Edesia

Meal planning made easy

Team Blue: Adam Bowers, Daniel Wingo, Stephen Bennett, and Jonathan Holley

**Table of Contents**

[**1. Project Definition:**](#_nm5x1x5enh62) **3**

[**2. Project Requirements:**](#_cmahib1ga0ky) **4**

[**3. Project Specification:**](#_pewapb4be0c7) **5**

[**4. System – Design Perspective:**](#_3lukddk1jowh) **5**

[**5. System – Analysis Perspective**](#_yvkz0nyckb8w) **17**

[**6. Project Scrum Report:**](#_u4l68b3m1wkh) **19**

[**7. Subsystems:**](#_r9jktq7kkmzg) **19**

[7.1 Subsystem 1 – Name 1 - Individual responsibility](#_ti0q2jevi8h) 19

[7.2 Subsystem 2 – Name 2 - Individual responsibility](#_9ktdp6logvas) 19

[7.3 Subsystem 3 – Name 3 - Individual responsibility](#_ixgslf9ibwc9) 20

[7.4 Subsystem 4 – Name 4 - Individual responsibility](#_urtcy0glp1qd) 20

[**8. Complete System – Group responsibility**](#_g5q1b03jeske) **21**

[**9. Sources: (Adam Bowers)**](#_vp23ucdprhgv) **21**

# 1. Project Definition:

* American’s eat out 4 to 5 times a week on average. Many people choose to eat out for the sake of convenience. It saves you time and a trip to the grocery store. Eating out, however, can be extremely expensive. The average American spends $232 eating out per month [*The Simple Dollar*]. The average cost of eating out is $12.75 per meal, while the average cost of cooking at home is $6.41 per meal [*Cheapism*]. Eating out is also much worse for your health. You take in an average of 200 more calories eating out than if you eat at home [*The Independent*].
* Planning meals can be an arduous and daunting task. Edesia aims to change that. Our goal is to make a user-friendly and intuitive meal planning application for the Android mobile platform. We hope to encourage people to eat healthier, save money, and improve their cooking ability.
* Edesia will allow the user to drag-and-drop different recipes into a day of the week. Users can search for recipes to add to their personal menu. If a user is unsure on what meal to cook, they can press a shuffle button and the application will select a variety of meals for that user.

# 2. Project Requirements:

* Functional:
  + Drag-and-drop user interface
  + Interactive calendar
  + Database of recipes
  + Users can save recipes
  + Users can upload recipes
  + Filter recipe search
  + Automated grocery list based on choices by user
* Potential Features:
  + Voting System for Meals for families
  + Taking pictures of food items using the Google Vision API
  + Gives you ideas based on the ingredients you already have
* Usability:
  + User interface:
    - Android Application, using a graphical interface with search, drag and drop features and calendar
  + Performance:
    - Needs to be able to support many users at one time
    - Should be able to give live feedback when votes are cast for meals for multiple users
    - Should be able to update recipe list based on the ingredients given quickly
    - Should be quick and clean with drop and drag features
* System:
  + Hardware: Mobile devices (Android)
  + Software: Java using Android Studio
  + Database: SQLite Database system seems to be one of the best options for Android.
* Security:
  + Hash users Sensitive data
    - Passwords, user’s secret recipes, and anything else that may want to be kept from others

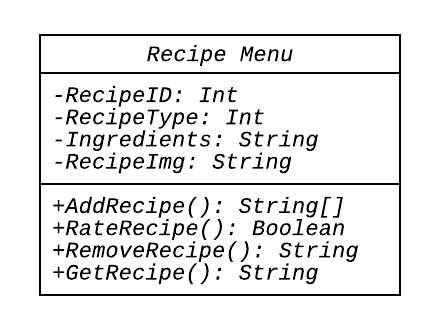
# 3. Project Specification:

* + Focus / Domain / Area:
    - Foods!
    - To help people organize their meals
    - People who can't figure out what to cook
* Libraries / Frameworks / Development Environment
  + Active android
    - Helps with communication between app and SQLite
  + Android Virtual Device (AVD)
    - Emulates Android OS within Android Studio
  + Web Scraping using Beautiful Soup, Selenium and Scrapy Python libraries
  + SQLite for the database
* Platform: Android Mobile
* Genre: Food/Planning Application

