RECIPES RECOMMENDATION SYSTEM BASED ON INGREDIENTS AVAILABILITY AND USER PREFERENCES

THESIS

Submitted as a Requirement for the Diploma IV Examination

State Polytechnic of Malang

By:

NABILAH ARGYANTI ARDYNINGRUM. NIM. 1941720083

DEPARTMENT OF INFORMATICS ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGY

STATE POLYTECHNIC OF MALANG

JULY 2023

RECIPES RECOMMENDATION SYSTEM BASED ON INGREDIENTS AVAILABILITY AND USER PREFERENCES

THESIS

Submitted as a Requirement for the Diploma IV Examination

State Polytechnic of Malang

By:

NABILAH ARGYANTI ARDYNINGRUM. NIM. 1941720083

DEPARTMENT OF INFORMATICS ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGY

STATE POLYTECHNIC OF MALANG

JULY 2023

APPROVAL PAGE

RECIPE RECOMMENDATION SYSTEM BASED ON INGREDIENT AVAILABILITY AND USER PREFERENCES

Prepared by:

NABILAH ARGYANTI ARDYNINGRUM. NIM. 1941720083

This Final Report has been examined on July 31, 2023

Approved by:

1.	Supervisor	: Yan Watequlis Syaifudin, S.T., M.M.T. Ph.D. NIP. 19810105 200501 1 005	.....
2.	Co-supervisor	: Dwi Puspitasari, S.Kom., M.Kom. NIP. 19791115 200501 2 002	.....
3.	Principal Examiner	: Meyti Eka Apriyani S.T., M.T. NIP. 19870424 201903 2 017	.....
4.	Assistant Examiner	: Mamluatul Hani'ah, S.Kom, M.Kom. NIP. 19900206 201903 2 013	.....

Acknowledged by,

Head of Major Information Technology	Head of Study Program Informatics Engineering
Dr. Eng. Rosa Andrie Asmara, S.T., M.T.	Dr. Ely Setyo Astuti, S.T., M.T.
NIP. 19801010 200501 1 001	NIP. 19760515 200912 2 001

STATEMENT PAGE

I hereby declare that this Thesis contains no work, either in whole or in part, that has been previously submitted for an academic degree at any other Higher Education Institution. To the best of my knowledge, there is also no work or opinions written or published by others, except those cited in the text and mentioned in the list of citations/references.

Malang, July 31, 2023

Nabilah Argyanti Ardyningrum.

ABSTRAK

Ardyningrum, Nabilah A.. "Sistem Rekomendasi Resep Masakan Berbasis Persediaan Bahan dan Preferensi Pengguna". **Pembimbing: (1) Yan Watequlis Syaifudin, S.T., M.M.T. Ph.D., (2) Dwi Puspitasari, S.Kom., M.Kom.**

Skripsi, Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang, 2023.

Memasak merupakan kegiatan penting dan menyenangkan yang memungkinkan individu menyajikan hidangan lezat. Namun, menemukan resep yang cocok dengan bahan yang ada dan preferensi pribadi sering menjadi tantangan. Untuk mengatasi ini, diperkenalkan sebuah sistem rekomendasi yang menggunakan pendekatan *content-based filtering* dengan *Jaccard similarity* untuk memanfaatkan bahan yang dimiliki pengguna. Selain itu, pendekatan *item-based collaborative filtering* dengan *cosine similarity* digunakan untuk merekomendasikan resep berdasarkan preferensi pengguna dan pengguna lain. Pendekatan ini menghasilkan rekomendasi resep yang akurat dan relevan, dengan memaksimalkan pemanfaatan bahan-bahan dan meningkatkan pengalaman memasak. Solusi optimal untuk rekomendasi berdasarkan bahan ditemukan dengan menggabungkan persentase kepemilikan bahan dan *Jaccard similarity*. Evaluasi rekomendasi berdasarkan preferensi pengguna juga mengungkapkan variasi hasil, dengan tingkat akurasi mencapai 70.76%.

Kata Kunci: *Content-Based Filtering, Cosine Similarity, Item-Based Collaborative Filtering, Jaccard Similarity, Recommender System*

ABSTRACT

Ardyningrum, Nabilah A.. *"Recipes Recommendation System Based on Ingredient Availability and User Preferences".* ***Supervisor: Yan Watequlis Syaifudin, S.T., M.M.T. Ph.D., Co-Supervisor: Dwi Puspitasari, S.Kom., M.Kom.***

Thesis, Informatics Engineering Study Program, Department of Information Technology, State Polytechnic of Malang, 2023.

Cooking is an important and enjoyable activity that allows individuals to present delicious dishes. However, finding recipes that match the available ingredients and personal preferences can be a challenge. To address this, an introduced recommendation system utilizes a content-based filtering approach with Jaccard similarity to leverage the ingredients owned by users. Additionally, an item-based collaborative filtering approach using cosine similarity is employed to suggest recipes based on user preferences and those of other users. This approach yields accurate and relevant recipe recommendations, maximizing ingredient utilization and enhancing the cooking experience. The optimal solution for ingredient-based recommendations is achieved by combining ingredient ownership percentage and Jaccard similarity. Evaluation of recommendations based on user preferences also reveals varying outcomes, with an accuracy rate reaching 70.76%.

Keywords: *Content-Based Filtering, Cosine Similarity, Item-Based Collaborative Filtering, Jaccard Similarity, Recommender System*

FOREWORD

Praise and gratitude are directed to the presence of Allah SWT for all His blessings and guidance, as the author completes the thesis entitled "RECIPE RECOMMENDATION SYSTEM BASED ON INGREDIENT AVAILABILITY AND USER PREFERENCES". This thesis is composed by the author as a requirement to complete the Diploma IV program of the Informatics Engineering Study Program, Department of Information Technology, State Polytechnic of Malang.

We realize that without the support and cooperation from various parties, this final report activity would not have been successful. Therefore, we would like to express our gratitude to:

1. Mr. Dr. Eng. Rosa Andrie Asmara, S.T., M.T., as the Head of the Department of Information Technology
2. Mrs. Ely Setyo Astuti, S.T., M.T., as the Head of the DIV Informatics Engineering Study Program
3. Mr. Yan Watequlis Syaifudin, S.T., M.M.T. Ph.D., as the supervisor
4. Mrs. Dwi Puspitasari, S.Kom., M.Kom., as the co-supervisor
5. My family who has consistently provided unwavering support
6. My friends, namely members of the Sesi Terapi Cringe, Abdul Cooking Class members, intern colleagues at PT Adma Digital Solusi, class TI-4H, and other friends who have provided technical and emotional support
7. And all parties who have assisted and supported the smooth completion of this Final Report, from start to finish, that we cannot mention one by one.

The author acknowledges that in the preparation of this final report, there are still many shortcomings and weaknesses in both writing systematics and language use. Therefore, the author hopes for constructive suggestions and criticisms from various parties for the improvement of this report. May this report be useful for readers in general and especially for the author. In conclusion, the author extends sincere thanks.

Malang, July 31, 2023

Author

TABLE OF CONTENTS

Pages

LIST OF FIGURES

Pages

LIST OF TABLES

LIST OF APPENDICES

CHAPTER I. INTRODUCTION

1.1 Background

Cooking is an essential activity in human daily life. Every day, people cook to meet their nutritional needs and create delicious and nutritious dishes. Moreover, cooking is necessary to transform raw ingredients into food that is easier and safer to consume. The cooking process helps eliminate bacteria and potential pathogens in food, thereby enhancing the safety and quality of the food consumed by humans (Doyle & Erickson, 2008).

During the cooking process, various ingredients are processed using different cooking tools and techniques, resulting in a variety of flavors and appearances of dishes. Cooking utensils such as pans, pots, ovens, blenders, and others play a crucial role in transforming ingredients into delicious and nutritious dishes. The use of appropriate cooking techniques, such as frying, boiling, steaming, or baking, produces different effects on the ingredients. In this process, people have the opportunity to be creative and explore flavors, ultimately contributing to their taste preferences and satisfaction with the food they prepare (Zhou et al., 2014).

Each individual has a unique palate influenced by various factors, including changes in the surrounding environment, physical needs, lifestyle, and personal experiences (Montanari, 2006). Some individuals tend to prefer foods with specific flavors, such as a preference for spicy, sweet, sour, or savory dishes (Anzman-Frasca et al., 2018). These food preferences often also resemble those of other individuals living in the same environment or sharing similar flavor preferences (Montanari, 2006).

When someone intends to consume food, they usually strive to find dishes that match the availability of ingredients, kitchen equipment, and their food preferences. Additionally, considerations about the shelf life of ingredients are important to avoid spoilage. Various methods are used in the search for suitable menus, including looking up recipes that meet these criteria. However, the limited information often found in these recipes often poses a challenge. As a result, individuals often must personally evaluate whether food choices are suitable for their situation and preferences. Moreover, the level of complexity in choosing

suitable food is further complicated by the fact that many dishes share similar ingredients. Individual appetites, the surrounding environment, and the influence of others also impact choices, creating uncertainty in decision-making. All these phenomena result in a less efficient and more confusing food search process.

To address these issues, several researchers have developed recipe recommendation systems to assist individuals in selecting suitable recipes to cook. For instance, a research team composed of Shilpa Chaudhari, Aparna R., Vinay G Tekkur, Pavan G L., and Shreekanth R Karki has developed an Ingredient/Recipe Algorithm using Web Mining and Web Scraping for Smart Chef. Through this system, recipe recommendations can be provided based on the ingredients entered the search column. However, in its implementation, this research requires users to repeatedly input these ingredients when searching for suitable recipes, ultimately consuming a significant amount of time (Chaudhari et al., 2020).

Another group of researchers, Ifeoma Adaji, Czarina Sharmaine, Simone Debrowney, Kiemute Oyibo, and Julita Vassileva, developed a Personality Based Recipe Recommendation Using Recipe Network Graphs. In their study, they constructed a recipe network graph based on users' personalities. They also tested the relationship between food types (e.g., meat or vegan) and five personality traits (openness, conscientiousness, extraversion, agreeableness, and neuroticism). The experimental results indicated that recipe categories based on food types did not significantly affect the assessors' personalities. The study also concluded that differences in extraversion and agreeableness personality traits did not significantly influence food preferences. Therefore, it can be inferred that individual personalities do not always have a significant impact on food preferences (Adaji et al., 2018).

In the context of the identified issues and based on the results of previous research studies, a recipe recommendation system will be developed with the goal of meeting users' needs. This system will assist users in finding recipes that match the ingredients they have available, enabling them to utilize ingredients more efficiently. Furthermore, the system will recommend other recipes that align with users' preferences or tastes, considering the food preferences of other users as well.

The proposed approach in developing this system involves the application of content-based filtering using Jaccard similarity. This method allows for the comparison of elements between two data sets, focusing on the presence of elements in both sets. Each element is examined, and a comparison is made by calculating matching elements. The similarity score is then generated by dividing the number of matched elements by the total number of elements in one data set (Yan et al., 2017). In the development of this system, the Jaccard similarity method is specifically used to compare the ingredients owned by users with the ingredients in the recipes. With this approach, the system can generate a score that reflects the extent of similarity between the ingredients owned by users and the ingredients in the recipes, which is ultimately used to recommend recipes that match the ingredients available to users.

Furthermore, the system adopts an item-based collaborative filtering approach using the cosine similarity method. The choice of cosine similarity as a suitable method for finding similarities in binary data is based on its ability to measure vector similarity without considering vector length. This indicates that cosine similarity remains valid even when binary vector lengths differ. An additional advantage of cosine similarity is its normalized characteristic, which is unaffected by vector size (Chakrabartty, 2022). This characteristic is highly beneficial in the context of binary data, as used in this research, where calculations are used to compare user recipe preferences with recipes favored by everyone. In this calculation, a value of 1 indicates a favored recipe, while a value of 0 indicates a disliked recipe. Through this interpretation, the result of the cosine similarity calculation will be a set of recipe recommendations tailored to the user's level of preference, based on the preferences of all users.

By adopting content-based filtering and item-based collaborative filtering approaches, this recommendation system is expected to provide relevant recipe recommendations based on the available ingredients and user taste preferences, thereby making the cooking process more efficient and enjoyable. This recommendation system is also anticipated to offer a practical solution for users who are uncertain about selecting food menus that match their circumstances and

preferences, making it easier for them to find inspiration to try new dishes that suit their personal preferences.

1.2 Problem Formulation

Based on the background, the problem formulation in this research includes:

1. How to provide a recipe recommendation system based on ingredient availability and user preferences?
2. How to implement the content-based filtering approach to recommend recipes with available ingredients?
3. How to implement the item-based collaborative filtering approach to recommend recipes based on user preferences?

1.3 Objectives

The objectives of creating the final project titled "**RECIPE RECOMMENDATION SYSTEM BASED ON INGREDIENT AVAILABILITY AND USER PREFERENCES**" are:

- Create a recipe recommendation system that provides relevant recommendations based on the ingredients available to the user, as well as aligned with user preferences.
- Implement the content-based filtering approach to recommend recipes based on the user's available ingredients, allowing the user to optimize the use of available food items.
- Apply the item-based collaborative filtering approach to find recipes relevant to those previously liked by the user, thereby recommending more personalized recipes that match the user's taste.

1.4 Problem Limitations

Problem limitations need to be established to define the scope of the research. The problem limitations that will be applied to the Recipe Recommendation System Based on Ingredient Availability and User Preferences are:

1. Recipe data is sourced from an Indonesian language recipe website, namely https://www.masakapahariini.com/.

2. Recommendations based on ingredient availability will use the content-based filtering approach with Jaccard similarity and will be recommended based on ingredient names, quantities, and units, following predetermined rules set by the developer.
3. Recommendations based on user preferences will use the item-based collaborative filtering approach with cosine similarity, aiming to recommend recipes that are similar to those previously liked by the user.
4. The developed system will be in the form of a website displaying a list of recommended recipes along with the completeness percentage of each recipe.
5. The system will display the top 12 recommended recipes based on ingredient availability and the top 12 recommended recipes based on user preferences in the form of a recipe list on the website page.

1.5 Report Structure

The description in this thesis report is organized with the following structure.

CHAPTER I INTRODUCTION

In this chapter, the general aspects are explained, including the background, problem formulation, problem limitations, research objectives, and the structure of the report.

CHAPTER II THEORETICAL FOUNDATIONS

This chapter contains the theories underlying and related to the planning and development of the application used to facilitate understanding and problem-solving.

CHAPTER III RESEARCH METHODOLOGY

In this chapter, the selected method, techniques, appropriate procedures, and tools used are explained so that each research phase can be carried out accurately.

CHAPTER IV ANALYSIS AND DESIGN

This chapter discusses the general design as well as further descriptions of the system design in software development. The description of this system design includes design methods, system requirement analysis, data design, process

design outlining how the processes work with specific procedures, and system interface design.

CHAPTER V IMPLEMENTATION AND TESTING

In this chapter, a description is given of the process of implementing the previously designed system into a programming language, accompanied by an overview of the application interface.

CHAPTER VI RESULTS AND DISCUSSION

This chapter discusses the research results from the testing process of the developed system.

CHAPTER VII CONCLUSION AND RECOMMENDATIONS

This chapter contains conclusions drawn from the testing of the developed system and recommendations that are needed.

CHAPTER II. THEORETICAL FOUNDATION

2.1 Previous Research Studies

In order to address the challenges faced in this research, several references from previous studies were utilized. These references were used to help define relevant problem limitations and to select appropriate methods to be applied in this research.

The first study was conducted by Shilpa Chaudhari, Aparna R., Vinay G Tekkur, Pavan G L., and Shreekanth R Karki, titled "Ingredient/Recipe Algorithm using Web Mining and Web Scraping for Smart Chef." In this research, they developed a recipe recommendation system that utilized web scraping techniques to assist users in searching for recipe recommendations based on the ingredients entered as search keywords (Chaudhari et al., 2020).

Additionally, Ifeoma Adaji, Czarina Sharmaine, Simone Debrowney, Kiemute Oyibo, and Julita Vassileva conducted a study titled "Personality Based Recipe Recommendation Using Recipe Network Graphs." This research proposed a recipe recommendation system based on users' personality types. The results of this study showed that users with similar personalities tended to have similar preferences for recipes, although some personality types did not exhibit significant differences. Furthermore, the recipe ingredient categories, such as vegan or meat-based, did not influence the users' personality types (Adaji et al., 2018).

Nilesh, Madhu Kumari, Pritom Hazarika, and Vishal Raman also contributed with a study titled "Recommendation of Indian Cuisine Recipes Based on Ingredients." In this research, they proposed a recipe recommendation system for Indian cuisine using web scraping and content-based methods. The system provided recommendations based on ingredients in recipes that shared similarities with other recipes (Nilesh et al., 2019).

2.2 Theoretical Basis

2.2.1 Cooking with Recipes

Cooking from a recipe involves steps to create a delicious and satisfying dish. Cooking with a recipe can start by gathering various elements such as ingredients,

equipment, and preferences. Next, a recipe can be searched for based on these elements. Once a recipe is chosen, it can be implemented using the listed ingredients and equipment. After the cooking process is completed, the food is ready to be enjoyed (Kadowaki et al., 2014).

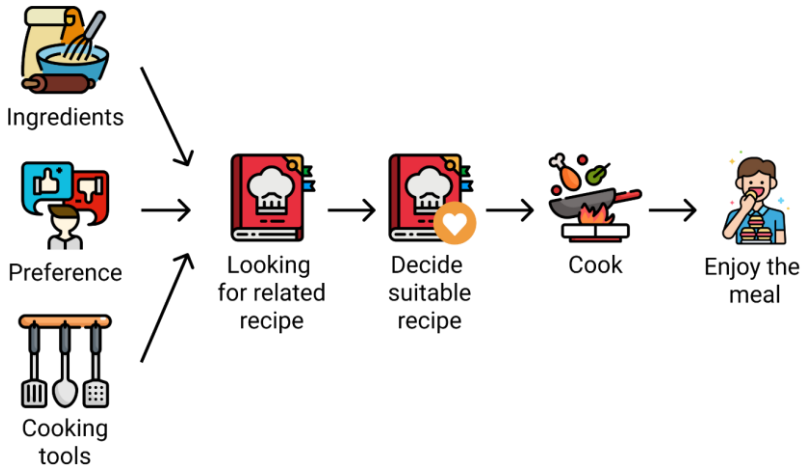


Figure 2.1 Steps in cooking with a recipe

The process of searching for a recipe requires a significant amount of time. The presence of similar recipes necessitates individuals to consider various factors to find a suitable one. Some individuals tend to consider the available ingredients before searching for a recipe, while others prioritize their personal preferences. This highlights the complexity of finding recipes that align with the needs and tastes of each individual (Tobey et al., 2019).

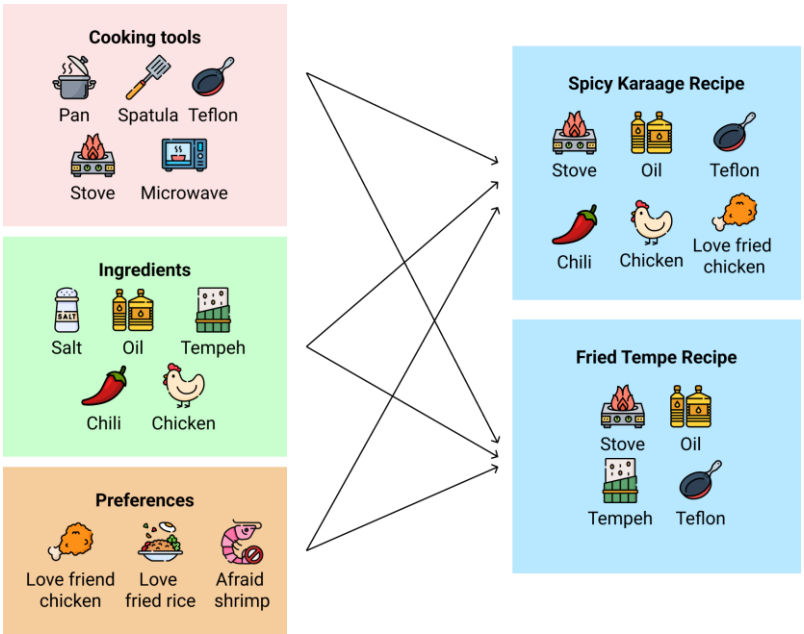


Figure 2.2 Considerations for recipe compatibility

2.2.2 Recipe Structure

The structure of a recipe is crucial in the culinary world, as it serves as a guide for food preparers throughout the cooking process. Generally, a recipe consists of several main components. First, there is a title that describes the dish to be prepared. Next, the ingredient section provides a list of required components with accurate measurements. Instructions form the core of the recipe, offering step-by-step guidance from preparation to serving, including cooking techniques and serving suggestions. Understanding the structure of a recipe allows individuals to create balanced and delicious dishes (Kadowaki et al., 2014).

2.2.3 Web Scraping

Web scraping is a technique for automatically extracting information from web pages using computer programs. This process allows access to and extraction of data present in the HTML structure of websites. Through web scraping, data from various sources can be collected quickly and accurately (Lawson, 2015).

One of the benefits of web scraping is its ability to store extracted data in a file or database. This enables users to have direct access and store data in a format that is easily accessible and analyzable for further purposes (Lawson, 2015).

When performing web scraping, a request is sent to the server, and the server responds with the required data. The data in the response is then parsed and extracted according to the needed information. This parsing technique allows the system to recognize and extract important parts of the web page, such as text, tables, images, and more (Zhao, 2017).

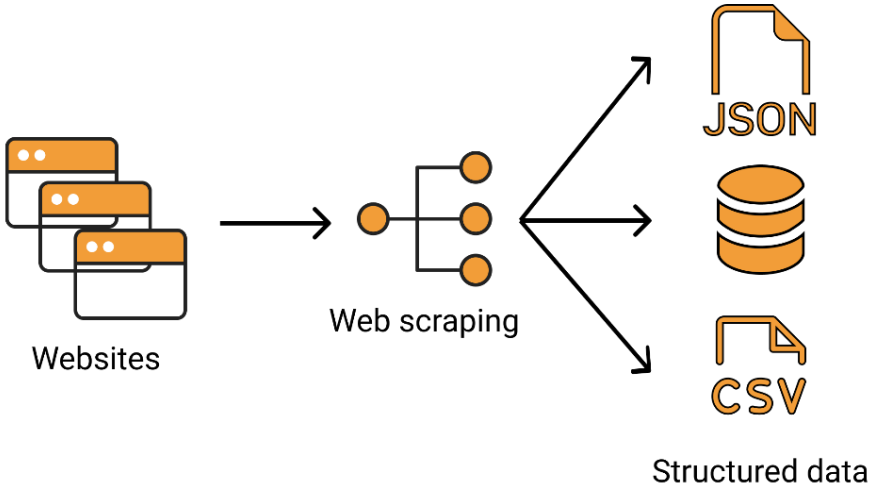


Figure 2.3 Web scraping process

Web scraping plays a crucial role in obtaining data from various sources for analysis, research, or business purposes (Hillen, 2019).The process is carried out automatically and efficiently, saving time and effort in manually collecting data. Successfully accessed and extracted data through web scraping can become a valuable source of information that supports better decision-making in various fields.

2.2.4 Text Preprocessing

Text preprocessing is the process of selecting and transforming raw text into a more structured and understandable form. Its purpose is to prepare text data for various text analysis tasks, thereby enhancing the performance and accuracy of analysis results (Bhujade & Janwe, 2011).

Several processes in text preprocessing include:

2.2.2.1 Case Folding

Case folding is the process of converting all letters in the text to lowercase or uppercase. The goal of case folding is to avoid unnecessary distinctions in text processing, allowing words that are essentially the same to be recognized correctly regardless of case differences (Manning et al., 2008).

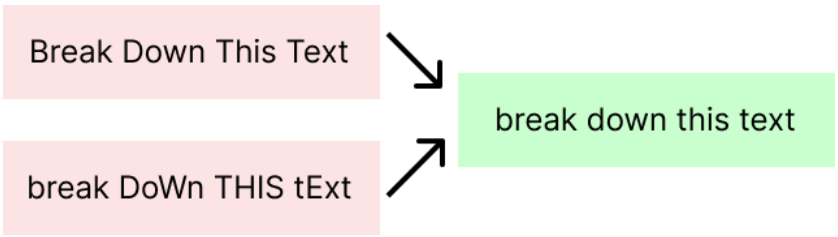


Figure 2.4 Example of lowercasing in case folding

2.2.2.2 Tokenization

Tokenization is the process of breaking down text into smaller units called tokens. Tokens can be words, phrases, sentences, or characters. Tokenization helps transform text into a more structured form, facilitating further processing (Manning et al., 2008).

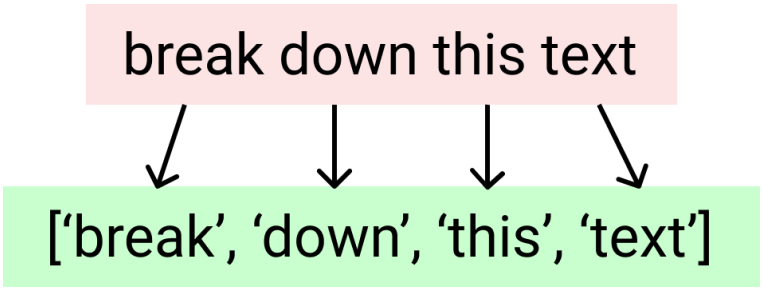


Figure 2.5 Example of tokenization

2.2.2.3 Filtering

In the context of text preprocessing, filtering refers to the process of removing or filtering out irrelevant or unwanted elements from the text. This can involve removing special characters, punctuation, numbers, emoticons, or text that lacks significant semantic meaning (Bhujade & Janwe, 2011). The goal of filtering is to reduce data dimensions and enhance processing efficiency, focusing only on important information for analysis.

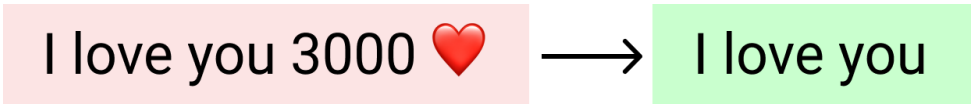


Figure 2.6 Example of filtering text

2.2.5 Recommender System

A recommender system is a technology used to provide recommendations or suggestions to users about products, services, or content that may be of interest or relevance to them. The goal of a recommender system is to enhance the user experience by presenting content that aligns with the user's preferences, needs, or behaviors. By understanding user preferences and analyzing previous user interactions, a recommender system can deliver personalized recommendations to each user (Burke et al., 2011).

There are several types of algorithms used in recommender systems, including:

2.2.3.1 Content-Based Filtering

Content-based filtering is a type of algorithm in a recommender system that provides recommendations based on the analysis of the content or attributes of items to be recommended. This method assumes that user preferences can be predicted

based on the similarity or suitability between the attributes of liked items and other items (Burke et al., 2011).

The process of content-based filtering begins by analyzing the attributes of items that are already known or liked by the user. These attributes can be features that describe the item, such as movie genres, book titles, song artists, or product descriptions. The algorithm then matches the attributes of the known items with attributes of other items (Shankar et al., 2020).

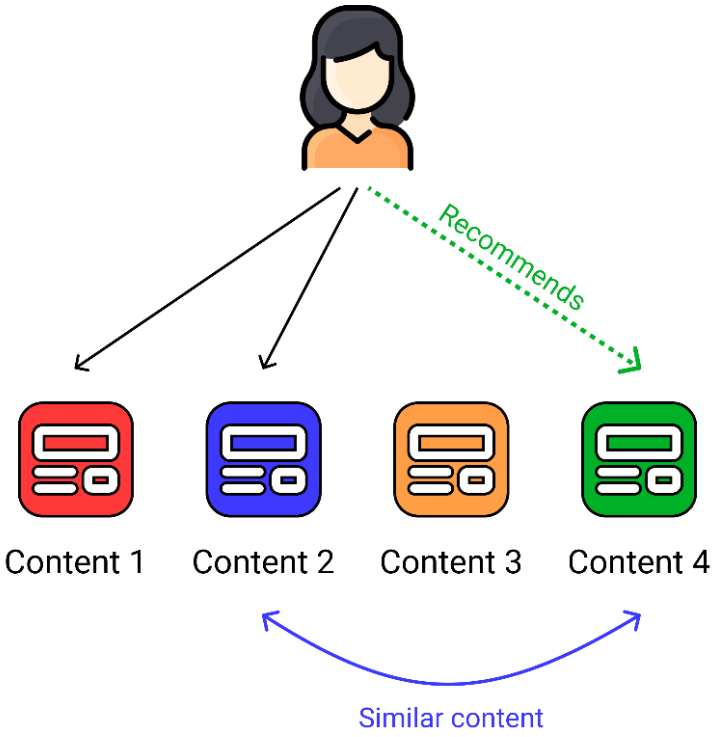


Figure 2.7 Content-based filtering illustration

2.2.3.2 Collaborative Filtering

Collaborative Filtering is a type of algorithm in a recommender system that provides recommendations based on the analysis of behavioral patterns and preferences of a group of users (Shankar et al., 2020). This method assumes that users who have similar preferences or behaviors in the past are likely to have similar preferences in the future.

