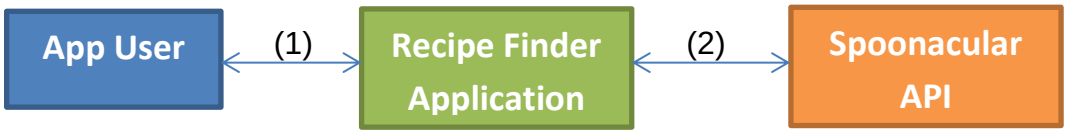
Technical Documentation

App Name: Aruna's Recipe Finder

High Level Design



(1) User interacts with application to find recipes online
(2) Recipe Finder application uses Spoonacular API to search the spoonacular's vast recipe collection and to find information of different ingredients required for the recipes

Code organization and structure

Recipe Finder is a simple console app. It has the following scripts:

startapp.py

- Is the entry point for the application
- Has code that receives user input, interacts with user
- Passes user input to recipefinder.py
- Displays results to user

recipefinder.py

- SpoonacularRecipeFinder class performs most of the required operations
- This class initializes configuration/constants from constant.py
- It passes user input to SpoonacularApi class to obtain recipes and ingredient information
- It maintains a buffer of result *'recipes'* and the shopping list *'shopping_list'*
- The result recipes are looped over to allow user to choose the recipes they like
- Missing items from liked recipes are added to the shopping cart
- Currently a set number recipes are maintained in the buffer, changeable by updating 'MAX_NUMBER_OF_RECIPES' in constant.py
  - Future extensions are discussed later
- Number if ingredients in the query is limited to 'MAX_NUMBER_OF_INGREDIENTS' defined in constant.py

spoonacularclient.py

- Is responsible for communicating with Spoonacular API

- Uses requests http library to talk to Spoonacular API
- Calls ‘recipes/findByIngredients’ and 'food/ingredients/{ingredient_id}/information' APIs
- ‘recipes/findByIngredients’ is called with ‘ignorePantry’ set to False

constant.py

- Defines constants and application-related configuration
- It is good to have these definitions in one file

Security (API key management)

Spoonacular API key is read by the application from the system environment variable SPOONACULAR_API_KEY
This is easier and more secure than having the API key hard-coded in the app code
The end-user has to use their own API key instead of the developer’s.
For a production solution, the API keys are better managed on AWS secret manager

Error handling

Exceptions are raised and caught as necessary in all scripts
Debug statements provide information about the errors

Testability

SpoonacularRecipeFinder class takes an instance of SpoonacularApi
This enables easy testing because SpoonacularApi can be mocked for unit-testing
The code is modular with each script, class, and function defined with clear separation in their functionality.
For ex:, SpooncularApi class is wholly responsible for communicating with Spoonacular

Improvements

Things I would do with more time:

1. Provide ways to delete the shopping list
2. Instead of restricting the recipes buffer to be ‘MAX_NUMBER_OF_RECIPES’, I would use explore fetching recipes using pagination
3. Stricter validation of input. Only allow valid ingredients (using parseIngredients Spoonacular API), remove duplicate ingredients, etc
4. Converting json recipe list and ingredient dictionary returned by spoonacular APIs to python objects for easier data access
5. Add unit tests