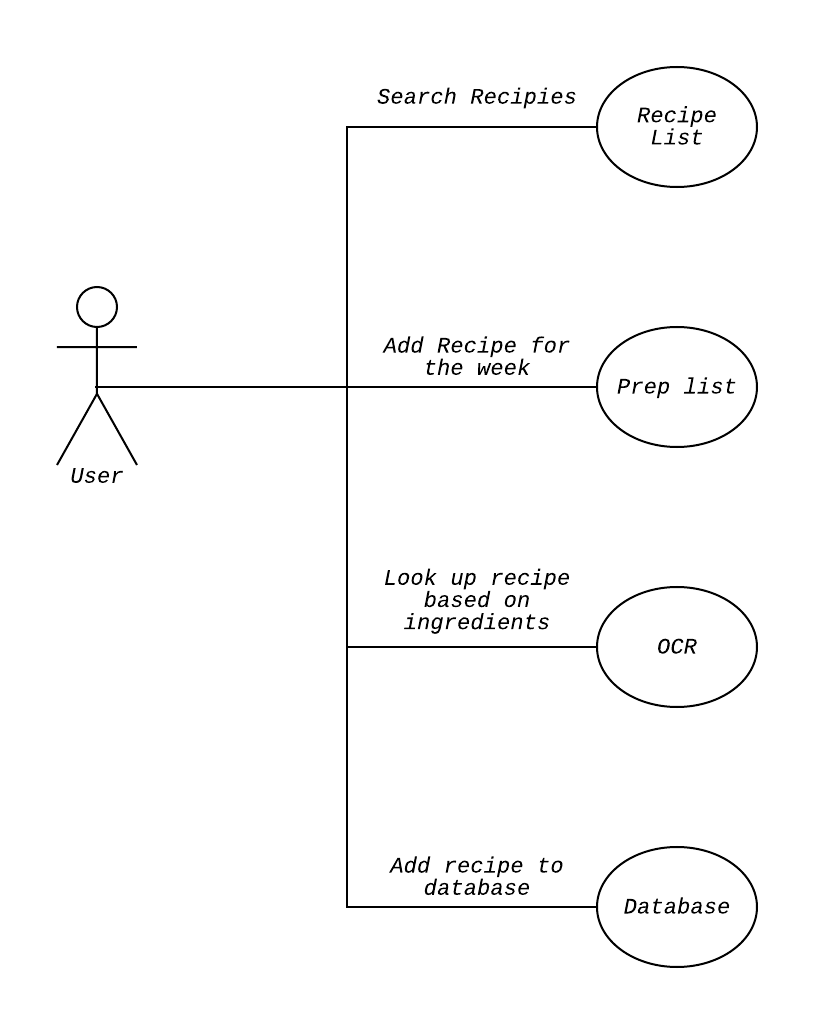
Edesia is a food planning and organization application for people who can’t figure out what it is they want to cook. It will help people organize and plan their meals for the week so that they can eat healthier as well as save money while they are at it. Edesia is a mobile application primarily focused on the Android platform. It will be designed using Android Studio and will use an SQLite database to store user information as well as the list of recipes. The recipes will be added to the Database using Web Scraping which will be accomplished using the Python Selenium and Scrapy libraries. (Jonathan Holley)

# 4. System – Design Perspective:

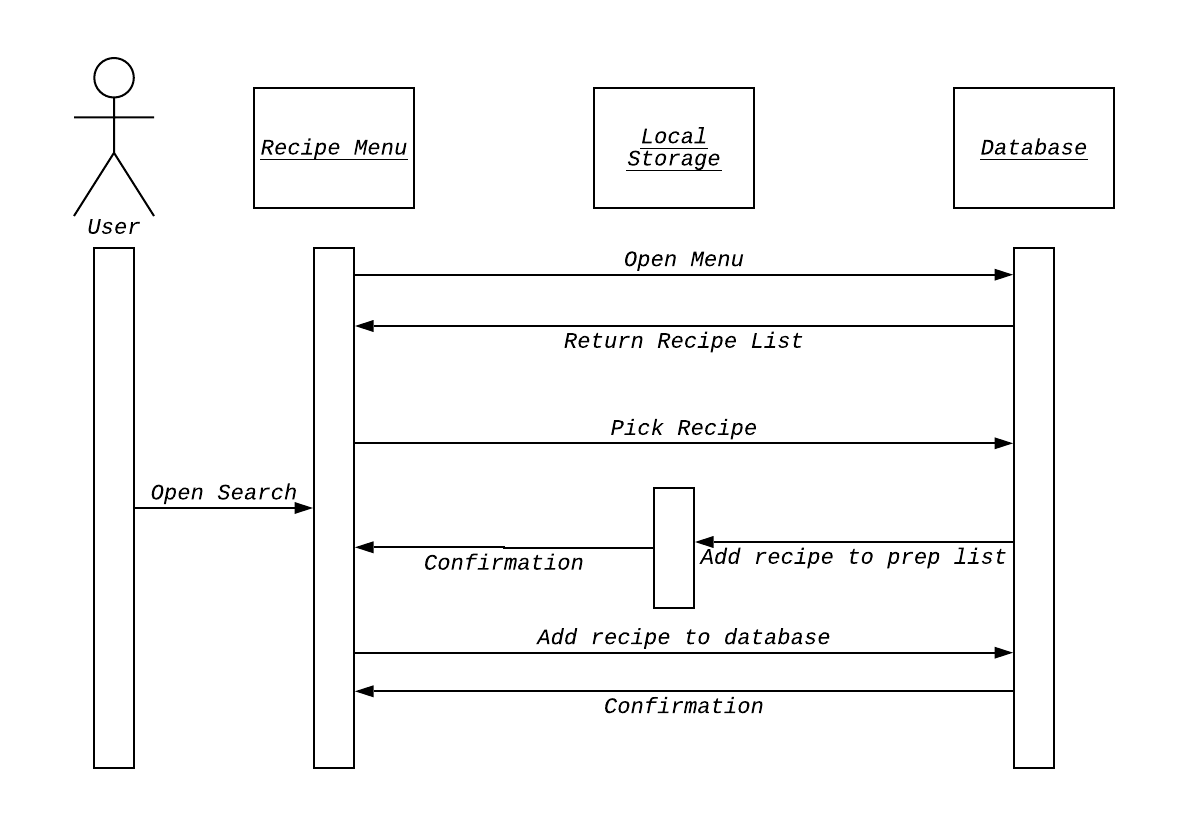
* Identify subsystems – design point of view
  + Recipe Menu - Jonathan Holley
  + Class Diagram



