KAMI: Kitchen Assistant and Meal Innovator

By: Noah Addie, Stephen Shuecraft, and Zhen Ze Ong

SCIS-4923 Senior Project

Dr. Ahmad Al Shami

Table of Contents

Abstract

This project aims to revolutionize home cooking by providing personalized recipes and cooking suggestions, thus reducing food waste in the kitchen. The project introduces KAMI, a kitchen assistant that leverages AI to generate recipes based on available ingredients. Users have the flexibility to customize these recipes according to their preferences, budget constraints, and dietary requirements. Additionally, KAMI allows users to manage their ingredient inventory, save generated recipes, and interact seamlessly through a user-friendly website interface. Emphasizing simplicity and user-friendliness, KAMI assists users in inputting ingredients and learning new recipes effortlessly. This report encompasses the project's mission, goals, strategies to accomplish them, research findings, and the final prototype of the KAMI system.

1. Introduction

1.1 Context and Motivation

We designed this project to answer two questions: Can AI help create proper dishes for someone with only certain available ingredients? Furthermore, can it adapt these dishes to the user's dietary restrictions and choice of cuisine? We are setting out to make an AI-based recipe generator that functions by feeding its available ingredients and establishing limits or constraints to the dish. We want to integrate this into a website that generates these recipes, considers dietary needs, generates dishes from different cultures, and stores an inventory of the ingredients in the user's kitchen. This project aims to significantly improve the at-home kitchen experience and reduce food waste in households everywhere.

1.2 Problem Statement

In a rapidly changing culinary landscape, individuals face challenges in planning and preparing meals that align with their dietary preferences, time constraints, and available ingredients. Traditional recipe generators cannot provide personalized, creative, and new recipes. This project aims to stand out by using AI to leverage state-of-the-art natural language processing and machine learning techniques to recommend, generate, and enhance culinary experiences. The system considers user preferences, dietary restrictions, ingredient availability, and cultural influences when generating recipes. AI is the solution to personalization as it can quickly adapt its output based on the scenario without extra work from the user or the system. It could inspire culinary innovation while promoting home cooking and reducing food waste.

1.3 Project Objectives

According to the U.S. Department of Agriculture (USDA), only 60% of women and 33% of men do food preparation at home on an average day (Zeballos, 2020). The project hopes to incentivize more of the population to cook at home, as home cooking can result in healthier and cheaper meals. People with dietary needs can also benefit from preparing their meals. “...the best part about cooking at home is controlling the ingredients so you know exactly what you're eating” (Wicks, 2022).

The main goal of this project is to provide a possible recipe regardless of the available ingredients while accommodating changes in dietary restrictions, type of cuisine, and ingredient supplemental options. We can achieve this goal by implementing an AI API, database, and user interface. These components work together to form a convenient and personalized system for each user.

Eventually, we plan to create a website to house all the functions we want incorporated with our AI kitchen assistant. We came up with the name for the project, KAMI, which stands for Kitchen Assistant and Meal Innovator. We aim to have the prototype functional by the end of the year, merging an AI API, database, and user interface using Python code. KAMI can make a recipe from the available ingredients or have it randomized from a master list of known ingredients. It will also have an inventory for a user’s ingredients, a way to put in criteria (dietary needs and cuisine preferences), and a user account feature to save personal preferences and data. If we complete the project earlier than expected, we have numerous next steps that we could take, such as adding a budget API to estimate the cost of the meal, a chatbot capable of general cooking help, and allowing users to save recipes and make digital cookbooks from them.

1.4 Project Scope

1.4.1 Main Objectives

We intend for KAMI to work in home kitchens of all levels, shapes, and sizes. It should also benefit beginner and expert cooks by reducing waste and teaching them new recipes. To fit our limits of resources and time, this is our list of priority objectives:

- Implement a way to generate recipes by calling multiple APIs for different functions
- Create a database to implement with the app and the user account.
- Create user accounts that store recipes, allergies, preferences, and inventory.
- Create a website to house and act as a man-in-the-middle to the AI and database, allowing for easy and quick inputs for recipe creation and general kitchen help.

1.4.2 Requirements

The project requires users to access KAMI on a website through a browser. We plan to take advantage of the flexibility of a website as it allows access to it on both mobile and desktop devices. This option would allow users to be flexible on what kind of device they prefer to use. The device must have a connection to the internet to access the website, along with the database and AI API. The user does not need to create an account or have actual ingredients to use the kitchen assistant.

1.4.3 Constraints

We only designed KAMI to be a digital assistant in the kitchen. It cannot physically help or guide a user with cooking. Although the recipe contains detailed instructions, the user must understand the culinary and kitchen tools they are using. KAMI does not purchase or physically manage ingredients for the user. The user will still have to purchase ingredients and determine their meal budget. The user must also understand the limitations of the KAMI and ensure that the

inputted information is correct. AI is not perfect and can make mistakes, which must be taken into account by the user.

It may also be impossible for the AI to generate a recipe with certain ingredients. For example, if a user inputs salt and pepper as their only ingredients, the AI has no choice but to deny their request. The AI is not required to use every ingredient the user has unless specified otherwise. The recipe may turn out differently than intended, especially if the user puts the AI under unreasonable conditions.

KAMI will not track the user's data for marketing or advertising purposes. Users can save their ingredients, allergies, or cuisine choices to make it more convenient, but they will not be shared or used outside their intended purpose.

1.5 Stakeholders

This project targets beginner and expert home cooks alike. We aim to develop KAMI to reduce the waste of food, time, and money needed to prepare meals. It is also a guide for aspiring cooks who may have few ingredients to work with or are inexperienced in the kitchen. The recipes generated by the AI must be detailed in every step so that the user gets all crucial information during the process.

Food businesses are also dealing with the problem of food waste. An online article states, “...around 40% of all food is wasted—with 66 billion pounds consisting of commercial leftover food waste” (Mettler, 2023). If we wanted to branch out our audience to locally owned or franchise restaurants, we could add features to help create daily menu schedules based on ingredient inventory, implement inventory access to distributors for seamless inventory updates, and add methods for finding more cost-effective recipes.

1.6 Report Structure

We compiled this report to document every stage of the project's timeline, providing a detailed account of its inception and conclusion. This comprehensive documentation is designed to assist in replicating or deriving insights for similar product development. It serves as a chronological guide and a deep dive into the intricacies of the project's system, elucidating our methodologies, challenges faced, and strategic decisions made along the way. We have included detailed descriptions of our results and findings, highlighting the advantages and disadvantages encountered throughout the project. Additionally, this report outlines potential future directions and next steps, offering recommendations based on our experiences and the outcomes achieved. This should empower readers with the knowledge and tools necessary to undertake a similar endeavor or to build upon the foundation we established.

2. Domain-Related Concepts and Systems

2.1 Introduction

This project aims to help decrease food waste and increase home cooking in households worldwide. We considered different variations of research, development, and analysis methods to determine the best practices for this project. This section will discuss the project's requirements, risk management, and feasibility.

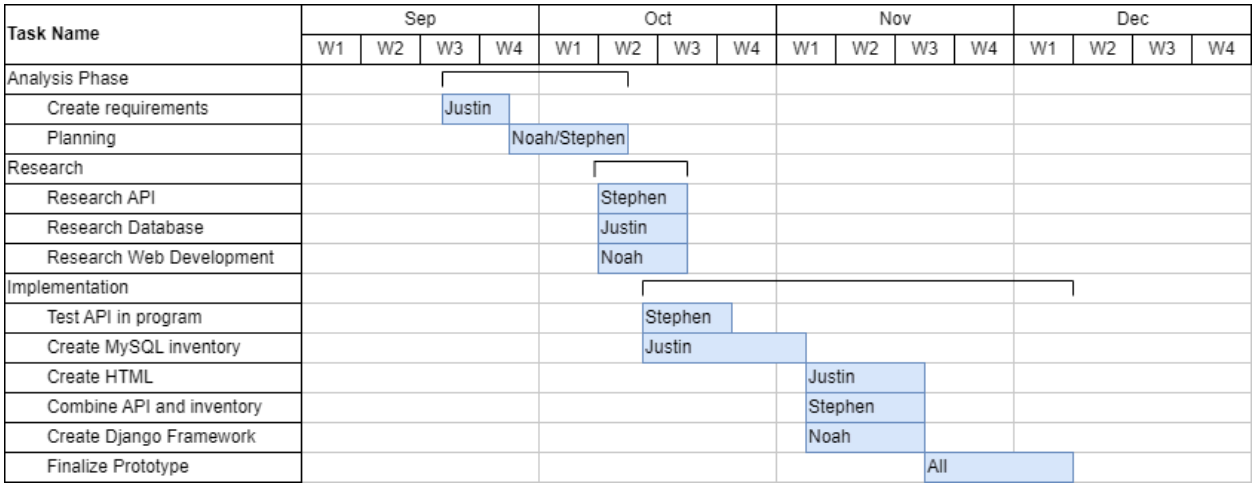
2.2 Adopted Methodology

Every successful project requires a well-planned methodology. No single methodology is the best, as each may be more effective in different scenarios. For this project, we adopted Agile Methodology, which is particularly suited to environments requiring flexibility and iterative progress. Given that each team member would be working on different components of the project, Agile's iterative and incremental approach allowed us to adapt and evolve our project dynamically. "Teams that adopt the Agile methodology are able to complete work faster, adapt to changing project requirements, and optimize their workflow" (Agile Methodology: Project Management, 2022). This model supports frequent reassessment and adaptation, which is crucial in responding to changing project requirements and stakeholder input effectively.

