type: none"> <li>- CookBookActivity</li> </ul>

	- CookBookPresent
--	-------------------

Class name:	<b>CookBookActivity</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- Acts as the view for the activity that handles viewing the user's cookbook</li> <li>- Detects if a user clicks on a recipe. If so, the recipe title gets passed on to the associated presenter for this activity</li> </ul>
Collaborators:	<ul style="list-style-type: none"> <li>- CookBookContract</li> <li>- CookBookPresent</li> </ul>

Class name:	<b>CookBookPresent</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- Acts as the presenter for the activity that handles viewing the user's cookbook.</li> <li>- Takes in recipe title clicked from the View, fetches the associated data from the database, and ensures the user is brought to a screen that displays the recipe that was clicked.</li> </ul>
Collaborators:	<ul style="list-style-type: none"> <li>- CookBookActivity</li> <li>- CookBookContract</li> </ul>

Class name:	<b>ViewRecipeActivity</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- Displays the recipe the user has clicked on the CookBook screen.</li> <li>- This should populate the recipe title, ingredients, steps, and serving size.</li> </ul>
Collaborators:	<ul style="list-style-type: none"> <li>- CookBookActivity</li> <li>- CookBookPresenter</li> </ul>

Class name:	<b>Ingredient</b>
Parent class (if any):	
Classname Subclasses (if any):	

Responsibilities:	<ul style="list-style-type: none"> <li>- Stores information of a single ingredient in a recipe.</li> <li>- Takes in the ingredient name, quantity, and quantity_type, and able to modify and update these values</li> </ul>
Collaborators:	

Class name:	<b>Recipe</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- Stores information of a single recipe.</li> <li>- Takes in the recipe name, username who created the recipe, serving_size, list of ingredients, list of tags and list of steps, and able to modify and update these values</li> </ul>
Collaborators:	

Class name:	<b>RecipeDao (<i>Interface</i>)</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- contains function declarations for inserting a new recipe and getting all recipes stored in the database</li> </ul>
Collaborators:	

Class name:	<b>RecipeDatabase(<i>abstract</i>)</b>
Parent class (if any):	<ul style="list-style-type: none"> <li>- RoomDatabase</li> </ul>
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- Instantiate a database object</li> </ul>
Collaborators:	<ul style="list-style-type: none"> <li>- RecipeDao</li> <li>- RecipeListConverter</li> </ul>

Class name:	<b>RecipeListConverter</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- a utility class that is responsible for converting types between strings and string lists for the ingredients</li> </ul>
Collaborators:	

Class name:	<b>RecipeModel</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- able to return all recipes stored in the database through RecipeRepository</li> <li>- able to insert a new recipe into the database through RecipeRepository</li> </ul>
Collaborators:	<ul style="list-style-type: none"> <li>- RecipeRepository</li> <li>- Recipe</li> </ul>

Class name:	<b>RecipeRepository</b>
Parent class (if any):	
Classname Subclasses (if any):	
Responsibilities:	<ul style="list-style-type: none"> <li>- able to get all recipes from the database</li> <li>- able to insert a new recipe into the database</li> </ul>
Collaborators:	<ul style="list-style-type: none"> <li>- RecipeDao</li> <li>- RecipeDatabase</li> </ul>

## System Interaction with Environment

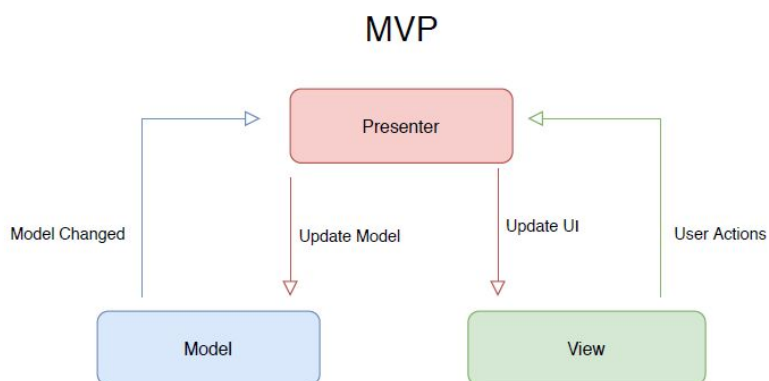
OS	Android Marshmallow 6.0.1 and greater
Programming Language	Java
Emulators	Emulator OS: Marshmallow 6.0.1 Emulator Device: Pixel 2
Database	Firebase/ Room DB (local)
Dependencies	roomDB, data binding, MaterialUI style themes, Firebase
Assumptions	<ul style="list-style-type: none"><li>• The device the app runs on has an OS API level of 16 or greater.</li><li>• The device must have enough storage to save the data.</li></ul>

# Architecture of the System

## Architecture high-level explanation:

The application follows the Model-View-Presenter(MVP) design pattern, which can be seen to be Android's adaptation of the Model-View-Controller design pattern. A diagram of the architecture is included below. In this architecture, the Presenter is responsible for interpreting any interaction the user has with the view and working with the Model accordingly. Note that the model does not communicate with the View. Our team decided to use this architecture as it is very commonly used among Android developers for its ease of testing capabilities.

## Image of the MVP design pattern:



## Links that describes the MVP:

- <https://android.jlelse.eu/architectural-guidelines-to-follow-for-mvp-pattern-in-android-2374848a0157>
- <https://www.raywenderlich.com/7026-getting-started-with-mvp-model-view-presenter-on-android>
- <http://www.gwtproject.org/articles/mvp-architecture.html>
- <https://www.martinfowler.com/eaDev/uiArchs.html#Model-view-presentermvp>

## MVP Components:

- CreateRecipePresent.java, CookBookPresent.java – the presenter
- ViewRecipeActivity.java, CookbookActivity.java, CreateRecipeActivity.java – the view
- CreateRecipeContract.java, CookBookContract – defines contract between view and presenter
- CookBookModel.java, CreateRecipeModel.java – the model



**Room Components (Part of Model):**

- Recipe.java – this is the entity which represents a relational table within the database
- RecipeDatabase.java – a Room Database
- RecipeDao.java - Contains the methods to access the database
- RecipeRepository.java - is a class for the repository in the viewModel + repository framework for roomDB

**Anticipated Errors and Exceptions:**

The user is required to fill out all fields when creating a recipe. If they fail to do this a toast notifies them of this.

In the upcoming sprints, we also plan to handle cases where the user inputs the incorrect data type. For example, if the user inputs a String into the textfield intended for a quantity of an ingredient, a toast will inform them that their input is invalid and they must enter a number into that field.