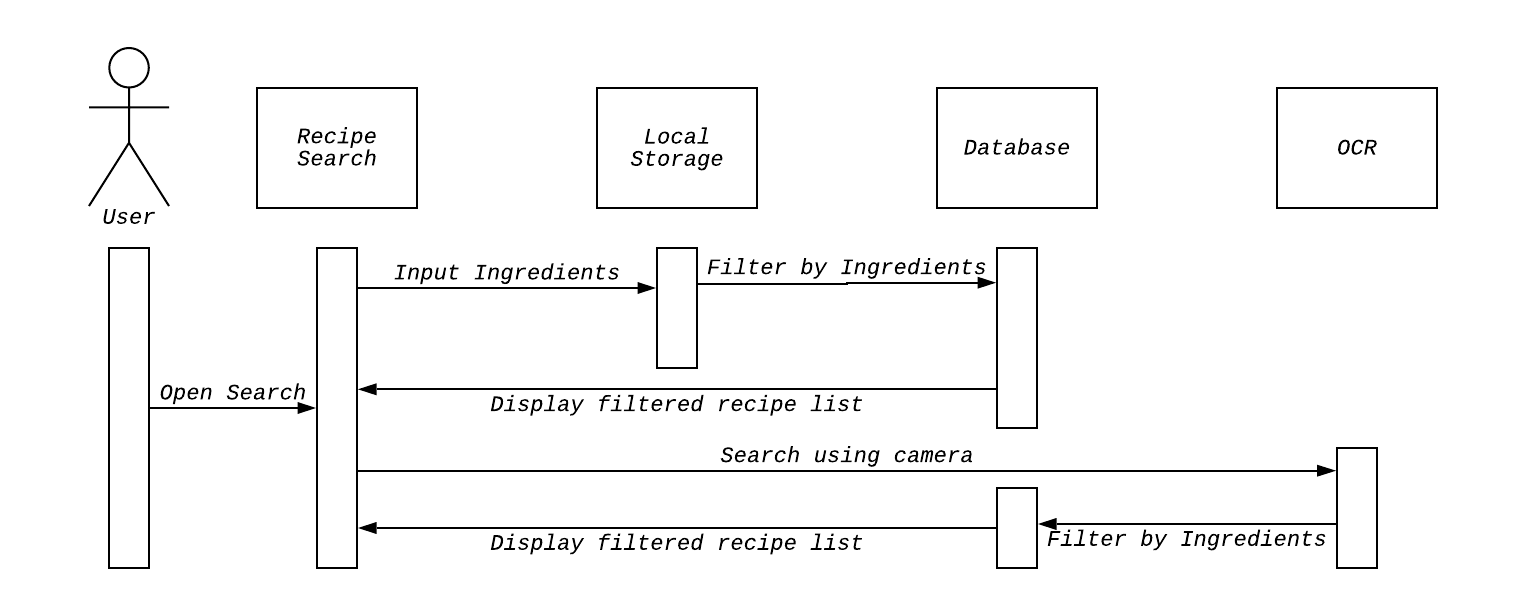
* + Use-case Diagram



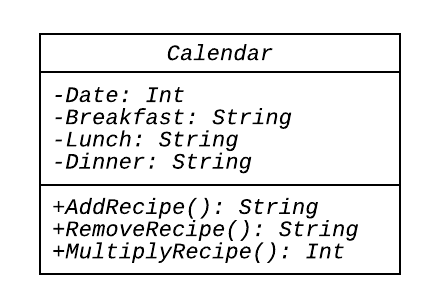
* + Sequence



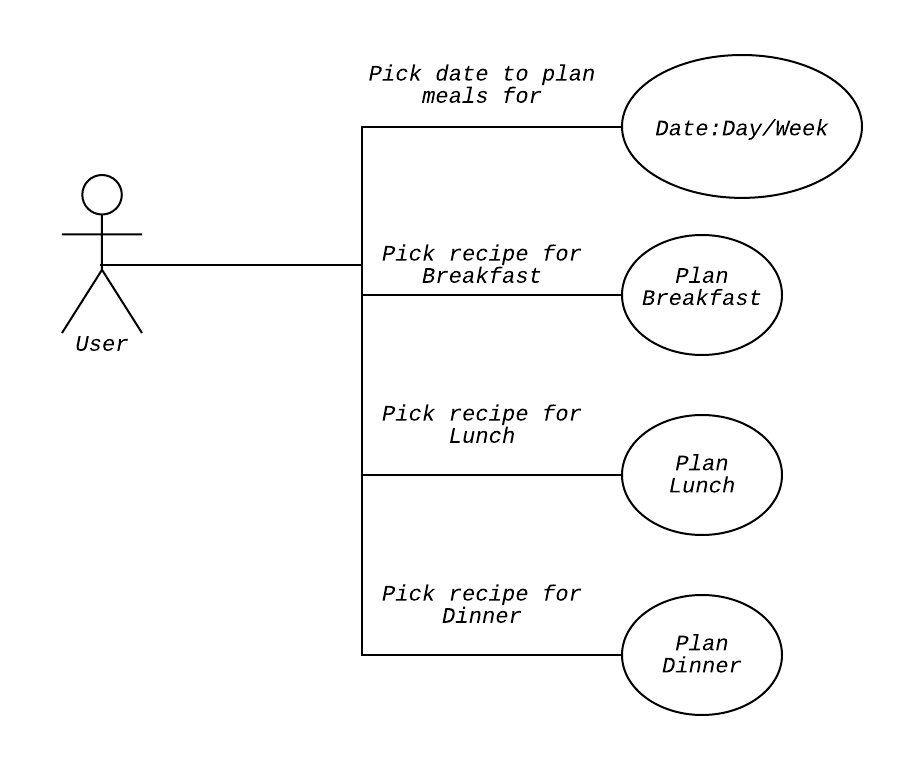
* + OCR - Jonathan Holley
    - Sequence Diagram