The process of collaborative filtering begins by analyzing the interaction data between users and items. This data includes information about what users have liked or rated, such as purchased products, watched movies, or listened-to songs. Based on this interaction data, collaborative filtering algorithms seek similarities or matches between users and items (Shankar et al., 2020).

There are two common types of collaborative filtering:

2.4.2.1 User-based collaborative filtering

This method looks for other users with similar behavior patterns to a particular user. If users A and B have similar preferences or interactions with a certain item, the algorithm will recommend items liked by user B to user A and vice versa (Shankar et al., 2020).

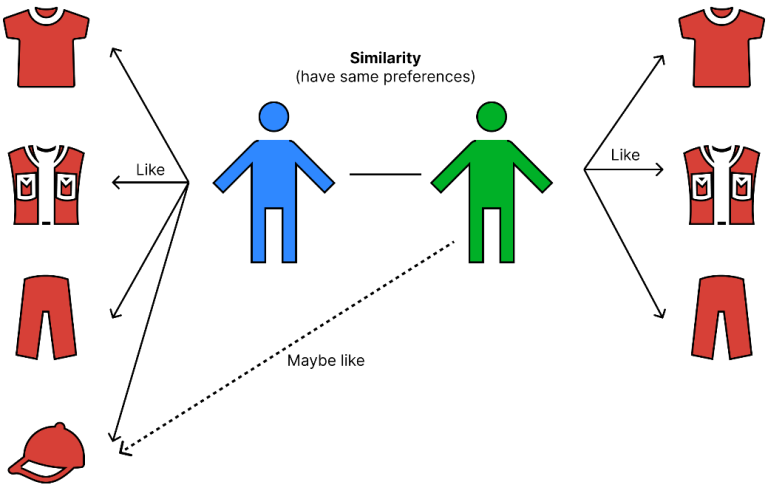


Figure 2.8 User-based collaborative filtering illustration

2.4.2.2 Item-based collaborative filtering

Item-Based Collaborative Filtering

This method looks for other items that have similarity or similarity in interaction patterns with a specific user. If user A likes item X, and item X is similar to item Y based on interaction patterns, the algorithm will recommend item Y to user A (Shankar et al., 2020).

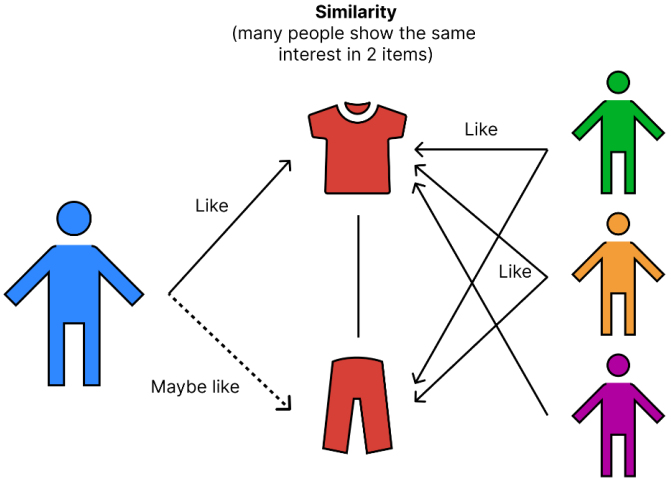


Figure 2.9 Item-based collaborative filtering illustration

2.2.6 Jaccard Similarity

Jaccard similarity is a metric for measuring the similarity between two sets. Jaccard similarity can be used to compare the elements of two sets and calculate how similar the two sets are based on the percentage of shared or similar elements between them (Yan et al., 2017).

The Jaccard similarity formula is defined as the number of elements in the intersection of the two sets divided by the number of elements in either one or both sets (union). Mathematically, the Jaccard similarity between two sets A and B can be calculated using the following formula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Di mana:

- $|A \cap B|$ is the number of elements in the intersection of the two sets.
- $|A \cup B|$ is the number of elements in either one or both sets (union).

The result of this calculation will be in the range of 0 to 1. The closer the value is to 1, the more similar the two sets are. If the Jaccard similarity value is 0, it means the two sets have no shared elements (Grygorian & Iacob, 2018).

Jaccard similarity is commonly used in various fields, especially in text mining and recommendation systems. In text mining, this metric compares the similarity between two documents or texts based on the sets of words they contain. Meanwhile, in recommendation systems, Jaccard similarity is used to recommend similar items based on their attributes. By using Jaccard similarity, both fields can gain insights into similarity and provide recommendations or analysis results that are more relevant to user preferences or the context being studied (Ertl, 2019).

2.2.7 Cosine Similarity

Cosine similarity is a method for measuring the similarity between two vectors in a multidimensional space. This method calculates the angle between two non-zero vectors and produces a value within the range of -1 to 1. If the cosine similarity value approaches 1, then the two vectors have a high similarity based on their vector directions. Conversely, if it approaches -1, the vectors are in opposite directions. The cosine similarity value approaches 0 if the vectors are nearly perpendicular or have little similarity (Lahitani, 2022).

The Cosine Similarity formula between two vectors A and B can be calculated using the dot product and the magnitudes of the vectors:

$$cos(\theta) = \frac{A \cdot B}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Where:

- $A \cdot B$ is the result of the dot product between vectors A and B
- $\| A \| \| B \|$ is the magnitude (length) of each vector A and B

Cosine similarity has a wide range of applications, especially in text analysis and recommendation systems. In text analysis, this method is used to compare the similarity between two documents or texts by measuring the vector representation of words or features within them, enabling the identification of similar documents based on the vocabulary used. Meanwhile, in recommendation systems, cosine similarity is used to recommend items similar to a user's preferences. By representing user and item data as feature vectors or attributes, the cosine similarity method measures the similarity between user preferences and the attributes possessed by items. This allows the recommendation system to provide recommendations based on items that have a high cosine similarity value with the user's preferences (Dangeti, 2017).

CHAPTER III. RESEARCH METHODOLOGY

3.1 Research Phases

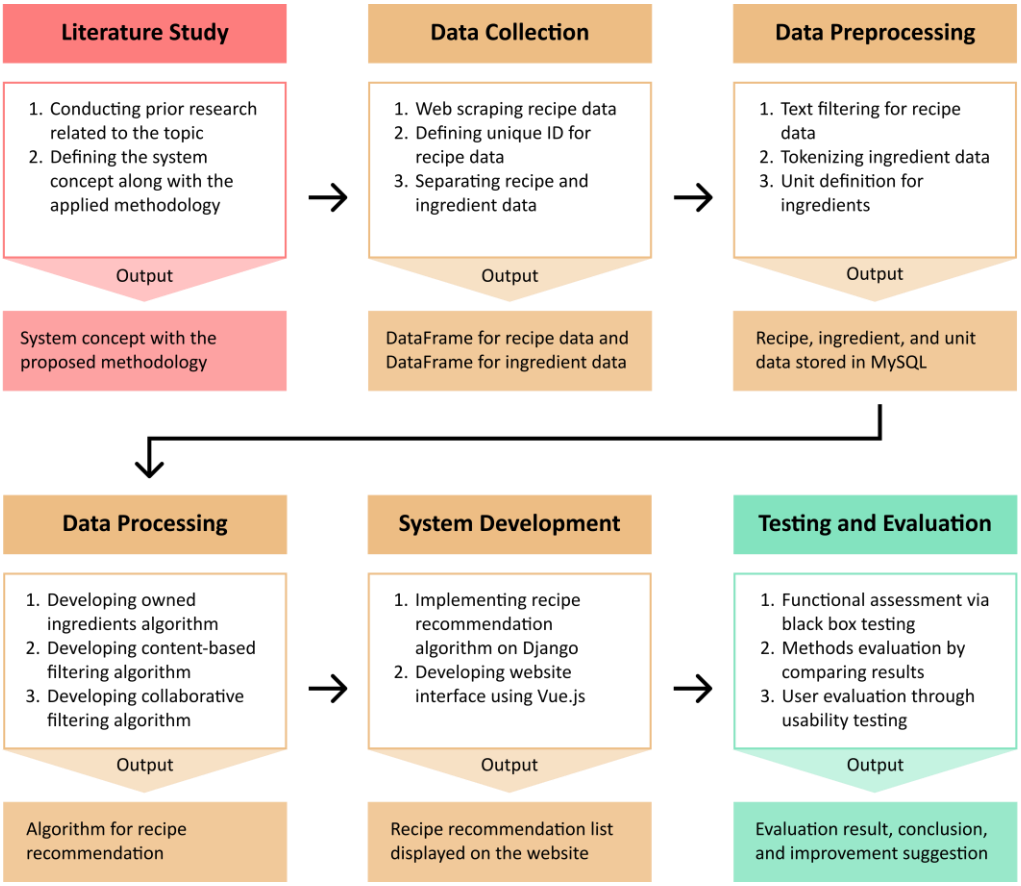


Figure 3.1 Research methodology

The development of a Recipe Recommendation System Based on Ingredient Availability and User Preferences involves several stages and methods to achieve the established goals. The initial phase involves a literature review to identify suitable methods for system development. Subsequently, data collection is carried out through web scraping to retrieve recipe information from various sources. The collected data undergoes data preprocessing, including text filtering to clean the data and tokenization to break the text into smaller units. In the data processing phase, recipe recommendation algorithms are developed based on ingredients using a content-based filtering approach, as well as recipe recommendations based on user preferences through collaborative filtering.

The next step is the development and implementation of the system using Django and Vue.js. The developed system is then tested to evaluate its functionality through black box testing, assess the performance of recipe recommendation methods by comparing the results among users, and evaluate user experience through usability testing.

3.2 Data Collection Techniques

To operate the recipe recommendation system, initial data consisting of Indonesian-language recipes sourced from the masakapahariini website (https://www.masakapahariini.com/) are required, which are divided into recipe data and ingredient data obtained through web scraping techniques.

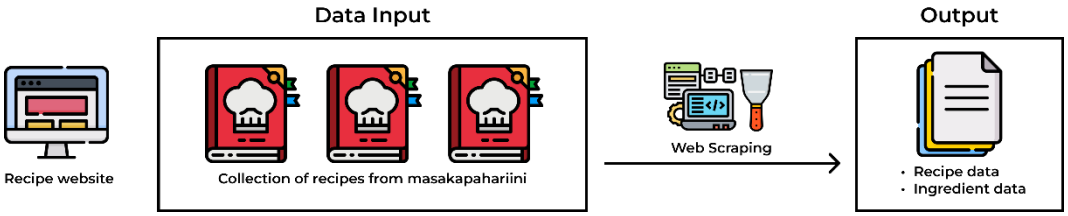


Figure 3.2 Data collection process

3.2.1 Cooking Recipe Data

Cooking recipe data is used to display the steps for preparing a dish. The collection of recipe data and ingredient data is conducted in three stages. Firstly, web scraping is performed to obtain recipe URLs from the recipe website. Then, another web scraping process is carried out to extract component data from each recipe. Lastly, recipe IDs are identified to differentiate between different recipes.

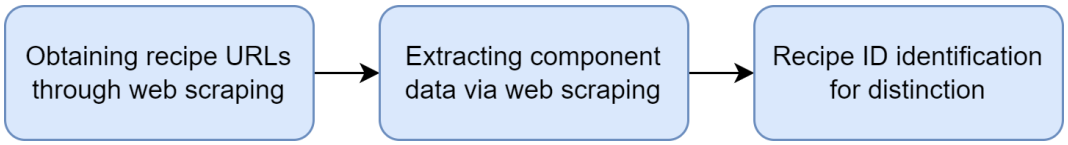


Figure 3.3 Cooking Recipe Data Collection Stages

3.2.1.1 Obtaining Recipe URLs

This stage is conducted to gather all recipe links from the masakapahariini website that led to recipe detail pages. The URL data is then stored in an array and will be used as target pages for extracting recipe data in the next stage. Figures 3.4 and 3.5 display examples of the masakapahariini website layout. In the HTML

structure of the recipe list, there are several elements, including anchor (<a>) elements that contain URL data to be extracted using web scraping methods.

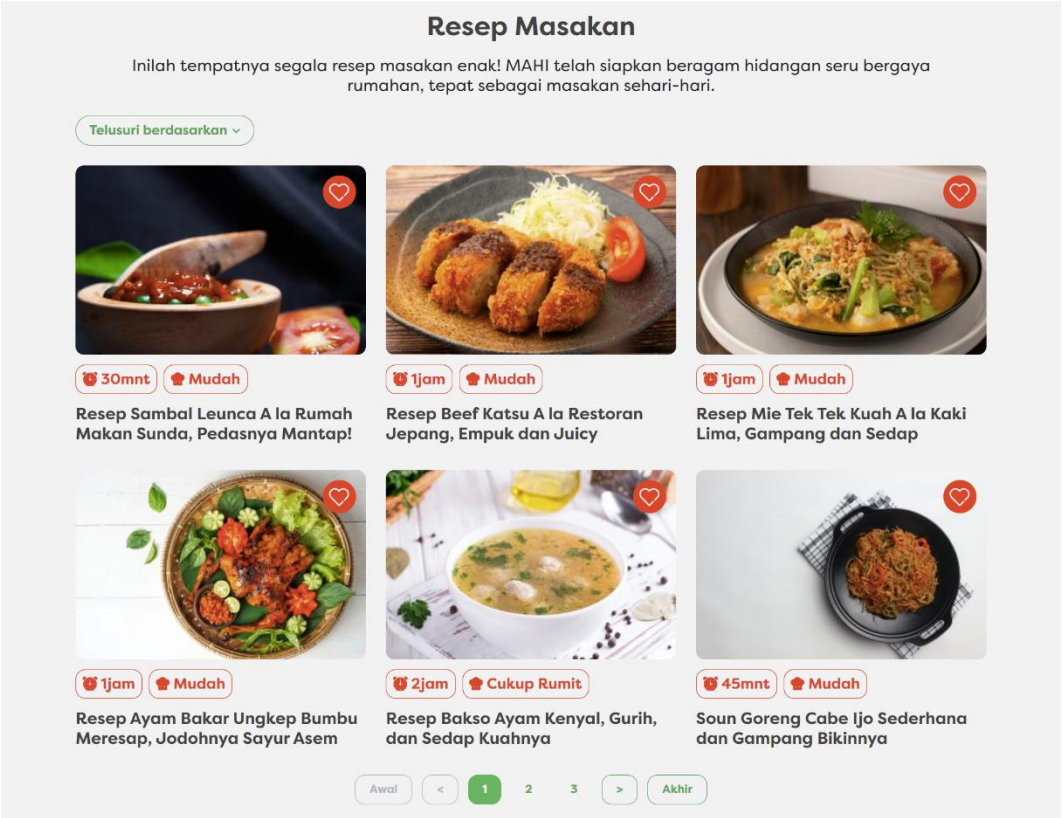


Figure 3.4 Display of recipe page on masakapahariini

```
<div class="card-body w-100 px-0 pt-0 pb-0">
  <div class="_recipe-features position-relative mb-3">...</div>
  <!-- TITLE -->
  <h3 class="card-title">
    <a href="https://www.masakapahariini.com/resep/resep-sambal-leunca-a-la-rumah-makan-sunda-pedasnya-mantap/"
    class="stretched-link text-decoration-none" data-tracking-value="Resep Sambal Leunca A la Rumah Makan Sunda,
    Pedasnya Mantap!">...</a>
  </h3>
</div>
```

Figure 3.5 Example of HTML elements storing recipe URLs

To perform web scraping on the masakapahariini recipe pages, the following steps need to be followed. Firstly, the masakapahariini recipe page is accessed through the URL https://www.masakapahariini.com/resep/. On this page, a list of available recipes is presented. Subsequently, the page displaying the recipe list is loaded for scraping purposes. During the scraping process, elements containing URLs for each recipe page are identified, and requests are sent to each URL. After receiving the server's response, parsing is carried out to extract URLs from each recipe. Finally, the recipe URLs are stored for further data processing and analysis.

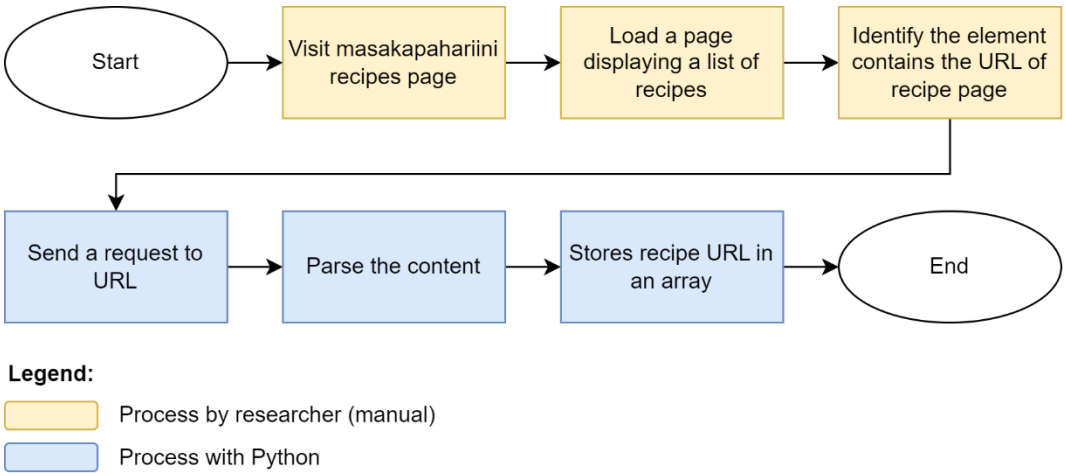


Figure 3.6 Web scraping process to obtain URLs leading to recipe details

Here is an example of successfully extracted URLs using web scraping techniques.

- https://www.masakapahariini.com/resep/resep-sambal-leunca-a-la-rumah-makan-sunda-pedasnya-mantap/
- https://www.masakapahariini.com/resep/resep-beef-katsu-a-la-restoran/
- https://www.masakapahariini.com/resep/resep-mie-tek-tek-kuah-a-la-kaki-lima/

3.2.1.2 Obtaining Recipe Detail Information

In this stage, specific information about each recipe is extracted. This information is used to present comprehensive details about each recipe on the website. The information includes:

1. Title: recipe title on the website.
2. Source URL: the source from where the recipe is obtained.
3. Image URL: link to the dish image.
4. Calories: the number of calories produced by the dish.
5. Servings: the number of portions produced from the recipe.
6. Time: the time required for cooking the dish.
7. Difficulty: the level of difficulty in cooking.
8. Ingredients: a list of ingredients required for cooking.
9. Instructions: the steps to follow in cooking.

Figures 3.7 and 3.8 show examples of the display of a cooking recipe on the masakapahariini website. In the HTML structure of the recipe detail, there are several elements used to display recipe information. These pieces of information will be extracted using web scraping methods.

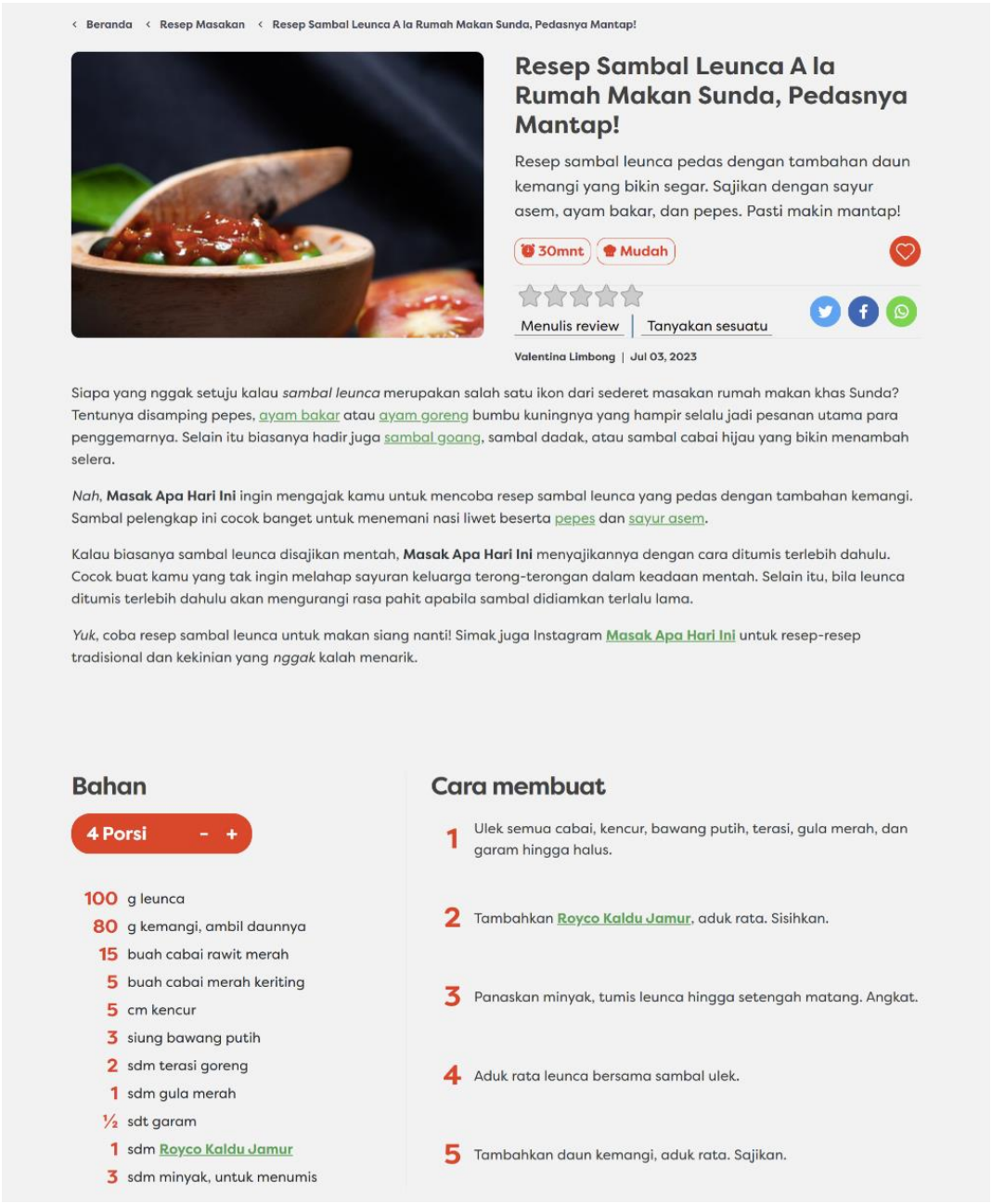


Figure 3.7 Display of recipe detail page in masakapahariini

```
<div class="d-flex"> flex
  <div class="part fw-bold me-3" data-value="20" data-base-measurement-unit="g " data-base-
  quantity="80"> 80 </div>
  <div class="item"> g kemangi, ambil daunnya </div> flex
</div>
<div class="d-flex"> flex
  <div class="part fw-bold me-3" data-value="3.75" data-base-measurement-unit="buah " data-base-
  quantity="15"> 15 </div>
  <div class="item"> buah cabai rawit merah </div> flex
</div>
<div class="d-flex"> flex
  <div class="part fw-bold me-3" data-value="1.25" data-base-measurement-unit="buah " data-base-
  quantity="5"> 5 </div>
  <div class="item"> buah cabai merah keriting </div> flex
</div>
```

Figure 3.8 Example of HTML structure displaying ingredient list

To perform web scraping, the following steps need to be followed. Firstly, one of the recipe URLs is selected from the collection of recipe URLs obtained earlier. Next, access to the recipe detail page is made using the selected URL. The content of the page is then parsed to identify the essential elements to be extracted. Subsequently, variables are defined to store the data to be extracted from the page. For example, a "title" variable is created to store the recipe title, an "ingredients" variable for storing the list of ingredients, and an "instructions" variable for storing the cooking instructions. Finally, the successfully extracted data is stored in a suitable format as required, enabling it to be used and analyzed according to the specific needs.

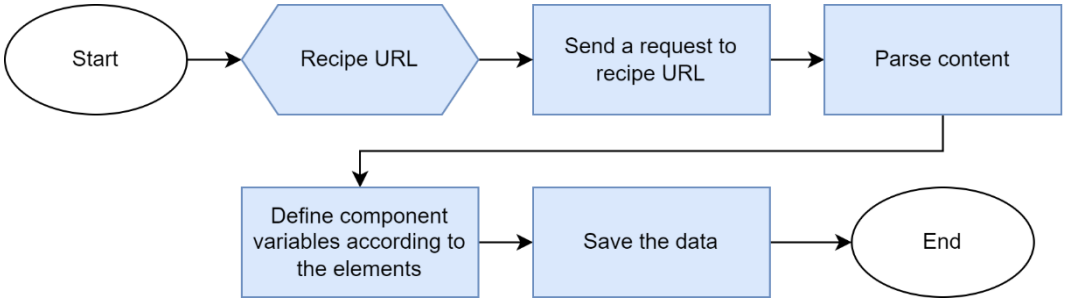


Figure 3.9 Web scraping process to obtain recipe data

An example of the successfully extracted recipe data from web scraping is shown in Table 3.1. Through this scraping process, several components have been defined, such as title, source URL, image URL, calories, servings, time, difficulty, ingredients, and instructions, which have been successfully extracted.

Table 3.1 Example of recipe data obtained through web scraping

title	source_url	image_url	time	servings	calories	difficulty	ingredients	instructions
Resep Sambal Leunca A la Rumah Makan Sunda	https://www.masakapahariini.com/resep/resep-sambal-leunca-a-la-rumah-makan-sunda-pedasnya-mantap/	https://www.masakapahariini.com/wp-content/uploads/2023/06/shutterstock_2074261048-780x440.jpg	30mnt	4	0	Mudah	['100 g leunca','80 g kemangi, ambil daunnya','15 buah cabai rawit merah','5 buah cabai merah keriting','5 cm kencur','3 siung bawang putih','2 sdm terasi goreng','1 sdm gula merah','1 / 2 sdt garam','1 sdm Royco Kaldu Jamur','3 sdm minyak, untuk menumis']	['Ulek semua cabai, kencur, bawang putih, terasi, gula merah, dan garam hingga halus.','Tambahkan Royco Kaldu Jamur, aduk rata. Sisihkan.','Panaskan minyak, tumis leunca hingga setengah matang. Angkat.','Aduk rata leunca bersama sambal ulek.','Tambahkan daun kemangi, aduk rata. Sajikan.']

3.2.1.3 Defining Recipe IDs

The process of defining recipe IDs is carried out to distinguish between one recipe and another, thus preventing duplication. This is accomplished by assigning random ID values to each recipe. Table 3.2 provides an example of recipe data that has been assigned unique IDs. Each recipe has its own distinct ID, setting it apart from other recipes within the dataset.

Table 3.2 Example of recipe data with assigned recipe IDs

id	title	source_url	image_url	time	servings	calories	difficulty	ingredients	instructions
890942566 16	Resep Sambal Leunca A la Rumah Makan Sunda	https://www.masakapahariini.com/resep/resep-sambal-leunca-a-la-rumah-makan-sunda-pedasnya-mantap/	https://www.masakapahariini.com/wp-content/uploads/2023/06/shutterstock_2074261048-780x440.jpg	30mnt	4	0	Mudah	['100 g leunca','80 g kemangi, ambil daunnya','15 buah cabai rawit merah','5 buah cabai merah keriting','5 cm kencur','3 siung bawang putih','2 sdm terasi goreng','1 sdm gula merah','1 / 2 sdt garam','1 sdm Royco Kaldu Jamur','3 sdm minyak, untuk menumis']	['Ulek semua cabai, kencur, bawang putih, terasi, gula merah, dan garam hingga halus.','Tambahkan Royco Kaldu Jamur, aduk rata. Sisihkan.','Panaskan minyak, tumis leunca hingga setengah matang. Angkat.','Aduk rata leunca bersama sambal ulek.','Tambahkan daun kemangi, aduk rata. Sajikan.']

3.2.2 Recipe Ingredient Data

After obtaining recipe data through web scraping techniques, the next step is to process the recipe ingredient data. In this stage, the recipe IDs and ingredient lists from the recipe data are extracted and divided into individual ingredients per row, then stored as separate data. The purpose of this process is to facilitate the identification of each ingredient individually and enable more effective analysis and processing of ingredient data.

Table 3.3 illustrates an example of data before and after being split into individual rows, where the recipe ID accompanies each ingredient.

Table 3.3 Example of ingredient data before and after being separated

Before

id	ingredients
89094256616	['100 g leunca','80 g kemangi, ambil daunnya','15 buah cabai rawit merah','5 buah cabai merah keriting','5 cm kencur','3 siung bawang putih','2 sdm terasi goreng','1 sdm gula merah','1 / 2 sdt garam','1 sdm Royco Kaldu Jamur','3 sdm minyak, untuk menumis']

After

id	ingredients
89094256616	100 g leunca
89094256616	80 g kemangi, ambil daunnya
89094256616	15 buah cabai rawit merah
89094256616	5 buah cabai merah keriting
89094256616	5 cm kencur
89094256616	3 siung bawang putih
89094256616	2 sdm terasi goreng
89094256616	1 sdm gula merah
89094256616	1/2 sdt garam
89094256616	1 sdm Royco Kaldu Jamur
89094256616	3 sdm minyak, untuk menumis

3.3 Data Processing Techniques

Data processing aims to generate recipe recommendations that are suitable based on ingredient availability and user preferences. In the process, the recipe and ingredient data possessed by the user will be processed using content-based filtering with Jaccard similarity. This approach seeks to find similarities between the ingredients present in recipes and the ingredients available in the user's kitchen, thereby providing relevant recommendations based on ingredient availability.