As depicted in the updated Gantt chart (Figure 1), our project comprises the AI API, database, and user interface. Each team member focuses on one main component but is also engaged in regular sprint reviews and retrospectives to ensure integration and coherence across the components. We further break these components down into smaller, manageable tasks, allowing us to address issues and refine the product incrementally. Unlike the Waterfall model, Agile does not require the completion of one phase to begin another, facilitating a more fluid and

overlapping project development. This flexibility ensures that the project adapts to new insights and changes in requirements without significant setbacks. The dynamic nature of Agile fits well with the project's evolving needs, enabling a more collaborative and adaptable development environment. Through the implementation of Agile practices, we aim to ensure that our research and development processes remain responsive and efficient throughout the project's lifecycle.

Figure 1



Gantt Chart of the Project

2.3 Research

Websites are very accessible at this age, and we plan to create one that can improve the quality of life of both beginner and expert home cooks. We initially wanted to develop an AI specializing in generating recipes based on cooking websites and videos. After a quick online search, we discovered many other recipe-generating AIs, such as ChefGPT and DishGen, existed on the market. To succeed in a capstone project, we needed to build a better AI or have our project do something the others did not. Since experts fully developed these AIs, they will likely outperform any AI we can create. We decided to create KAMI, which implements an AI API instead.

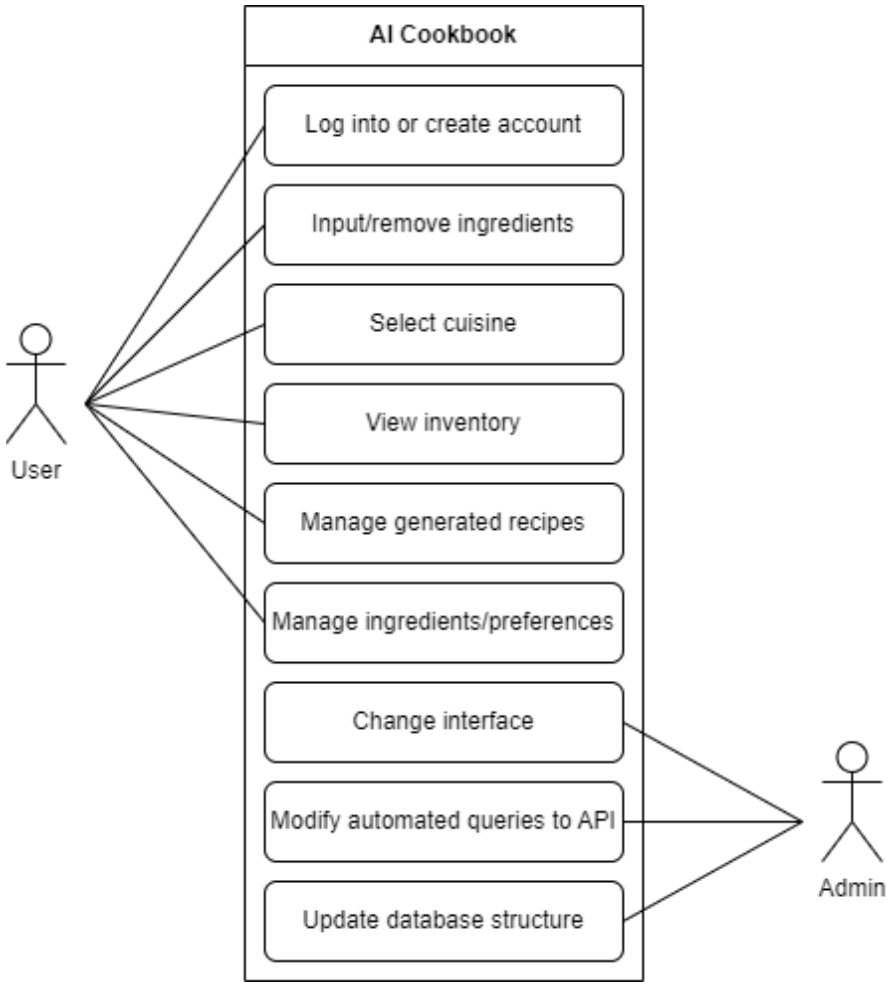
The original idea was to have the kitchen assistant as a mobile application. However, after extensive research and work, connecting many APIs to an application became too complicated and would take many extra steps. The project shifted towards hosting the kitchen assistant on a website instead. Having the prototype on a website would provide more advantages and fewer disadvantages than a mobile application.

Inputting the same ingredients for every use would be too troublesome for a consumer, so a database was needed to store the user's inventory. Dietary needs would also be considered, as users might have allergies or observe a particular diet for health or religious reasons. KAMI should output a recipe with detailed steps that a beginner cook can understand while not being too trivial for an expert cook. We aim to use it to elevate a user's cooking experience in their kitchen. After extensive research, we decided to use OpenAI's ChatGPT API for the AI, MySQL for the database, and an HTML website for the user interface.

2.3.1 Resulting Diagrams

This project used the following diagrams to represent different components of the system. The use case diagram illustrates the high-level functions of the project, representing the interactions between actors and the system. ER diagrams represent the structure of the database, showing the relationships between tables and values.

