ReciPlan

Requirements Analysis

Version 1.2

CMSC 495

For: Dr. Hung Dao

Due: 6 March 2022

Authors:

Team 6 – Josh Coe, Danita Hodges, Scott McClure

Version History:

Revision	Date	Author	Changes
1	29 Jan 22	Josh	Initial creation of document
1.1	30 Jan 22	Scott	Added in the sub-system context diagram
1.2	1 Feb 22	Danita	Formatting for readability and review

Data format questions:

Outside system	User
Input Data	Account information Recipe Search Parameters Recipe metadata: name, image, url, description, instructions, original yield Ingredient data: name, amount, unit of measure Grocery list items add
Output Data	Session authentication / user creation / password resets Search query results Original recipe confirmation/display Adjusted recipe display Complete grocery list items display
Data processing	User registration/authentication/authorization Recipe search by term in title Recipe creation and storage Ingredient conversion metric to imperial Ingredient conversion for yield Shopping cart item list

Context Diagrams:

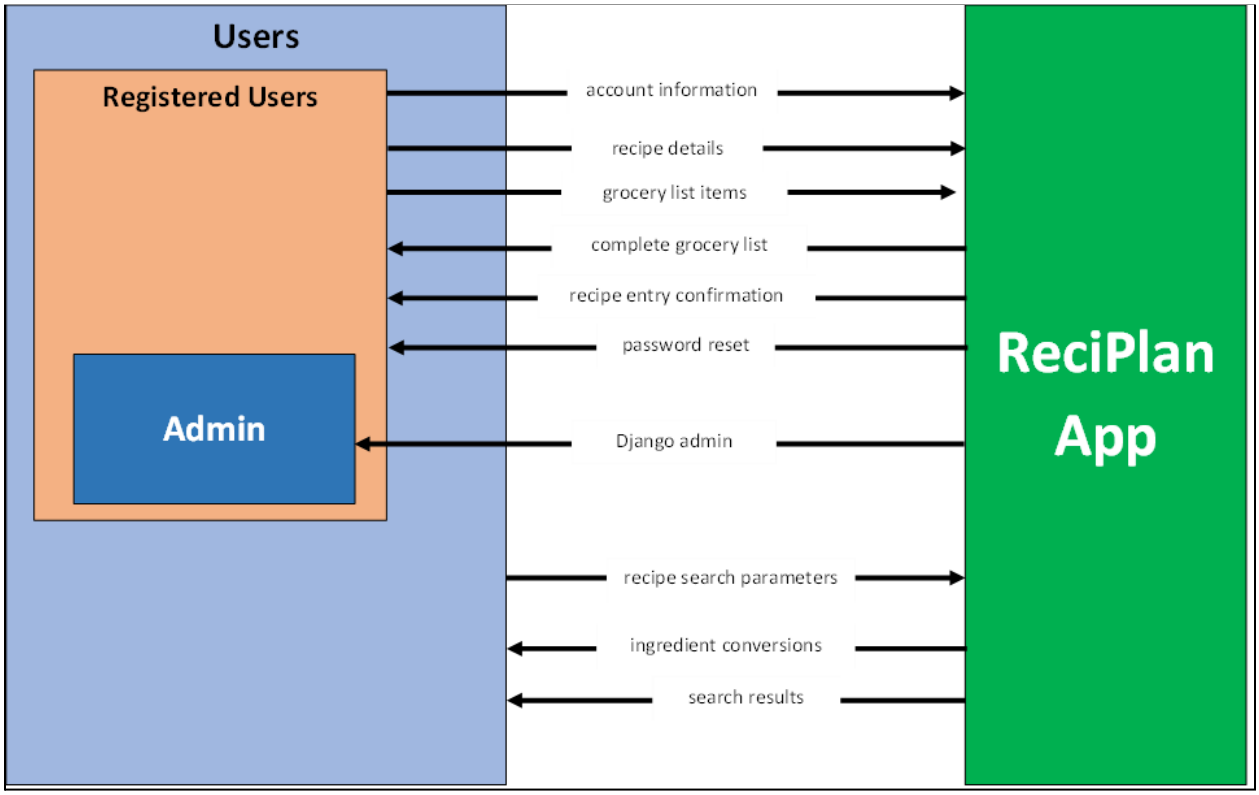

Figure 1: User Interface Context Diagram

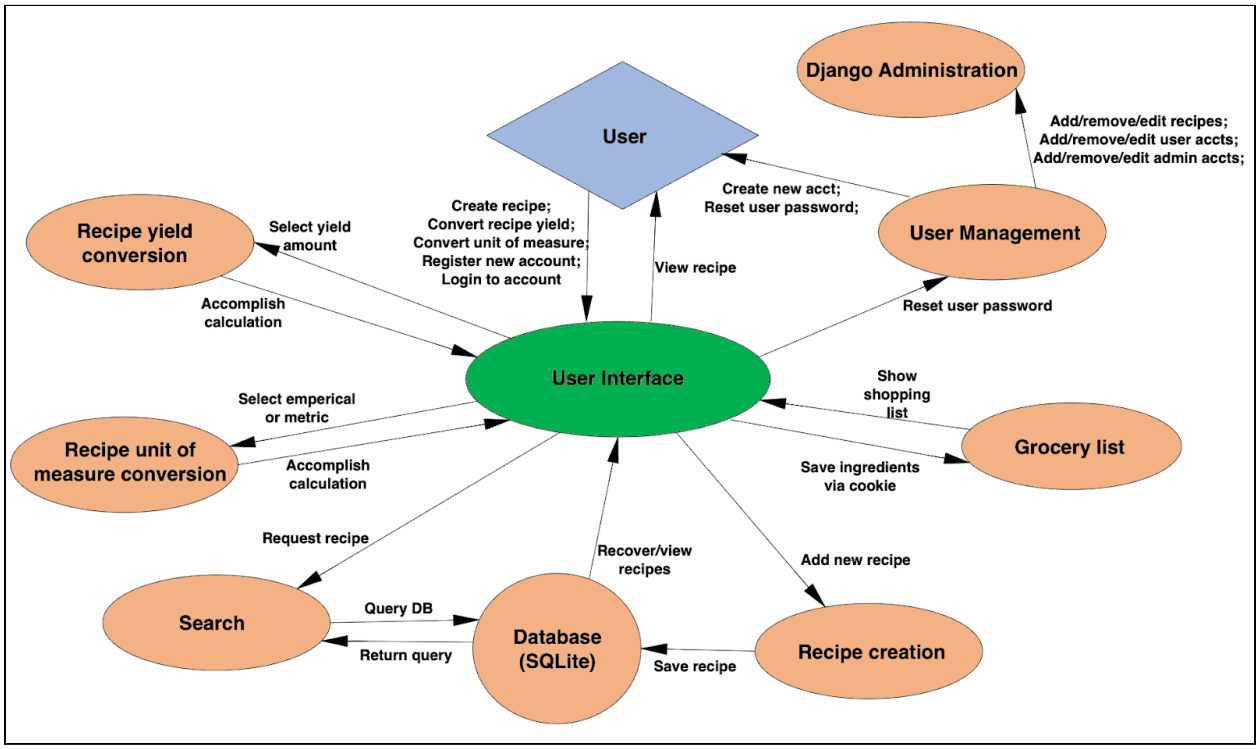

Figure 2: Subsystem Context Diagram

Sub-system descriptions:

a) Input
The input subsystem consists of the GUI based web application and all input form fields contained therein. This system takes information for each of the following subsystems from the user, sanitizes it programmatically, and passes it to the correct function to complete the desired task. Any field listed as input information is handled by this subsystem.