Additionally, the recipe data that the user has liked will also be processed using item-based collaborative filtering with cosine similarity. This approach aims to find similarities between recipes that the user has liked and other recipes, thus offering recommendations for recipes that are similar to the ones previously liked.

3.3.1 Text Preprocessing

Text preprocessing in this stage aims to generate clean, structured, and ready-to-process data. The data to be processed includes recipe data, ingredient data, and unit data. This process helps simplify the data and enhance the quality of information used in the recipe recommendation system.

3.3.1.1 Recipe Data

Before processing recipe data, the first step is to perform text filtering on the recipe data. This process involves examining sentence patterns used in each data column. Subsequently, data cleaning is carried out using string splitting and text replacement methods to efficiently remove irrelevant details. This helps eliminate unnecessary text and retains only relevant numbers in the data format.

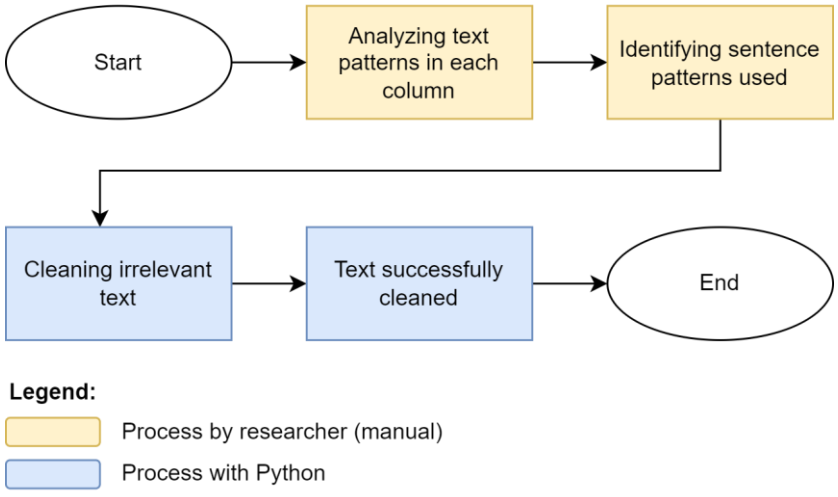


Figure 3.10 Text filtering process for recipe data

Table 3.4 presents an example of data before and after the text cleaning and transformation process. In the initial data, several shortcomings are evident, such as recipe titles using excessive non-essential phrases, unnecessary calorie units, and time units still in text format. However, after processing, the data becomes more organized and easier to understand.

Table 3.4 Example of recipe data before and after text filtering

Before

title	calories	time
Resep Ayam Sambal Geprek, Menggugah Selera	400Kkal	30mnt
Cara Membuat Ketupat, Sajian Wajib Saat Lebaran	700Kkal	1jam 30 mnt

After

title	calories	time
Ayam Sambal Geprek	400	30
Ketupat	700	90

3.3.1.2 Recipe Ingredients Data

Before using ingredient data as parameters to determine food recommendations and ingredient availability percentages, the initial step involves separating the ingredient data into four main components: ingredient name, quantity, unit, and state, using tokenization methods. This separation process is carried out using regular expressions (regex) to organize the data with an appropriate structure and pattern.

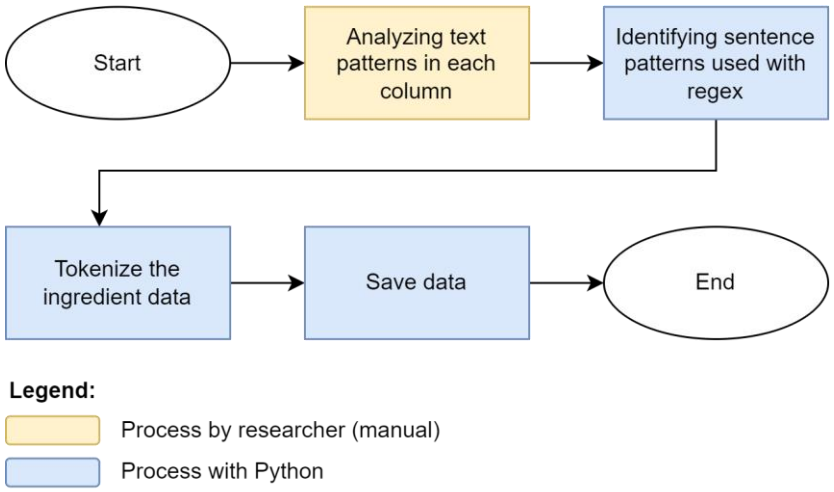


Figure 3.11 Tokenization process for ingredients data

Table 3.5 provides an example of ingredient data before and after tokenization, where the tokenized ingredient data is divided into ingredient name, quantity, unit, and state, facilitating further data processing.

Table 3.5 Example of ingredient data before and after tokenization

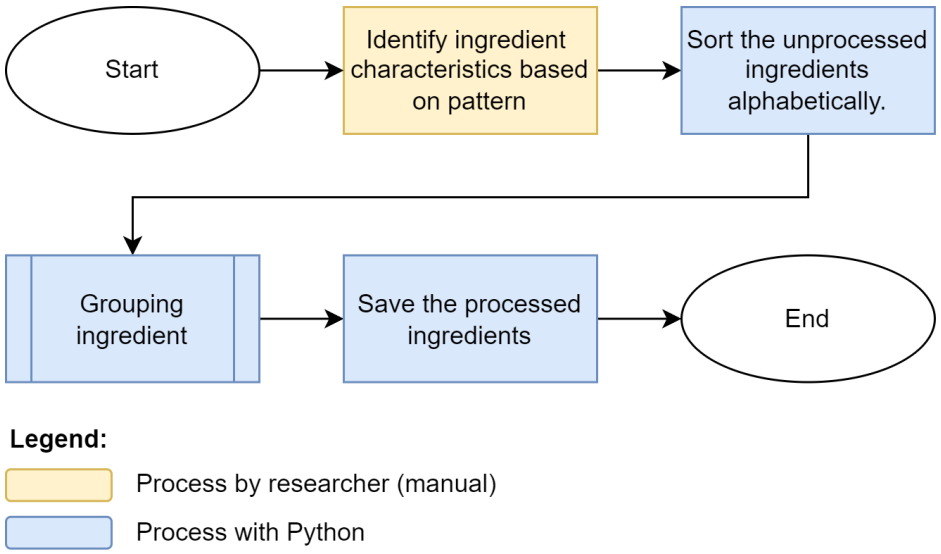
Before

id	ingredients
89094256616	100 g leunca
89094256616	80 g kemangi, ambil daunnya
89094256616	1/2 sdt garam

After

id	name	quantity	unit	state
89094256616	leunca	100	g	
89094256616	kemangi	90	g	ambil daunnya
89094256616	garam	1/2	sdt	

After the data has been separated, the next step is to perform ingredient name grouping to identify the basic names of cooking ingredients. This is done to enable the system to recognize that certain ingredients are actually the same, only written differently. The grouping of basic ingredient names involves identifying common writing patterns, such as size or color indicators. Once these patterns are identified, the ingredient data is grouped into several categories based on their characteristics. This grouping allows the system to recognize similar ingredients and optimize the analysis and food recommendations provided more effectively.



In the ingredient data sourced from the masakapahariini website, there are several characteristics based on which ingredients can be categorized. These characteristics include:

- Difficult Crafting Dish: Referring to ingredients that undergo processing altering their original form. Example: *saos, sambal, saus, tepung*.
- Size: Words used to indicate ingredient size. Example: *besar, kecil, sedang*.
- Different Meaning Part: Part of the ingredient name that, when combined with other parts, forms a sentence with a different meaning. Example: *daun (daun pisang, daun jeruk), has (daging has dalam)*.
- Easy Crafting Dish: Ingredients processed without changing their original form or taste. Example: *asap, goreng, kukus, larutan, perasan, serut*.
- Brands: Ingredients that are also specific brand names. Example: Bango, Buavita, Jawara, Royco, Wall's.
- Ingredient Forms: Different forms of ingredients before processing. Example: *basah, dingin, es, giling, halus, kering, matang*.
- Lost Mean Words: Ingredients that lose meaning if their names are removed. Example: *bawang, ikan*.
- Multi Meaning Dish: Ingredients that, when combined with other ingredients, have a different meaning. Example: water (plain water, orange juice).

- Irrelevant Words: Words that do not significantly contribute to ingredient naming. Example: meat, as needed, eat.
- Conjunction: Words used to connect multiple ingredients in the naming. Example: and, or.

The process of grouping ingredient names to obtain basic ingredient names follows certain rules or steps:

1. First, check the number of words in the ingredient name. If it consists of only one word, the ingredient will form its own group in the final outcome.
2. If the ingredient name consists of more than one word, process the words. Remove irrelevant words such as size indicators, easy crafting terms, or non-essential words in ingredient naming.
3. After removing irrelevant words, recount the number of words in the processed ingredient name.
4. Next, determine the category based on the following rules:
    - If there is a brand keyword in the ingredient name, group the ingredient into the corresponding brand category.
    - If there is a keyword indicating a difficult crafting dish, group the ingredient into the difficult crafting category.
5. If there are no brand or difficult crafting categories, check against existing categories. If the ingredient has multi-meaning and more than one word in the processed name, ignore it at this stage.
6. Count the number of words in the processed ingredient name that match the keywords found in the previous step.
7. Compare the number of matched words from step 4 with the previous keywords. If the matched word count is higher or equal, select the keyword with the highest word count.
8. If a suitable keyword is found, add the ingredient name to the corresponding category. If a suitable category exists, add it if it meets specific criteria related to word count.
9. If no suitable keyword or category is found, add the ingredient name to a new category.

10. Repeat the above steps for all ingredients.

After going through the processing steps, the data is stored in JSON format. If anomalies occur within the data, manual handling is necessary. This handling is performed by editing the exported JSON data and manually correcting its categories.

For example, in the case of the ingredient "*daging kepiting*," some entries labeled as "*daging kepiting*" have been categorized with different ingredient names such as "*daging kepiting rebus*" or "*daging kepiting segar ukuran besar*." However, all three of these entries refer to the same ingredient, which is "*daging kepiting*," but they might have undergone different initial processing methods. In this scenario, manual editing is required to adjust these categories so that they all fall under the category of "*daging kepiting*."

```
# Not yet corrected data
"daging kepiting": {
    "daging kepiting": [
        "daging kepiting"
    ],
},
"daging kepiting rebus": {
    "daging kepiting rebus": [
        "daging kepiting rebus"
    ]
},
"kepiting segar ukuran besar": {
    "kepiting segar ukuran besar": [
        "kepiting segar ukuran besar"
    ]
},
# Manually corrected data
"daging kepiting": {
    "daging kepiting": [
        "kepiting segar ukuran besar",
        "daging kepiting rebus",
        "daging kepiting",
    ]
},
```

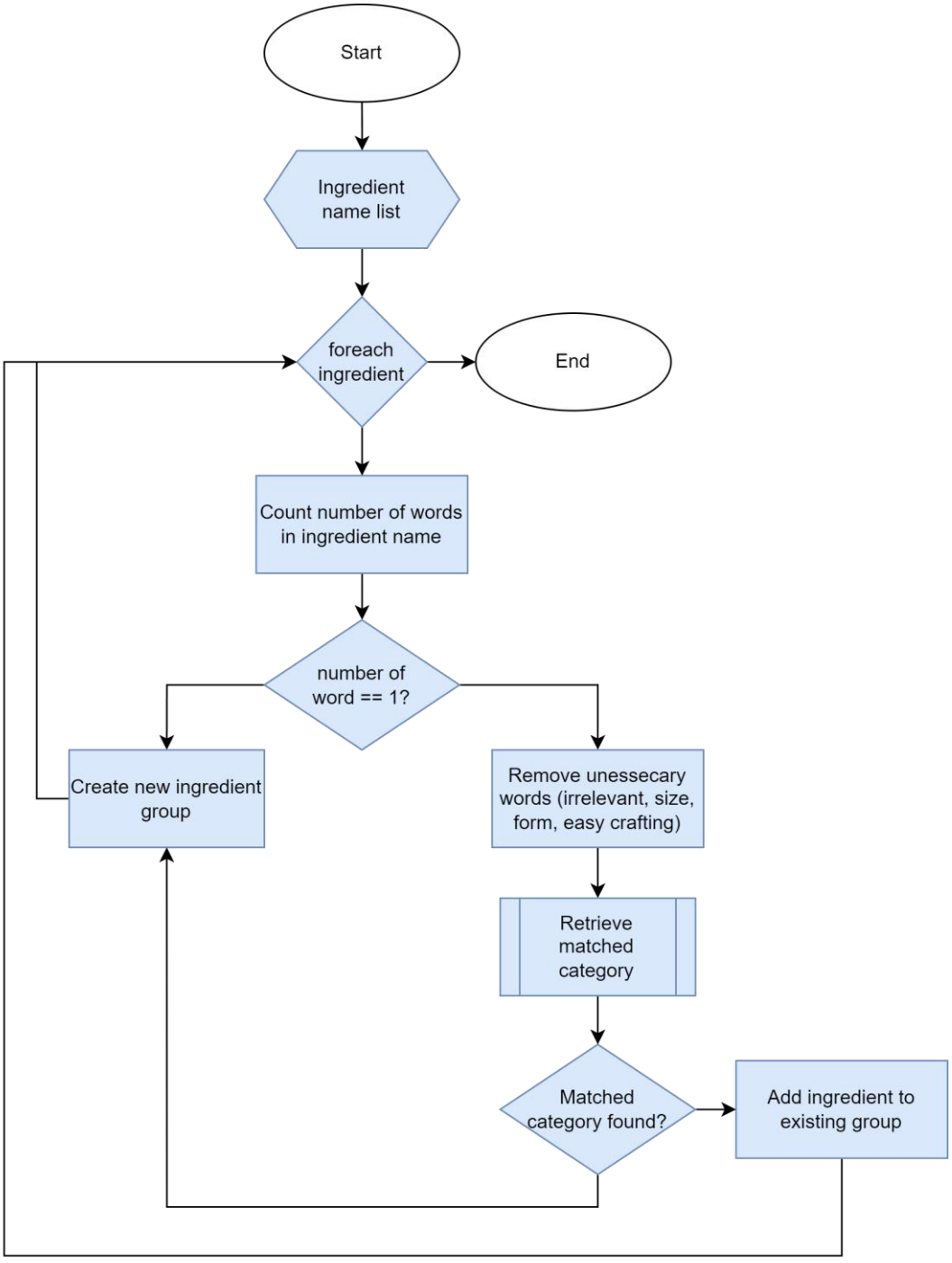


Figure 3.12 Grouping ingredient names process

Table 3.6 illustrates an example of ingredient data that has been grouped based on basic names. This basic ingredient data is used to identify the intended ingredient names in the recipe (for example, "*ayam ukuran besar*" is identified as

"*ayam*") and to become the ingredient names entered by users who want to add items to the available ingredient list.

Table 3.6 Example of ingredient data with basic names

ID	Name	Quantity	Unit	State	Basic name
89094256616	ayam ukuran besar	100	g	potong	ayam
42368728282	dada ayam rebus	90	g		dada ayam
10923281922	Ayam	1/2	kg	kupas	ayam
89094256616	pisang kukus	3	buah		pisang
42368728282	daun pisang	1	lembar	sobek	daun pisang

3.3.1.3 Unit Data

The unit data is derived from the ingredient data that has undergone the previous processing steps. This unit data is then categorized based on its type, which includes volume, weight, and items. Additionally, each unit is assigned a unit type, which could be informal or measured, indicating whether the unit is countable or informal. Furthermore, a hierarchical grouping is applied, using numbers where smaller numbers indicate a higher hierarchy, to determine whether the unit value is higher or lower than other units.

Below is an example table containing ingredients along with hierarchical, type, and category information for the identified units, as demonstrated in Table 3.7.

Table 3.7 Example of unit data with hierarchy, type, and category information

Code	Name	Category	Hierarchy	Type
ikat	ikat	item	1	informal
siung	siung	item	2	informal
batang	batang	item	3	informal
buah	buah	item	3	informal
biji	biji	item	3	informal
butir	butir	item	3	informal
ekor	ekor	item	3	informal
kotak	kotak	item	3	informal

pak	pak	item	3	informal
papan	papan	item	3	informal
ruas	ruas	item	3	informal
stik	stik	item	3	informal
bungkus	bungkus	item	3	informal
lembar	lembar	item	3	informal
potong	potong	item	3	informal
sachet	sachet	item	3	informal
tangkai	tangkai	item	3	informal
scoop	scoop	item	4	informal
cm	centimeter	item	3	measured
tetes	tetes	volume	3	informal
L	liter	volume	1	measured
cc	cc	volume	2	measured
ml	mililiter	volume	2	measured
porsi	porsi	weight	2	informal
kg	kilogram	weight	1	measured
ons	ons	weight	3	measured
sdm	sendok makan	weight	4	measured
sdt	sendok teh	weight	5	measured
g	gram	weight	6	measured

After the identification is done, the next step is to define the data for unit conversion that has standard measurements. This table of unit conversion for ingredients will be utilized to assist users in converting ingredient units according to their needs and preferences.

Table 3.8 shows the unit conversion table that will be implemented in the system.

Table 3.8 Unit conversion table

From	**To**	**Value**
kilogram	gram	1000

liter	mililiter	100
ons	gram	100
cc	ml	1
kilogram	ons	10
liter	cc	1000
Sendok makan	gram	15
Sendok makan	mililiter	15
Sendok teh	gram	5
Sendok teh	mililiter	5

3.3.2 Ownership of User Ingredients for Recipe Ingredients

The user ingredient ownership algorithm is employed to assist users in identifying the ingredients they possess from a selected recipe. This algorithm also aids users in recognizing any shortages of required ingredients for cooking a chosen recipe.

The algorithm process begins by converting the list of ingredients owned by the user and the list of ingredients in the recipe. Subsequently, the algorithm groups these ingredients based on the following rules:

1. If the user's ingredient lacks a unit, add it to the user's ingredient list.
2. If the recipe ingredient lacks a unit, move to the next ingredient in the recipe.
3. If both the user's ingredient and recipe ingredient share the same unit, check if the quantity of the ingredient in the pantry is sufficient to meet the required amount in the recipe. If sufficient, add the ingredient to the user's ingredient list.
4. If both ingredients have a "measured" unit type and the user's ingredient hierarchy is lower or equal to the recipe ingredient hierarchy, convert the user's ingredient quantity to the appropriate unit according to the recipe. Then, check if the converted quantity is sufficient to meet the required amount in the recipe. If sufficient, add the ingredient to the user's ingredient list.
5. If both ingredients share the same unit category, compare the hierarchy levels of the user's ingredient and recipe ingredient. If the user's ingredient

hierarchy is lower, add the ingredient to the user's ingredient list. If hierarchy levels are the same, check if the quantity of the user's ingredient is sufficient to meet the required amount in the recipe. If sufficient, add the ingredient to the user's ingredient list.

6. If both ingredients belong to the "item" unit category, add the ingredient to the user's ingredient list.

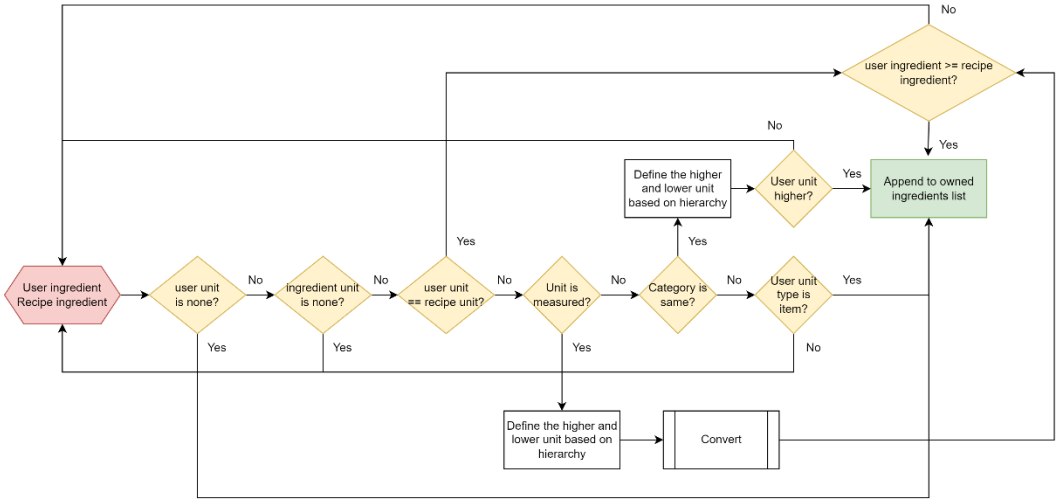


Figure 3.13 The conversion of user and recipe ingredient unit's flowchart

The result of this algorithm is a list of ingredients owned by the user for each recipe. This outcome can be used as input to run the recipe recommendation algorithm that aligns with the user's available ingredients. Additionally, this ingredient list can be displayed to users to provide information about the ingredients they possess for the selected recipe.

3.3.3 Percentage of User Ingredient Ownership for Recipe

This algorithm aims to determine how much of the user's ingredients correspond to the ingredients in the recipe, presented as a percentage. The percentage of ingredient ownership for a specific recipe is obtained by dividing the quantity of user-owned ingredients (attained through the user ingredient ownership algorithm) by the total quantity of recipe ingredients, then multiplying the result by 100.

$$Owned\ ingredient\ percentage = \frac{Quantity\ of\ user\ ingredients}{Total\ recipe\ ingredients} \times 100\%$$

By employing this simple formula, the percentage of ingredient ownership by the user can be determined, helping understand the extent to which their ingredients are sufficient for cooking the selected recipe. A higher ownership percentage indicates a more complete collection of ingredients, facilitating the user in achieving the desired cooking outcome.

3.3.4 Recipe Recommendations

The goal of this recipe recommendation algorithm is to provide users with food menu recommendations that align with their criteria. The methods utilized to achieve this involve content-based filtering and item-based collaborative filtering. Through content-based filtering, the system analyzes recipe data and user-owned ingredients to offer relevant recommendations. On the other hand, item-based collaborative filtering utilizes user-preferred recipe data to discover similar recipes that might interest them.

3.3.4.1 Recommendations Based on Ingredient Availability

This method is used to provide recipe recommendations based on the ingredients available to the user. By analyzing the similarity between the user's ingredients and the ingredients in recipes, this method generates a list of recipes that are most suitable for the user's ingredients.

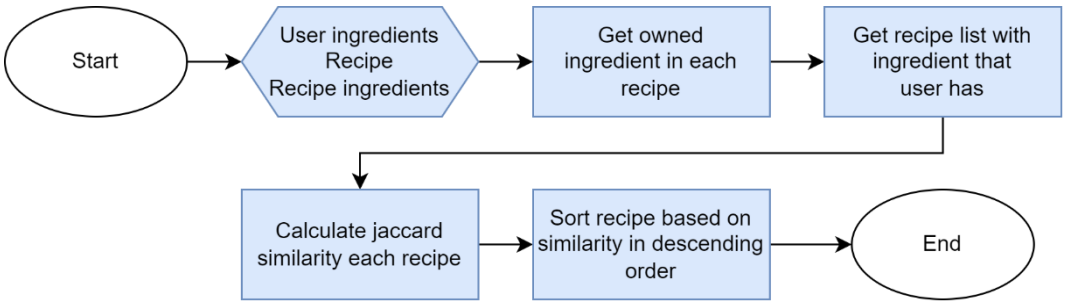


Figure 3.14 Process of recipe recommendations based on ingredient availability

The process begins by obtaining a list of ingredients owned by the user to search for relevant recipes. Next, the list of recipes is filtered specifically for recipes that use the user's ingredients. In this process, irrelevant recipes are ignored. After that, the Jaccard similarity between the user's ingredients and the required ingredient in each relevant recipe is calculated. This step aims to determine how similar each recipe is to the user's ingredients. Finally, the list of recipes is sorted based on the decreasing similarity score.

To provide weighting to the recommendations, this method combines the Jaccard similarity value with the percentage of ingredient ownership by the user. The Jaccard similarity value depicts the similarity of ingredients owned by the user with the ingredients in the recipe, while the percentage of ingredient ownership indicates how much of the user's ingredients match the recipe's ingredients. In this case, the Jaccard similarity value is weighted at 80%, while the percentage of ingredient ownership is weighted at 20%.

By assigning weights to both factors, this method ensures that the recommendations remain relevant to the user's ingredient inventory while considering the level of similarity with existing recipes. The result of this method is a list of recipes that best match the user's ingredient inventory, with the goal of providing more personalized and relevant recommendations for selecting recipes based on the ingredients they have.

For example, a user has several ingredients such as carrots, corn, tofu, and tomatoes. Furthermore, there are five recipes with various ingredients, as explained in Table 3.9.

Table 3.9 List of user ingredients and recipe along with ingredient names

User Ingredients
Carrot
Corn
Tofu
Tomato

Recipe	**Ingredients**
Recipe 1	Carrot, Chili, Peanut
Recipe 2	Corn, Carrot
Recipe 3	Carrot, Tempeh
Recipe 4	Tofu, Oil, Tomato
Recipe 5	Tempeh, Oil, Chili

The initial step to be taken is to prepare the initial data by detailing the list of recipes and the ingredients involved in each recipe. Next, the formation of ingredient sets is carried out for each recipe, involving the grouping of related ingredients into one set for each recipe.

The next step is to calculate the Jaccard Similarity for each recipe. In this stage, the Jaccard Similarity formula is used to calculate the ratio of the number of shared elements between the user's ingredients and the recipe's ingredients to the total number of elements in the union of the user's ingredients and the recipe's

ingredients. For example, for Recipe 1, there is one ingredient from the recipe that is also owned by the user, namely carrots. This element represents part of the intersection set between the two sets. This intersection value, namely carrots, is also included in the set of ingredients in Recipe 1 and the set of user's ingredients. The union set of these two sets consists of carrots, corn, tofu, tomatoes, chili, and peanuts.

This process is repeated for each recipe, so the calculations can be explained as follows:

- $Jaccard\ similarity(Recipe\ 1) = \frac{1}{6} = 0.167$
- $Jaccard\ similarity(Recipe\ 2) = \frac{2}{4} = 0.5$
- $Jaccard\ similarity(Recipe\ 3) = \frac{1}{5} = 0.2$
- $Jaccard\ similarity(Recipe\ 4) = \frac{2}{6} = 0.333$
- $Jaccard\ similarity(Recipe\ 5) = \frac{0}{7} = 0$

Next, the calculation of the percentage of completeness of ingredients owned by the user with respect to the ingredients in each recipe is performed. This provides an overview of how well the user's ingredients match the ingredient composition in the recipe.

- $Completeness\ ratio(Recipe\ 1) = \frac{1}{3} = 0.333$
- $Completeness\ ratio(Recipe\ 2) = \frac{2}{2} = 1$
- $Completeness\ ratio(Recipe\ 3) = \frac{1}{2} = 0.5$
- $Completeness\ ratio(Recipe\ 4) = \frac{2}{3} = 0.667$
- $Completeness\ ratio(Recipe\ 5) = \frac{0}{3} = 0$

To generate a recommendation score, the next step is to combine these two aspects by assigning weights to each recipe. This weighting is achieved through a combination of the completeness percentage of ingredients (20%) and the Jaccard Similarity value (80%). As an illustration, the weight calculation is done as follows:

- $Weight\ score(Recipe\ 1) = 0.167 \times 80\% + 0.333 \times 20\% = 0.2$
- $Weight\ score(Recipe\ 2) = 0.5 \times 80\% + 1 \times 20\% = 0.6$
- $Weight\ score(Recipe\ 3) = 0.2 \times 80\% + 0.5 \times 20\% = 0.26$

- $Weight\ score(Recipe\ 4) = 0.333 \times 80\% + 0.667 \times 20\% = 0.4$
- $Weight\ score(Recipe\ 5) = 0 \times 80\% + 0 \times 20\% = 0$

Based on this calculation, the recommended recipes are as follows: Recipe 2 receives the highest score of 0.6, followed by Recipe 4 with a score of 0.4, then Recipe 3 with a score of 0.26, and finally Recipe 1 with a score of 0.2. Recipe 5 does not receive a recommendation because there are no ingredients owned by the user.

Table 3.10 Recipe recommendation ranking based on user-owned ingredients

Ranking	Recipe Name	Score
1	Recipe 2	0.6
2	Recipe 4	0.4
3	Recipe 3	0.26
4	Recipe 1	0.2

3.3.4.2 Recommendations Based on User Preferences

This method aims to provide recipe recommendations similar to recipes that have already been liked by the user. By analyzing preference patterns from liked recipe data using cosine similarity, this method generates recommendations of recipes that have similarities with the user's preferences as choices that align with their taste.