Figure 2



Use Case Diagram of the System

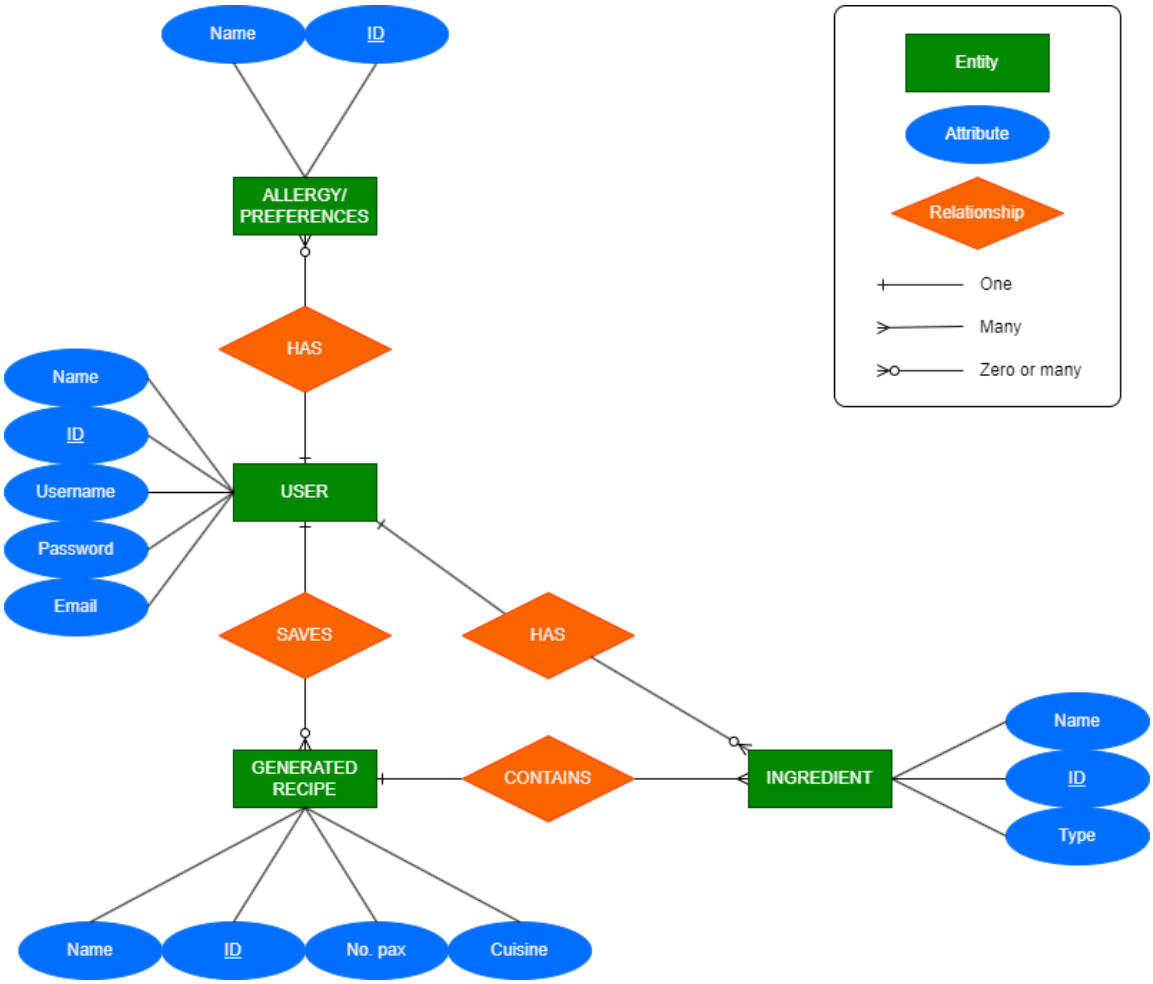
The use case diagram above (Figure 2) represents the high-level functions of our system. It summarizes the interactions of the actors (users and admins) with our system, like logging into or creating an account and inputting or removing ingredients in their inventory. Users will have access to specific interactions but not others, indicated by the lines drawn to each interaction. Admins have backend access to the interface, API, and database, also shown by the drawn lines. We can further break down each user case, as shown below for the “Log into or create account” use case.

Figure 3

Use Case Description		
Use Case Name		Log into or Create Account
Actors		User and device of choice
Preconditions		The user must have a device, be connected to Wi-Fi, and access to the website. If the user is wants to login, they must have created an account.
Normal Flow	Description	If the user is creating an account, the user must select the option, below the login form, of "Create an Account". Then they must enter their name, email, and desired username and password. Then they will select create account. If the user wants to login, they will enter their email or username and password, then select login.
	Postconditions	If the user is creating an account, it will print out "Thank you for creating an account!", then login the user. If the user just logins, then it will pull their ingredients and preferences from their account and give them options about creating recipes.
Alternative flows and expectations		The user may enter an invalid email, password, or username. The user may try to enter a email or username that is already in use when they are creating an account. A user forgetting their password.
Nonfunctional requirements		The user must remember their password and email they used. They must know their available ingredients and preferences in order to enter them into the system.

Use Case Description for Logging into or Creating a User Account

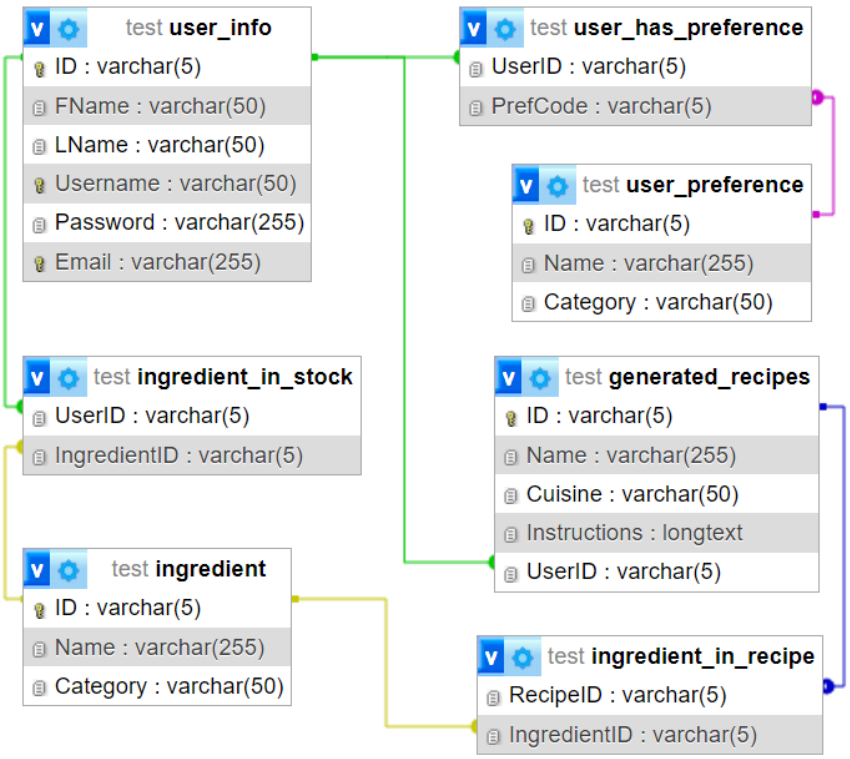
Figure 4



Early ER Diagram Prototype of the Database

We designed an early version of the database using this ER (Entity Relationship) diagram (Figure 4). It represents what attributes each entity has, as well as the relationships between the entities. Relationships can be one-to-one, one-to-many, many-to-many, or optional, depending on the requirements. Each entity has at least one key attribute used to identify the entity. We also assign unique IDs to each entity that cannot be modified unless we remove the entire entity from the database.

Figure 5



Final 3NF ER Diagram of the Database

Figure 5 illustrates the relationships between the tables. The colored links show which columns were connected. For example, with the ID from “user_info,” the IngredientID of “ingredient_in_recipe” can be accessed through multiple links. On KAMI, this will allow the user to view the ingredients in the recipes they had previously generated and saved. The Key icon represents each table’s key attributes. The diagram also includes the data type and size of each value in the table. We used 5-letter strings like “F0001” and “U0001” as unique IDs. Doing so prevented any confusion in tables containing multiple unique IDs.

2.4 Project Feasibility

2.4.1 Technical Feasibility

The system is composed of the AI API, database, user interface, and user accounts. The problem with the user interface is easily solvable; it will house all the functions in the website using a web framework. We chose Django as this project's web framework for different reasons, but we will go further into that later, as the AI and database are the priority. The website uses HTML, which can be easily imported into the web framework. Lastly, other AI tools, such as image generation, can further personalize the recipes that the users generate. These components must work together to provide a good user experience with a developed A.I. kitchen assistant.

The user requires a functional device with a stable Wi-Fi connection. The best solution for ensuring the device is functional and has a Wi-Fi connection is to host the KAMI on a website. Only when a user has a device such as a smartphone, tablet, or laptop with a connection to the internet can they access a website. Adding new criteria or changing information like available ingredients or cuisine can also be easily updated through an exemplary user interface.

2.4.2 Operational Feasibility

This project will use existing AI and database structures, OpenAI's ChatGPT and MySQL, respectively. We chose these products because they are popular in the market, easy to implement, and have heavy documentation. This choice allows us to focus on the system of the project rather than "reinventing the wheel" by creating a new AI or database structure. Popular products also ensure high performance and reliability, but a common downside is extra cost for the project. Using HTML for the website is also efficient, as many templates and tutorials are available to the public. The user accounts will also be easy to implement because most web

frameworks have that built into their system. Python will be the primary programming language as it contains many libraries that connect the OpenAI ChatGPT and MySQL APIs to the system.

2.4.3 Schedule Feasibility

We researched and brainstormed project ideas during the first two weeks. Then, we researched our steps for the next two weeks to ensure every project step was feasible. After that, it was a cycle of learning and using all the models chosen to complete a prototype within the timeframe. The project code and presentation were due a week before the end of the semester.

2.4.4 Economic Feasibility

The only main cost of the current project prototype is $10 to test and run the OpenAI APIs. The APIs are charged according to tokens, where 1000 tokens are about 750 words. Different OpenAI AI models have different token rates. This project uses the following models:

Model	Input	Output
GPT-4	$0.03 / 1K tokens	$0.06 / 1K tokens

Model	Quality	Resolution	Price
DALL·E 3	Standard	1024×1024	$0.040 / image

The APIs are reasonably priced and only cost a few cents after multiple queries, so $10 should be sufficient for prototype testing for both semesters. The web framework has no or negligible cost. Other APIs like MySQL and webpage development also have no cost.