b) User Management
This subsystem allows a new user to register a new user level account by providing a unique username, email address, and password. The system creates the user, stores a new user object in the database based on the custom user model. The system also handles authentication by allowing the user to enter a password and receive a valid session cookie. The system handles password management by allowing users to enter a previously registered email to reset their password with a one-time use link. Finally, this system handles authorization by protecting views inside the framework with the @login_required Django decorator.

c) SQLite DB
This subsystem is responsible for storing all user entered recipes and user accounts.

d) Recipe creation
This system takes user input and places it in the SQLite system for storage. It allows a user to enter the recipe metadata and ingredient data for as many ingredients as are required and stores them in two tables in the SQLite DB.

e) Search
The search subsystem allows a user to enter a search term, and returns a list of recipes in which each title contains the search term along with a link to that recipe's detail view.

f) Measurement Conversion
This system takes both user input and input from the SQLite system. It allows the user to choose an adjusted measurement system and performs the calculations to output the desired new amounts.

g) Yield Conversion
This system takes both user input and input from the SQLite system. It allows the user to choose an adjusted yield and perform the calculations to output the desired new amounts.

h) Grocery List
This system takes both user input and input from the SQLite system. It allows the user to add items to a grocery list from the detailed view of recipes, and allows the user to view and print their final grocery list.

i) Django Administration
This subsystem is used to manage administrative functions of the web application. It is used to create new admin accounts; add, remove, modify or delete accounts; add, remove, modify or delete recipe and ingredient objects; and make administrative changes to the database. This system is a built-in function of the Django web framework.

Subsystems by Requirement:

Requirement ID	Subsystem	Description
RP-USER001	Input; user mgmt	New users shall be able to navigate to the website and create a new user profile consisting of a unique email address, unique user ID and a password.
RP-USER002	Input; user mgmt	Registered users shall be able to navigate to website and gain access by entering a valid user ID and password combination
RP-USER003	Input; user mgmt	Registered users shall reset password via email confirmation
RP-USER004	Input; SQlite DB	Any user shall be able to search the included recipe library by recipe name using a search box.
RP-USER005	Input; SQlite DB	Registered users shall be able to create their own recipe and add it to the library.
RP-USER006	Input	Registered users shall be able to clear a recipe while creating it in order to start over.
RP-USER007	Input; recipe conversion	Any user shall be able to click a button that will convert ingredient amounts between imperial and metric in the recipe detail view.
RP-USER008	Input; recipe conversion	Any user shall be able to change the target yield on any recipe to accommodate for the amount of portions required in the recipe detail view.

RP-USER009	Input; shopping cart	Registered users shall be able to add ingredients to a shopping list directly from the recipe detail view.
RP-USER0010	Input; shopping cart	Registered users shall be able to view their shopping list.
RP-USER011	Input; shopping cart	Registered users shall be able to remove items from their shopping list.
RP-ADMIN001	Input; Django Administration	Administrators shall delete user profiles as needed.
RP-ADMIN002	Input; Django Administration	Administrators shall reset user passwords as needed.
RP-ADMIN003	Input; Django Administration	Administrators shall be able to modify any recipe in the library.
RP-ADMIN004	Input; Django Administration	Administrators shall be able to delete any recipe in the library.
RP-ADMIN005	Input; Django Administration	Administrators shall create new admin accounts

Table 1: Requirements and Sub-systems

Possible Enhancements:

- Shopping cart aggregation function to add all items of same type together
- Online shopping API integration to direct order from Walmart/Instacart or similar
- Recipe scraping from popular sites such as recipes.com
- Comment box for recipes to share how you liked them

Possible Risks and Mitigations:

Injection attacks on input system: These are mitigated by using Django's built in forms which automatically sanitize data inputs.

Database overload/storage space restriction: If the application receives more than the expected traffic a postgresql AWS database will be added to handle the overflow. Migration will be performed to ensure no loss of user accounts or recipe data.