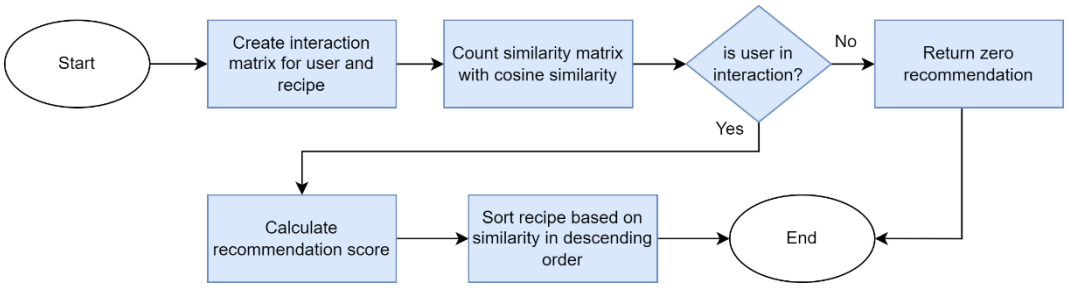


Figure 3.15 Process of recipe recommendations based on user preferences

The process begins with defining an interaction matrix between all users and recipes. This matrix stores values indicating whether a user has liked or disliked a particular recipe. Next, a similarity matrix is calculated using cosine similarity to measure the similarity between recipes based on the interaction patterns recorded in the matrix.

If the user being processed is present in the interaction matrix, the next step is to calculate the recommendation score. This score is obtained by performing a

dot product between the interaction matrix and the similarity matrix, adjusted according to the user's preferences for the existing recipes. The recommendation score is then used to determine the ranking of recipes in the recommendation list.

Finally, a list of recommended recipes is generated by sorting the recipes based on the recommendation scores in descending order. Recipes with higher scores are ranked higher in the recommendation list, allowing users to easily identify recipes that best match their positive interaction patterns with previous recipes.

For example, there are several users who like certain recipes, as detailed in Table 3.11.

Table 3.11 Example of user data who liked certain recipes

User	**Liked Recipes**
User 1	Recipe 1, Recipe 3, Recipe 4
User 2	Recipe 3, Recipe 5
User 3	Recipe 1, Recipe 2, Recipe 4
User 4	Recipe 1, Recipe 5

The initial step in applying cosine similarity involves analyzing the recipe preferences of each user based on the data presented in Table 3.11. This data provides a clear picture of the recipes liked by each user, such as User 1 who is interested in Recipe 1, Recipe 3, and Recipe 4, while User 2 tends to prefer Recipe 3 and Recipe 5. This information serves as an important starting point in the cosine similarity calculation, as it forms the basis for the subsequent preference user-recipe matrix calculation.

The next step is to create a matrix representing user preferences for recipes. In this matrix, each row represents a user, and each column represents a specific recipe. The values in the matrix indicate whether the user likes (value 1) or dislikes (value 0) that recipe. This process helps organize user preferences into a more structured format.

Table 3.12 Example of user-recipe matrix for cosine similarity

	Recipe 1	**Recipe 2**	**Recipe 3**	**Recipe 4**	**Recipe 5**
User 1	1	0	1	1	0

User 2	0	0	1	0	1
User 3	1	1	0	1	0
User 4	1	0	0	0	1

The next step is to calculate the cosine similarity between the recipes, using the previously formed user-recipe preference matrix. For example, the cosine similarity calculation for Recipe 1 and Recipe 2 is as follows:

$$Similarity(Recipe\ 1, Recipe\ 2) = \frac{(1 \times 0 + 0 \times 0 + 1 \times 1 + 1 \times 0)}{\sqrt{1^2 + 0^2 + 1^2 + 1^2} \times \sqrt{0^2 + 0^2 + 1^2 + 0^2}}$$
$$= 0{,}577$$

In Table 3.13, the results of the cosine similarity calculations between recipes are shown, where each value in the matrix indicates the level of similarity between the two respective recipes.

Table 3.13 Results of cosine similarity calculations between recipes

	Recipe 1	**Recipe 2**	**Recipe 3**	**Recipe 4**	**Recipe 5**
Recipe 1	1	0,577	0,408	0,816	0,408
Recipe 2	0,577	1	0	0,707	0
Recipe 3	0,408	0	1	0,5	0,5
Recipe 4	0,816	0,707	0,5	1	0
Recipe 5	0,408	0	0,5	0	1

After obtaining the cosine similarity matrix, the next step is to calculate the dot product between user preferences and the cosine similarity matrix. In this process, each user's preference value is multiplied by the corresponding cosine similarity value, and the results are summed for each recipe. The outcome of this calculation yields a user-specific score per recipe, which indicates the extent to which a recipe aligns with the user's preferences.

For example, for User 4, the process can be defined as follows:

- $Recipe\ 1 = 1 \times 1 + 0 \times 0.577 + 0 \times 0.408 + 0 \times 0.816 + 1 \times 0.408 = 1.408$
- $Recipe\ 2 = 1 \times 0.577 + 0 \times 1 + 0 \times 0 + 0 \times 0.707 + 1 \times 0 = 0.577$
- $Recipe\ 3 = 1 \times 0.408 + 0 \times 0 + 0 \times 1 + 0 \times 0.5 + 1 \times 0.5 = 0.908$

- $Recipe\ 4 = 1 \times 0.816 + 0 \times 0.707 + 0 \times 0.5 + 0 \times 1 + 1 \times 0 = 0.816$
- $Recipe\ 5 = 1 \times 0.408 + 0 \times 0 + 0 \times 0.5 + 0 \times 0 + 1 \times 1 = 1.408$

After obtaining the per-user scores, the next step is to compile recipe recommendations for each user. These recommendations are organized based on the highest to lowest scores, providing users with guidance on recipes that best match their positive interaction patterns with previous recipes. For User 4, the recommended recipes obtained are Recipe 3, Recipe 4, and Recipe 2. This is because User 4 liked Recipe 1 and Recipe 5.

Table 3.14 Recipe recommendation ranking for User 4

Ranking	Recipe Title	Score
1	Resep 3	0.908
2	Resep 4	0.816
3	Resep 2	0.577

3.4 System Testing

After the development of the system, it needs to undergo testing to ensure its success and reliability. Three testing methods are used in this phase: functional testing with black box testing, performance testing by comparing results and calculating accuracy, and user feedback testing using usability testing.

3.4.1 Black Box Testing

Black box testing is a software testing method that focuses on the external functionality of a system without considering its internal structure. This testing aims to ensure that the system functions correctly according to the specified requirements and specifications from the end-user perspective (Verma et al., 2017).

The black box testing process begins with a thorough analysis of the system's specifications and requirements to be tested. The system's functionality and expected behavior in various situations are examined. After the analysis is complete, test cases or testing scenarios are created based on these specifications. Test cases include the steps to be followed to systematically test the system's functions. Subsequently, test cases are executed using the system's interface, where data is input, and the output results from the system are examined.

The results of the testing are compared to the expected outcomes according to the specifications. If the results match the expectations, then the system is considered to function properly and meet the requirements. However, if there are differences between the test results and expectations, potential bugs or issues in the system that need to be addressed are identified. Black box testing helps ensure that the system functions well and meets the needs of end users, thereby enhancing the overall quality of the system.

3.4.2 Accuracy Prediction

Accuracy level is an important measure for evaluating how well the predictions match the actual data. It is used to determine the level of accuracy of predictions made by a system. The common formula used to calculate accuracy is:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \times 100\%$$

In this formula, "Number of Correct Predictions" refers to the number of predictions that match the actual data, and "Total Data" is the total number of data points evaluated. The accuracy level provides an overview of how accurately the system can make relevant predictions based on the available data (Hasty, n.d.).

3.4.3 Usability Testing

Usability testing is an evaluation method used to measure the usability of a system or technological product by observing and collecting user feedback as they interact with the system. During usability testing, users are asked to complete a series of tasks or scenarios relevant to the system's main functions. Throughout the testing process, users use the provided system interface, and they provide feedback and impressions about their experience interacting with the system (Yigitbas et al., 2019).

When collecting user opinions, a rating scale is needed to measure quantitatively. One commonly used rating scale is the Likert scale with values from 1 to 5. The Likert scale is used to measure opinions or responses by assigning numerical values to levels of agreement or disagreement. On the Likert scale of 1 to 5, a value of 1 represents "Strongly Disagree," and a value of 5 represents "Strongly Agree." This Likert scale is applied to questions relevant to the usability of the system being tested (Joshi et al., 2015).

After users have completed usability testing and provided feedback using the Likert scale of 1 to 5, the data from their responses is processed by calculating the average value of all scores given by users. The average value results provide a quantitative overview of the level of satisfaction and usability of the system according to users. The average score can help the development team evaluate how well the system meets user needs and provides a satisfactory experience.

CHAPTER IV. SYSTEM ANALYSIS AND DESIGN

4.1 System Description

EasyCook is a website that offers personalized recipe recommendations based on ingredient availability and user food preferences. In this system, users can input information about the ingredients available in their kitchen for the recommended recipes. After users provide this input, the system will undergo data analysis using content-based filtering and item-based collaborative filtering algorithms. The system will match the available ingredients with the recipe ingredients and take user preferences into account to determine the most suitable menus.

Upon completing the analysis process, EasyCook will display personalized food menu recommendations based on ingredient availability and user preferences. Users can view a list of dishes that can be prepared with the available ingredients and learn the corresponding recipes. If users find interesting recipes, they can save them as favorites for future reference.

With EasyCook, users can easily discover meal ideas that align with their ingredient availability and preferences. This system assists users in planning and preparing dishes that suit their needs, making it easier to manage daily meal menus.

4.2 System Architecture

EasyCook adopts an integrated and modern system architecture to provide the best user experience. In the backend, EasyCook utilizes Django as the framework for building the server-side, with support from Django Rest Framework (DRF). Django DRF, consisting of models, views, serializers, and URL routing, processes client requests, manages business logic, and provides the required APIs.

In the frontend, EasyCook uses Vue.js as the JavaScript framework that enables the development of responsive and interactive user interfaces. To interact with the Django DRF backend, EasyCook utilizes Axios, an HTTP client library for Vue.js.

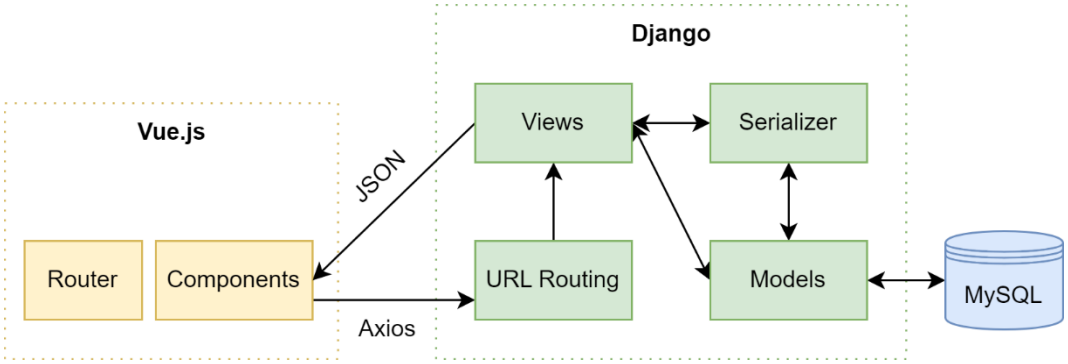


Figure 4.1 EasyCook system architecture

Data exchange in EasyCook begins when users interact with the user interface. For example, if a user wants to input their available ingredients, they can access the web page through the /mykitchen URL, which is set in the Vue.js Router. In this process, Vue.js will display the /mykitchen page view using appropriate components. When the data needs to be displayed, Vue.js sends an HTTP request through Axios to the Django backend using the specified endpoints in the Django DRF URL routing. These requests can be GET requests to fetch data or POST requests to send data to the server.

The Django DRF backend receives and processes these requests according to the defined business logic. This business logic involves data processing using Django models, transformed using serializers, and interaction with the MySQL database. After processing is complete, the Django DRF backend generates a response that is sent back to the Vue.js frontend. Vue.js uses Axios to receive the response from the backend and update the user interface according to the received data. The received response, in JSON object format, can be used to update the display, fill out forms, or show other relevant information.

4.3 System Requirement Analysis

System requirement analysis is the process of identifying and defining the requirements needed for the development of a system. For the EasyCook website, there are functional and non-functional requirements that need to be considered.

4.3.1 Functional Requirements

Functional requirements relate to the features and functions expected from the website. These requirements include:

1. The system can facilitate the registration of new users.
2. The system can enable users to log in to the system.
3. The system can provide access to the Home page.
4. The system can provide recipe recommendations to users.
5. The system can provide access to the recipe detail page.
6. The system can provide access to the History page.
7. The system can provide access to the Favorites page.
8. The system can allow users to add recipes to their favorites list.
9. The system can allow users to remove recipes from their favorites list.
10. The system can allow users to manage ingredient availability.
11. The system can provide recipe search results to users.

4.3.2 Non-Functional Requirements

Non-functional requirements do not directly relate to features but are crucial for the system's performance, security, user interface, performance, and compatibility. These requirements include:

1. The system is equipped with authentication and authorization procedures that ensure only authorized users can access recipe recommendations, manage ingredients, and adjust user preferences.
2. The system can be accessed responsively and is compatible across various devices, enabling users to access it seamlessly on desktops, tablets, and smartphones.
3. The system can provide a user-friendly interface, allowing users to easily access available features.
4. The system can be always accessed by users, ensuring optimal service availability for users.

4.4 Use Case Diagram Modeling

A use case diagram is used to depict and define the features that will be utilized in the system. This aids in system modeling by visualizing the relationships between actors and use cases.

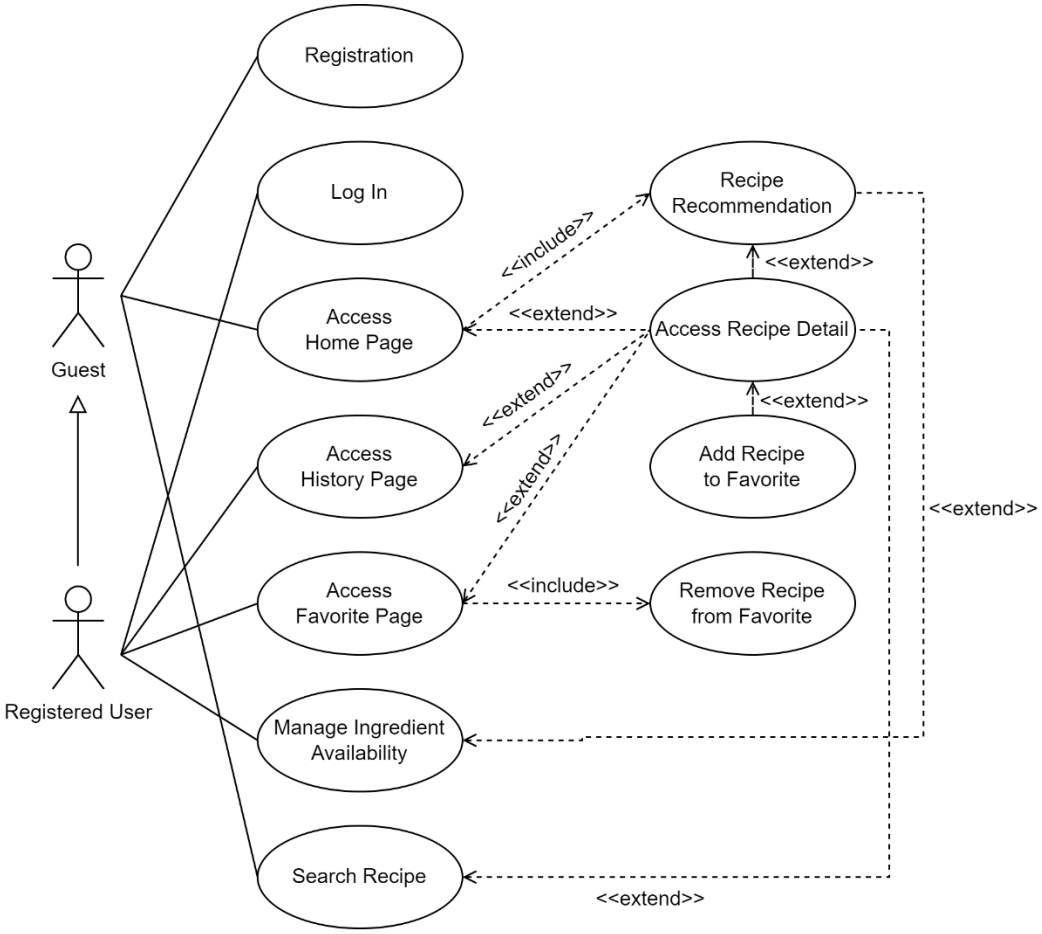


Figure 4.2 EasyCook use case diagram

4.4.1 Actor Definition

Actor definitions describe the role, responsibilities, and interactions between actors and the system. This aims to provide a clearer understanding of how actors are involved in using the system and how they interact with existing features.

Table 4.1 Actor definition

Actor	**Definition**
Guest	A user who is not registered in the system, with limited access to certain features within the system.
Registered User	A user who has registered and has an account in the system. Users access all features and participate more fully in interactions with the system.

4.4.2 Use Case Definition

Use case definitions are used to depict the actions performed by the system and how the interaction between actors and the system occurs.

Table 4.2 EasyCook use case definition

No	Actor	Use Case	Definition
1	Guest	Registration	The process of registering a new user in the system.
2	Guest	Access Home Page	Allows the Guest to access the Home page of the system.
3	Guest	Recipe Recommendation	Provides recipe recommendations based on popularity.
4	Guest	Access Recipe Detail	Views detailed information about a specific recipe.
5	Guest	Search Recipe	Searches for recipes based on keywords.
6	Registered User	Log In	The process of authenticating and logging in a registered user into the system.
7	Registered User	Access Home Page	Allows the Registered User to access the Home page of the system.
8	Registered User	Recipe Recommendation	Provides recipe recommendations based on ingredient availability and user preferences, including ingredient completeness percentage.
9	Registered User	Access Recipe Detail	Views detailed information about a specific recipe.
10	Registered User	Access History Page	Views a list of recipes previously visited by the Registered User.
11	Registered User	Access Favorite Page	Views a list of favorite recipes belonging to the Registered User.

12	Registered User	Add Recipe to Favorite	Adds a recipe to the Registered User's list of favorite recipes.
13	Registered User	Remove Recipe from Favorite	Removes a recipe from the Registered User's list of favorite recipes.
14	Registered User	Manage Ingredient Availability	Manages ingredient availability in the kitchen of the Registered User.
15	Registered User	Search Recipe	Searches for recipes based on keywords, including ingredient completeness percentage.

4.4.3 Use Case Scenario

Use case scenarios are used to depict the sequence of steps or concrete scenarios that describe the interaction between actors and the system in a use case. This helps in understanding in detail how the use case functions and how actors interact with the system in real-world contexts.

4.4.3.1 Registration

Table 4.3 Use case scenario - registration

Use Case Name	**Registration**
Actor	Guest
Goal	Guests can create an account and change their status to active users
Pre-conditions	The user does not have a registered account in the system
Main Flow	1. The user clicks the "Log In" button on the navigation bar. 2. The system displays the Log In page. 3. The user clicks the "Sign Up Now!" button. 4. The system displays the registration page. 5. The user fills out the registration form with required information, such as full name, username, email, and password.

	6. After the user completes all the required information, they click the "Register Account" button to proceed. 7. The system validates the information entered by the user. 8. If the information is valid, the system creates a new account for the user and redirects them back to the Log In page.
Post-conditions	The system can create an account for the guest user and change their status to active users after the registration process is complete

4.4.3.2 Log In

Table 4.4 Use case scenario – log in

Use Case Name	**Log In**
Actor	Registered User
Goal	Allowing registered users to log in to the system
Pre-conditions	The user already has a registered account in the system
Main Flow	1. The user opens the Log In page. 2. The system displays a login form, prompting the user to enter their username and password. 3. The user enters their email address and password. 4. The user clicks the "Log In" button. 5. The system validates the login credentials entered by the user. 6. If the credentials are valid, the system authenticates the user and directs them to the Home page. 7. If the credentials are invalid, the system presents an appropriate error message and requests the user to enter the correct credentials.
Post-conditions	A registered user successfully logs in to the system and is directed to the Home page after logging in

4.4.3.3 Access Home Page

A. Guest

Table 4.5 Use case scenario – access home page for guest

Use Case Name	**Access Home Page**
Actor	Guest
Goal	Displaying recipe recommendations on the home page
Main Flow	1. The user accesses the Home page. 2. The system checks the user's status (Guest or Registered user). 3. If the user is a Guest: a. The system displays recipe recommendations based on the most visited recipes. b. The system displays recipe recommendations based on the most liked recipes by Registered Users.
Post-conditions	The Home page is successfully loaded, showcasing recipe recommendations based on the most visited and liked recipes

B. Registered User

Table 4.6 Use case scenario - access home page for registered user

Use Case Name	**Access Home Page**
Actor	Registered User
Goal	Displaying recipe recommendations on the home page
Main Flow	1. The user accesses the Home page. 2. The system checks the user's status (Guest or Registered User). 3. If the user is a Registered User: a. The system displays recipe recommendations based on the most visited recipes. b. The system displays recipe recommendations based on the most liked recipes by Registered Users.

	c. The system displays additional recipe recommendations that the user might like based on recipes they have previously liked.
Post-conditions	The Home page is successfully loaded, showcasing recipe recommendations based on the most visited and liked recipes. Additionally, the recommendations include other recipes that share similarities with those previously liked by the user

4.4.3.4 Access Recipe Detail

A. Guest

Table 4.7 Use case scenario – access recipe detail for guest

Use Case Name	**Access Recipe Detail**
Actor	Guest
Goal	Guests can view complete details of the selected recipe, including the title, difficulty level, time, servings, food image, recipe source, ingredients, and instructions
Main Flow	1. The system displays the recipe list page. 2. The user clicks on a recipe that catches their attention. 3. The system displays the selected recipe's detail page. 4. The user can view the following information: • Recipe title • Recipe difficulty level • Time required to prepare and cook the recipe • Number of servings produced • Food image • Recipe source • List of required ingredients along with quantities and sizes • Step-by-step instructions to follow the recipe
Post-conditions	The recipe's detail page is successfully loaded and displays complete details of the selected recipe, including the title,

	difficulty level, time, servings, food image, recipe source, ingredients, and instructions

B. Registered User

Table 4.8 Use case scenario – access recipe detail for registered user

Use Case Name	**Access Recipe Detail**
Actor	Registered User
Goal	Registered users can view complete details of the selected recipe, including the title, difficulty level, time, servings, food image, recipe source, ingredients, and instructions
Main Flow	1. The user opens the website. 2. The system displays the list of recipes. 3. The user clicks on a recipe that catches their attention. 4. The system displays the selected recipe's detailed page. 5. The user can view the following information: • Recipe title • Recipe difficulty level • Time required for preparation and cooking • Number of servings produced • Tempting food image • Recipe source • List of required ingredients with quantities and sizes • Step-by-step instructions to follow the recipe 6. The user can add this recipe to their favorite recipe list by pressing a love-shaped button. 7. The user can see a list of owned ingredients marked with a checkmark and those not owned with a cross mark.
Post-conditions	The recipe detail page is successfully loaded and displays complete details of the selected recipe, including the title, difficulty level, time, servings, food image, recipe source, ingredients, and instructions

4.4.3.5 Access History Page

Table 4.9 Use case scenario – access history page

Use Case Name	**Access History Page**
Actor	Registered User
Goal	Registered users can access a history page to view a list of previously visited recipes
Main Flow	1. The user clicks on their profile picture in the navbar. 2. The system displays a dropdown menu with several options. 3. The user selects the "History" option from the dropdown menu. 4. The system processes the request and displays the "History" page with a list of previously visited recipes by the user. 5. The user can view the list of visited recipes.
Post-conditions	The system successfully loads a page displaying a list of recipe history that has been previously visited by the registered user

4.4.3.6 Access Favorite Page

Table 4.10 Use case scenario - access favorite page

Use Case Name	**Access Favorite Page**
Actor	Registered User
Goal	Registered users can access a favorites page to view a list of recipes they have liked
Main Flow	1. The user clicks on their profile picture in the navbar. 2. The system displays a dropdown menu with several options. 3. The user selects the "Favorites" option from the dropdown menu.

	4. The system processes the request and displays the "Favorites" page with a list of recipes liked by the user. 5. The user can view the list of liked recipes. 6. To remove a recipe from the favorites list: • The user clicks the "Love" button or heart icon on the recipe they want to remove from their favorites. • The system processes the request and removes the recipe from the user's favorites list.
Post-conditions	The system successfully loads the favorites page to view a list of recipes that have been liked by the registered user

4.4.3.7 Add Recipe to Favorite

Table 4.11 Use case scenario – add recipe to favorite

Use Case Name	**Add Recipe to Favorite**
Actor	Registered User
Goal	Registered users can easily add recipes to their favorites list, both from the recipe list and from the recipe detail page
Main Flow	1. The user views the available recipe list. 2. The user finds a recipe that catches their interest in the recipe list. 3. The user clicks the "Love" button located near the image of the selected recipe. 4. The system processes the request and adds the recipe to the user's favorites list. 5. If the user wants to view the recipe details first: a. The user clicks the recipe title to open the recipe detail page. b. The system displays the recipe detail page containing complete information about the recipe. c. The user sees the "Love" button on the recipe detail page.

	d. The user clicks the "Love" button on the recipe detail page to add the recipe to their favorites list.
Alternative Flow	1. If the selected recipe is already in the user's favorites list: • The "Love" button will indicate that the recipe has been marked as a favorite. • If the user clicks the "Love" button on a recipe that is already a favorite, the recipe will be removed from the favorites list.
Post-conditions	The system successfully adds recipes to the user's favorites list, both from the recipe list and from the recipe detail page

4.4.3.8 Manage Ingredient Availability

Table 4.12 Use case scenario – manage ingredient availability

Use Case Name	**Manage Ingredient Availability**
Actor	Registered User
Goal	Registered users can manage the availability of ingredients in "My Kitchen," including adding and removing ingredients from the list along with their quantities. The "My Kitchen" page will also display recipe recommendations based on the ingredients available to the registered user
Main Flow	1. The user clicks on the "My Kitchen" menu in the navbar. 2. The system displays the "My Kitchen" page containing the list of ingredients owned by the user. 3. The user can perform the following actions: • Add a new ingredient by entering the ingredient name along with its quantity. • Remove ingredients that are no longer owned by clicking the cross button next to the ingredient to be removed from the list. 4. After the user adds or removes ingredients, the system will save these changes to the ingredient list in "My Kitchen."

	5. The "My Kitchen" page also displays recipe recommendations based on the ingredients available to the user. 6. The user can view the list of recipe recommendations and click on a recipe title to view its details.
Alternative Flow	1. If the ingredient list in "My Kitchen" is empty: • The system will not be able to display recipe recommendations as there are no ingredients owned by the user. • However, the user can still add new ingredients.
Post-conditions	The system successfully loads the "My Kitchen" page, and the registered user can manage their list of owned ingredients. The system also successfully displays recipe recommendations based on the ingredients available to the registered user

4.4.3.9 Search Recipe

A. Guest

Table 4.13 Use case scenario – search recipe for guest

Use Case Name	**Search Recipe**
Actor	Guest
Goal	Guests can search for recipes based on keywords through the search bar in the navbar
Main Flow	1. The user sees the search bar located in the navbar. 2. The user enters the desired keyword into the search bar. 3. The user presses the "Enter" key on the keyboard. 4. The system processes the request and performs a recipe search based on the entered keyword. 5. The system displays search results in the form of a list of recipes that match the searched keyword. 6. The user can view recipe titles and click on a recipe title to view its details.

Post-conditions	The system successfully displays recipe search results based on the entered keyword

B. Registered User

Table 4.14 Use case scenario – search recipe for registered user

Use Case Name	**Search Recipe**
Actor	Registered User
Goal	Registered users can search for food recipes based on keywords through the search bar in the navbar. The search results will be sorted based on ingredient availability
Main Flow	1. The user sees the search bar located in the navbar. 2. The user enters the desired keyword into the search bar. 3. The user presses the "Enter" key on the keyboard. 4. The system processes the request and performs a recipe search based on the entered keyword. 5. The system displays search results in the form of a list of recipes that match the searched keyword, sorted based on the ingredient availability of the user. 6. The user can view recipe titles and click on a recipe title to view its details.
Post-conditions	The system successfully displays recipe search results based on the entered keyword and sorts the results based on the ingredient availability of the registered user

4.5 Pemodelan Activity Diagram

Activity diagrams are a visual representation illustrating the sequence of steps in an activity within a use case. They can also model a series of actions that occur during the execution of an operation, depicting the outcomes of these actions. Through these diagrams, the activities within a system can be understood, from their start to their conclusion.

4.5.1 Account Registration

This activity diagram illustrates the process of registering a new account in the system. The user clicks the "Log In" button to access the Log In page. Then, the user clicks the "Sign up now!" button to navigate to the registration page. On the registration page, the user fills in the required information and clicks the "Register account" button. The system validates the entered information. If the information is valid, the system creates a new account and redirects the user to the Log In page.

