

Team Project - D.4 Design

Jasque Saydyk, Maria Granroth, Jasmine Mitchell, Matt Rittenback

Nosh

<https://github.com/clockian-NAU/Nosh>

CS 386 - Software Engineering

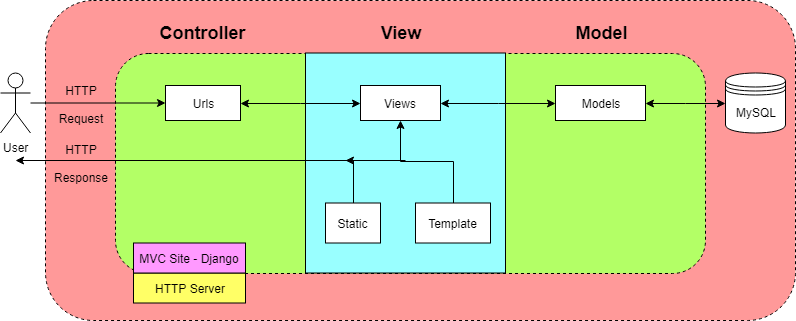
Spring 2018

Marco Aurélio Gerosa

Description

For those who would regard themselves as beginner cooks, Nosh is a recipe search website that can ease the burden of finding use for leftover ingredients. It allows for greater specificity on what recipes are available given an assortment of random ingredients. The system allows for users to search for recipes based on limited ingredients, time to prepare, materials, and dietary restrictions; they are given privileges to rate these recipes after completing them. A moderator is able to verify that the recipes are legitimate and reject them otherwise. Some main features include user accounts and recipe search functionality.

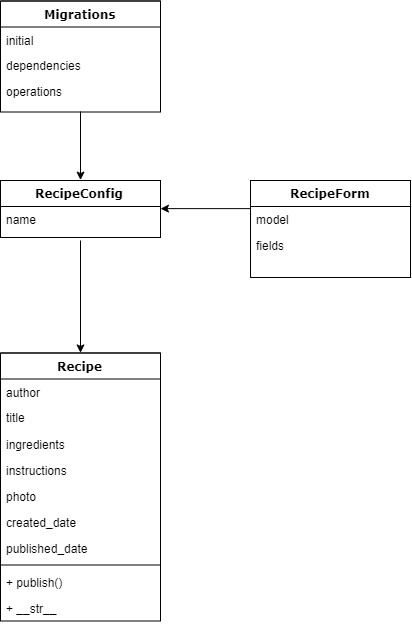
Architecture



The website works with a MVC paradigm, which is intrinsic with using Django. The Controller is responsible for taking the user requests and sending an action to the appropriate View to construct the webpage they are requesting, using the template and any static files, and giving the response. The View also makes updates to the Model, which is responsible for querying the mySQL database, and storing any business logic for the application.