2.5 Risk Management

While there would not be much personal or identifiable information tied to each user's account, it would still be in our best interest to ensure the privacy and security of the user's data. Any data collected from an individual must not be used as a marketing tool. These include the

ingredients and dietary habits of the user. The data and communication between the user and KAMI must be private and secure.

Should the user enter unneeded personal data, the system should not store it in any shape or form. The database should have a proper system and be regulated often to minimize unintended data sharing. As the APIs used in the project are external sources, it must be in our best interest to ensure that those sources do not collect a user's data without their consent. In the current state of this project, it does not share information with any third-party organizations. The system should inform users about third-party changes and allow them to decline those services.

2.6 Requirements

Most of the requirements for this project are self-evident. The user must have a device and a list of available ingredients and preferences to input into the website. The device will require an internet connection. The user will have to determine if the recipe properly considers allergies and other food issues, update their inventory, and provide their data for input. Users who intend to remove or edit their data in the system should be allowed to do so.

2.7 Conclusion

In conclusion, our project follows an Agile model as its methodology. After extensive research, we shifted the project's focus from creating an AI to integrating existing APIs into KAMI, a digital kitchen assistant. The use case and entity relationship diagrams illustrate the system's functionalities. Before starting the project's development, we addressed the technical, operational, schedule, and economic feasibilities. We considered privacy and security measures to mitigate potential risks. The system requires user input and discretion to function as intended. The project aims to offer a user-friendly and personalized kitchen assistant, contributing to a more enjoyable cooking experience.

3. Computing-based Solution/System Analysis

3.1 Introduction

To determine the effectiveness of the envisioned project system, the domain, functional, and non-functional requirements had to be accessed appropriately. All these requirements are vital to the final product's success and can lead to complications if unchecked.

3.2 Domain Requirements

We decided to operate KAMI on a website accessed through a browser. The website should be compatible with mobile and desktop devices with similar functionality. A user should be able to access their data across multiple devices. The device must have a connection to the internet. This section also discusses other domain requirements: inputs, outputs, processes, performance, controls, and scalability.

3.2.1 Inputs

The inputs for this project are the available ingredients and preferences (allergies and diets) for each individual and the individual’s choice of cuisine for the recipe. All inputs are sent to the AI API to determine what recipes it can generate. The user can also select a “go wild” option, which sends a list of roughly 130 commonly known ingredients to the AI API. The AI can use any combination of ingredients from that list to generate a recipe while considering the user’s preferences and cuisine. Users may also input what ingredients they want to be added or removed from their inventory. Inputs can be saved to user accounts, allowing users to save their inventory and preferences to generate future recipes.

3.2.2 Outputs

The main output of the system is the recipe. The generated recipe contains the ingredients from the user's inventory, the necessary cooking steps, and additional notes like warnings and substitute ingredient options. Another output is an AI-generated image of the envisioned meal. Similar to the inputs, generated recipes and images can also be saved by users to their accounts.

3.2.3 Processes

Inputs can be taken directly from the database and the user. The system formats these inputs into a string and sends them to the AI API as a query. String manipulation is vital for the whole system to work. The query should prompt the AI to return an output that follows a consistent format. A consistently formatted output will allow the system to send the needed information to the website to display the output expectedly. The system has an error-checking process to ensure an expected AI output. If the output is not formatted correctly, it may cause the website to display the results incorrectly. This error is especially evident in the AI image generation. As the image generator should only receive the recipe's title, receiving any other part of the recipe, such as raw ingredients or warnings, could result in a completely unexpected image.

The recipe output from the AI is returned as one string and must be separated (or sliced) into portions before the system can use it. A section of the code slices it into a list of strings: the title, ingredients, and recipe steps. As mentioned above, the recipe's title is sent to the AI image generator to return an image. The recipe and generated image are returned to the system and sent to the website where users can access them. The system then stores the title, ingredients, and cooking steps to their respective tables in the database.

3.2.4 Performance

Since the project is composed of several APIs, the overall performance strongly depends on each API's speed. API calls to the database are nearly instant. The API generates the text output using “autoregression.” It generates the first word, predicts the next word, and then the next. It repeats the process until it completes the whole recipe. The website should display the text as it is generated and should start within an estimated 5 seconds. Popular AI chatbots like ChatGPT and Google Bard return outputs in a similar fashion.

3.2.5 Controls

The user should be given adequate control over the website so that it can still function properly when errors or mistakes occur. The option to regenerate a similar recipe or add an unknown ingredient should be available for the user. The unknown ingredient may be foreign to the initial database because it is rare or culturally exclusive. Examples could be oyster sauce or miso, which do not exist in the initial database due to their exclusivity to Asian meals. However, the user should be able to input them as text manually, and the AI should be able to use the ingredient in the recipe as long as there is information about it on the internet. The initial database gives the AI a list of common ingredients for the random recipe generation.

3.2.6 Scalability

In the future, many new functions could be added to the website. We considered several future functions that could improve the system:

- A budgeting system that considers ingredient prices
- Inventory management regarding ingredient expiration dates
- Image detection for easy input of ingredients
- Business accounts with functions tailored toward restaurants and food stalls

If more users begin using the website, both the database and the AI may need to be upgraded to handle the increased demand. The database may require cloud storage services, and the AI may require an upgrade to make more simultaneous input and output calls to keep up with the increased demand. These upgrades will increase the cost of running the project.

Turning the website into an application for iOS and Android devices is also a possible addition to this project in the future. Having it as an application may encourage its popularity and marketability. Users should be able to use their accounts interchangeably between the website and application versions. An offline version was considered but may not be possible, as although the user can download a personal database to the device, the AI will still need an internet connection to function.

3.3 Functional Requirements

Several things are required for the whole system to function correctly. The user's browser should be able to run the website as intended, and it must establish a connection to the database and the AI to begin any process. These will all require a stable internet connection. The queries sent to and from the AI must be formatted correctly. Otherwise, the results might be inaccurate.

3.4 Non-Functional Requirements

The need to save inventory, preferences, and recipes is an extended functionality that is not required. The user is not required to create an account for the system to function. Image generation is also a non-functional requirement, only providing the user with an idea of how the recipe might turn out without any other benefit.

Other features may be added to the system to improve reliability and efficiency. We considered examples such as image detection through a camera or receipt scanning to automatically input detected ingredients. Another would be a page on the website where users

can share and rate each other’s generated recipes. Kitchen tools like timers could be built into the website as well. However, these ideas are not required and thus not prioritized. So, it was decided not to include them in the project's current timeline.

3.5 Conclusion

In conclusion, a successful project requires proper planning for its requirements. The project took detailed consideration of domain, functional, and non-functional so that setbacks would be minimal. Domain requirements included the system's inputs, outputs, processes, performance, and controls. Functional requirements are the components the system must contain to function, such as the AI and database. Non-functional requirements do not relate to functionality; they only improve functionality in reliability and efficiency. These include the implementation of user accounts and image generation.

4. Computing-Based Solution/System Design, Implementation, and Testing

4.1 Introduction

The design process of this project was complicated as it required various development tools to work in harmony for the final product to function. User inventory and preferences were to be stored in and retrieved from a database, formatted adequately by a program, and sent to an AI API to return a desired result. The result can be saved into the database and retrieved for future use. We do this behind a user-friendly interface, where the user can access all of KAMI's functions.

4.2 Programming Environment

4.2.1 Visual Studio

Visual Studio Code is an easy-to-use, well-organized code editor. Its design allows the user to have easy access to the entire directory. We thought this would be an excellent IDE (integrated development environment) for this project. It can work efficiently with Andriod Studio to create a working application. However, due to our decision to no longer develop an application, we decided to find another software to develop our project.

4.2.2 PyCharm

PyCharm is a popular Python IDE. The team member who worked on the system code preferred it over Visual Studio, so we used it as the IDE instead. PyCharm's integrated terminal streamlines downloading dependencies, which are essential for the system's functionality. The AI and database APIs were connected through Python code, allowing convenient prototype testing prototype within the IDE.