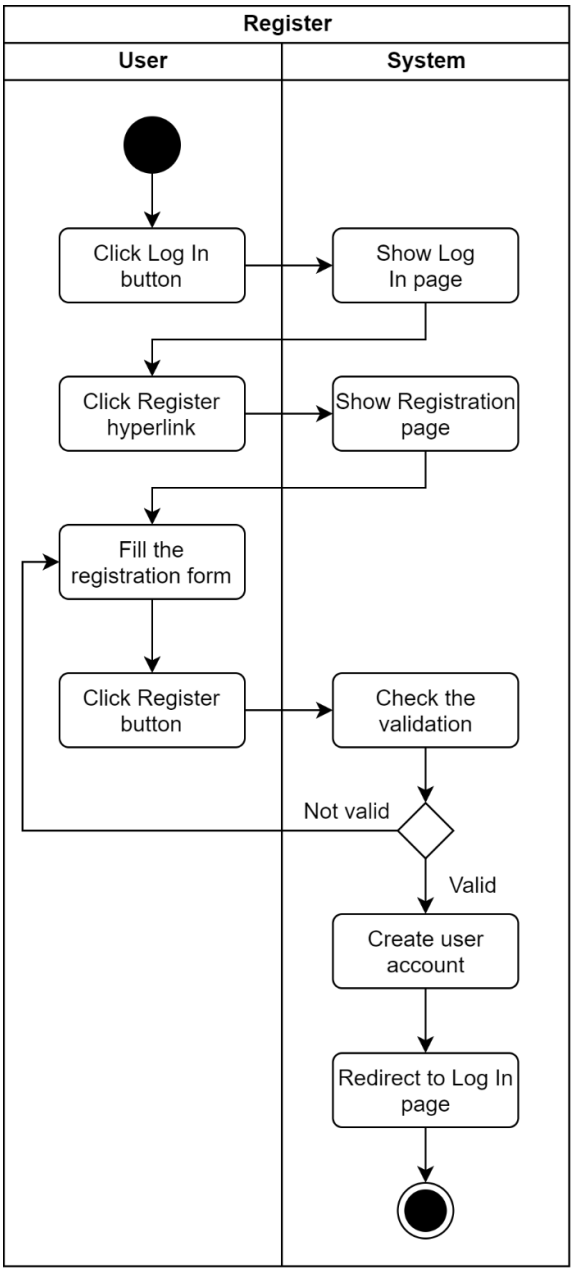


Figure 4.3 Activity diagram - registration

4.5.2 Log In Account

This process illustrates the steps when a user logs in to the system. The user opens the Log In page and enters their email address along with a password. After clicking the "Log In" button, the system validates the login credentials. If the credentials are valid, the user is authenticated and directed to the Home page. If the credentials are invalid, the system displays an error message and prompts the user to enter correct credentials.

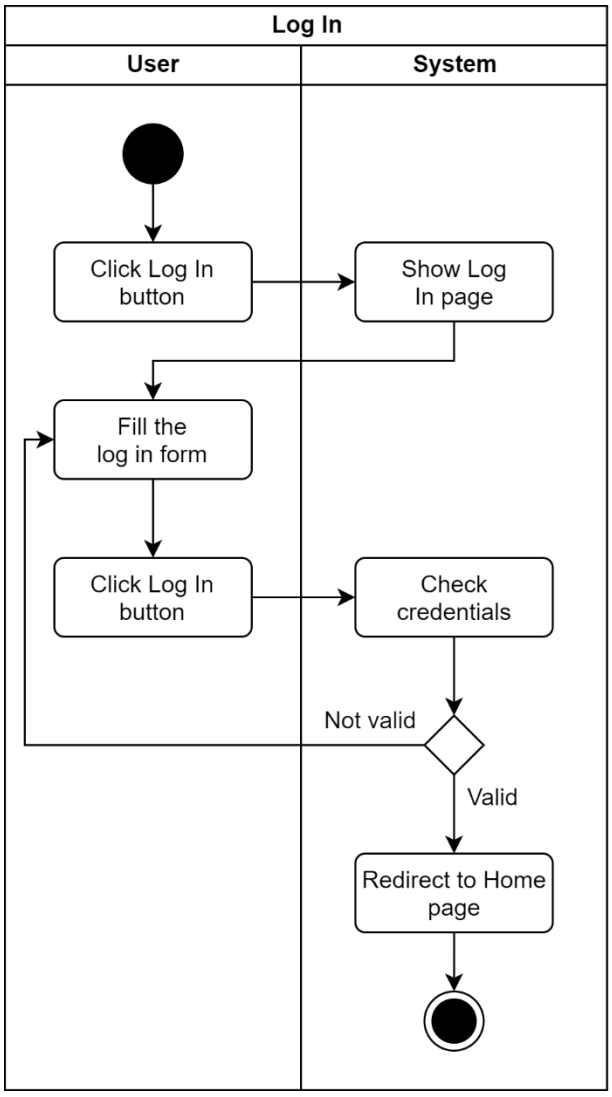


Figure 4.4 Activity diagram – log in

4.5.3 Accessing the Home Page

A. Guest

In this step, the user accesses the Home page. The system then checks the user's status, whether the user is a Guest or a Registered User. If the user is a Guest,

the system will display recipe recommendations based on the most visited recipes and the most liked recipes by Registered Users.

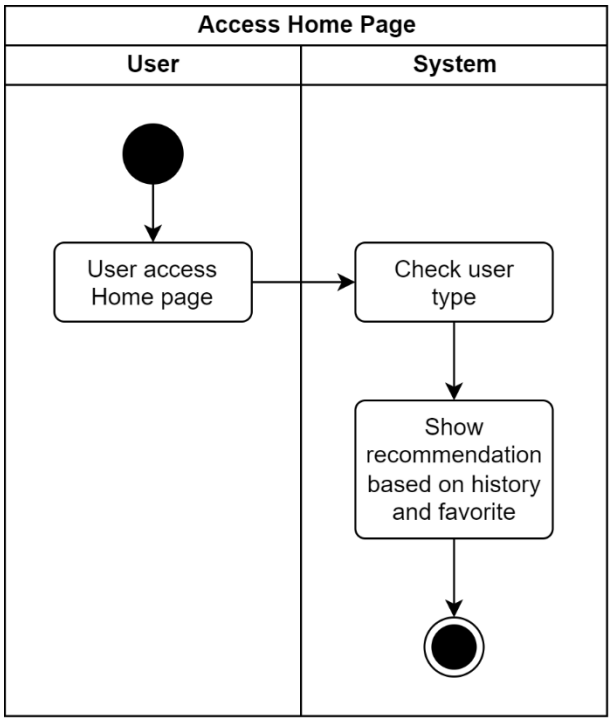


Figure 4.5 Activity diagram – access home page for guest

B. Registered User

In this step, the user accesses the Home page. The system then checks the user's status, whether the user is a Guest or a Registered User. If the user is a Registered User, the system will display recipe recommendations based on user preferences, the most visited recipes, and the most liked recipes.

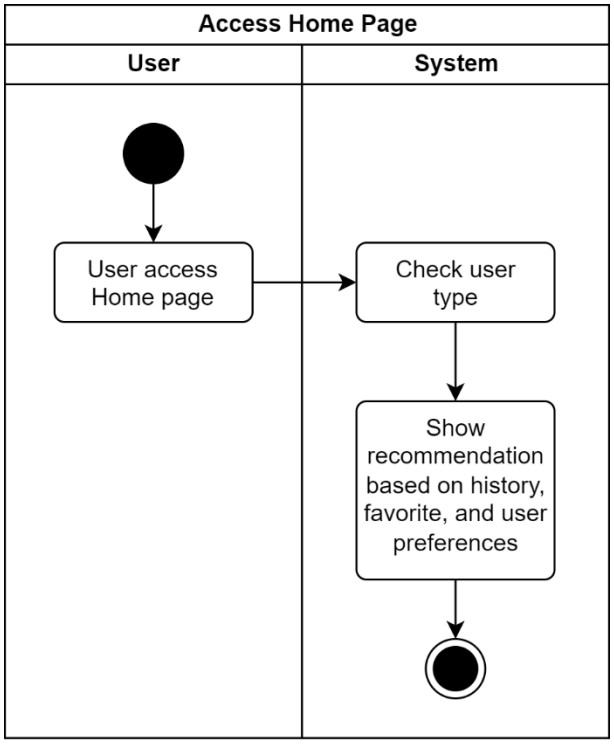


Figure 4.6 Activity diagram – access home page for registered user

4.5.4 Accessing Recipe Details

A. Guest

In this process, the user opens the website, and the system displays a page listing available recipes. Next, the user clicks on a recipe that catches their interest. The system then displays the selected recipe's detail page.

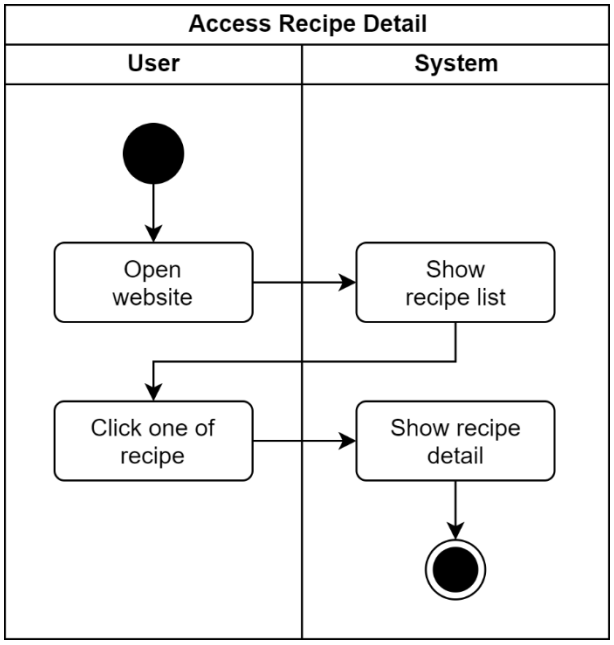


Figure 4.7 Activity diagram – access recipe detail for guest

B. Registered User

In this process, the user opens the website, and the system displays a page listing available recipes. Next, the user clicks on a recipe that catches their interest. The system then displays the selected recipe's detail page. Furthermore, the user can add this recipe to their list of favorite recipes. The user can also view a list of ingredients with checkboxes indicating owned ingredients and crosses indicating missing ones.

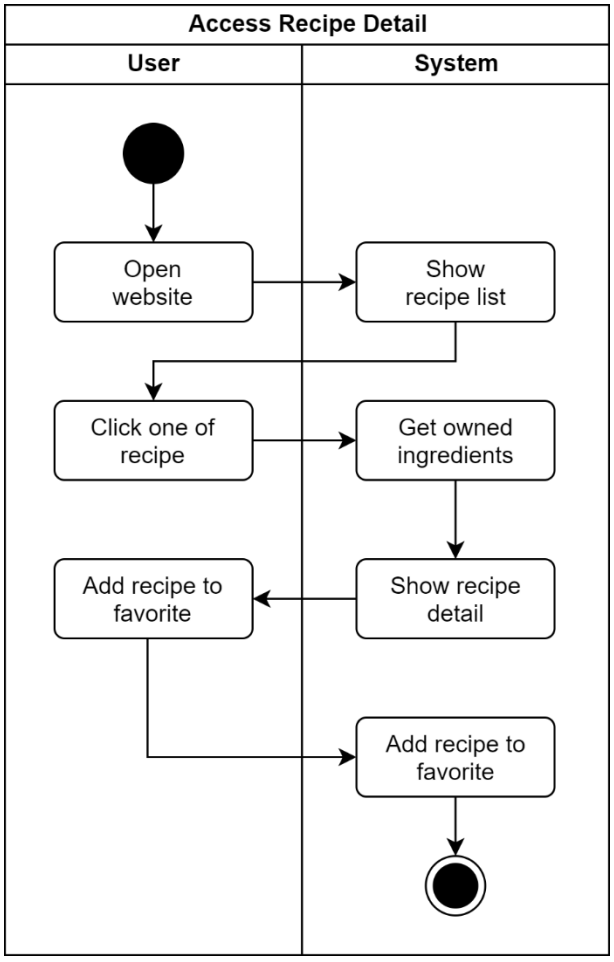


Figure 4.8 Activity diagram – access recipe detail for registered user

4.5.5 Accessing the History Page

In this step, the user clicks on their profile picture in the navigation bar, and the system displays a dropdown menu with several options. The user then selects the "History" option. The system processes the request and displays the "History" page containing a list of recipes the user has previously visited. On this "History" page, the user can view a list of recipes they have visited before.

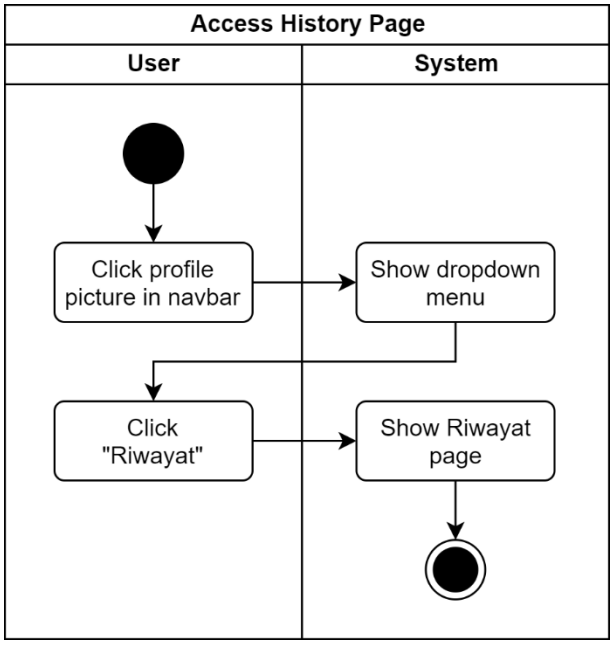


Figure 4.9 Activity diagram - access history page

4.5.6 Accessing Favorite Page

In this step, the user clicks on their profile picture in the navigation bar, and the system displays a dropdown menu with several options. The user then selects the "Favorites" option from the dropdown menu. The system processes the request and displays the "Favorites" page containing a list of recipes liked by the user. On this "Favorites" page, the user can view a list of previously liked recipes and can also remove recipes from the favorites list.

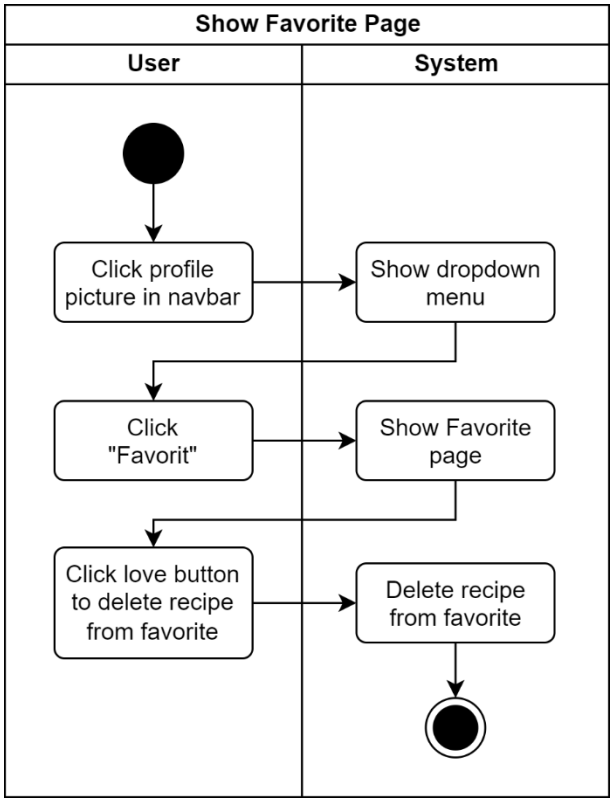


Figure 4.10 Activity diagram – show favorite page

4.5.7 Adding a Recipe to Favorites

In this step, the user views a list of available recipes and finds a recipe that interests them. The user then clicks the "Love" button. The system processes the request and adds the recipe to the user's favorite list. If the user wants to add a recipe to their favorites from the recipe detail page, they must first open the recipe detail and then click the "Love" button to add it to their favorites.

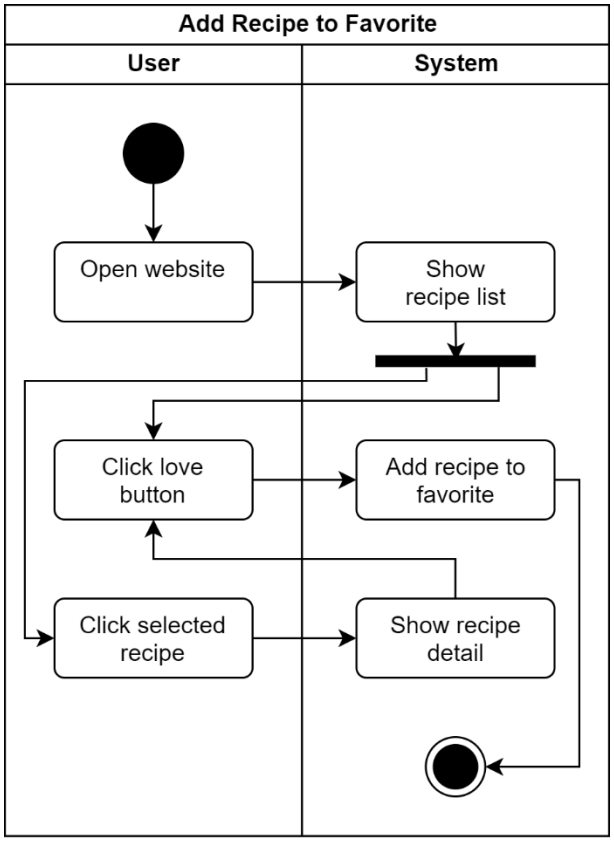


Figure 4.11 Activity diagram – add recipe to favorite

4.5.8 Managing Ingredient Availability

In this step, the user clicks on the "My Kitchen" menu in the navigation bar, and the system displays the "My Kitchen" page containing a list of ingredients owned by the user. The user can manage the owned ingredients and can also view recipe recommendations based on the owned ingredients.

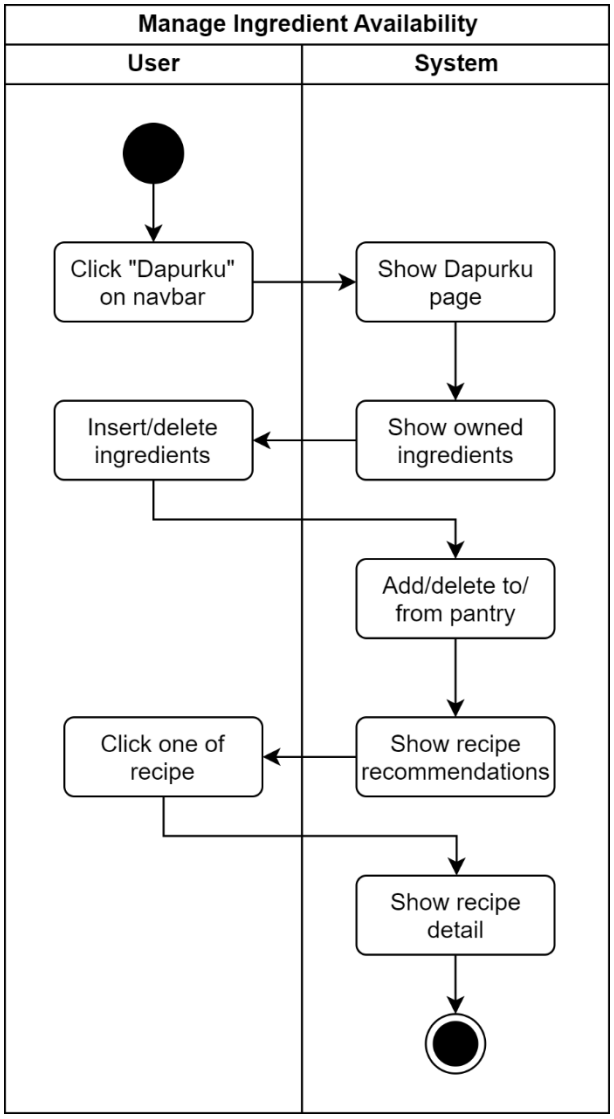


Figure 4.12 Activity diagram – manage ingredient availability

4.5.9 Searching for Recipes

A. Guest

In this step, the user sees a search bar located in the navigation bar and enters a keyword they want to search for into the bar. After entering the keyword, the user presses the "Enter" key on the keyboard. The system then processes the request and searches for recipes based on the entered keyword. After that, the system displays search results in the form of a list of recipes that match the searched keyword.

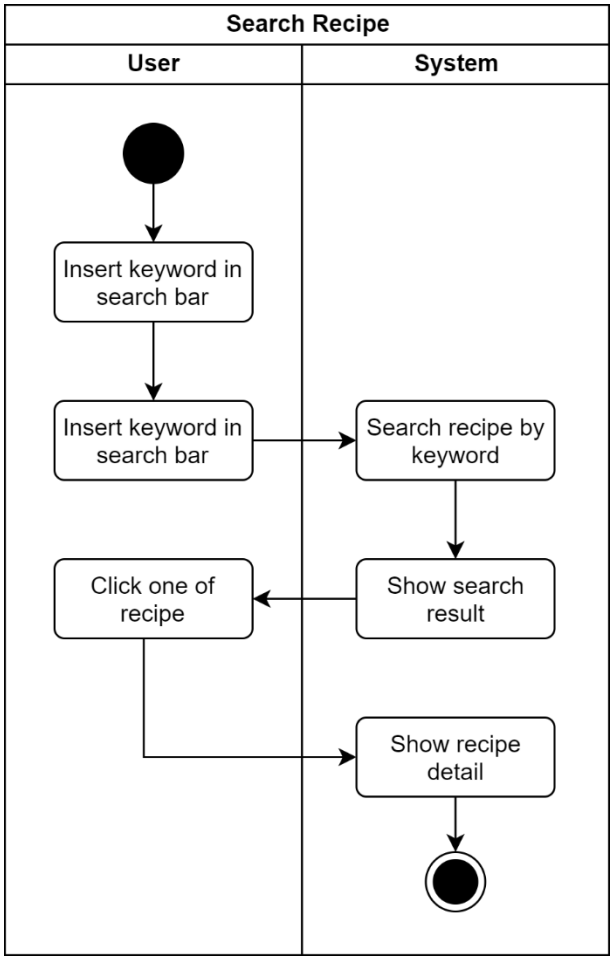


Figure 4.13 Activity diagram – search recipe for guest

B. Registered User

In this step, the user sees a search bar located in the navigation bar and enters a keyword they want to search for into the bar. After entering the keyword, the user presses the "Enter" key on the keyboard. The system then processes the request and searches for recipes based on the entered keyword. After that, the system displays search results in the form of a list of recipes that match the keyword, sorted by the percentage of ingredient availability for the user.

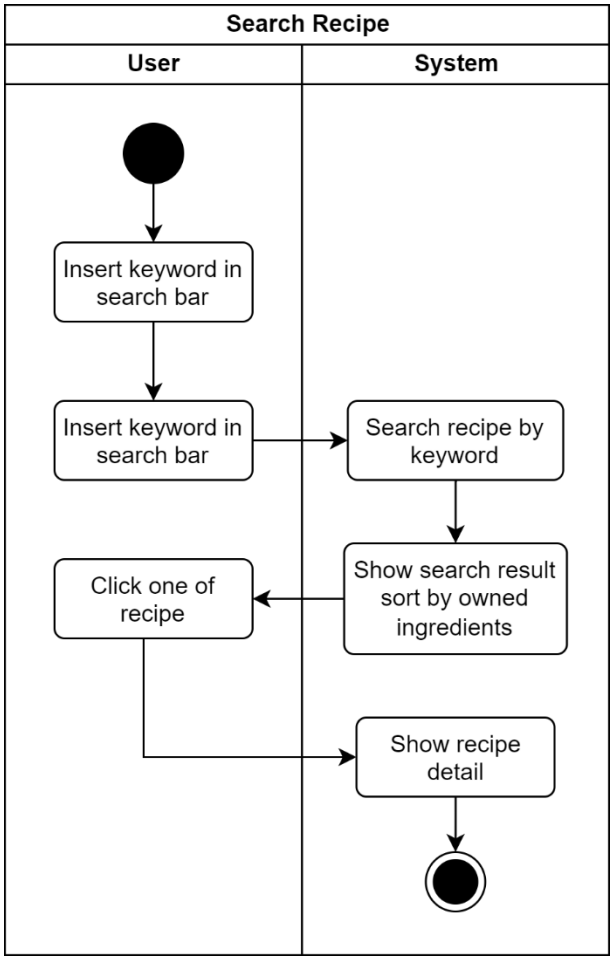


Figure 4.14 Activity diagram – search recipe for registered user

4.6 Entity Relationship Diagram Modeling

The Entity Relationship Diagram (ERD) is used to depict relationships between entities in a database modeling context. In the context of the EasyCook application, the ERD helps visualize relationships between entities such as User, User Profile, and Recipe. By categorizing these entities into distinct groups, the database structure can be efficiently designed, making data management easier for presenting recipe recommendations that align with user preferences.

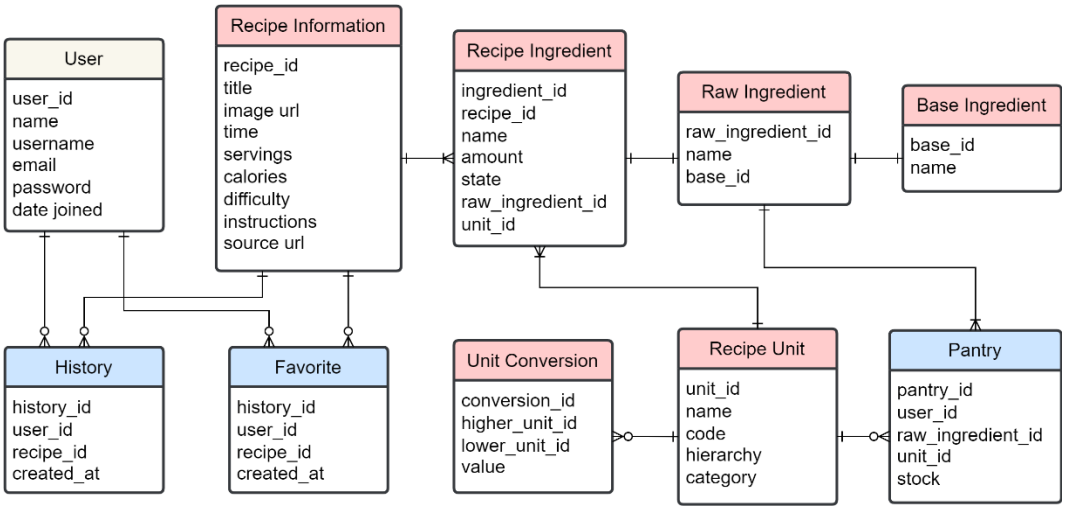


Figure 4.15 EasyCook ERD Design

CHAPTER V. IMPLEMENTATION AND TESTING

5.1 Web Scraping Implementation

Web scraping is a technique used to automatically extract information from web pages. In its implementation using Python, essential modules such as Pandas, BeautifulSoup, and Requests are needed to manage and process the data extracted from the web pages.

5.1.1 Obtaining Recipe URLs

To obtain a list of recipe URLs from the website https://www.masakapahariini.com/resep/, the initial step is to define the number of pages to be explored and the base URL of the website. Next, a looping process is performed to identify the "href" attribute of each recipe on these pages using BeautifulSoup. Once found, these URLs are stored in an array for processing in the next step.

```
# Function to get recipe's URL from each pages
def get_recipe_urls(url):
  page = requests.get(url)
  soup = BeautifulSoup(page.content, 'html.parser')

  urls = []
  recipe_cards = soup.find_all('div', class_='_recipe-card')

  for recipe_card in recipe_cards:
    link_element = recipe_card.find('h3', class_='card-
title').find('a')
    urls.append(link_element['href'])

  return urls

# Get all URL
recipe_urls = []
base_url = 'https://www.masakapahariini.com/resep/'
page_number = 134

for n in range(1, page_number):
  url = base_url if n == 1 else base_url + 'page/' + str(n)
```

```
  recipe_urls += get_recipe_urls(url)
```

5.1.2 Obtaining Detailed Recipe Information

After obtaining the list of recipe URLs from the masakapahariini website, the next step is to perform web scraping to extract recipe components from each recipe page. This web scraping process involves parsing or extracting information from each recipe component found.