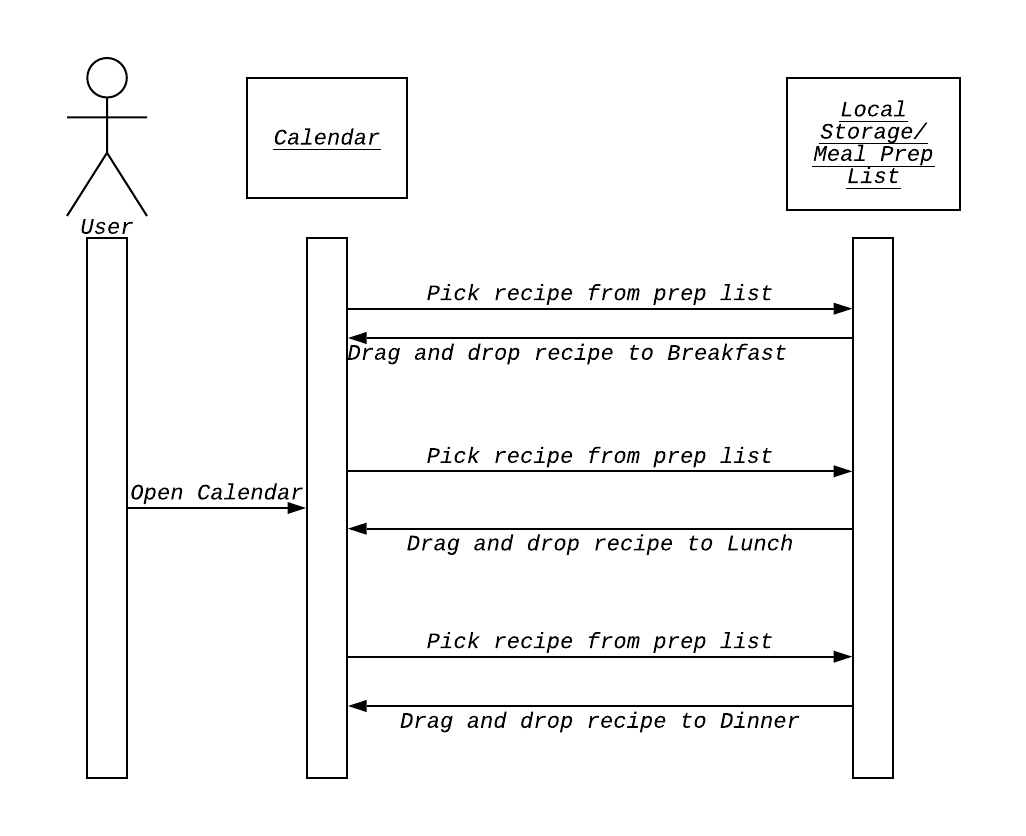
* + Calendar - Jonathan Holley
  + Class Diagram



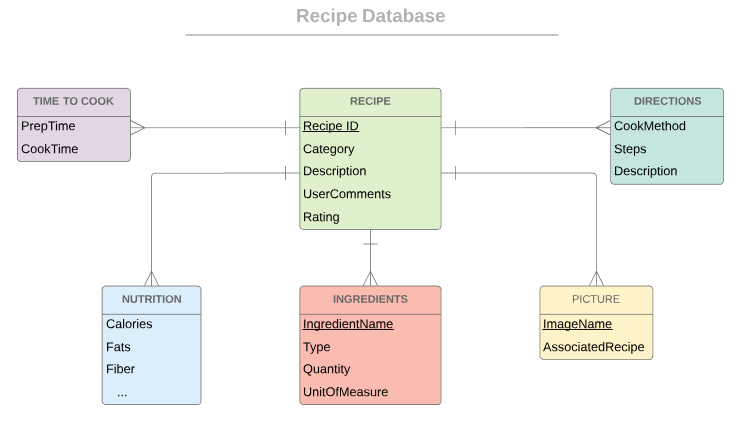
* + Use-case Diagram

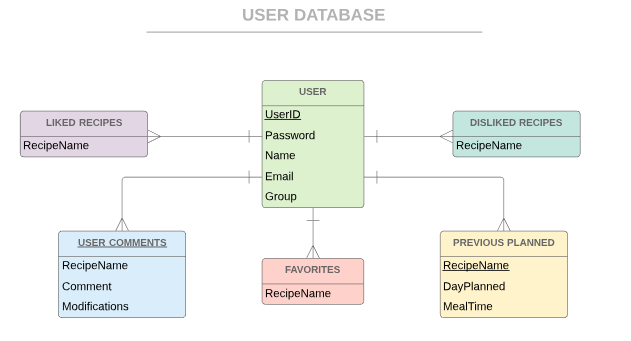


* + Sequence

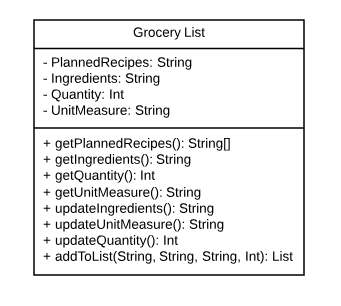


* + Database Entity-Relationship Models (Stephen)