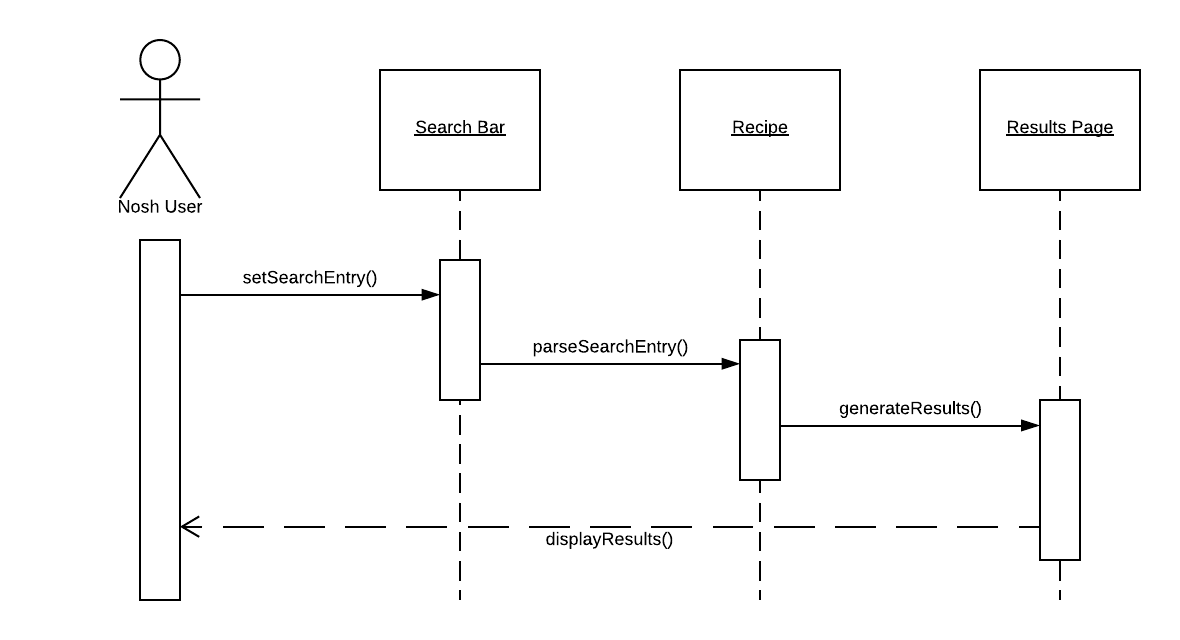
This is the view of a single server serving the website. We don’t expect to have enough traffic to warrant designing for multiple servers communicating with one database, with load balancers and dynamic creation of servers to demand.

Class Diagram



Note: Types are not included in the class diagram as they are not included in the source code written in the Python language.

Sequence Diagram

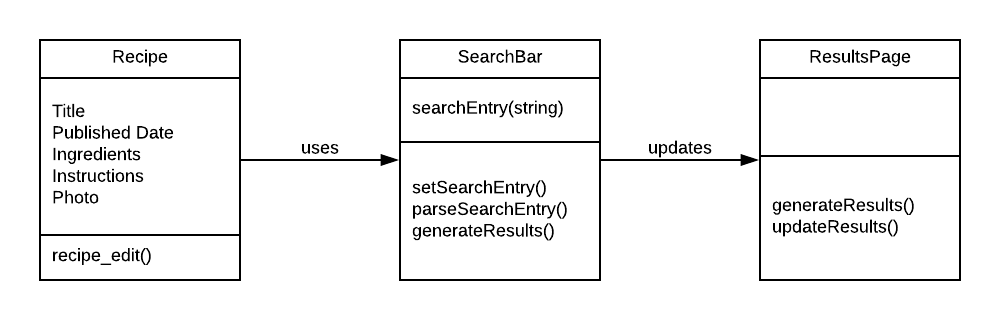


**Use Case Description (Search Feature)**

This use case describes how a Nosh User will search for recipes and reach the results page. Once the user enters input in the search bar, the search bar will set the input at the entry for the search. After the entry has been set from the user’s input, the system will parse the entry searching for all relevant recipes and generate results. Once the results have been generated, the results page will be displayed for the user.

Design Patterns

MVC Design Pattern

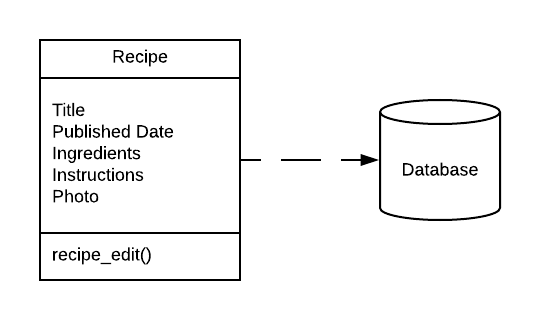


Recipe: <https://github.com/clockian-NAU/Nosh/blob/master/recipe/templates/recipe/recipe_edit.html>

SearchBar: Not yet implemented.