Each successfully extracted recipe component is stored in the form of a dictionary (dict). This dictionary is then used to form a dataframe, where each recipe component becomes a row in that dataframe.

```
# RecipeScraper code in appendix 1
# Get all recipe detail
scraper = RecipeScraper()
df = pd.DataFrame([scraper.scrape_recipe(url) for url in
recipe_urls]) df.drop_duplicates(subset=['title', 'source_url'],
inplace=True)
df.reset_index(drop=True, inplace=True)
```

5.1.3 Defining Recipe IDs

IDs for each recipe are generated randomly based on the recipe title using the random.randint() function. Subsequently, each generated ID is verified to ensure its uniqueness and stored in the 'id' column of the created DataFrame.

```
# Generate ID by title
def generate_id_by_title(df):
  unique_titles = df['title'].unique()

  new_ids = {title: random.randint(10_000_000_000,
99_999_999_999) for title in unique_titles}

  while len(set(new_ids.values())) != len(unique_titles):
    for title in unique_titles:
      if new_ids.count(title) > 1:
        new_ids[title] = random.randint(10_000_000_000,
99_999_999_999)

  df['id'] = df['title'].map(new_ids)
  return df
```

```
# Generate the id by title and save to df
df = generate_id_by_title(df)
```

5.1.4 Obtaining Recipe Ingredient Data

To store recipe ingredient data, a new dataframe named df_ingredients is created. This dataframe is specifically used to store information about recipe ingredients. Within this dataframe, there are two columns: "id" and "ingredients." The "id" column contains unique identifications for each ingredient, while the "ingredients" column contains an array listing the ingredients for each recipe.

Furthermore, to structure and manage the ingredient data more effectively, the df_ingredients.explode() process is performed on the "ingredients" column. This process splits the data in the "ingredients" column into several new rows, so that each new row contains one ingredient from a specific recipe. Additionally, the "id" column, which previously originated from recipe data, is renamed to "recipe_id" to better fit the context of the ingredient dataframe.

```
# Create new dataframe for ingredients
df_ingredients = df.loc[:, ['id', 'ingredients']]

# Explode array
df_ingredients['ingredients'] =
df_ingredients['ingredients'].apply(lambda x:
ast.literal_eval(x))
df_ingredients = df_ingredients.explode('ingredients')

# Reset index
df_ingredients = df_ingredients.reset_index()
df_ingredients = df_ingredients.drop('index', axis=1)
df_ingredients.rename(columns = {'id':'recipe_id',
'ingredients': 'name'}, inplace = True)
df_ingredients['id'] = df_ingredients.index + 1
```

5.2 Text Preprocessing Implementation

Text preprocessing in this phase aims to produce clean, structured data ready for further processing. The data to be processed includes recipe data, ingredient data, and unit data.

5.2.1 Recipe Data

To clean the text in the recipe data, a cleaning process is performed using the str.split method in Python. This cleaning process involves splitting the text in the title column and performing splits on calorie data as well as converting time data formats. The first step is to clean the recipe title column. By using str.split, the text in the title column can be separated based on spaces or specific characters, allowing it to be accessed and manipulated further. Subsequently, str.split is also used to separate data in the calorie column and convert the time format. The manipulated data is then stored in a .csv format and imported into MySQL.

Text preprocessing pada tahap ini bertujuan untuk menghasilkan data yang bersih, terstruktur, dan siap untuk diproses lebih lanjut. Data yang perlu diolah adalah data resep, data bahan, serta data satuan.

```
# Clean title
df_recipes['title'] = df_recipes['title'].str.split(',',
n=1).str[0]
df_recipes['title'] = df_recipes['title'].str.split('yang',
n=1).str[0]
df_recipes['title'][df_recipes['title'].str.contains("Resep")] =
df_recipes['title'].str.split('Resep', n=1).str[1]
df_recipes['title'][df_recipes['title'].str.contains("Cara
Memasak")] = df_recipes['title'].str.split('Cara Memasak',
n=1).str[1]
df_recipes['title'][df_recipes['title'].str.contains("Cara
Masak")] = df_recipes['title'].str.split('Cara Masak',
n=1).str[1]
df_recipes['title'][df_recipes['title'].str.contains("Cara
Membuat")] = df_recipes['title'].str.split('Membuat',
n=1).str[1]

# Clean calories
df_recipes['calories'] =
df_recipes['calories'].str.split('Kkal', n=1).str[0]

# Transform time format
df_recipes['time'] = df_recipes['time'].str.split('mnt',
n=1).str[0]
```

```
df_recipes['jam'] = 0
df_recipes['jam'][df_recipes['time'].str.contains("jam|j")] =
df_recipes['time'].str.split('jam|j', n=1).str[0]
df_recipes['time'][df_recipes['time'].str.contains("jam|j")] =
df_recipes['time'].str.split('jam|j', n=1).str[1]
df_recipes['time'][df_recipes["time"] == ''] = 0
df_recipes['time'] = df_recipes['time'].astype("int32") +
(df_recipes['jam'].astype("int32") * 60)
```

5.2.2 Recipe Ingredient Data and Unit Data

Once the recipe ingredient data is obtained, tokenization is performed using regex, which is divided into several parts:

1. recipe_id: ID resep.
2. amount: The quantity of ingredients required for the recipe.
3. name: The name of the ingredient used in the recipe.
4. unit: The unit of measurement for the ingredient, such gram, *sendok teh*, etc.
5. state: The state of the ingredient, such as "*cincang halus*", "*potong kecil*", etc.

To manage unit data in the ingredient data, a new dataframe called df_unit is created. After separating the unit data from the "unit" column in the original dataframe, the next step is to format the unit names to match the desired format. Once a new dataframe is created, reference is made to the ingredient data using the unit_id. Finally, this data is stored in .csv format and imported into MySQL.

```
# Tokenization the amount and ingredients
df_ingredients["amount"] =
df_ingredients["name"].str.extract(r'^(\d+(?:\.\d+)?\s?\d*/?\d*)
')
df_ingredients["name"] =
df_ingredients["name"].str.replace(r'^\d+(?:\.\d+)?\s?\d*/?\d*\s
?', '')

# code is from dict
# Get unit
df_unit = pd.DataFrame.from_dict(code, orient='index')
df_unit = df_unit.reset_index()
```

```
df_unit = df_unit.rename(columns={"index": "code", 0: "name"})
df_unit['id'] = df_unit.index + 1
df_unit = df_unit[['id', 'code', 'name']]
code_map = df_unit.set_index('code').T.to_dict('index')

# Split ingredient based on unit
df_ingredients['unit_id'] = df_ingredients['name'].str.split('
').str[0].str.split(',').str[0]
df_ingredients.head()

# replace unit name into default unit name
df_ingredients['unit_id'] =
df_ingredients['unit_id'].replace({'liter': 'L'}, regex=True)
df_ingredients['unit_id'] =
df_ingredients['unit_id'].replace({'gram': 'g'}, regex=True)
df_ingredients['unit_id'] =
df_ingredients['unit_id'].replace({'gr': 'g'}, regex=True)

# Set unit to unit_id
code_map_lower = {k.lower(): v for k, v in
code_map['id'].items()}
df_ingredients['unit_id'] =
df_ingredients['unit_id'].astype(str).str.lower().map(code_map_l
ower)

# If unit id is not null, delete first word on ingredient name
df_ingredients['name'] =
np.where(df_ingredients['unit_id'].notna(),
df_ingredients['name'].str.split(n=1).str[1],
df_ingredients['name'])
```

5.2.3 Ingredient Data Grouping

Grouping ingredient names is done to identify the basic names of cooking ingredients. This grouping process involves searching for common writing patterns, such as ingredient size or color, in the ingredient data. Once these patterns are identified, the ingredient data is categorized into several groups according to their purpose. This is done to simplify and organize the ingredient data, making it easier to analyze and utilize ingredient information in various recipes.

After coding the grouping process to identify the basic names of cooking ingredients, the next step is to import a list of ingredient names in list format, convert the names to lowercase, and then perform grouping using the previously created IngredientGrouper(). Once these patterns are identified, the ingredient data will be categorized into several groups according to their purpose. The results of this grouping will then be exported in JSON format and imported into the MySQL database.

```
# Read the CSV file into a DataFrame
df = pd.read_csv('ingredients-c5.csv', sep=',')
df = df.dropna(subset=['name'])

# Select only the 'ingredients' column
df = df[['name']]

# Apply lowercasing to the 'ingredients' column
df['name'] = df['name'].str.lower()

# Remove duplicates from the DataFrame
df.drop_duplicates(subset=['name'], inplace=True)

# Change ingredients to list
ingredients = df['name'].tolist()
ingredients = sorted(set(ingredients), key=lambda x:
(get_word_count(x), x))

# IngredientGrouper() in appendix 2
# Grouping the ingredients
grouper = IngredientGrouper()
result = grouper.group_ingredients(ingredients)
json_str = json.dumps(result, indent=4)

# Save the result as a JSON file
with open('output.json', 'w') as json_file:
    json.dump(result, json_file, indent=4)
```

5.3 Implementation of User Ingredient Ownership of Recipes

This algorithm is designed to obtain a list of ingredients owned by the user, which will be compared with the number of ingredients in a recipe. It starts by converting the list of user-owned ingredients and the list of recipe ingredients. Next, this algorithm will group these ingredients based on predetermined rules. This algorithm is used in various functions, such as displaying the list of owned ingredients on the recipe detail or getting the number of ingredients owned by the user. The algorithm can be seen in appendix 3.

5.4 Implementation of User Ingredient Ownership Percentage

This algorithm displays the percentage of ingredient ownership the user has for the required ingredients in a recipe on the list of recipes. This information allows users to quickly determine how many of the required ingredients they already possess for a particular recipe.

```
def get_completeness_percentage(self, instance):
    user = self.context['request'].user

    if user.is_authenticated:
        owned_ingredient_ids = len(get_owned_recipe_ingredients(
            user, instance.pk))

        total_ingredient_count = instance.ingredients.count()

        if total_ingredient_count > 0:
            completeness_percentage = (
                owned_ingredient_ids / total_ingredient_count *
100
            )
            completeness_percentage = round(
                completeness_percentage, 2)
            return completeness_percentage

    return 0
```

5.5 Recipe Recommendation Implementation

Recipe recommendation is a feature provided to offer users personalized recipe recommendations based on their preferences, and the results are displayed in the form of a list of recipes tailored to the user's preferences.

5.5.1 Recommendation Based on Ingredient Availability

To provide recipe recommendations based on the user's owned ingredients, this algorithm compares the user's owned ingredients with the required ingredients in a recipe. This comparison is done using the Jaccard Similarity method to calculate the similarity level between two sets of ingredients. Jaccard Similarity calculates the percentage of similarity between two sets by dividing the number of common elements in both sets by the total number of unique elements from both sets.

```
def jaccard_similarity(set1, set2):
    intersection = len(set1.intersection(set2))
    union = len(set1.union(set2))
    similarity = intersection / union
    return similarity


def calculate_top_recipes_by_ingredients(user):
    owned_raw_ingredient = set(get_owned_raw_ingredients(user))

    # Get valid recipes
    recipe_ingredients = Ingredient.objects.filter(
        raw_ingredient_id__in=owned_raw_ingredient,
        raw_ingredient__pantry__user=user
    ).values('recipe_id', 'raw_ingredient_id',
user_id=F('raw_ingredient__pantry__user_id'))

    data = {}
    for ingredient in recipe_ingredients:
        user_id = ingredient['user_id']
        recipe_id = ingredient['recipe_id']
        raw_ingredient_id = ingredient['raw_ingredient_id']

        if recipe_id in data:
```

```
data[recipe_id]['raw_ingredient_ids'].add(raw_ingredient_id)
        else:
            data[recipe_id] = {
                'user_id': user_id,
                'recipe_id': recipe_id,
                'raw_ingredient_ids': {raw_ingredient_id}
            }
    for recipe_id, recipe_data in data.items():
        raw_ingredient_ids = recipe_data['raw_ingredient_ids']
        similarity = jaccard_similarity(
            owned_raw_ingredient, raw_ingredient_ids)
        recipe_data['similarity'] = similarity * 100

    sorted_recipes = sorted(
        data.values(), key=lambda x: x['similarity'],
reverse=True)

    return sorted_recipes
```

In views.py, this algorithm is called when the "My Kitchen" page is accessed. The results of this similarity calculation are sorted based on the highest similarity level and structured in JSON format. Subsequently, the JSON data containing the list of recipe recommendations based on the user's owned ingredients will be sent as output to the "My Kitchen" page.

```
class PantryRecommendationView(generics.GenericAPIView):
    permission_classes = [IsAuthenticated]

    def get(self, request):
        user = request.user

        top_recipes =
calculate_top_recipes_by_ingredients(user)[:20]
        top_recipe_ids = [recipe['recipe_id'] for recipe in
top_recipes]
        recipes =
Information.objects.filter(id__in=top_recipe_ids).annotate()
        recipes_list = RecipeInformationSerializer(
```

```
            recipes, context={'request': request, 'top_recipes':
top_recipes}, many=True).data

        filtered_recipes = [
            recipe for recipe in recipes_list if
recipe['completeness_percentage'] > 0]

        sorted_recipes = sorted(
            filtered_recipes,
            key=lambda x: (0.8 * next(
                (item['similarity'] for item in top_recipes if
item['recipe_id'] == x['id']), 0)) +
            (0.2 * x['completeness_percentage']),
            reverse=True
        )

        sorted_recipes = sorted_recipes[:12]
        return Response(sorted_recipes,
status=status.HTTP_200_OK)
```

5.5.2 Recommendation Based on User Preferences

Recipe recommendations based on user preferences are obtained from a list of recipes that the user has liked. This algorithm compares the recipes liked by the user with recipes liked by other users. This comparison is done using the Cosine Similarity method to calculate the similarity level between two vectors representing the recipes that have been liked by users.

```
# Interaction matrix
def create_interaction_matrix():
    favorites = Favorite.objects.all()
    user_ids = []
    recipe_ids = []

    for favorite in favorites:
        user_ids.append(favorite.user_id)
        recipe_ids.append(favorite.recipe_id)

    unique_user_ids = list(set(user_ids))
    unique_recipe_ids = list(set(recipe_ids))
```

```
    interaction_matrix = pd.DataFrame(
        0, index=unique_user_ids, columns=unique_recipe_ids)

    for i in range(len(user_ids)):
        user_id = user_ids[i]
        recipe_id = recipe_ids[i]
        interaction_matrix.loc[user_id, recipe_id] = 1

    return interaction_matrix

# Calculate cosine similarity
def calculate_cosine_similarity():
    interaction_matrix = create_interaction_matrix()
    similarity_matrix = cosine_similarity(interaction_matrix.T)
    similarity_df = pd.DataFrame(
        similarity_matrix, index=interaction_matrix.columns,
columns=interaction_matrix.columns)
    return similarity_df, interaction_matrix

def get_recommendations_for_user(user_id):
    similarity_matrix, interaction_matrix =
calculate_cosine_similarity()

    if user_id in interaction_matrix.index:
        scores = np.dot(interaction_matrix.values,
similarity_matrix)

        recommended_items = [interaction_matrix.columns[i]
                             for i in
np.argsort(scores[interaction_matrix.index == user_id,
:])[0][::-1]]

        recommendations = []

        min_score = np.min(scores)
        max_score = np.max(scores)

        if min_score == max_score:
```

```
            return []

        # Normalize the scores to a range of 0-100
        normalized_scores = 100 * \
            (scores - min_score) / (max_score - min_score)

        for item in recommended_items:
            score = scores[interaction_matrix.index ==
                           user_id, interaction_matrix.columns
== item][0]
            similarity_score =
normalized_scores[interaction_matrix.index ==
                                                  user_id,
interaction_matrix.columns == item][0]
            if score > 0:
                recommendation = {
                    'recipe_id': item,
                    'similarity': round(similarity_score, 2),
                    'score': round(score, 2)
                }
                recommendations.append(recommendation)
        return recommendations
    else:
        return []
```

Once the Cosine Similarity calculation is complete, the list of recipes is sorted based on the highest similarity level. These recipe recommendations are organized as a list and called in views.py to be displayed on the Home page.

```
class CFRecommendationView(generics.GenericAPIView):
    permission_classes = [IsAuthenticated]
    serializer_class = RecipeInformationSerializer

    def get(self, request):
        user_id = self.request.user.id
        top_recipes = get_recommendations_for_user(user_id)[:20]
        top_recipes_ids = [recipe['recipe_id']
                           for recipe in top_recipes]

        recipes = Information.objects.filter(
```

```
            id__in=top_recipes_ids)
        recipes_list = RecipeInformationSerializer(
            recipes, context={'request': request, 'top_recipes':
top_recipes}, many=True).data
        print(recipes_list)
        sorted_recipes = sorted(
            recipes_list, key=lambda x: x['similarity'],
reverse=True)
        return Response(sorted_recipes,
status=status.HTTP_200_OK)
```

5.6 Database Implementation

The database implementation is based on the previously designed ERD. This database is created through migration from the Django framework, connected to MySQL. However, table names in the database differ from the ERD due to Django's specific table naming format. Initial data required for the database includes recipe information, ingredients, and units. This data is obtained through data processing from the previous stages in CSV format. Afterward, this data is imported into MySQL using the data import feature provided by MySQL.

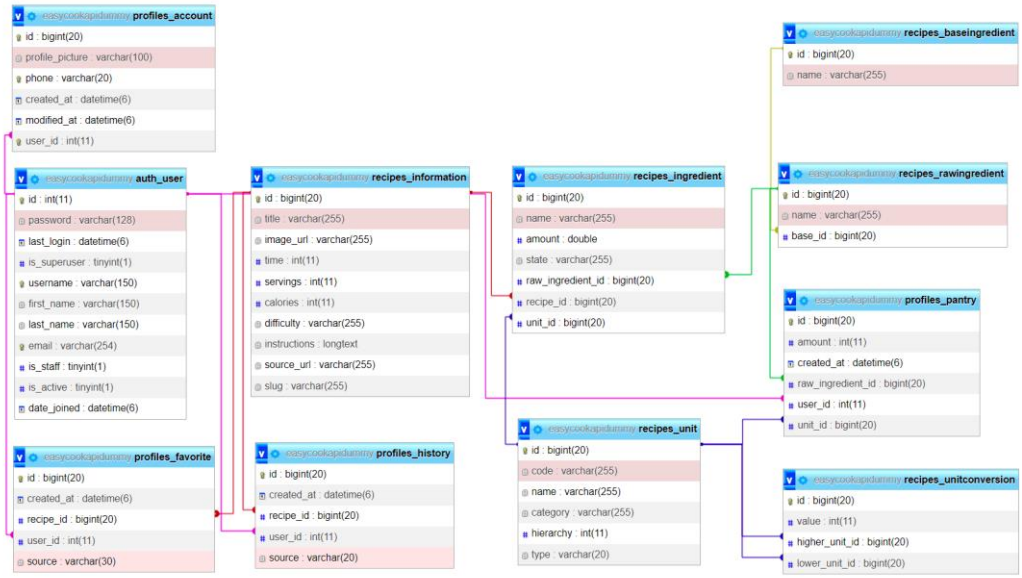


Figure 5.1 Database diagram on MySQL

5.7 Interface Implementation

The interface is used to present the system's usage to users. The developed interface is web-based and implemented using Vue.js, connected to the EasyCook

API developed using the Django framework. Through this integration, users can easily access and interact with various features provided by the system.

5.7.1 Register Page

The register page allows new users to create accounts in the system. Users are required to fill out a form with information such as name, username, email, and password. After filling out all the necessary information and submitting the form, the data is processed, and a new account is created for the user in the system.

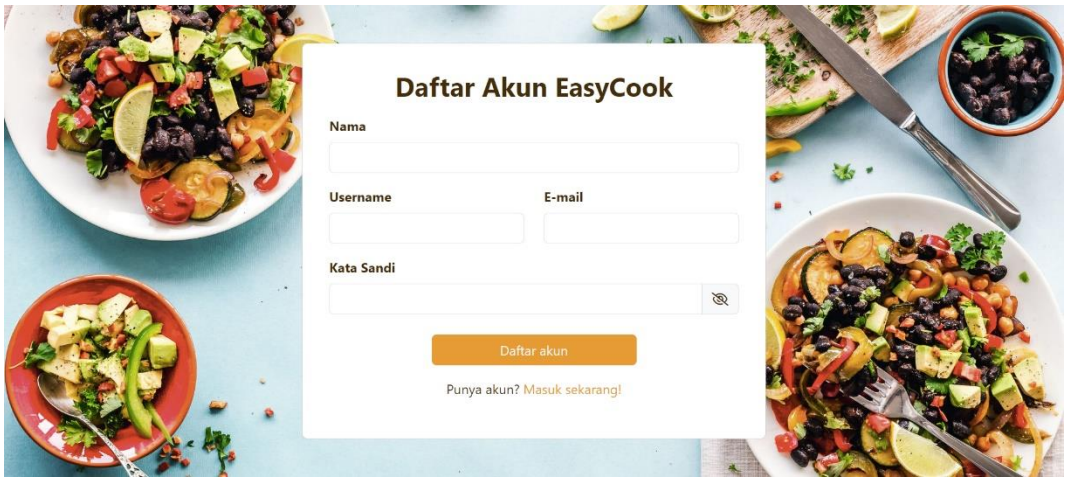


Figure 5.2 Register page view

5.7.2 Log In Page

The login page is used to allow users who already have registered accounts to log into the system. On the login page, users are prompted to fill out a form with the required information, namely their username and password. If the entered information matches the existing data, the user will be directed to the Home page.

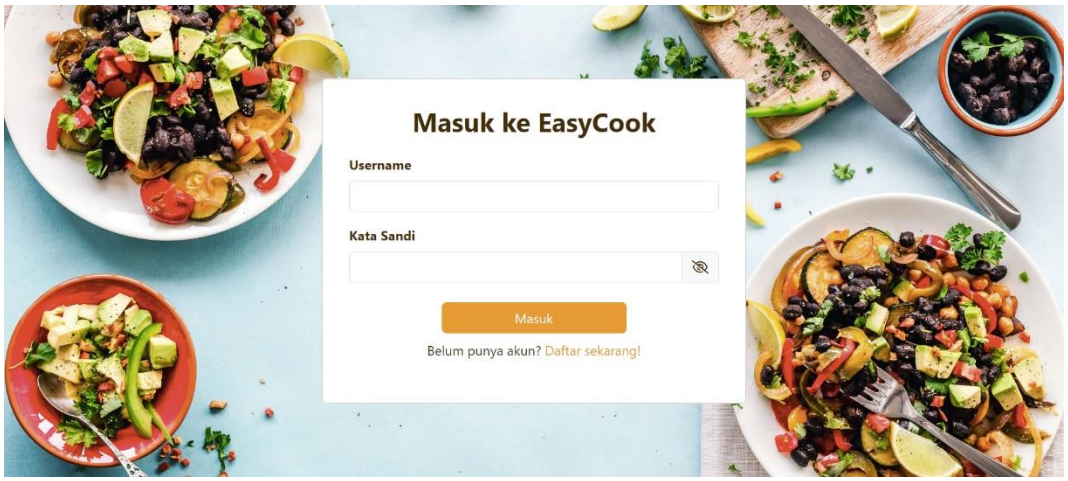


Figure 5.3 Login Page View

5.7.3 Home Page

The Home page is designed to display various recipe recommendations to users. These recommendations include:

1. "Mungkin Kamu Suka": The system provides recipe recommendations based on the preferences of the respective user. The system analyzes recipes that the user has previously liked using the item-based collaborative filtering method with cosine similarity. Based on this analysis, the system offers recipe recommendations that are highly likely to be liked by the user.
2. "Paling Disukai Pengguna": Recipe recommendations based on recipes that have been frequently added to the favorites list by users. The system presents recipes that many users have included in their favorites.
3. "Paling Sering Dilihat": Recipe recommendations based on the highest number of visits or views by all users.

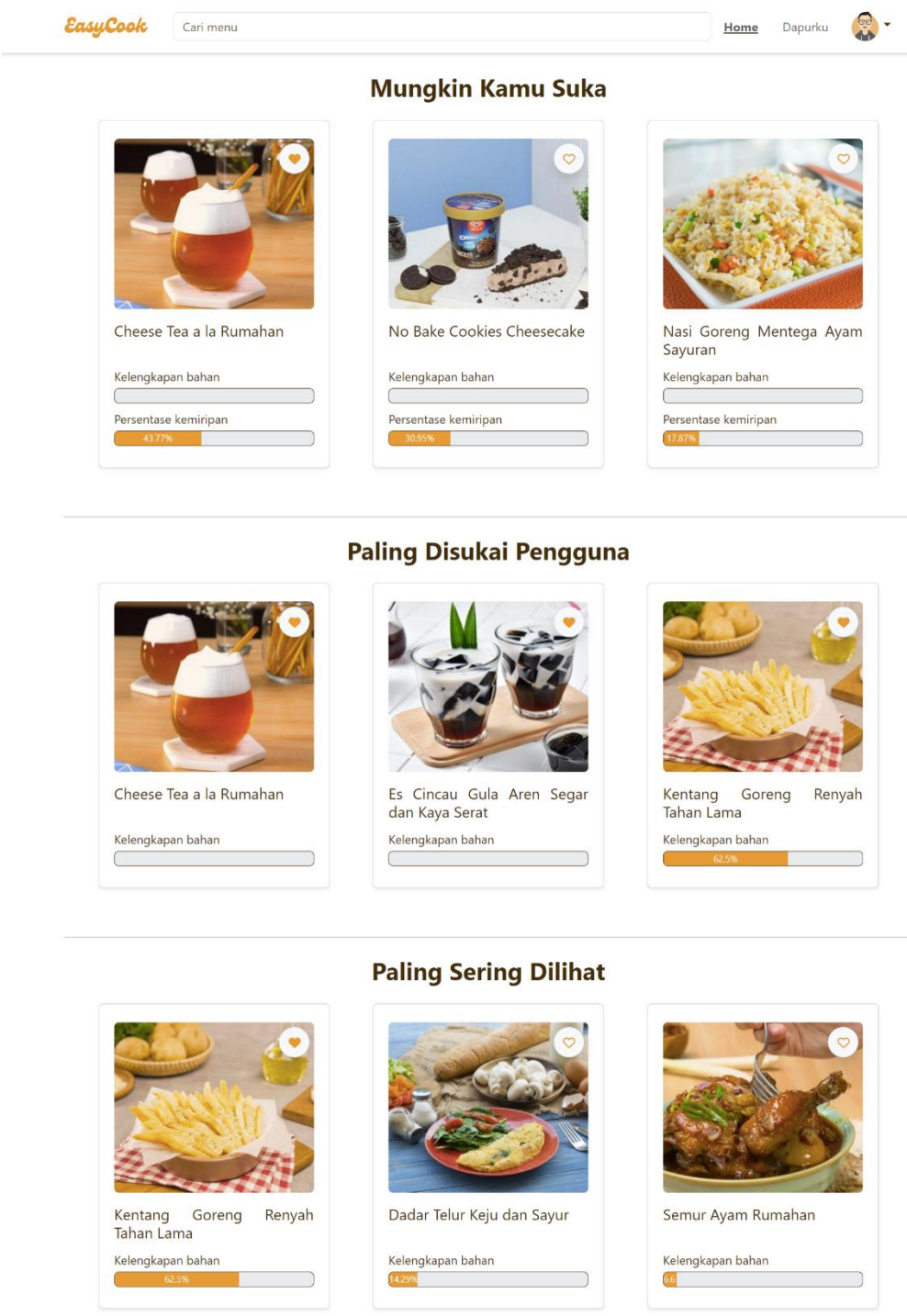


Figure 5.4 Home page view

5.7.4 My Kitchen Page

The Kitchen page is a feature that allows users to store a list of ingredients they own. Users have the flexibility to add or remove ingredients according to their kitchen inventory and can also input the quantity of available ingredients. Additionally, this page presents relevant recipe recommendations based on the

user's owned ingredients. These recommendations are generated using content-based filtering with Jaccard similarity.