Users can interact with the system through a terminal menu, where their selections influence the generated string. The generated string is a combination of preprogrammed phrases and user input. It dynamically incorporates details like dietary restrictions, ingredient preferences, and the cuisine of choice. This personalized string is then sent to the AI API, starting the recipe generation process.

4.2.3 XAMPP MySQL

The inventory and user information database was set up on XAMPP, a popular web development tool. XAMPP provided many development tools, including MySQL, the database application used in the backend of our project. XAMPP allowed us to simulate a server environment on our local machines so we could quickly make SQL queries locally in the early stages of the project. Having a local server makes it very useful for prototyping and debugging before deploying the database to a live server. XAMPP uses phpMyAdmin, a web-based interface for MySQL database management. We used SQL queries on phpMyAdmin to create tables, populate them with sample data, and set relationships between them. We integrated MySQL into Python by importing the “MySQL.connector” library, allowing the Python program to send SQL queries in a string format to the database. These program queries function identically to queries on phpMyAdmin.

4.3 Tools and APIs

4.3.1 OpenAI GPT-4 API

The core of the KAMI lies in the capabilities of OpenAI's GPT-4. “GPT-4 is OpenAI’s most advanced system, producing safer and more useful responses” (OpenAI). It is an improved version of older versions like GPT-3.5. By importing the “openai” library, the system can send

queries and receive responses from GPT-4 through the AI API. To be operational, the API requires a key generated on the OpenAI website, and the account associated with that key must have an active subscription. In the code, we can prompt the API with how GPT-4 is required to act, along with a level of randomness determined by a factor called “temperature.” The temperature ranges from 0 to 2, with 0 being completely unrandom (giving the same output for the same initial prompt) and 2 being complete random gibberish.

4.3.2 OpenAI’s DALL-E 3 API

The OpenAI subscription also includes DALL-E 3, an AI image generator. It uses the same “openai” library, and we can send a text prompt to the API in Python code. The API returns an image based on the prompt, which we determined to add as a feature for the project.

Figure 6

AI-Generated Images during Testing

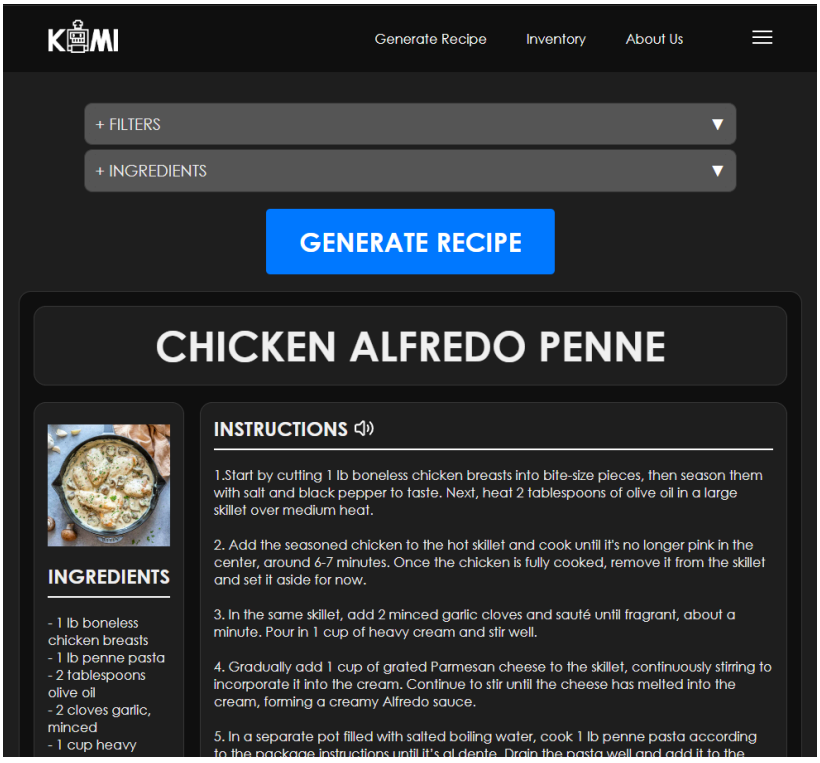
4.3.3 HTML, CSS, and JavaScript

HyperText Markup Language (HTML) is the standard markup language for webpages. It defines the structure and design of web pages. HTML often implements Cascading Style Sheets (CSS) and JavaScript, creating 95.5% of all websites on the World Wide Web (*Usage Statistics of HTML for Websites*, n.d.).

CSS allows websites to achieve more complicated designs by laying out and structuring HTML code. It allows the grouping of webpage contents, keeping the design neat and formatted. It enables the webpage to adapt its width and height as the window changes. It also allows for interactive buttons that change color when hovered upon, drop-down menus, and other valuable functions.

JavaScript is responsible for the behavior of the website, allowing it to react to specific user actions. It enables the website to accept input when users click a button or tick a checkbox. JavaScript often incorporates third-party libraries to perform its required tasks. With the help of ChatGPT and extensive online documentation, we generated the necessary HTML, CSS, and JavaScript code for a functional website.

Figure 7



Screenshot of the Prototype Website

4.3.4 Django

Django is a web application framework that uses Python as its basis. We used this to expand our API application to a broader audience. Django allows us to host our program on a web server accessible through a browser. We specifically chose this framework due to a focus on easy Python implementation, the ability to implement an SQL database, and hosting the HTML for the website.

4.3.5 Dependency Downloads

We installed several other libraries for the project to work. They provided the required functions or allowed the APIs to connect with the code. We ran the following "import" statements in our Python code:

- `import openai` - OpenAI API library
- `import PIL` - Pillow, an image displaying library (only used for the prototype)
- `import mysql.connector` - database API library
- `import requests` - HTTP library

4.4 Testing Techniques

This development process adhered to a test-driven development (TDD) approach, where each newly implemented feature underwent immediate testing to verify the correct output. This testing technique was consistently applied, encompassing functionalities like layouts, button clicks, scanners, and database queries and extending to HTML testing and API calls.

For every introduced feature, the initial focus was on validation testing. This step was critical to ensure that each segment functioned correctly and that modifications to one feature did not adversely impact others. Following successful validation, the testing process delved into

defect testing. This phase aimed to confirm that results are accurate and guarantee that any application misuse would not lead to issues and bugs.

As an example, during the implementation of the database feature, thorough testing was conducted to ensure the proper functioning of retrieving and inputting data from the database. Beyond this, the testing procedure extended to checking other tables in the database to confirm that these queries did not inadvertently affect unrelated tables. This comprehensive testing strategy, encompassing HTML, API calls, and various functionalities, played a pivotal role in ensuring the robustness and reliability of the application throughout its development.

4.5 Computing-Based Solution/System Design

The project's development process underwent multiple iterations to refine the application's design, focusing on determining the feasibility and outlining the future user interface (UI). Throughout these iterations, we embraced an object-oriented approach to craft various data objects essential for implementing the required features. Ensuring a well-organized structure in both organization and implementation, we recognized the necessity of a robust design architecture.

Django, the web framework of choice, emphasizes a Model View Template (MVT) structure. This determines the architectural pattern of the project. Django's MVT structure divides the application into three key components: the Model, which represents the data and database interactions; the View, which is responsible for handling user interface logic and rendering; and the Template, which defines the structure and presentation of the UI.

Figure 8

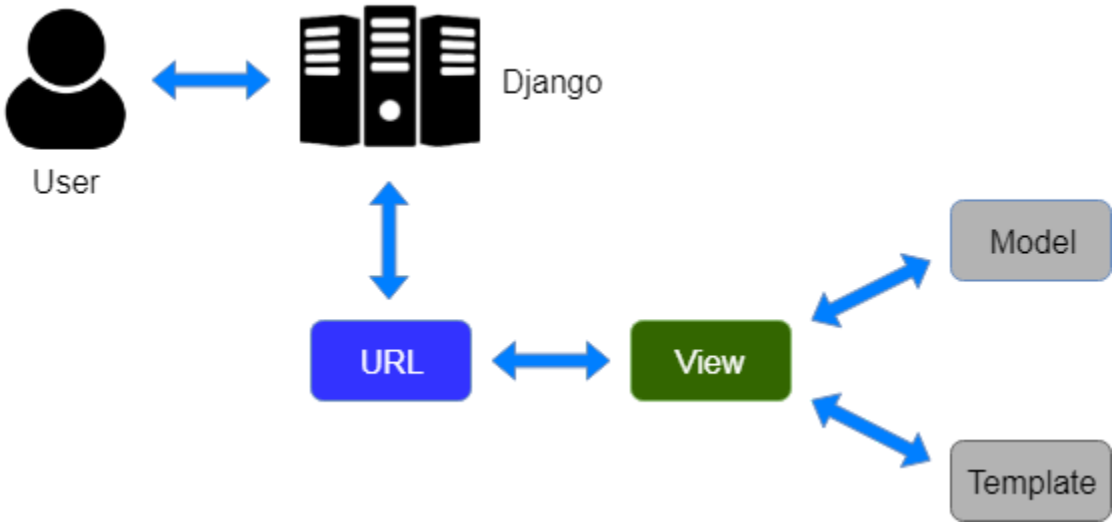


Diagram of a Web Framework Architecture (*Django MVT*, n.d.)