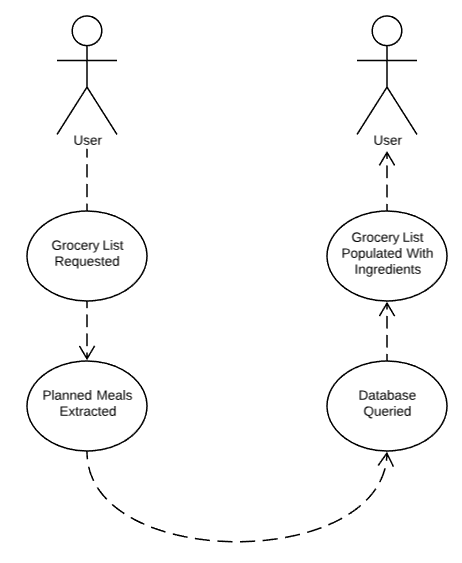




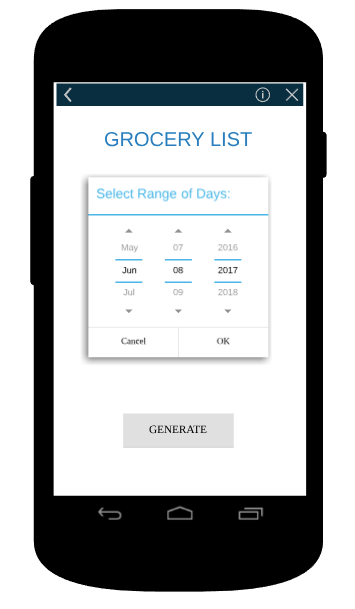
* Our system will run with two primary databases. One will contain all relevant information to recipes, and the other having all relevant information to users. These diagrams display a general layout, but changes may be necessary depending on the exact data extracted via the web scraper subsystem. (Stephen)
  + Grocery List generator
  + Class Diagram (Stephen)



* + Flow Diagram (Stephen)



* + Mockup (Stephen)



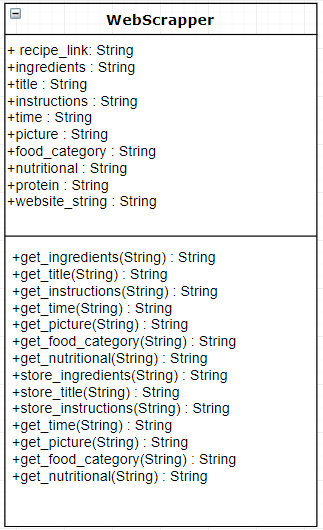
* The figure above is a mockup example of how the user will select a range of days for meals that have been planned. Based on this selection the generator will add all necessary ingredients to cook those meals to the grocery list. (Stephen)
  + Web Scraper: (Daniel)
  + Class Diagram
  + 

