**CS673 Software Engineering** 

**Team 1 - Cheffy**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Justin Fanning | Team Leader | *Justin Fanning* | 9/18/21 |
| Kyle Mabry | Backup Team Leader | *Kyle Mabry* | 9/18/21 |
| Andrew Fish | Requirement Leader | *Andrew Fish* | 9/18/21 |
| Jean Shalenkova | Design and Implementation Leader; Configuration Leader | *Jean Shalenkova* | 9/18/21 |
| Yanru Zhu | QA Leader | *Yanru Zhu* | 9/18/21 |
| Aidan Duffy | Security Leader | *Aidan Duffy* | 9/18/21 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 1 | Team 1 | 9/27/2021 | First Draft |
|  |  |  |  |

[Introduction](#_heading=h.gjdgxs)

[Software Architecture](#_heading=h.17dp8vu)

[Design Patterns](#_heading=h.3znysh7)

[Key Algorithms](#_heading=h.tyjcwt)

[Classes and Methods](#_heading=h.3dy6vkm)

[References](#_heading=h.4d34og8)

[Glossary](#_heading=h.2s8eyo1)

# Introduction

*In this section, give an overview of this document, and also address the design goals of your software system.*

This document is used to describe the high-level design goals of the cheffy web application. The backend system of cheffy will use Python, Flask, Flask-related libraries and the frontend will use HTML5 and Jinja templates. SQLite will be used to build the database, which will be the storage of user data. The Spoonacular API will be used as our source of recipes.

Two of our main design goals are maintainability and reusability, which can be achieved by using the MVC architecture (discussed in the next item), because the front-end templates are separated from the backend business logic, keeping everything easier to be maintained, and the model is separated from views and controllers, making it easier to be used with other controllers if we want.

# Software Architecture

*In this section, you will describe the decomposition of your software system, which includes each component (which may be in terms of package or folder) and the relationship between components. You shall have a diagram to show the whole architecture, and class diagrams for each component. The interface of each component and dependency between components should also be described. If any framework is used, it shall be defined here too. Database design should also be described if used.*

Cheffy is using a MVC architecture that decomposes the system into three main logical components: the model, the view, and the controller. So far the View and Controller have been developed but the Model has not yet. The view contains an [index.html](https://github.com/BUMETCS673/BUMETCS673OLF21P1/blob/main/src/templates/index.html) page which serves as the homepage and the user will be directed to the [recipe.html](https://github.com/BUMETCS673/BUMETCS673OLF21P1/blob/main/src/templates/recipe.html) page once clicks on the “search” button. The [app.py](https://github.com/BUMETCS673/BUMETCS673OLF21P1/blob/main/src/app.py)is the controller that serves as a bridge between the view and model. The cheffy API is defined in app.py by using @app.route("/recipe") served as the API interface and def showRecipes() being the implementations, and this is where the main recipe search function gets developed.

Here is an example scenario of how these three components interact (this feature has not been developed yet): The user clicks on the “save it” button in the recipe page (View) -> App.py gets that user action (Controller) -> App.py asks the Database to store that recipe into the user’s account (Model) -> Database (Model) notifies app.py (Controller) the recipe is saved successfully -> app.py (Controller) updates the recipe page (View) by notifying the user the success of saving the recipe.

# 

# Database Design (if applied)

*In this section, you shall describe any database if used in your software system.*

Cheffy will implement a SQLite database. As of now, the database is still in development, but will likely be used to track the following:

* + User Ids - connecting the username/email with password
  + User pantry items - connecting ingredient API ids with users
  + User favorite recipes - connecting recipe API ids with users

# Security Design

*In this section, you shall describe any security design in your software system.*

Here is a general overview: Using Auth0 in the application makes the most sense to me here. Auth0 allows us to utilize their server to store user info, if need be, and it seamlessly integrates into web applications (it is as simple as an API call, if we choose this route). Auth0 will help in generating access tokens and signatures. If we decide to hardcord everything ourselves, all account information, (login, storing data, and creation), we will primarily utilize AES-256 for encryption and HMAC-SHA256 for hashing, in addition to a password salt. We also need to introduce input sanitization for when a user adds ingredients to their pantry, in order to avoid any form of an injection into the database. The web application will also utilize HTTPS requests, not HTTP. Finally, we need to integrate cookies into our design for the purpose of session management; this would be for both users and non-users.