Incorporating Django's MVT architecture ensures a clean and modular design, promoting scalability, maintainability, and readability. The Model component encapsulates the business logic and database interactions, facilitating effective data management. The View handles user interface logic, ensuring seamless interaction between the user and the application, while the Templates define the visual presentation of the UI.

The decision to adopt the MVT architecture aligns with best practices for Django projects, providing a solid foundation for development and enhancing the overall structure and organization of the project. This approach promotes code reusability, ease of maintenance, and a clear separation of concerns, contributing to a more efficient and sustainable project development lifecycle.

Figure 9

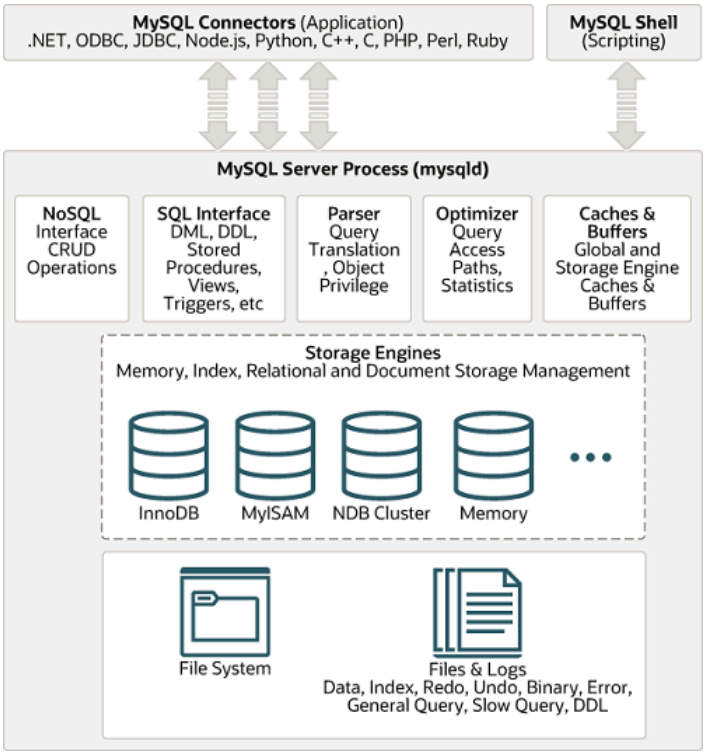


Diagram of the Database Architecture (*Overview of MySQL Storage Engine Architecture*, n.d.)

The MySQL pluggable storage engine architecture offers a versatile solution for a database by allowing the user to access their desired data storage without the need for extra coding at the application level. The system uses a consistent and easy-to-use application model, and the API provides a standardized interface, isolating users from the need to manage the backend of the database.

As illustrated in Figure 9, the MySQL architecture contains pluggable storage engines. This architecture introduces a standard set of management and support services to all storage engines, enabling efficient reading and writing of data at the physical server level. The design is beneficial for tailored application requirements, such as data warehousing, transaction processing, or high availability scenarios, without necessitating significant coding changes (*Overview of MySQL Storage Engine Architecture*, n.d.). Through connector APIs and service

layers, users can interact with the MySQL database, maintaining flexibility for potential changes in storage engines without compromising the consistency of the user interface.

4.6 Implementation and Testing

We initially chose Microsoft Azure for the database but later changed to XAMPP MySQL. MySQL was more suitable for the project because Azure was too heavily commercialized. Other than the database, it also required the management of the server and database subscriptions. MySQL provided all the necessary functions for this project while remaining simple to set up and manage. We can quickly transfer databases between team members who use MySQL. We used simple unit testing in the calls between the code and database to ensure the system receives the expected output. For example, if our database contains a user with the username “JamesBond007”, the code should only have access to the ingredients and preferences associated with that username.

Python was the programming language of choice because it contained all the needed libraries for this project. The initial choice of a platform to run the project’s code was Google Colab. However, it presented many challenges in handling and integrating the needed APIs. It was especially slow and unresponsive to the OpenAI API. We tested both Visual Studio and PyCharm as alternatives, and PyCharm ended up being the IDE of the prototype. PyCharm's ability to easily download dependencies was why we chose it over Visual Studio.

The selection of a language model played a crucial role in the development process. Initially employing GPT-3.5 Turbo, issues arose as the model proved slower than desired. We decided to upgrade to GPT-4, which offered enhanced speed and improved accuracy in generating recipes. However, a challenge arose when the model generated the same output repeatedly when receiving the same prompt, compromising the randomness we expected an AI to

have. Further research led to the discovery of GPT-4's built-in function, "temperature," which determines randomness in the AI's output. We could set it between 0 and 2 in the Python code. 0, the default setting, meant that the AI would have no randomness. 2 was complete randomness, resulting in the AI returning gibberish in our tests. After several query tests, we adjusted this parameter to 1, a balance between the two. Adjusting this parameter proved key to injecting variety into the model's outputs, ensuring our goal of generating unique and new recipes.

We decided to extend the project's scope beyond text-based AI generation to image generation. We chose OpenAI's DALL-E as the subscription and code already included it and its libraries. We initially started using DALL-E 2, an older image generation model, which is the cheaper option. Limitations in image quality and aesthetics were quickly apparent, particularly with the smaller size of images. We upgraded to DALL-E 3, the latest version of the DALL-E series. This shift significantly enhanced the visual appeal of the generated images, aligning them more closely with the project goals.

4.7 Conclusion

This project used many advanced tools to make a working website. We used PyCharm to write the code as it could easily integrate all the required dependencies. We used advanced architectures like Django's project architecture and MySQL's pluggable storage engine architecture. We chose Django to build the website, ensuring smooth communication between the AI API and databases. XAMPP allowed us to set up local MySQL servers on our computers, which helped test the prototype before the final product. ChatGPT-4 and DALL-E 3 were the main components implemented into the system to allow KAMI to generate recipes and give them a visual appeal. We implemented our research and tested KAMI to ensure its functionality and reliability.

5. Conclusion

5.1 Summary

This project focused on enhancing users' experience in the kitchen, whether for home cooks or professional chefs. Home cooks may have limited ingredients and may want ideas for using them. On the contrary, more experienced chefs may have too many ingredients and want to try exploring new dishes. This project is an AI-powered kitchen assistant, which we named KAMI. Through implementing AI, KAMI can assist the user by generating recipes with all the available ingredients while keeping dietary needs in mind. To further convenience the user, they can save their ingredient inventory and preferences to their account. Generating recipes based on only the available ingredients can also reduce food waste and boost productivity by providing practical solutions that cater to the user’s individual needs. We have high hopes for the future development of KAMI, as it has the potential to be a beneficial product in the market.

5.2 Reflections

A semester-long project was no small feat, and it displayed the need for proper planning and adaptability to setbacks. Using the Waterfall model as the development methodology, each team member could work on a specific project component. This methodology allowed each member to specialize in a field. One could focus on the AI API, another on the database, and another on the application interface. This division of work allowed for specialization and efficiency within the team. The use case and entity relationship diagrams, essential components of the project, were developed collaboratively, reflecting the collective effort in creating a clear and comprehensive system design.

This project also proved the importance of a robust research phase. It forced us to adapt early in the timeline, even before the project's development started. For example, when faced

with the realization that building a superior recipe-generating AI might be impractical compared to existing APIs like ChefGPT and DishGen, we quickly shifted towards a kitchen that integrates these APIs instead. This change prevented us from wasting valuable development time and allowed us to avoid unforeseen issues.

Proper collaboration was required to develop this team project. We used Discord for online communication and Google Drive for file sharing. These popular platforms ensured efficient communication and access to the most updated files between team members. Using Discord also allowed us to meet virtually and work together outside of class hours.