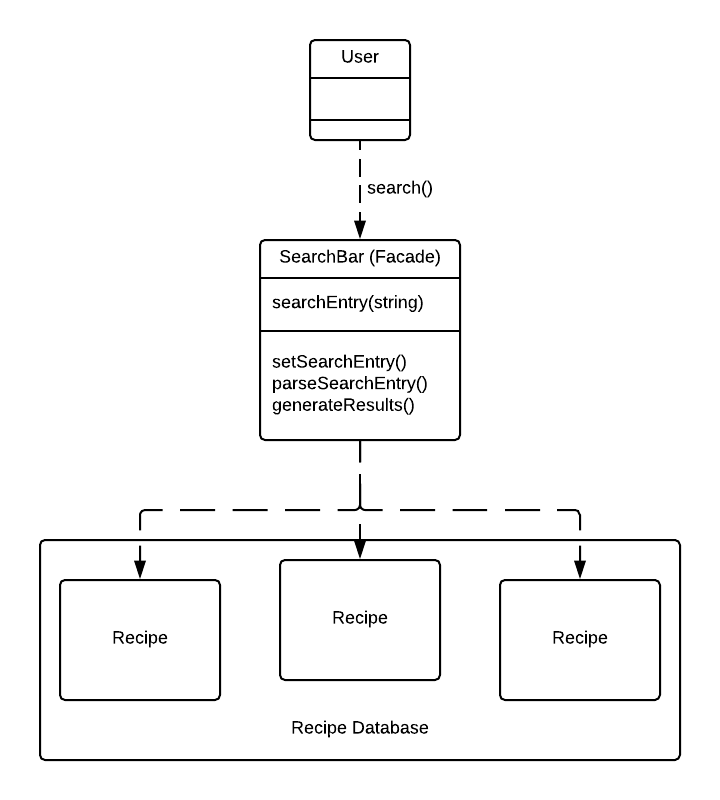
ResultsPage: <https://github.com/clockian-NAU/Nosh/tree/master/recipe>

Active Record Design Pattern



Recipe: <https://github.com/clockian-NAU/Nosh/blob/master/recipe/templates/recipe/recipe_edit.html>

Facade Design Pattern



User: Not yet implemented.

SearchBar: Not yet implemented.

Recipe: <https://github.com/clockian-NAU/Nosh/blob/master/recipe/templates/recipe/recipe_edit.html>

Design Principles

When considering web applications, the SOLID design principles do not typically apply in the traditional sense. Our project attempts to observe the SOLID principles by focusing on the single responsibility principle and the organization that tends to follow. The single responsibility principle refers to the idea that each class should only do one thing. For example, our recipe page has the responsibility of allowing the user to submit a recipe, and our user account creation page (not yet implemented) will only serve the purpose of creating a user account. However, the rest of the principles are not prominently applied in our project*.* The open-closed interface principle requires that classes are open for extension and closed for modification. Normally, this would be achieved through the use of interfaces in the code, which may be redundant for Nosh*.* The Liskov substitution principle suggests that subtypes should be substitutable for their base types, and while this is done in Python in a looser way than the principle states, we can’t foresee this being done in this particular project*.* The interface segregation principle says that no client should be forced to depend on methods it does not use. Instead, the interface should be made up of smaller, more specific interfaces. Our project implements it in the same way it implements the single responsibility principle, in that the user creation page is not included with the recipe submission page or the search bar. The final principle is the dependency inversion principle, which refers to the idea that high level modules should not depend on low level modules.Nosh implements this through the Model-View-Control framework it is implemented under. This separates the business logic in the model from the construction of the webpage in the view to support loose coupling, which separates the policy of Nosh, like the definition of what a recipe contains, from the implementation of the webpage that the user sees.

Group Participation

Nick Reader — Absent with no contribution or reciprocation to communication - 0%

Jasque Saydyk — Architecture diagram - 25%

Matt Rittenback — Sequence Diagram and Design Patterns - 25%

Jasmine Mitchell — Introduction section and Class diagram - 25%

Maria Granroth — Design Principles - 25%