Figure WS.1

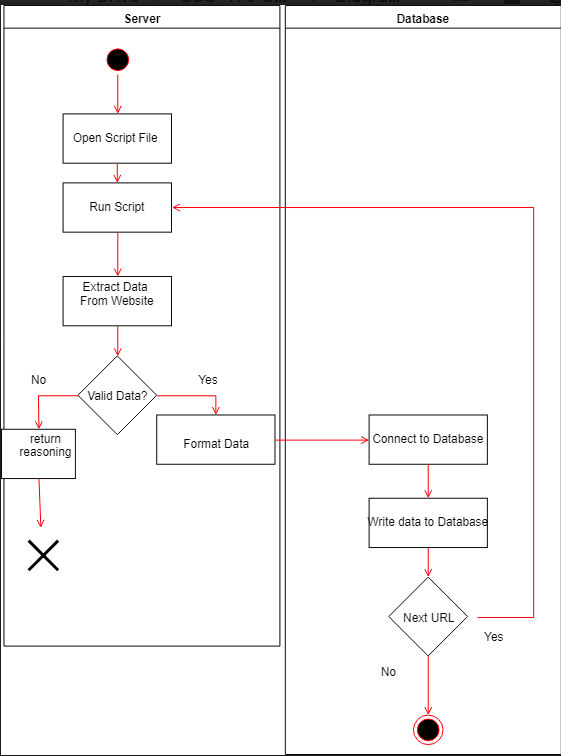
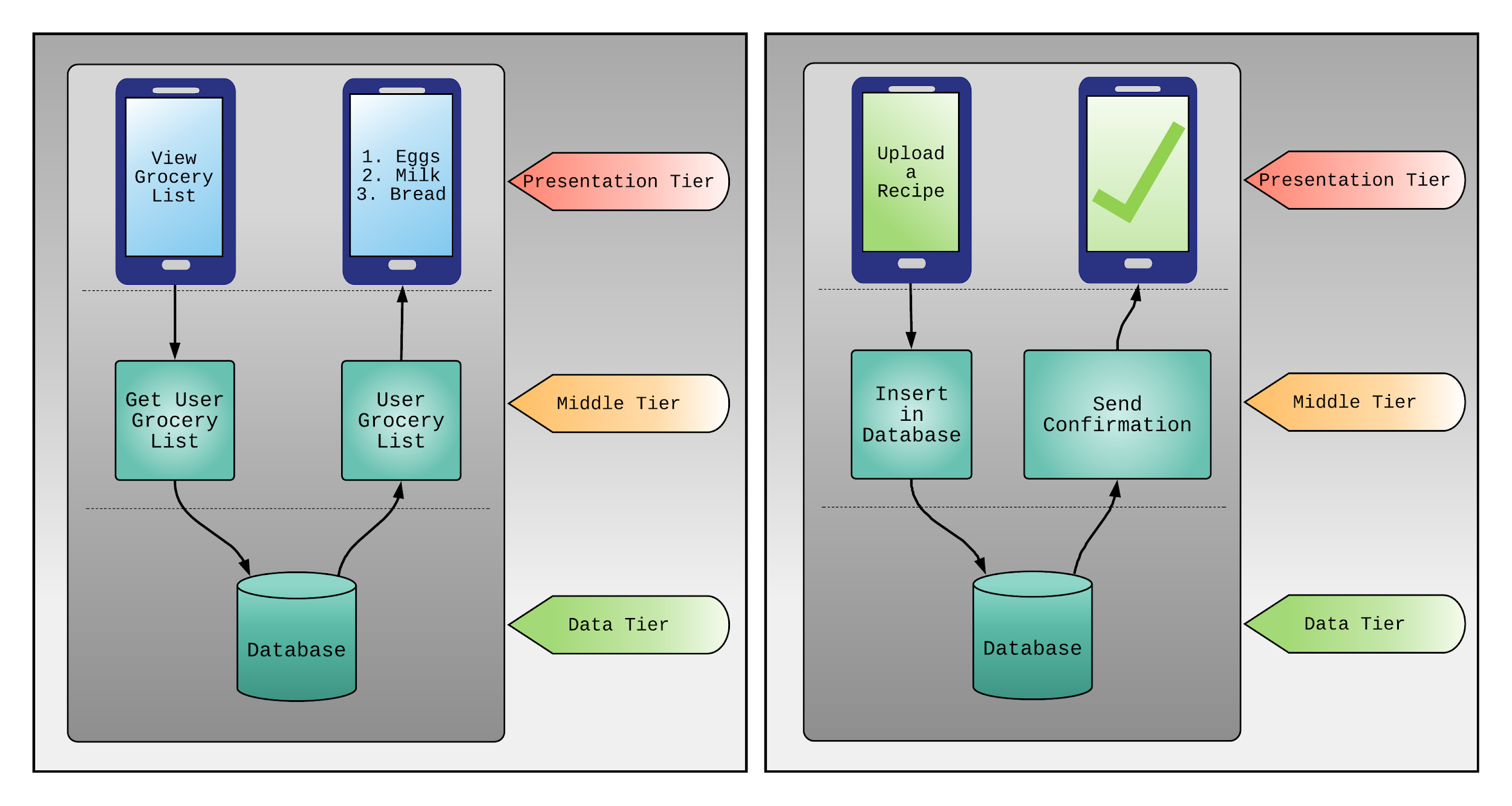
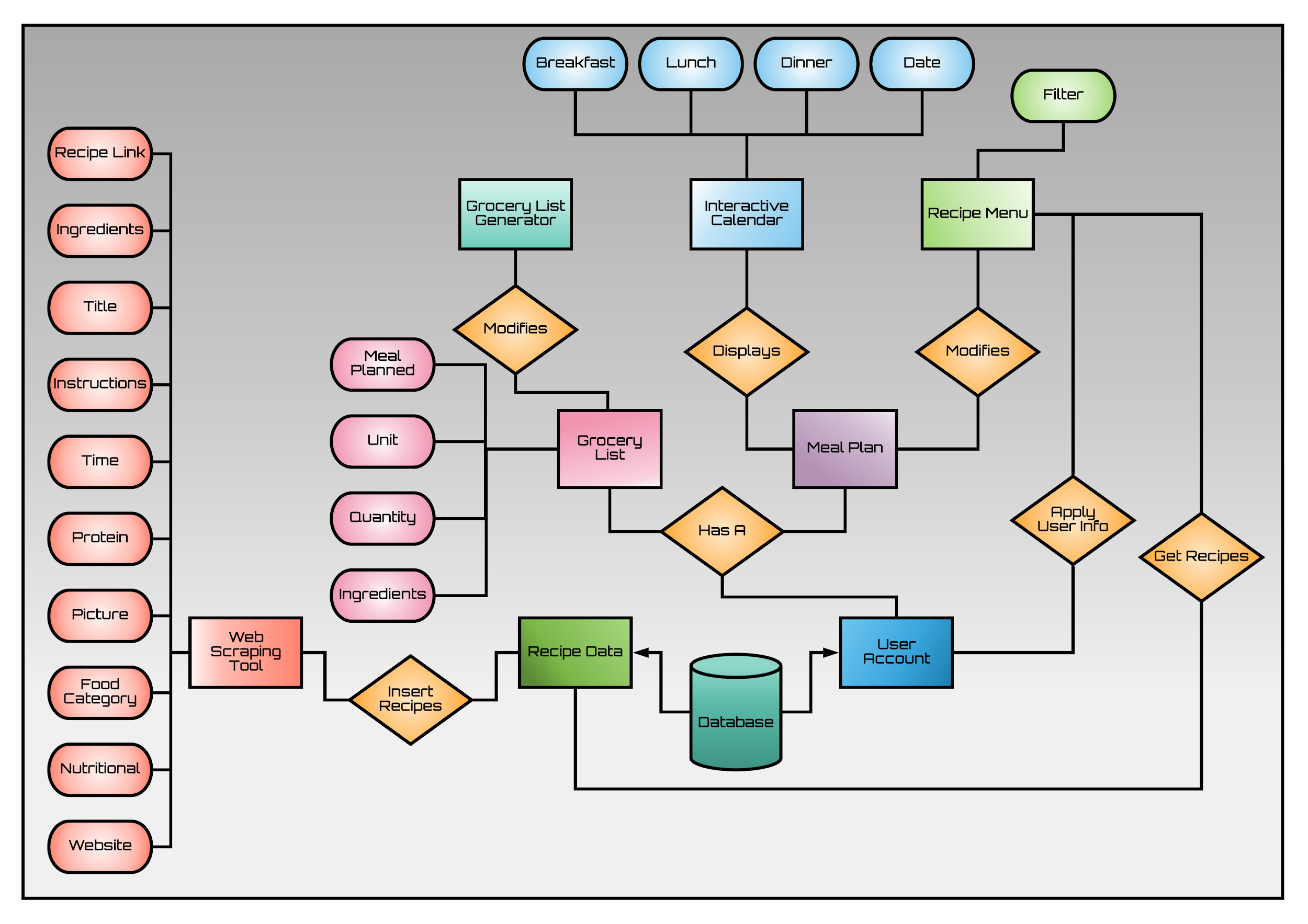
* Activity Diagram