In conclusion, our teamwork was instrumental in successfully navigating the complexities of development, from strategic decision-making to the practical implementation of each project component. The synergy of individual contributions within the framework of a well-organized team played a pivotal role in achieving our goal of creating KAMI.

5.3 Future Development

Based on the insights gained from the capstone project, a strategic future development action plan emerges to enhance the KAMI further. Introducing a budgeting system that factors in ingredient prices would empower users to make cost-effective culinary choices. Implementing inventory management features, mainly focusing on ingredient expiration dates, would add a valuable layer of practicality, reducing food waste and ensuring user safety.

Expanding the platform to facilitate recipe sharing among users cultivates a sense of community and encourages culinary creativity. Moreover, tailoring KAMI's functionalities to accommodate business accounts for restaurants and food stalls opens up new possibilities for commercial applications. To support inexperienced cooks, integrating video tutorials and helpful links would provide visual guidance and supplementary resources. This approach ensures

continuous improvement and user satisfaction, positioning the kitchen assistant AI website as an evolving and indispensable tool in the culinary landscape.

The same framework can be replicated to create similar AI-based applications. The utility of the web application can be changed without changing the core system architecture. Our team can modify KAMI to fit the needs for potential future clients, outside of AI recipe generation.

References

AI Won't Replace Humans — But Humans With AI Will Replace Humans Without AI. (2023, August 4). Harvard Business Review. Retrieved December 1, 2023, from https://hbr.org/2023/08/ai-wont-replace-humans-but-humans-with-ai-will-replace-humans-without-ai

Boyanton, M. U.-L. (2023, March 3). *Is ChatGPT Coming for Your Kitchen?* Eater. Retrieved December 1, 2023, from https://www.eater.com/23620766/chatgpt-ai-recipes-versus-chefs-tiktok-who-made-it-better

DALL·E 3. (n.d.). OpenAI. Retrieved December 1, 2023, from https://openai.com/dall-e-3

Davis, M. E., & Phillips, J. A. (2007). *Learning PHP & MySQL: Step-by-Step Guide to Creating Database-Driven Web Sites*. O'Reilly Media.

Difference between MVC and MVT design patterns. (2022, September 14). GeeksforGeeks. Retrieved December 1, 2023, from https://www.geeksforgeeks.org/difference-between-mvc-and-mvt-design-patterns/

Django documentation. (n.d.). Django. Retrieved December 1, 2023, from https://docs.djangoproject.com/en/4.2/

Django MVT. (n.d.). Javatpoint. Retrieved December 1, 2023, from https://www.javatpoint.com/django-mvt

Django Project MVT Structure. (2021, August 16). GeeksforGeeks. Retrieved December 1, 2023, from https://www.geeksforgeeks.org/django-project-mvt-structure/

Food Waste FAQs. (n.d.). USDA. Retrieved December 1, 2023, from https://www.usda.gov/foodwaste/faqs

GPT-4 is OpenAI's most advanced system, producing safer and more useful responses. (2023, March 13). OpenAI. Retrieved December 1, 2023, from https://openai.com/gpt-4

How to Use ChatGPT API in Python? (2023, April 8). GeeksforGeeks. Retrieved December 1, 2023, from https://www.geeksforgeeks.org/how-to-use-chatgpt-api-in-python/

Introduction - OpenAI API. (n.d.). Platform OpenAI. Retrieved December 1, 2023, from https://platform.openai.com/docs/introduction

Kumar, B. (2023, October 11). *Outputting Python To HTML In Django*. Python Guides. Retrieved December 3, 2023, from https://pythonguides.com/outputting-python-to-html-django/

Marcus, G., & Davis, E. (2020). *Rebooting AI: Building Artificial Intelligence We Can Trust*. Knopf Doubleday Publishing Group.

Mettler, A. (2023, May 2). *Food waste in restaurants: What we know — Fourth*. Fourth. Retrieved December 1, 2023, from https://www.fourth.com/article/how-much-food-restaurants-waste

Overview of MySQL Storage Engine Architecture. (n.d.). MySQL. Retrieved December 1, 2023, from https://dev.mysql.com/doc/refman/8.0/en/pluggable-storage-overview.html

Suehring, S. (2002). *MySQL Bible*. Wiley.

Usage statistics of HTML for websites. (n.d.). W3Techs. Retrieved December 1, 2023, from https://w3techs.com/technologies/details/ml-html_any

Beginner's Guide to Agile Project Management. (2022, March 18). Adobe Experience Cloud. Retrieved December 2, 2023, from https://business.adobe.com/blog/basics/agile

Wicks, L. (2022, December 7). *Why Cooking—No Matter the Recipe—Is Better for Your Health*. EatingWell. Retrieved December 1, 2023, from

https://www.eatingwell.com/article/291719/why-cooking-no-matter-the-recipe-is-better-for-your-health/

Zeballos, E. (2020, April 6). *More Americans Spend More Time in Food-Related Activities Than a Decade Ago*. USDA ERS. Retrieved December 1, 2023, from https://www.ers.usda.gov/amber-waves/2020/april/more-americans-spend-more-time-in-food-related-activities-than-a-decade-ago/

Acronyms

AI - Artificial Intelligence

API - Application Programming Interface

ChatGPT - Chat Generative Pre-Trained Transformer

CSS - Cascading Style Sheets

HTML - HyperText Markup Language

KAMI - Kitchen Assistant and Meal Innovator

MVT - Model View Template

MySQL - My Structured Query Language

TDD - Test-Driven Development

UI - User Interface

USDA - U.S. Department of Agriculture

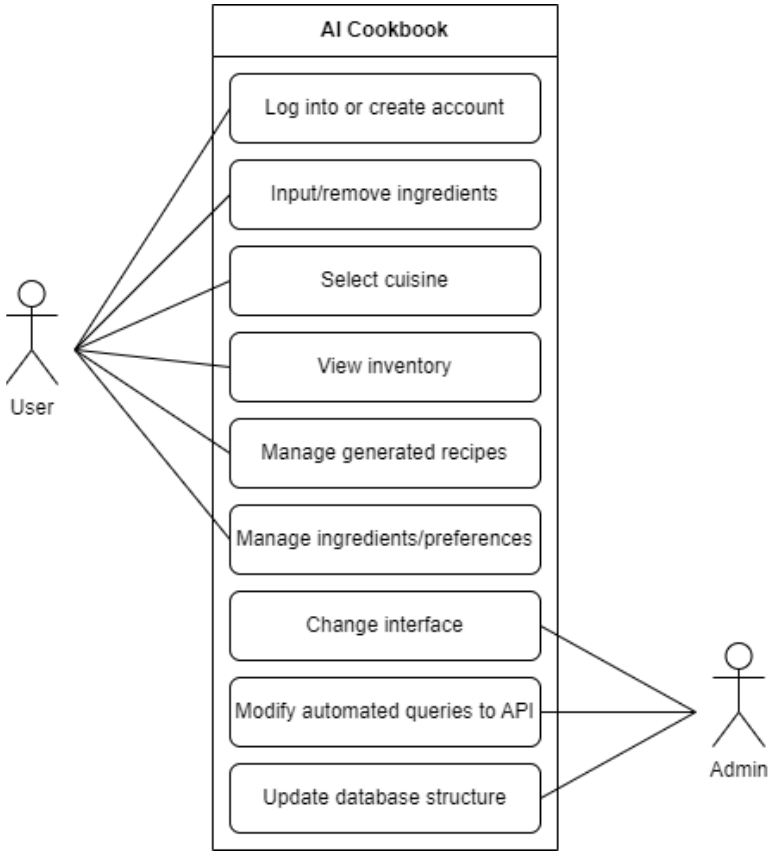
XAMPP - Cross-Platform, Apache, MySQL, PHP, and Perl

List of Figures

Gantt Chart of the Project

Task Name	Sep				Oct				Nov				Dec			
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
Analysis Phase																
Create requirements			Justin													
Planning				Noah/Stephen												
Research																
Research API						Stephen										
Research Database						Justin										
Research Web Development						Noah										
Implementation																
Test API in program							Stephen									
Create MySQL inventory							Justin									
Create HTML									Justin							
Combine API and inventory									Stephen							
Create Django Framework									Noah							
Finalize Prototype											All					