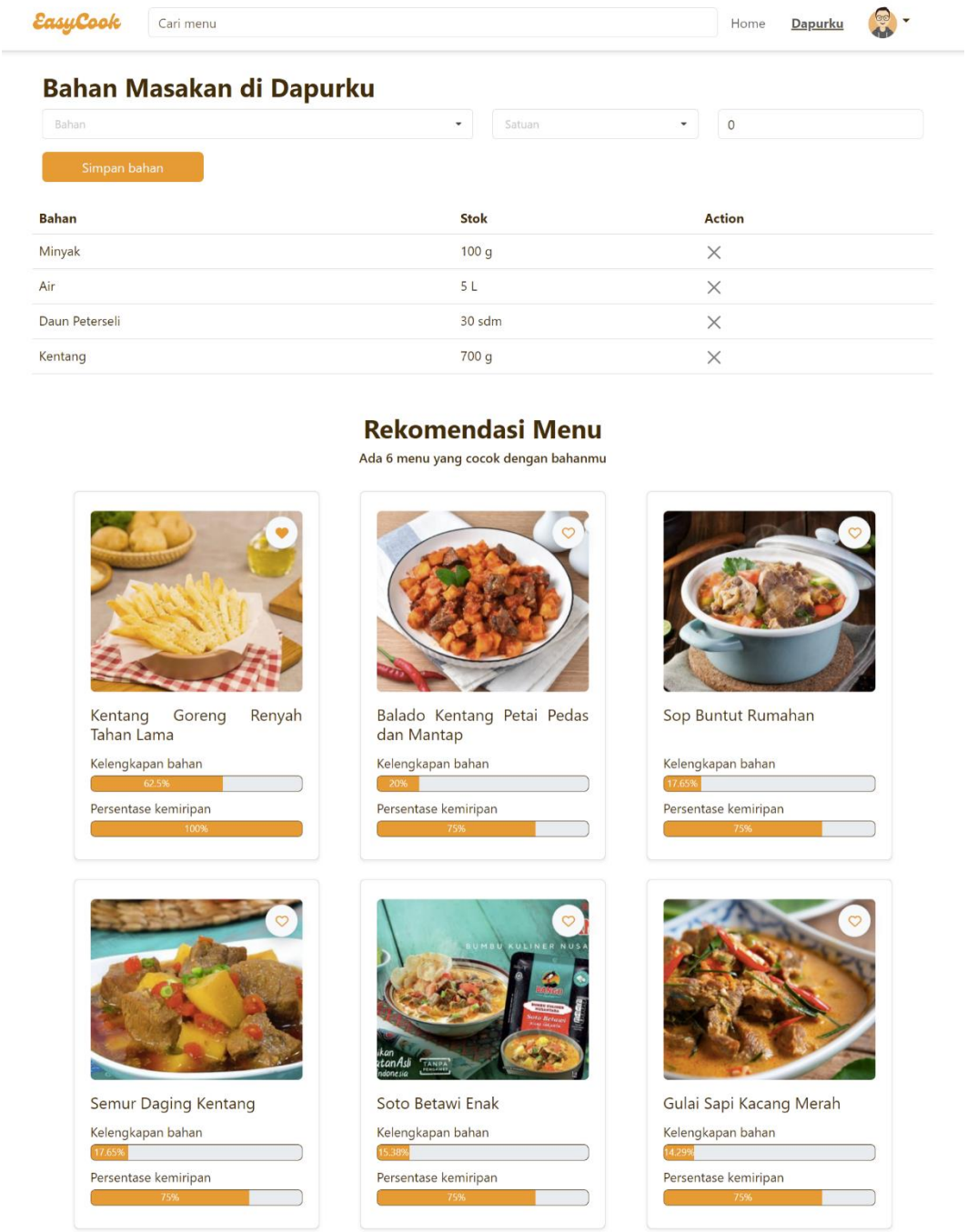


Figure 5.5 My Kitchen page view

5.7.5 Recipe Detail Page

The recipe detail page provides comprehensive information about a recipe. Users can view the recipe's title, cooking difficulty level, estimated time required, serving size, list of required ingredients, and step-by-step instructions for preparing

the dish. Additionally, users can save recipes to their favorites list. This feature allows users to easily access and refer to recipes they like without having to search for them again.

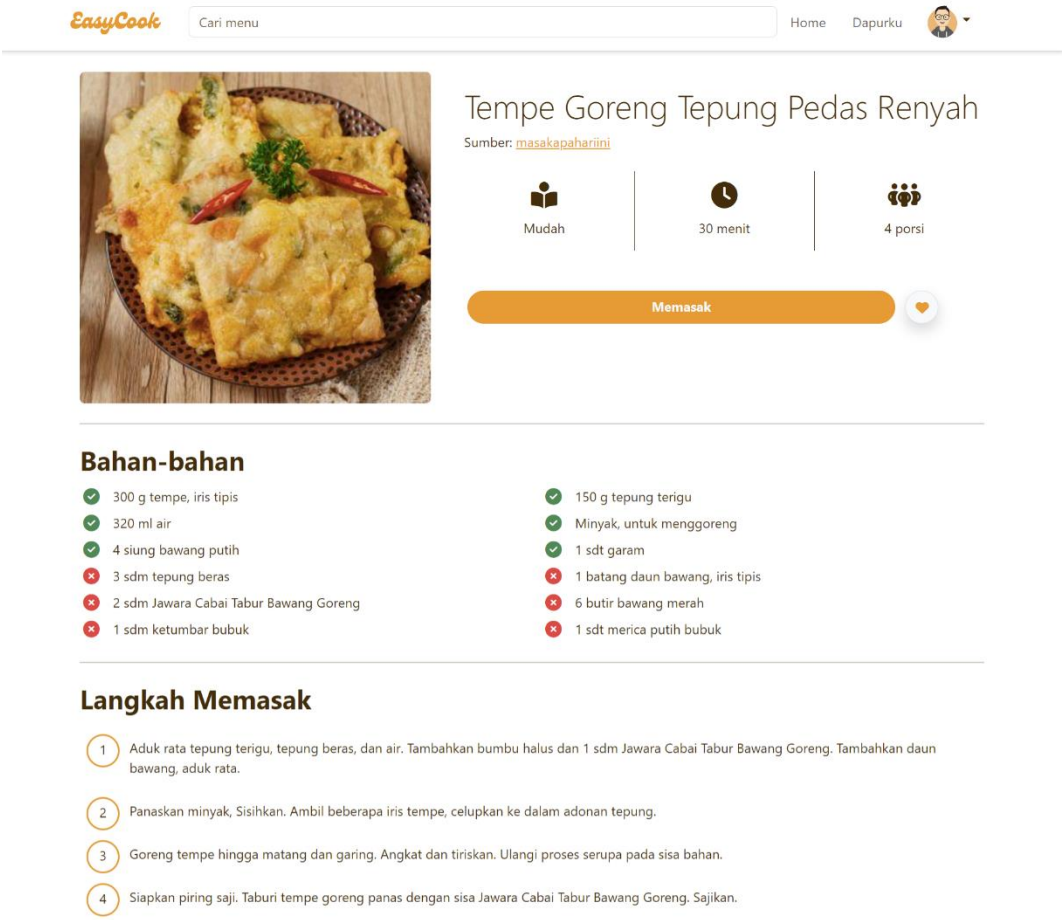


Figure 5.6 Recipe Detail page view

5.7.6 Search Page

The search page allows users to search for recipes by specific dish names. Search results are sorted based on the percentage of ingredient completeness, enabling users to quickly find recipes that match the ingredients available in their kitchen.

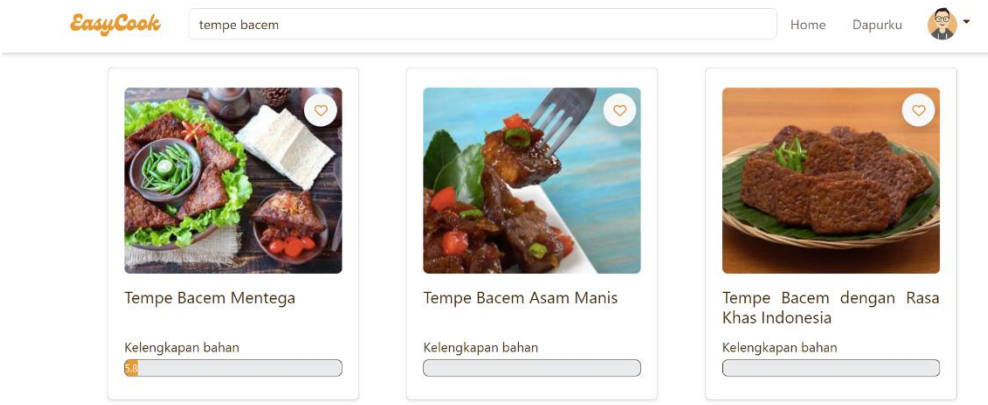


Figure 5.7 Search page view

5.7.7 User Profile Page

The user profile page displays account details such as name, username, and registered email address.

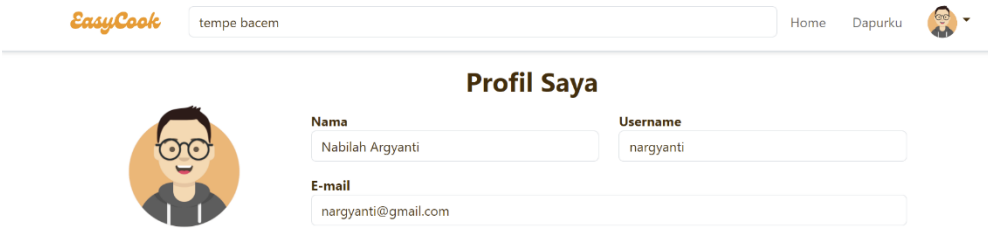


Figure 5.8 Profile page view

5.7.8 History Page

The history page displays a list of recipes that the user has visited. This list is sorted by the most recent visits, allowing users to easily view recipes they have recently seen or visited.

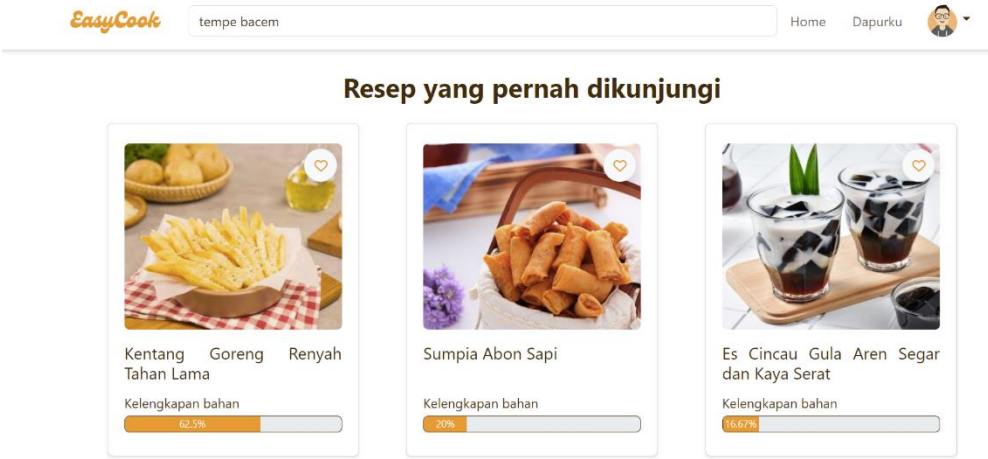


Figure 5.9 History page view

5.7.9 Favorites Page

The favorites page displays recipes that are liked by the user, allowing them to easily access their favorite recipes whenever needed. With this feature, users can save and revisit their favorite recipes without having to search again.

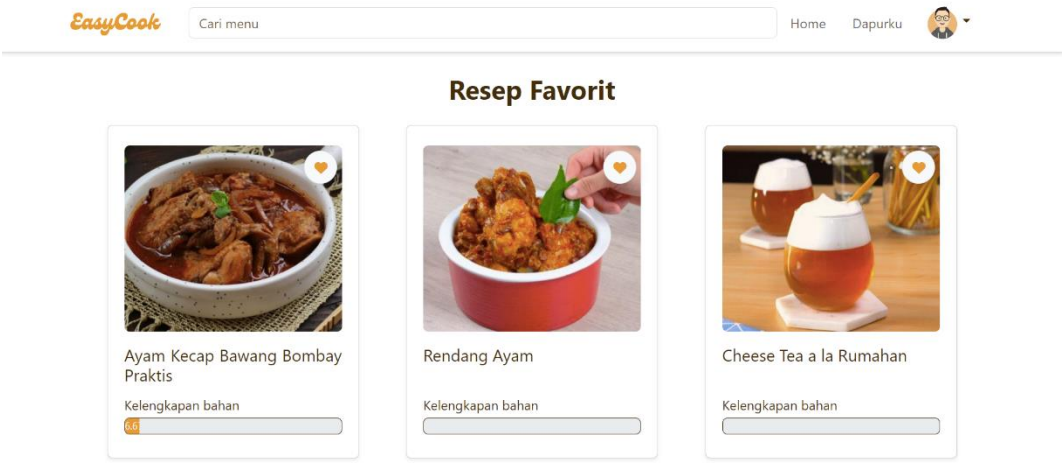


Figure 5.10 Favorites page view

5.8 System Testing

5.8.1 Functional Testing

Functional testing is used to test whether the features of the website are functioning according to the predefined functional requirements of the system. Testing is conducted using a black box testing approach, where testing is performed without knowledge of the internal implementation details of the program code or system.

The stages of black box testing in this system start with preparing a series of test cases that cover various scenarios and system features to be tested. These test cases serve as a guide for testing the system's functionality without considering the internal implementation details. After the test cases are prepared, the next step is to perform feature testing one by one according to the created test cases. In the testing process, the results of each feature will be carefully analyzed to determine whether the feature functions correctly or not. If the feature meets the expectations, it is given a "PASSED" status, but if there are errors or inconsistencies with the specifications, it is given a "FAILED" status.

The evaluation of test results is the next step in this black box testing process. From the test data that has been conducted, the system's success rate will be calculated using the formula:

$$System\ success\ rate = \frac{Number\ of\ successfull\ features}{Number\ of\ test\ cases} \times 100\%$$

5.8.2 Method Testing

Method testing is used to assess the system's success in applying and executing its methods accurately and effectively. This testing aims to verify whether the methods used by the system can produce the expected output and fulfill the established objectives. Additionally, method testing helps identify potential errors or issues in method implementation, allowing for improvement and enhancement.

5.8.2.1 Recipe Recommendation Testing Based on Ingredient Availability

To determine the extent to which the recipe recommendation system optimally provides recommendations based on user-owned ingredients, a method comparison is used as the main evaluation tool. This testing will be conducted in two stages. The first stage is testing to determine the most optimal method for recommending recipes based on ingredients, which will be performed three times: testing recipe recommendations with ingredient ownership percentage, testing recipe recommendations with Jaccard similarity, and testing recipe recommendations with both Jaccard similarity and ingredient ownership percentage. The following steps will be taken in the first stage of testing:

1. Users are asked to explore the EasyCook website and open the Kitchen page. Users are asked to add several ingredients they own.
2. After entering several owned ingredients, the system will calculate Jaccard similarity for each recipe. The higher the Jaccard similarity value between the user's ingredients and the recipe's ingredients, the more user ingredients match the recipe's ingredients.
3. The Kitchen page will display recipe recommendations based on the user's owned ingredients.
4. Steps 1-3 will be repeated three times with different algorithms, and the test results will be compared to determine the ranking of ingredient

recommendations and which algorithm is the most optimal for recommending recipes based on user-owned ingredients.

The second stage of testing involves multiple users to analyze the generated recipe recommendations and identify factors influencing the results. This stage focuses on two important aspects: the completeness of ingredients entered by users and the number of recommended recipes. The steps for the second stage of testing include:

1. Users are asked to explore the EasyCook website and open the Kitchen page. Users are asked to add several ingredients they own.
2. The Kitchen page will display recipe recommendations based on the user's owned ingredients.
3. The given recommendations will be recorded and analyzed. The analysis parameters include the number of ingredients entered by users, the percentage of ingredient completeness in the top recipes, the number of ingredients matching user preferences in the top recipes, the total number of ingredients in the top recipes, and the names of ingredients matching user preferences in the top recipes.

The results of both evaluations will provide an overview of how well the recipe recommendation system functions and how closely the recommendations align with the user's owned ingredients.

5.8.2.2 Recipe Recommendation Testing Based on User Preferences

To determine the extent to which the recipe recommendation system optimally provides recommendations based on user preferences, a comparison of results among four users is used as the main evaluation tool. This comparison will help assess the testing accuracy and analyze the results. The following steps will be taken in the method testing for recipe recommendations based on user preferences:

1. Users are asked to explore the EasyCook website and add several recipes they like to the favorites list.
2. After collecting data on the recipes liked by several users, the system will calculate cosine similarity for each recipe. The higher the cosine similarity value between two recipes, the more similar the recipes are in terms of preference.

3. From all participating users, four users will be selected as the data sample: two married users and two unmarried users. This sample data will be used as test data to assess the performance of the recommendation system.
4. Sample users are asked to like several recipes.
5. After liking some recipes, the Home page will display recipe recommendations in the "You Might Like" section based on the previously calculated cosine similarity.
6. Sample users are asked to select recipes that match their preferences and those that do not match. The results provided by each user will be compared and analyzed to determine the determining factors of recommendation results and the accuracy of recommendations.

The results of this evaluation will provide an overview of how well the recipe recommendation system functions and how closely the recommendations align with user preferences.

5.8.3 User Testing

Usability testing is used as a method to gather valuable feedback from users regarding their experience in using the system. In this testing, usability testing is conducted by asking users to try the website and fill out a feedback questionnaire as part of the system performance evaluation.

The steps in this usability testing start with asking several questions to fill out the user profile in a table. The information from this profile will help understand user characteristics and map different user groups.

Once the user profile is completed, users will be asked to try the website with the following steps:

1. Users are asked to click the Log In button on the navigation bar, then click Register Account.
2. After that, users are asked to fill out their name, username, email, and password. Next, users click the Register Account button.
3. Users are asked to log in by entering their username and password.
4. Users are asked to explore some recipes through the Home page or perform a recipe search.

5. Next, users are asked to visit the Kitchen page, add some ingredients they want to have, and click the Save Ingredients button.
6. Users are asked to select the most suitable recipe recommendations by pressing the heart-shaped favorite button on the recipe.
7. Finally, users are asked to return to the Home page. In the "You Might Like" section, users are asked to select several recipes that match their preferences by pressing the heart-shaped favorite button.

After trying the website, users will be asked to fill out a feedback questionnaire with the following questions:

Table 5.1 List of user feedback questions

No	Question
1	How easy is it to use this recipe website with your device?
2	How helpful are the most popular recipe recommendations and favorite recipe recommendations in helping you find desired recipes?
3	To what extent do you feel that the recipe search page accurately recommends recipes based on your preferences?
4	To what extent do you feel that the "Dapurku" page accurately recommends recipes based on the ingredients you own?
5	To what extent do you feel that the "Mungkin Kamu Suka" feature on the Home page accurately recommends recipes based on your preferences
6	To what extent does the ingredient completeness feature in recipes help you see ingredient availability for cooking?
7	To what extent does the feature of saving recipes to the favorites list help you save liked recipes?
8	How satisfied are you with the features provided by this website, both in terms of appearance and functionality?

These questions will cover aspects such as layout clarity, availability of necessary information, ease of use, recommendation quality, and overall user satisfaction with the website experience.

The feedback collected from the questionnaire will be calculated using a Likert scale, where the scale values will represent the level of satisfaction or user

perception for each question. From this calculation, insights will be obtained about the performance and shortcomings of the system as experienced by users.

Table 5.2 Likert scale value weights

Answer	**Weight**
Strongly agree / very easy / very good / very satisfied	5
Agree / easy / good / satisfied	4
Neutral	3
Disagree / difficult / bad / not satisfied	2
Strongly disagree / very difficult / very bad / very unsatisfied	1

Based on the results of this usability testing, information about user satisfaction with the system, the extent to which the website meets their needs and expectations, and which aspects need improvement or enhancement to enhance the overall user experience will be obtained.

CHAPTER VI. RESULTS AND DISCUSSION

6.1 Results

6.1.1 Functional Testing

Functional testing was conducted by developers by trying out various scenarios on the EasyCook website, and the following results were obtained.

Table 6.1 Black box testing result

No	Feature	Input	Scenario	Result	Status
1	Registration	Name, username, email, and password	Unique username or email	Successful registration and redirected to Log In page	Passed
			Existing username or email	Registration failed and displayed an alert that the username or email is already in use	Passed
2	Log in	Username and password	Correct username and password	Successful log in and redirected to Home page	Passed
			Incorrect username or password	Login failed and displayed an alert that the username or password is incorrect	Passed
3	Most liked recipe recommendations	Recipes liked by all users	Recommended recipes present	List of recommended recipes displayed on the Home page	Passed

			No recommended recipes	Section for recipes liked by all users not displayed	Passed
4	Recipe recommendations based on user visits	Recipes visited by all users	Recommended recipes present	List of recommended recipes displayed on the Home page	Passed
			No recommended recipes	Section for most visited recipes not displayed	Passed
5	Recipe recommendations based on user preferences	Recipes liked by the user	Recommended recipes present	List of recommended recipes displayed on the Home page	Passed
			No recommended recipes	Section for recommendations based on user preferences not displayed	Passed
6	Accessing recipe details	Selected recipe	Recipe exists in the database	Website displays recipe details	Passed
			Recipe does not exist in the database	Website displays an empty page	Passed
7	Get user-owned ingredients for a recipe	User-owned ingredients	User-owned ingredients match recipe	Website displays a checkmark on owned ingredients	Passed

		and recipe ingredients	User-owned ingredients do not match recipe	Website displays a cross mark on non-owned ingredients	Passed
8	Add user-owned ingredients	User-owned ingredients	Ingredients are available in the database	Ingredients can be added to the user's ingredient list	Passed
			Ingredients are not available in the database	Ingredients cannot be added to the user's ingredient list	Passed
9	Delete user-owned ingredients	User-owned ingredients	Ingredients can be deleted	Ingredients successfully removed from the user's ingredient list	Passed
			Ingredients cannot be deleted	Displays an error message that ingredients cannot be deleted	Passed
10	Display recipe recommendations based on user-owned ingredients	User-owned ingredients	Recommended recipes present	Website displays recipe details	Passed
			No recommended recipes	Website displays empty recommendations	Passed
11	Search for recipes	Keyword	Recipe found	Displays a list of recipes matching the keyword	Passed
			Recipe not found	Displays an empty page	Passed

12	Accessing visited recipes	Visited recipes	Visited recipes exist	Displays a list of visited recipes	Passed
			No visited recipes	Displays an empty page	Passed
13	Accessing favorite recipes	Recipes liked by the user	Favorite recipes exist	Displays a list of user-liked recipes	Passed
			No favorite recipes	Displays an empty page	Passed
14	Add recipe to favorites	Recipe liked by the user	Recipe to be liked is available	Recipe successfully added to favorites list	Passed
			Recipe to be liked is not available	No recipes added to favorites	Passed
15	Remove recipe from favorites	Recipe not liked by the user	Recipe to be disliked is available	Recipe successfully removed from favorites list	Passed
			Recipe to be disliked is not available	No recipes removed	Passed

$$System\ success\ rate = \frac{15}{15} \times 100\% = 100\%$$

Based on the above testing results, it was found that the tested features function well and meet the established expectations, with a success rate of 100%. The system successfully handled each test case and is ready to be used more extensively by users.

6.1.2 Recipe Recommendation Testing

6.1.2.1 Recipe Recommendations Based on Ingredient Availability

1) Testing the Most Optimal Recommendation Method

This testing involved 1 user who was asked to input several ingredients they owned to receive suitable recipe recommendations. A sample of 25 recipes and 123 ingredients were used to facilitate testing. The user-entered ingredients are described in Table 6.2.

Table 6.2 List of ingredients entered by the user

Ingredient	**Unit**	**Quantity**
Susu cair	Mililiter	1000
Gula pasir	Gram	500
Bawang merah	Siung	10
Bawang putih	Siung	10
Tomat	Buah	20
Bawang daun	Batang	100

After inputting the ingredients into the user's ingredient list, they were matched with the existing recipe database. Out of the 25 available recipes, it was found that 20 recipes contained at least one or more ingredients that the user had. The detailed results of this matching process are shown in Table 6.3.

Table 6.3 List of recipes along with user's ingredient quantity and recipe ingredient quantity

No	**Title**	**User Ingredient Quantity**	**Recipe Ingredient Quantity**
1	Avocado Mocha Ice Cream Sandwich	1	10
2	Ayam Goreng Thailand	1	10
3	Bakwan Jagung Sayuran	4	16
4	Bebek Rica Pedas	4	18
5	Cheese Tea	2	8
6	Cireng Keju Sambal Kecap	5	13
7	Kambing Masak Asam Pedas	4	19
8	Kepiting Saus Tiram Pedas	2	17

9	Nugget Tempe	1	9
10	Pempek Panggang Istimewa	2	14
11	Rabeg Daging Sapi Banten	2	16
12	Sate Kambing Bumbu Kacang Kurma	4	16
13	Sate Telur Gulung Sehat	2	10
14	Semur Daging Khas Pontianak	4	22
15	Semur Kambing Cincang Gulung	3	18
16	Semur Kambing Giling Kacang Merah	3	16
17	Sop Buntut Asam Pedas	3	22
18	Sop Kacang Merah Fusilli	3	14
19	Sup Ikan Asam Pedas	1	10
20	Terung Kecap Pedas	3	12

Testing the recipe recommendation method based on ingredients was conducted three times using different approaches. First, using the ingredient ownership percentage to sort recipes based on ingredients that nearly meet the user's requirements. Second, utilizing Jaccard similarity to assess the similarity of user ingredients with recipe ingredients. Lastly, combining Jaccard similarity and ingredient ownership percentage to achieve a balanced recommendation. Each test selected the top 12 recommendations for comparison.

A. Testing Recommendation with Ingredient Ownership Percentage

In this test, recipe recommendations were sorted based on the ingredient ownership percentage, which is the ratio of the number of ingredients owned by the user to the total number of ingredients required in the recipe. In this recommendation order, recipes with the highest ingredient ownership percentage received the top rank.

The test results presented in Table 6.4 indicate that the top-ranked recipe recommendations are those recipes that almost meet all the user's ingredient requirements. Conversely, recipes with lower ingredient ownership percentages are ranked lower in the recommendation order.

Table 6.4 List of Recipe Recommendations Based on Ingredient Ownership Percentage

No	Title	User Ingredients Quantity	Missing Ingredients Quantity	Ingredient Ownership Percentage
1	Cireng Keju Sambal Kecap	5	8	38.46%
2	Bakwan Jagung Sayuran	4	12	25%
3	Cheese Tea	2	6	25%
4	Sate Kambing Bumbu Kacang Kurma	4	12	25%
5	Terung Kecap Pedas	3	9	25%
6	Bebek Rica Pedas	4	14	22.22%
7	Sop Kacang Merah Fusilli	3	11	21.43%
8	Kambing Masak Asam Pedas	4	15	21.05%
9	Sate Telur Gulung Sehat	2	8	20%
10	Semur Kambing Giling Kacang Merah	3	13	18.75%
11	Semur Daging Khas Pontianak	4	18	18.18%
12	Semur Kambing Cincang Gulung	3	15	16.67%

B. Testing Recommendation with Jaccard Similarity

In this test, recipe recommendations were sorted based on the similarity of user-owned ingredients to the ingredients in the recipes, calculated using Jaccard similarity. In this recommendation order, recipes with the highest similarity to the user's ingredients received the top rank. The results of this test are presented in Table 6.5, where recipes with the most user-owned ingredients are ranked at the top.

Table 6.5 List of Recipe Recommendations with Jaccard Similarity

No	Title	User Ingredients Quantity	Missing Ingredients Quantity	Similarity Percentage
1	Cireng Keju Sambal Kecap	5	8	35.71%
2	Sate Kambing Bumbu Kacang Kurma	4	12	25%
3	Bakwan Jagung Sayuran	4	12	22.22%
4	Terung Kecap Pedas	3	9	21.43%
5	Bebek Rica Pedas	4	14	21.05%
6	Kambing Masak Asam Pedas	4	15	19.05%
7	Semur Kambing Giling Kacang Merah	3	13	17.65%
8	Sop Kacang Merah Fusilli	3	11	17.65%
9	Semur Daging Khas Pontianak	4	18	17.39%
10	Cheese Tea	2	6	16.67%
11	Semur Kambing Cincang Gulung	3	15	14.29%
12	Sate Telur Gulung Sehat	2	8	14.29%

C. Testing Recommendation with Jaccard Similarity and Ingredient Ownership Percentage

In this test, recipe recommendations were sorted based on the similarity of user-owned ingredients to the ingredients in the recipes, calculated using Jaccard similarity, as well as the ingredient ownership percentage with an 80% weight for ingredient similarity and a 20% weight for ingredient ownership percentage. In this recommendation order, recipes with the highest similarity and ingredient ownership percentage received the top rank. This means that the recommended recipes not only have a high similarity to the user's ingredients but also fulfill the ingredient requirements quickly. The results of this test are presented in Table 6.6, providing

a comprehensive overview of optimal recipe recommendations considering both important aspects.

Table 6.6 List of recipe recommendations with Jaccard similarity and ingredient ownership percentage

No	Title	User Ingredients Quantity	Missing Ingredients Quantity	Ingredient Ownership Percentage	Similarity Percentage
1	Cireng Keju Sambal Kecap	5	8	38.46%	35.71%
2	Sate Kambing Bumbu Kacang Kurma	4	12	25%	25%
3	Bakwan Jagung Sayuran	4	12	25%	22.22%
4	Terung Kecap Pedas	3	9	25%	21.43%
5	Bebek Rica Pedas	4	14	22.22%	21.05%
6	Kambing Masak Asam Pedas	4	15	21.05%	19.05%
7	Sop Kacang Merah Fusilli	3	11	21.43%	17.65%
8	Cheese Tea	2	6	25%	16.67%
9	Semur Kambing Giling Kacang Merah	3	13	18.75%	17.65%
10	Semur Daging Khas Pontianak	4	18	18.18%	17.39%

11	Sate Telur Gulung Sehat	2	8	20%	14.29%
12	Semur Kambing Cincang Gulung	3	15	16.67%	14.29%

2) Testing the Impact of Recommendation Differences on User-Owned Ingredients

This system was tested on 4 out of 47 users selected based on users who input the highest number of ingredients on the EasyCook website, resulting in 1591 recipes and 22,606 ingredients grouped into 1,591 ingredients. During the evaluation process, the system's performance in recommending recipes based on user-entered ingredients was observed. In this evaluation, two important aspects received special attention: ingredient completeness and the number of recipe recommendations.