Figure WS.2

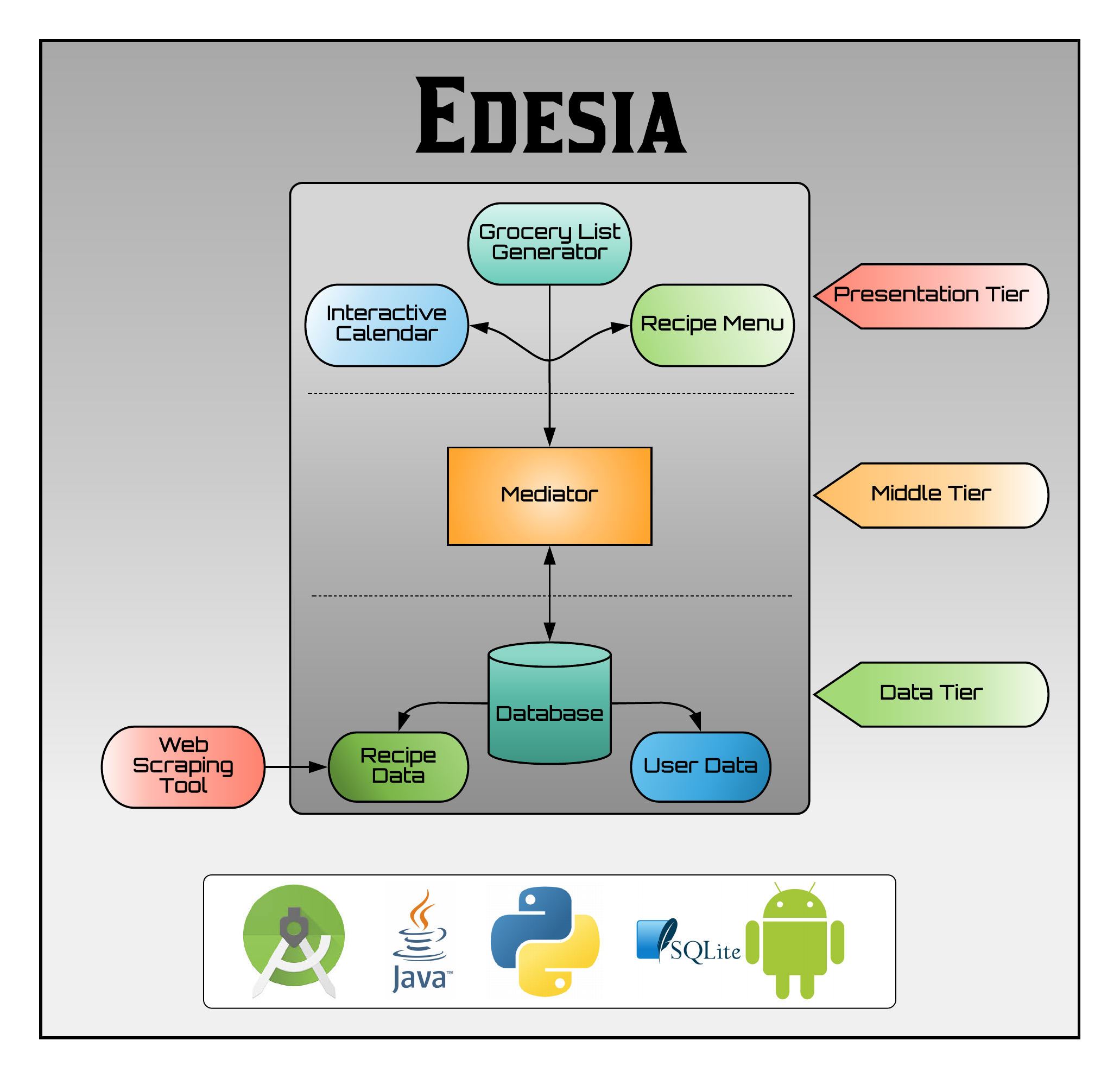
* Subsystem Communication: (Adam Bowers)



* + The image above shows an example of how Edesia’s subsystems interact with one another. In the left image; the user clicks a button to view their grocery list. This control is managed on the presentation tier, which makes a call to the middle tier to get the user’s grocery list. The middle tier acts as a mediator between the subsystems. From here, a query is made to the database on the data tier. The database sends the queried data back up to the middle tier, which will then format the data for the user. Lastly, the grocery list is sent to the presentation tier for the user to view.
  + A similar process happens in the right image. The user inputs a request to upload a recipe to the database. This request is passed to the middle tier, which will then validate the data before sending it to the database. Upon success, the database will send a confirmation to the mediator, which will then send the output to the user.
  + These two images give a simplified idea of how controls, input/output, and dataflow is handled by Edesia’s subsystems.
  + DataFlow
    - Note: For web scraper dataflow, see Figure WS.2 (Daniel Wingo)
* System Entity Relationship Model (Adam Bowers)



* + The above ER diagram shows each subsystem’s relationship between each other, as well as the attributes associated with those subsystems.
    - Note: The attributes for the database have been omitted from this diagram. They are shown in detail in the database ER diagram.
* Overall operation - System Model (Adam Bowers)
  + Simplified Sub-system to System interaction

****

* The above system model shows Edesia’s overall operation, architecture, and subsystem to system interaction. A three-tier architecture design is being implemented. The presentation tier will consist of user interfaces and “listen” for user input. Upon an event, the presentation tier will make the appropriate call to the middle tier. All the meaningful processing and data traffic coordination will occur here. This middle tier serves as a mediator between the presentation tier and the data tier, and on a deeper level it handles all interactions between subsystems. This allows for cleaner code design. All subsystems can be independent of one another. So if a subsystem is added, removed, or modified, other subsystems will not be affected. The final tier is the data tier. Edesia’s database of recipes and user data is stored here. This is also the destination for all data gathered by the web scraping tool subsystem.