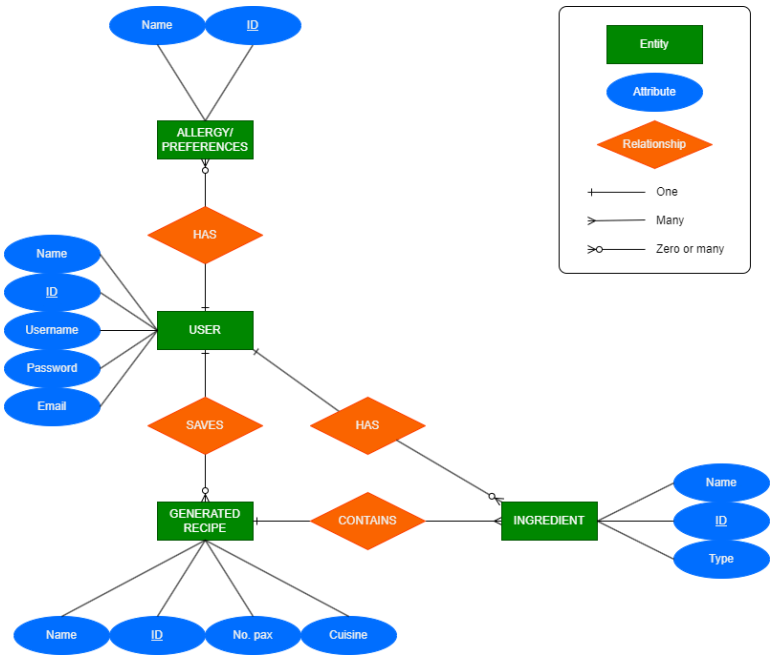
Use Case Diagram of the System



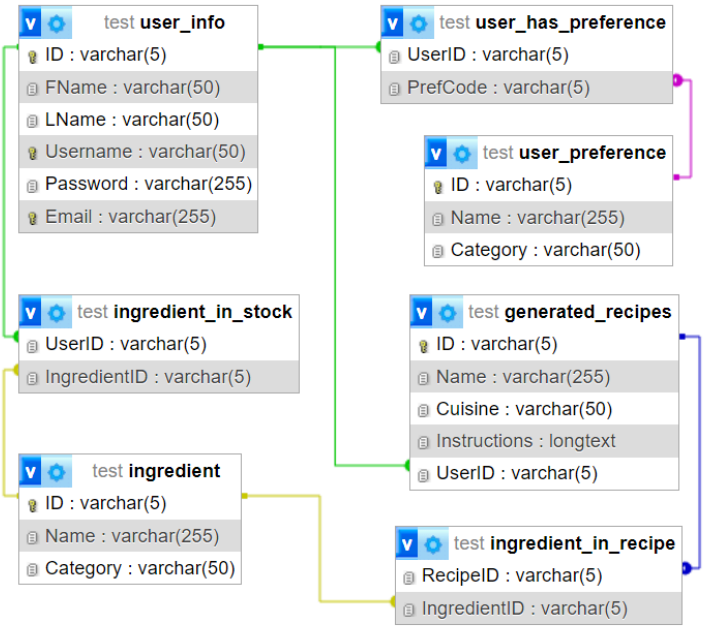
Use Case Description for Logging into or Creating a User Account

Use Case Description		
Use Case Name		Log into or Create Account
Actors		User and device of choice
Preconditions		The user must have a device, be connected to Wi-Fi, and access to the website. If the user is wants to login, they must have created an account.
Normal Flow	Description	If the user is creating an account, the user must select the option, below the login form, of "Create an Account". Then they must enter their name, email, and desired username and password. Then they will select create account. If the user wants to login, they will enter their email or username and password, then select login.
	Postconditions	If the user is creating an account, it will print out "Thank you for creating an account!", then login the user. If the user just logins, then it will pull their ingredients and preferences from their account and give them options about creating recipes.
Alternative flows and expectations		The user may enter an invalid email, password, or username. The user may try to enter a email or username that is already in use when they are creating an account. A user forgetting their password.
Nonfunctional requirements		The user must remember their password and email they used. They must know their available ingredients and preferences in order to enter them into the system.

Early ER Diagram Prototype of the Database



Final 3NF ER Diagram of the Database



AI-Generated Images during Testing

Screenshot of the Prototype Website

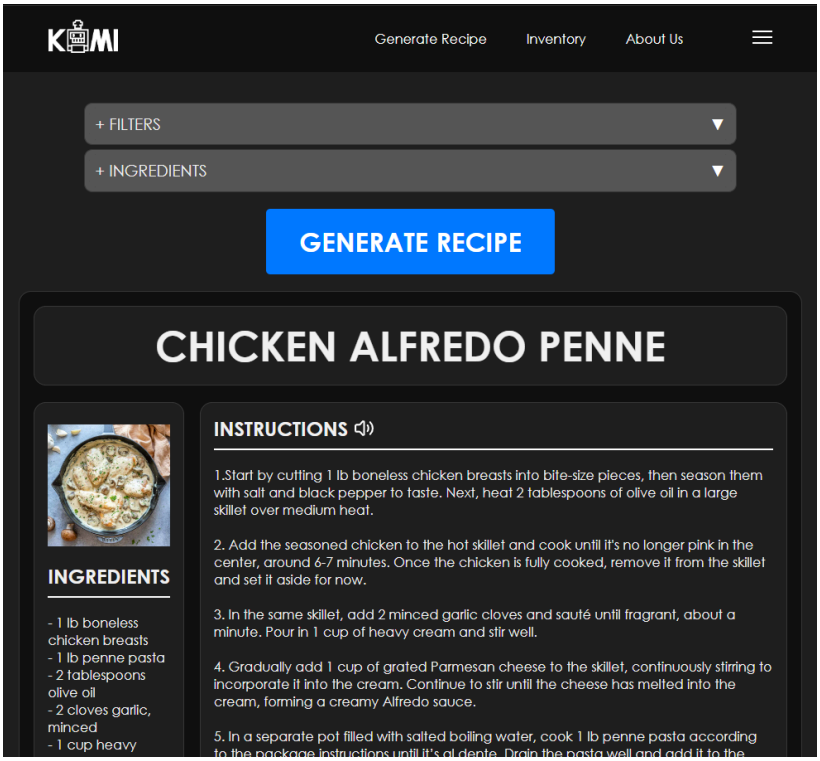


Diagram of a Web Framework Architecture (*Django MVT*, n.d.)

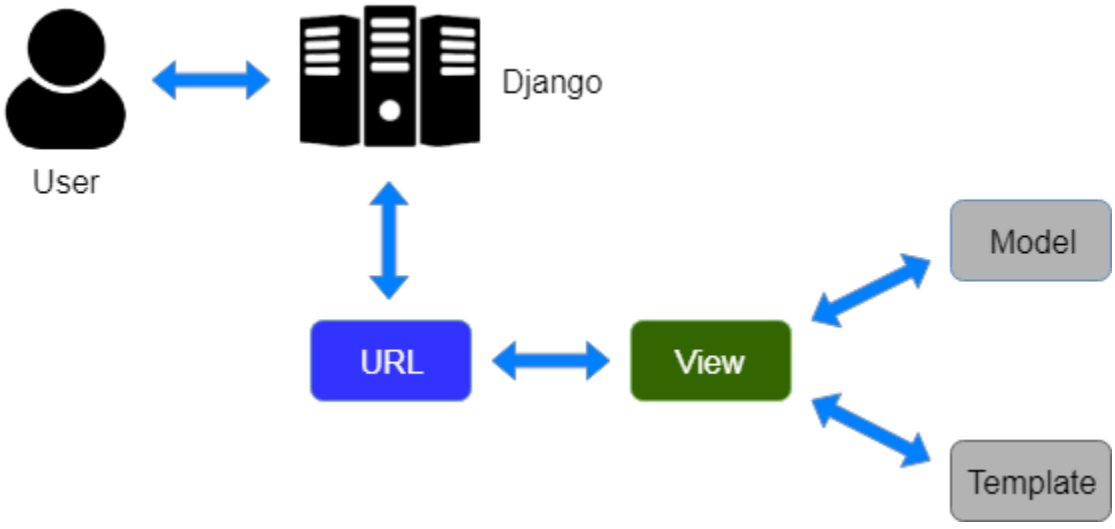


Diagram of the Database Architecture (*Overview of MySQL Storage Engine Architecture*, n.d.)