# Design Patterns

*In this section, you shall describe any design patterns used in your software system.* **[LATER]**

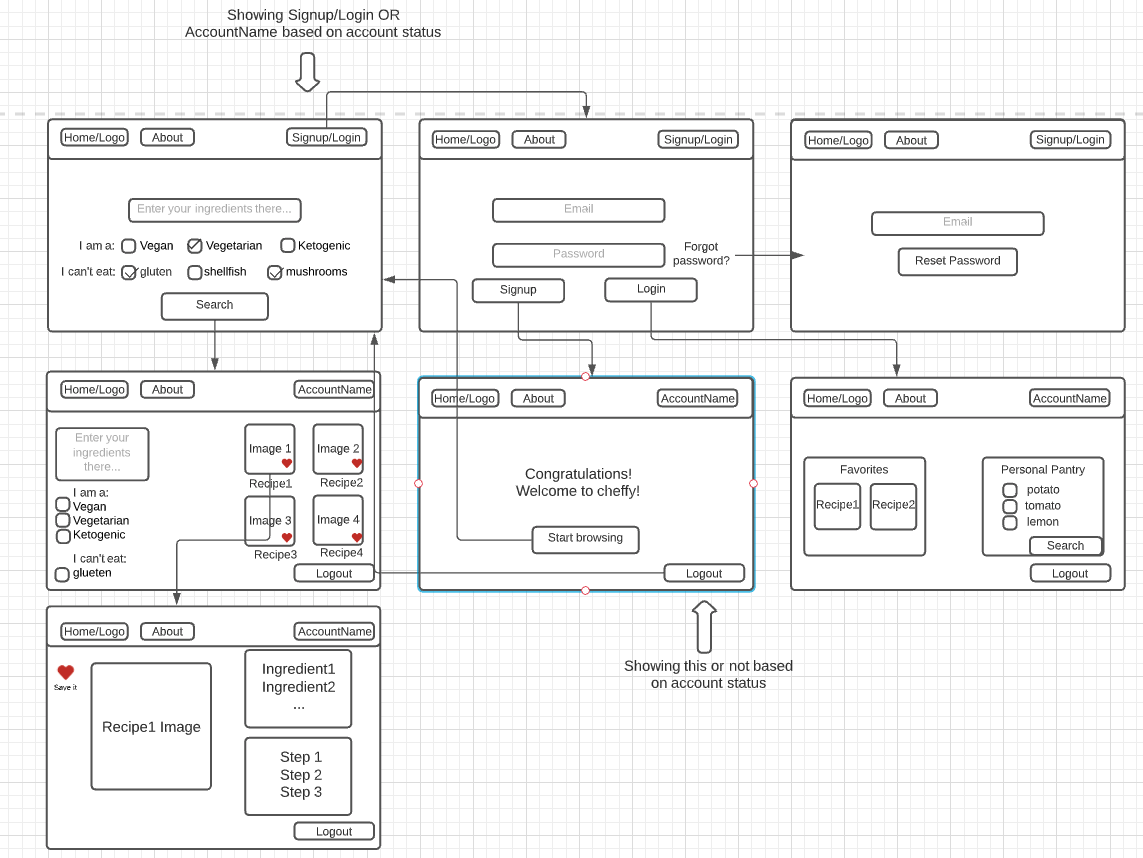
# Key Algorithms

*In this section, you shall describe any key algorithms used in your software system, either in terms of pseudocode or flowchart.* **[LATER]**

# UI Design

*In this section, you can describe your UI design.*

Below is the UI wireframe of cheffy. To take a look at it in a larger view please click [here](https://lucid.app/lucidchart/e43bb18e-998e-412d-8a71-ebcdbb20a884/edit?viewport_loc=-830%2C-463%2C3465%2C1799%2C0_0&invitationId=inv_3a0b20f5-932b-495a-878b-23f26f5b59da). Basically all of the pages will have a navigation bar that includes items home, about, login/register/accountname (if logged in), and logout. There will be an index/home page that has the search bar. Then the pages will be divided into two main sections based on if the user clicks on the login/register button or the search button: 1. Account-related pages: Account registration/login and account information, which will include the saved favorite recipes and personal pantry. 2. Recipe-related pages: The recipe list and a specific recipe page.



# Classes and Methods

*Please provide a link to your application API document which should be automatically generated.* **[LATER]**

# References

-https://spectralops.io/resources/how-to-choose-a-secret-scanning-solution-to-protect-credentials-in-your-code/

# Glossary