# 

# 5. System – Analysis Perspective

* Identify subsystems - (Jonathan Holley)
  + Subsystems
    - Grocery List Generator (Stephen)
    - Interactive Calendar
    - Recipe Menu
    - Database (Stephen)
      * User Data
      * Recipe Data
        + Web Scraper (Daniel)
* System (Tables and Description)
  + Data analysis
    - Data dictionary (Table - Name, Data Type, Description) (Daniel)

|  |  |  |
| --- | --- | --- |
| Columns | Data Type | Description |
| username | VARCHAR | This is the user's login name |
| email | VARCHAR | User's email |
| password | VARCHAR | User's password |
| protein | VARCHAR | Food's Protein Source |
| title | VARCHAR | Title of the recipe |
| recipe\_link | VARCHAR | recipe URL |
| ingredients | LONGTEXT | list of ingredients |
| instructions | LONGTEXT | list of steps to follow for the recipe |
| cook\_time | DECIMAL | estimated time to cook the meal |
| prep\_time | DECIMAL | estimated time to prep meal |
| total\_time | DECIMAL | calculated cook\_time+prep\_time |
| food\_category | VARCHAR | category the recipe belongs to |
| nutritional | MEDIUMTEXT | nutritional information about the meal |
| image | BLOB(binary large object) | Image of the recipe |
| user\_comment | VARCHAR | User's comment on the recipe |
| rating | DECIMAL | rating for a recipe |

* Algorithm Analysis
  + Big - O analysis of overall System and Subsystems
    - For the web scraper we are more worried about accuracy than speed. The store for the data will be O(1). (Daniel)
    - SQLite Database uses a B-Tree so the time complexity is O(log n) for insert, delete and search. (Daniel)
    - The Recipe menu subsystem will have the same worst case time complexity as the database as it will require calls to the database to fill out the menu. (Jonathan Holley)
    - The calendar will be based locally so the time complexity should be low. I couldn’t find something similar but I figure it should be less than O(log n). (Jonathan Holley)
    - Time complexity for the grocery list generator will be O(log n), due to the database queries needed for ingredient lookup. Accessing the planned recipes will be constant time O(1), since that information is stored locally in the calendar subsystem. (Stephen)

# 6. Project Scrum Report:

* Product Backlog (Table / Diagram)
* Sprint Backlog (Table / Diagram)
* Burndown Chart

# 7. Subsystems:

## 7.1 Subsystem 1 – Name 1 - *Individual responsibility*

* Initial design and model
  + Illustrate with class, use-case, UML, sequence ..... diagrams
  + Design choices
* Data dictionary
* If refined (changed over the course of project)
  + Reason for refinement (Pro versus Con)
  + Changes from initial model
  + Refined model analysis
  + Refined design (Diagram and Description)
* Scrum Backlog (Product and Sprint - Link to Section 6)
* Coding
  + Approach (Functional, OOP)
  + Language
* User training
  + Training / User manual (needed for final report)
* Testing

## 7.2 Subsystem 2 – Name 2 - *Individual responsibility*

* Initial design and model
  + Illustrate with class, use-case, UML, sequence ..... diagrams
  + Design choices
* Data dictionary
* If refined (changed over the course of project)
  + Reason for refinement (Pro versus Con)
  + Changes from initial model
  + Refined model analysis
  + Refined design (Diagram and Description)
* Scrum Backlog (Product and Sprint - Link to Section 6)
* Coding
  + Approach (Functional, OOP)
  + Language
* User training
  + Training / User manual (needed for final report)
* Testing

## 7.3 Subsystem 3 – Name 3 - *Individual responsibility*

* Initial design and model
  + Illustrate with class, use-case, UML, sequence ..... diagrams
  + Design choices
* Data dictionary
* If refined (changed over the course of project)
  + Reason for refinement (Pro versus Con)
  + Changes from initial model
  + Refined model analysis
  + Refined design (Diagram and Description)
* Scrum Backlog (Product and Sprint - Link to Section 6)
* Coding
  + Approach (Functional, OOP)
  + Language
* User training
  + Training / User manual (needed for final report)
* Testing

## 7.4 Subsystem 4 – Name 4 - *Individual responsibility*

* Initial design and model
  + Illustrate with class, use-case, UML, sequence ..... diagrams
  + Design choices
* Data dictionary
* If refined (changed over the course of project)
  + Reason for refinement (Pro versus Con)
  + Changes from initial model
  + Refined model analysis
  + Refined design (Diagram and Description)
* Scrum Backlog (Product and Sprint - Link to Section 6)
* Coding
  + Approach (Functional, OOP)
  + Language
* User training
  + Training / User manual (needed for final report)
* Testing

# 8. Complete System – *Group responsibility*

* Final software/hardware product
* Source code and user manual – screenshots as needed - Technical report
  + Github Link
* Evaluation by client and instructor
* Team Member Descriptions

# 9. Sources: (Adam Bowers)

* Hamm, Trent. “Don't Eat Out as Often (188/365).” *The Simple Dollar*, TheSimpleDollar.com, 18 Oct. 2017, [www.thesimpledollar.com/dont-eat-out-as-often-188365/](http://www.thesimpledollar.com/dont-eat-out-as-often-188365/).
* Lampert, Tess Rose. “Is Cooking at Home Really Cheaper Than Eating Out?” *Cheapism*, 11 July 2018, blog.cheapism.com/eating-at-home-vs-eating-out/.
* Cha, Ariana Eunjung. “Eating out at Restaurants Is Always Healthier than Guzzling Fast Food, Right? Wrong.” *The Independent*, Independent Digital News and Media, 16 July 2015, www.independent.co.uk/life-style/food-and-drink/news/why-eating-out-at-restaurants-may-be-just-as-bad-for-your-health-as-grabbing-fast-food-10394392.html.

***This is just a guide, and use it to create/improve your report. Feel free to add sections. You are responsible for your own subsystem/s, not other members. You have to contribute to the team’s goals and objectives, and develop your subsystem/s, write your documents and slides.***