Table 6.7 Evaluation results of recipe recommendations based on user's ingredient availability

	User 1	**User 2**	**User 3**	**User 4**
Number of ingredients entered	16	8	6	6
Number of recommendations displayed	969	1149	533	1171
Highest ingredient completeness percentage	46.15%	38.89%	28.57%	42.86%
Ingredients owned by user in top recipe	Udang, daun bawang, wortel, minyak	Kentang, garam, ayam, mentega,	Ayam, cabai	Ayam, telur, nasi

	wijen, telur, jamur shiitake	bawang putih		
Number of ingredients in top recipe	13	16	7	7
Number of ingredients owned in top recipe	6	5	2	3

The results of the evaluation, as shown in Table 6.7, reveal interesting findings in the analysis of recipe recommendations for specific users.

First, User 1 had the highest ingredient completeness at 46.15%, followed by User 4 with 42.86%. Ingredient completeness is influenced by how well the user's ingredients match the recipe's ingredients. For example, User 1 has 6 matching ingredients out of a total of 13 ingredients in the recipe, resulting in an ingredient completeness percentage of 46.15%. Meanwhile, User 4 has 3 matching ingredients out of a total of 7 recipe ingredients, resulting in an ingredient completeness percentage of 42.86%. Despite the slight difference, both have matching ingredients compared to the total number of ingredients in the recipe.

Second, it can be observed that the number of entered ingredients does not always affect the number of recipe recommendations. For instance, both User 3 and User 4 entered 6 ingredients, but the number of recipe recommendations for them differs significantly, with 533 and 1171 recipes, respectively. Other factors such as similarity weight and ingredient suitability also play a role in determining the recommendation outcomes.

Third, it is noteworthy that User 1 has almost twice the number of ingredients compared to User 2, yet User 2 receives more recipe recommendations. User 2 entered 8 ingredients and received 1149 recipe recommendations, while User 1, with 16 ingredients, only received 969 recipe recommendations. This suggests that the number of ingredients entered by the user is not always the main factor in determining the number of recipe recommendations.

6.1.2.2 Recipe Recommendations Based on User Preferences

This test involved 43 users with different backgrounds. Users were asked to add several recipes to their favorite list, and the data from their participation was used as initial or training data. After these 43 users added recipes to their favorite list, four users were selected, consisting of two active homemakers cooking for their families and two single individuals living on their own. Each user was asked to like some recipes as part of the testing process. Subsequently, the system provided recommendations based on user preferences, and users were asked to select recommendations that matched their preferences. The results of this evaluation were analyzed and compared to assess how well the system provides accurate and personalized recipe recommendations.

Table 6.8 Table of recipe recommendation testing results based on user preferences

	User 1	**User 2**	**User 3**	**User 4**
User status	Married	Married	Single	Single
Number of user's favorite recipes	10	6	3	3
Number of recommendations displayed	8	11	11	8
Number of relevant recommendations	6	7	9	5
Recipe recommendation accuracy	75%	63.6%	81.8%	62.5%

Based on the testing data provided in Table 6.9, four users participated, each with different marital statuses: two married users (User 1 and User 2) and two single users (User 3 and User 4).

Firstly, in terms of the number of recommendations provided by the system, there was significant variation. User 2 and User 3 received the highest number of recommendations, each getting 11 recipes. On the other hand, User 1 and User 4 received slightly fewer recommendations, with 8 recipes each. This indicates that the system provides different numbers of recommendations to each user, likely based on their preferences and history of recipe evaluations.

Secondly, the number of relevant recommendations based on user preferences also varied. User 3 had the highest number of relevant recommendations, which was 9 recipes, while User 4 had the lowest with only 5 recipes. This suggests that the system can provide more suitable recommendations for User 3 compared to User 4.

Thirdly, there was a noticeable difference in the number of favorite recipes liked by each user. User 1 had the most favorite recipes, totaling 10 recipes, while User 3 and User 4 only liked 3 recipes. User 2 fell in between with 6 favorite recipes. This indicates a variation in culinary preferences and interests among users.

6.1.3 Usability Testing

The testing was conducted involving 47 respondents representing various levels of cooking expertise. The results of this testing are presented in a questionnaire table containing questions related to user experience when using the recipe recommendation website. Respondents were asked to rate their satisfaction on a scale of 1 to 5, where 1 indicates the lowest level of satisfaction and 5 indicates the highest. The Likert scale was used to calculate the answers from the questionnaire, as shown in Table 6.9.

Table 6.9 User Questionnaire responses with likert scale calculation

No	**Question**	**Respondent**					**Score**	**Percentage**
		1	**2**	**3**	**4**	**5**		
1	How easy is it to use this recipe website with your device?	0	0	3	19	21	190	80.85%
2	How helpful are the most popular recipe recommendations and favorite recipe recommendations in helping you find desired recipes?	0	2	3	20	22	203	86.38%
3	To what extent do you feel that the recipe search page	0	1	5	20	15	172	73.19%

	accurately recommends recipes based on your preferences?							
4	To what extent do you feel that the "Dapurku" page accurately recommends recipes based on the ingredients you own?	0	2	4	21	20	200	85.11%
5	To what extent do you feel that the "Mungkin Kamu Suka" feature on the Home page accurately recommends recipes based on your preferences	0	2	11	18	16	189	80.43%
6	To what extent does the ingredient completeness feature in recipes help you see ingredient availability for cooking?	0	0	1	20	26	213	90.64%
7	To what extent does the feature of saving recipes to the favorites list help you save liked recipes?	0	0	2	20	25	211	89.79%
8	How satisfied are you with the features provided by this website, both in terms of appearance and functionality?	0	0	4	20	23	207	88.09%
Total							1585	84.31%

From the questionnaire results, the majority of respondents provided positive ratings for this recipe website. About 80.85% of respondents stated that using the website was easy with their respective devices. The most popular features, such as top recipe recommendations and favorite recipes, were deemed helpful in finding desired recipes by 86.38% of respondents.

Furthermore, 85.11% of respondents felt that the recommendations on the "My Kitchen" page were accurate enough in suggesting recipes based on their ingredients. Similarly, the "You Might Like" section on the Home page was considered accurate enough in recommending recipes based on preferences by 80.43% of respondents.

The ingredient completeness feature in recipes was found helpful by 90.64% of respondents for checking ingredient availability for cooking. Additionally, the feature to save recipes to the favorite list was found helpful by 89.79% of respondents in saving liked recipes.

Overall, 88.09% of respondents expressed satisfaction with the features provided by the website, both in terms of appearance and functionality.

6.2 Discussion

After conducting several testing phases, the overall results indicate that the recipe recommendation system performs very well. The testing was carried out using various methods, including functional testing, recipe recommendation testing, and user testing.

The results of functional testing showed that the tested features functioned well and met the established expectations, with a success rate of 100%. The system successfully passed each test case, indicating its readiness for broader use by users.

In the test for recipe recommendations based on ingredient availability, there were some notes. In the recommendation with ingredient ownership percentage, an imbalance in outcomes was observed, where ingredient utilization tends not to be maximized. There were cases where a recommendation received fewer recipe ingredients with minimal ingredient usage compared to recipes ranked lower. For instance, in the Cheese Tea recipe, there were only 2 ingredients owned by the user, while the user had 4 ingredients for the Sate Kambing Bumbu Kacang Kurma recipe.

In the recommendation with Jaccard similarity, a limitation was observed where recipes were still sorted based on the similarity of user ingredients to recipe ingredients. This approach did not consider the shortage of recipe ingredients. As a result, recipes with higher ingredient shortages could be recommended earlier than those with fewer shortages. For example, the Semur Kambing Giling Kacang Merah

recipe had 13 missing ingredients, while the Sop Kacang Merah Fusilli had only 11 missing ingredients.

The combination of ingredient ownership percentage and Jaccard similarity proved to be the optimal solution. This approach maximized ingredient usage while considering the relevant ingredient percentage. Higher similarity and ingredient ownership percentage were given priority in ranking recommendations higher. However, the sorting of recommendations still does not prioritize recipes with fewer missing ingredients, requiring users to check each recipe individually to identify those with fewer shortages.

The evaluation results of recipe recommendations based on user-entered ingredients among users revealed that while the number of entered ingredients played an important role, other factors such as ingredient suitability with the recipe also affected the number of recipe recommendations. Furthermore, the type of ingredients provided, whether basic ingredients like water, salt, and onions, or specific ingredients like carrots, red chili peppers, and shiitake mushrooms, also influenced the number of recipe recommendations. The more common the type of ingredients entered, the more recommendations appeared.

Additionally, in the evaluation of recipe recommendations based on user preferences, it was found that each user had different preferences for liked recipes. Some users might like many recipes, while others only prefer a few. An interesting fact was that the number of liked favorite recipes by users did not have a direct impact on the number of recommendations shown. Similar results were observed for user status, which did not significantly affect the recommendation outcomes. For example, User 1, who is married and liked 10 recipes, received only 8 recommendations, whereas User 3, who is single and liked only 3 recipes, received 11 recommendations.

Moreover, variations in the number of relevant recommendations indicated diverse user preferences. Some users had preferences almost identical to the provided recommendations, while others had slightly different preferences, affecting the accuracy of the recommendations. Overall, the evaluation results indicated that this recipe recommendation system is fairly accurate, with an average of 70.76% of recipes matching user preferences.

Based on user feedback, recipe recommendations based on ingredients were rated higher than those based on user preferences, with a satisfaction rate of 85.11%. This suggests that ingredient-based recommendations can provide more relevant results that match users' needs.

CHAPTER VII. CONCLUSION AND RECOMMENDATIONS

7.1 Conclusion

Based on the research and testing conducted, the following conclusions are:

1. The overall results of the testing indicate that the recipe recommendation system functions very well. Testing through various methods, including functional testing, recipe recommendation testing, and user testing, provides positive indications of the system's readiness for broader user utilization.
2. Recipe recommendations based on ingredient availability calculated using Jaccard similarity and user ingredient ownership percentage were identified as the optimal solution. While the number of ingredients entered by the user plays a significant role, the suitability of ingredients with the recipe also impacts the resulting recommendations. Moreover, the types of ingredients available, both basic and specialized, also influence the number of recommendations provided to the user.
3. Recipe recommendations based on user preferences reveal variations in user preferences when selecting favorite recipes. Although the number of favorite recipes and user status do not significantly affect the number of recommendations displayed, the evaluation results demonstrate that the system is capable of presenting recommendations that align with each user's unique preferences. The variation in the number of relevant recipes also reflects diverse preferences, indicating the system's ability to meet user preferences quite effectively.

7.2 Recommendations

From the development of recipe recommendations based on ingredient availability and user preferences, there are several areas that can be improved. Recommendations for further research include:

1. The system could be enhanced by providing users with the option to specify ingredients to avoid, such as allergens or dietary restrictions. This way, recipe recommendations containing ingredients to be avoided can be

filtered out, allowing users to find recipes that align with their preferences and dietary needs more safely and comfortably.

2. Further development of recipe recommendations based on ingredient availability could consider the number of ingredients not owned by the user. This would allow users to apply recipes by maximizing the use of ingredients they possess the most while identifying the fewest missing ingredients. With this information, users can efficiently choose recipes that match their available ingredients, simultaneously reducing unnecessary ingredient purchases.
3. Recipe recommendations based on user preferences could also be enhanced by combining calculations from ingredient-based recommendations with recommendations based on previously liked recipes. In this fusion, priority could be given to recipe recommendations relevant to the user's ingredient availability, enabling users to receive recommendations that align with their preferences while also considering available ingredients.

BIBLIOGRAPHY

Adaji, I., Sharmaine, C., Debrowney, S., Oyibo, K., & Vassileva, J. (2018). Personality Based Recipe Recommendation Using Recipe Network Graphs. In G. Meiselwitz (Ed.), *Social Computing and Social Media. Technologies and Analytics* (Vol. 10914, pp. 161–170). Springer International Publishing. https://doi.org/10.1007/978-3-319-91485-5_12

Anzman-Frasca, S., Ventura, A. K., Ehrenberg, S., & Myers, K. P. (2018). Promoting healthy food preferences from the start: A narrative review of food preference learning from the prenatal period through early childhood: Promoting healthy food preferences. *Obesity Reviews*, *19*(4), 576–604. https://doi.org/10.1111/obr.12658

Bhujade, V., & Janwe, N. J. (2011). Knowledge Discovery in Text Mining Technique Using Association Rules Extraction. *2011 International Conference on Computational Intelligence and Communication Networks*, 498–502. https://doi.org/10.1109/CICN.2011.104

Burke, R., Felfernig, A., & Göker, M. H. (2011). Recommender Systems: An Overview. *AI Magazine*, *32*(3), 13–18. https://doi.org/10.1609/aimag.v32i3.2361

Chakrabartty, S. N. (2022). Assessment of Item and Test parameters: Cosine Similarity Approach. *International Journal of Psychology and Educational Studies*, *8*(3), 28–38. https://doi.org/10.52380/ijpes.2021.8.3.190

Chaudhari, S., Aparna, R., Tekkur, V. G., Pavan, G. L., & Karki, S. R. (2020). Ingredient/Recipe Algorithm using Web Mining and Web Scraping for Smart Chef. *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 1–4. https://doi.org/10.1109/CONECCT50063.2020.9198450

Dangeti, P. (2017). *Statistics for machine learning: Techniques for exploring supervised, unsupervised, and reinforcement learning models with Python and R*. Packt Publishing.

Doyle, M. P., & Erickson, M. C. (2008). Summer meeting 2007 the problems with fresh produce: An overview. *Journal of Applied Microbiology*, *105*(2), 317–330. https://doi.org/10.1111/j.1365-2672.2008.03746.x

Ertl, O. (2019). *ProbMinHash—A Class of Locality-Sensitive Hash Algorithms for the (Probability) Jaccard Similarity*. https://doi.org/10.48550/ARXIV.1911.00675

Grygorian, A., & Iacob, I. E. (2018). A Concise Proof of the Triangle Inequality for the Jaccard Distance. *The College Mathematics Journal*, *49*(5), 363–365. https://doi.org/10.1080/07468342.2018.1526020

Hasty. (n.d.). Accuracy. *Hasty.Ai*. https://hasty.ai/docs/mp-wiki/metrics/accuracy

Hillen, J. (2019). Web scraping for food price research. *British Food Journal*, *121*(12), 3350–3361. https://doi.org/10.1108/BFJ-02-2019-0081

Joshi, A., Kale, S., Chandel, S., & Pal, D. (2015). Likert Scale: Explored and Explained. *British Journal of Applied Science & Technology*, *7*(4), 396–403. https://doi.org/10.9734/BJAST/2015/14975

Kadowaki, T., Yamakata, Y., Mori, S., & Tanaka, K. (2014). Recipe search for blog-type recipe articles based on a user's situation. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, 497–506. https://doi.org/10.1145/2638728.2641327

Lahitani, A. R. (2022). Automated Essay Scoring menggunakan Cosine Similarity pada Penilaian Esai Multi Soal. *Jurnal Kajian Ilmiah*, *22*(2), 107–118. https://doi.org/10.31599/jki.v22i2.1121

Lawson, R. (2015). *Web scraping with Python: Scrape data from any website with the power of Python*. Packt Publishing.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Montanari, M. (2006). *Food is culture*. Columbia University Press.

Nilesh, N., Kumari, M., Hazarika, P., & Raman, V. (2019). Recommendation of Indian Cuisine Recipes Based on Ingredients. *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*, 96–99. https://doi.org/10.1109/ICDEW.2019.00-28

Shankar, N., Nagaraj, S. H., & Swamy, S. (2020). Review of Recommendation System using Filtering based Concepts. *Research Journal of Engineering and Technology*, *7*(5), 1559–1564.

Tobey, L., Mouzong, C., Angulo, J., Bowman, S., & Manore, M. (2019). How Low-Income Mothers Select and Adapt Recipes and Implications for Promoting Healthy Recipes Online. *Nutrients*, *11*(2), 339. https://doi.org/10.3390/nu11020339

Verma, A., Department of Computer Science, Amity University, Gurgaon, India, Khatana, A., Department of Computer Science, Amity University, Gurgaon, India, Chaudhary, S., & Department of Computer Science, Amity University, Gurgaon, India. (2017). A Comparative Study of Black Box Testing and White Box Testing. *International Journal of Computer Sciences and Engineering*, *5*(12), 301–304. https://doi.org/10.26438/ijcse/v5i12.301304

Yan, Z., Liu, J., Li, G., Han, Z., & Qiu, S. (2017). *PrivMin: Differentially Private MinHash for Jaccard Similarity Computation*. https://doi.org/10.48550/ARXIV.1705.07258

Yigitbas, E., Hottung, A., Rojas, S. M., Anjorin, A., Sauer, S., & Engels, G. (2019). Context- and Data-driven Satisfaction Analysis of User Interface Adaptations Based on Instant User Feedback. *Proceedings of the ACM on Human-Computer Interaction*, *3*(EICS), 1–20. https://doi.org/10.1145/3331161

Zhao, B. (2017). Web Scraping. In L. A. Schintler & C. L. McNeely (Eds.), *Encyclopedia of Big Data* (pp. 1–3). Springer International Publishing. https://doi.org/10.1007/978-3-319-32001-4_483-1

Zhou, B., Yamanaka-Okumura, H., Seki, S., Tatano, H., Adachi, C., & Takeda, E. (2014). What influences appetite more: Eating approaches or cooking methods? *The Journal of Medical Investigation*, *61*(1.2), 118–125. https://doi.org/10.2152/jmi.61.118

APPENDIX

Appendix 1. Recipe Scraper Source Code

```
# Class for scrape the recipe detail
class RecipeScraper:
  def __init__(self):
    self.page = None
    self.soup = None

  def get_soup(self, url):
    self.page = requests.get(url)
    self.soup = BeautifulSoup(self.page.content, 'html.parser')

  def extract_text(self, scripts):
    return [script.text.strip() for script in scripts]

  def extract_amount(self, scripts):
    return [script.get('data-base-quantity') for script in
scripts]

  def extract_instructions(self, scripts):
    instructions = []
    for instruction in scripts:
        content = instruction.find('div', {'class': 'content'})
        if content.find('p'):
            p_element = content.find('p')
            text = p_element.get_text().strip()
        else:
            text = content.get_text().strip()
        instructions.append(text)

    return instructions

  def combine_ingredients(self, amounts, ingredients):
    combined = []
    for i in range(len(amounts)):
        data = str(amounts[i]) + " " + \
```

```
            ingredients[i].replace('\t', '').replace('\n', '')
        data = ' '.join(data.split())
        combined.append(data)
    return combined

  def scrape_recipe(self, url):
    self.get_soup(url)

    title = self.soup.find('title')
    image_url = self.soup.find("img", {"class": "image"})["src"]
    time = self.soup.find(
        'a', href=lambda href: href and 'time' in
href).text.strip()
    servings = self.soup.find("div", {"id": "portions-
value"}).text.strip()
    difficulty = self.soup.find(
        "a", {"class": "icon_difficulty"}).text.strip()
    calories = self.soup.find("a", {"class":
"icon_fire"}).text.strip(
    ) if self.soup.find("a", {"class": "icon_fire"}) else 0
    ingredients = self.extract_text(
        self.soup.find_all('div', {'class': 'item'}))
    amount = self.extract_amount(
        self.soup.find_all('div', {'class': 'part'}))
    ingredients = self.combine_ingredients(amount, ingredients)
    instructions = self.extract_instructions(self.soup.find(
        'div', {'class': '_recipe-steps'}).find_all('div',
{'class': 'step'}))
    tags = self.extract_text(self.soup.find_all("a", {"class":
"tag"}))

    return {
        'title': re.split(r" - | \| | untuk | yang", title.text,
1)[0],
        'image_url': image_url,
        'time': time,
        'servings': servings,
```

```
        'calories': calories,
        'difficulty': difficulty,
        'ingredients': ingredients,
        'instructions': instructions,
        'source_url': url,
        'tags': tags,
    }
```

Appendix 2. Ingredient Grouper Source Code

```
def get_word_count(ingredient):
    return len(ingredient.split())

# Grouping ingredients
class IngredientGrouper:
    def __init__(self):
        self.difficult_crafting_dish = ['saos', 'sambal',
'saus', 'tepung']
        self.size_words = ['besar', 'kecil', 'sedang']
        self.different_meaning_part = ['daun', 'has']
        self.easy_crafting_dish = ['asap', 'goreng', 'kukus',
'larutan', 'perasan', 'serut']
        self.brands = ['bango', 'buavita', 'jawara', 'royco',
"wall's", "sariwangi", "sapi", "kambing"]
        self.ingredient_forms = ['basah', 'dingin', 'es',
'giling', 'halus', 'kering', 'matang', 'mendidih', 'mineral',
'panas', 'parut', 'rebus', 'utuh', 'bermineral', 'dikupas',
'hangat']
        self.lost_mean_words = ['bawang', 'ikan']
        self.multi_meaning_dish = ['air']
        self.irrelevant_words = ['daging', 'secukupnya',
'santap', 'sudah', 'pakai', 'segar', 'tanpa', 'ukuran', 'untuk',
'yang', 'buah', 'dengan', 'siap', 'yang', 'potong', 'berukuran']
        self.conjunction = ['dan', 'atau', '/']

    def group_ingredients(self, ingredients):
        ingredient_groups = {}

        for ingredient in ingredients:
            word_count = len(ingredient.split())
            processed_word = ingredient

            if word_count == 1:
                ingredient_groups[ingredient] = {}
                ingredient_groups[ingredient][ingredient] = []
```

```
ingredient_groups[ingredient][ingredient].append(ingredient)
            else:
                processed_word = ' '.join(word for word in
ingredient.split() if word not in self.size_words and word not
in self.easy_crafting_dish and word not in self.irrelevant_words
and word not in self.ingredient_forms)
                word_count = len(processed_word.split())

                matching_key = None
                max_matching_words = 0
                matching_key_word_count = 0
                matching_words = 0

                if any(keyword in processed_word for keyword in
self.brands):
                    for brand in self.brands:
                        if brand in processed_word:
                            matching_key = brand
                            max_matching_words = matching_words
                            matching_key_word_count =
len(matching_key.split())
                            matching_words = 1
                            break
                elif any(keyword in processed_word for keyword
in self.difficult_crafting_dish):
                    for diff_craft in
self.difficult_crafting_dish:
                        if diff_craft in processed_word:
                            matching_key = diff_craft
                            max_matching_words = matching_words
                            matching_key_word_count =
len(matching_key.split())
                            matching_words = 1
                else:
                    for key in ingredient_groups.keys():
```

```
                        if key in self.multi_meaning_dish and
len(processed_word.split()) > 1:
                            continue

                        matching_words = sum(1 for word in
processed_word.split() if word == key)

                        if len(key.split()) == 1 and
len(processed_word.split()) > 1:
                            matching_words = sum(1 for word in
processed_word.split() if key == word)
                        elif len(key.split()) <=
len(processed_word.split()):
                            matching_words = sum(1 for word in
processed_word.split() if word in key)
                        else:
                            matching_words = sum(1 for word in
processed_word.split() if key in word)

                        if matching_words > max_matching_words:
                            matching_key = key
                            max_matching_words = matching_words
                            matching_key_word_count =
len(matching_key.split())
                        elif matching_key is not None:
                            if key in ingredient and
matching_key in ingredient and matching_words ==
max_matching_words:
                                if ingredient.index(key) <
ingredient.index(matching_key):
                                    matching_key = key
                                    max_matching_words =
matching_words
                                    matching_key_word_count =
len(matching_key.split())

                if matching_key is not None:
```

```
                    if matching_key not in ingredient_groups:
                        ingredient_groups[matching_key] = {}

ingredient_groups[matching_key][ingredient] = []

ingredient_groups[matching_key][ingredient].append(ingredient)
                    elif (matching_key_word_count > 1 and
max_matching_words > 1 and word_count > 1) or
(matching_key_word_count == 1 and max_matching_words == 1) or
(len(matching_key.split()) < word_count):
                        nested_matching_key = None
                        max_nested_matching_words = 0
                        for key in
ingredient_groups[matching_key].keys():
                            if key != matching_key:
                                if key in processed_word:
                                    nested_matching_key = key
                                elif (len(key.split()) ==
len(processed_word.split())) or (len(key.split()) ==
len(ingredient.split())):
                                    if all(word in key.split()
for word in processed_word.split()):
                                        nested_matching_key =
key
                            elif len(key.split()) ==
len(processed_word.split()):
                                if key in processed_word:
                                    nested_matching_key = key
                        if nested_matching_key is not None:

ingredient_groups[matching_key][nested_matching_key].append(ingr
edient)
                        else:

ingredient_groups[matching_key][ingredient] = []

ingredient_groups[matching_key][ingredient].append(ingredient)
```

```
                    else:
                        ingredient_groups[ingredient] = {}

ingredient_groups[ingredient][ingredient] = []

ingredient_groups[ingredient][ingredient].append(ingredient)
                else:
                    ingredient_groups[ingredient] = {}
                    ingredient_groups[ingredient][ingredient] =
[]

ingredient_groups[ingredient][ingredient].append(ingredient)
        return ingredient_groups
```

Appendix 3. Get Owned Ingredients Source Code

```
def get_owned_raw_ingredients(user):
    pantry_ids =
Pantry.objects.filter(user=user).values_list('id', flat=True)
    unit_conversions = UnitConversion.objects.select_related(
        'higher_unit', 'lower_unit').all()

    owned_raw_ingredient_ids = set()

    for pantry_id in pantry_ids:
        pantry = Pantry.objects.select_related(
            'unit', 'raw_ingredient').get(id=pantry_id)
        raw_ingredient_id = pantry.raw_ingredient_id

        ingredients = Ingredient.objects.filter(
            Q(raw_ingredient_id=raw_ingredient_id) &
            Q(raw_ingredient__pantry__user=user)
        ).select_related('unit')

        for ingredient in ingredients:
            if pantry.unit is None:
                owned_raw_ingredient_ids.add(raw_ingredient_id)
            elif ingredient.unit is None:
                continue
            else:
                if ingredient.unit == pantry.unit:
                    if pantry.amount >= ingredient.amount:

owned_raw_ingredient_ids.add(raw_ingredient_id)
                else:
                    if ingredient.unit.type == "measured" and
pantry.unit.type == "measured":
                        if pantry.unit.hierarchy <=
ingredient.unit.hierarchy:
                            higher = pantry
                            lower = ingredient
                        else:
```

```
                            higher = ingredient
                            lower = pantry

                        conversion = UnitConversion.objects.get(
                            higher_unit_id=higher.unit.pk,
lower_unit_id=lower.unit.pk)
                        if conversion:
                            converted_amount = conversion.value
* higher.amount
                            if higher == pantry and
converted_amount >= lower.amount:

owned_raw_ingredient_ids.add(raw_ingredient_id)
                            elif higher == ingredient and
lower.amount >= converted_amount:

owned_raw_ingredient_ids.add(raw_ingredient_id)
                        else:
                            common_unit_id = find_common_unit(
                                higher.unit.id, lower.unit.id,
unit_conversions
                            )
                            if common_unit_id:
                                converted_higher_amount =
convert_amount(
                                    higher.amount,
higher.unit.id, common_unit_id, unit_conversions
                                )
                                converted_lower_amount =
convert_amount(
                                    lower.amount, lower.unit.id,
common_unit_id, unit_conversions
                                )
                                if converted_higher_amount >=
converted_lower_amount:

owned_raw_ingredient_ids.add(
```

```
                                        raw_ingredient_id)
                    else:
                        if ingredient.unit.category ==
pantry.unit.category:
                            if pantry.unit.hierarchy <
ingredient.unit.hierarchy:

owned_raw_ingredient_ids.add(raw_ingredient_id)
                            elif pantry.unit.hierarchy ==
ingredient.unit.hierarchy:
                                if pantry.amount >=
ingredient.amount:

owned_raw_ingredient_ids.add(
                                        raw_ingredient_id)
                        elif pantry.unit.category == 'item':

owned_raw_ingredient_ids.add(raw_ingredient_id)

    return owned_raw_ingredient_ids


def find_common_unit(higher_unit_id, lower_unit_id,
unit_conversions):
    if higher_unit_id == lower_unit_id:
        return higher_unit_id

    conversion = unit_conversions.filter(
        higher_unit_id=higher_unit_id,
lower_unit_id=lower_unit_id
    ).first()
    if conversion:
        return lower_unit_id

    conversion = unit_conversions.filter(
        higher_unit_id=lower_unit_id,
lower_unit_id=higher_unit_id
```

```
    ).first()
    if conversion:
        return higher_unit_id

    for conversion in unit_conversions:
        if conversion.higher_unit_id == higher_unit_id:
            common_unit = find_common_unit(
                conversion.lower_unit_id, lower_unit_id,
unit_conversions
            )
            if common_unit:
                return common_unit
    return None


def convert_amount(amount, from_unit_id, to_unit_id,
unit_conversions):
    if from_unit_id == to_unit_id:
        return amount

    conversion = unit_conversions.filter(
        higher_unit_id=from_unit_id, lower_unit_id=to_unit_id
    ).first()
    if conversion:
        return amount * conversion.value

    conversion = unit_conversions.filter(
        higher_unit_id=to_unit_id, lower_unit_id=from_unit_id
    ).first()
    if conversion:
        return amount / conversion.value

    common_unit_id = find_common_unit(
        from_unit_id, to_unit_id, unit_conversions)
    if common_unit_id:
        return convert_amount(
            convert_amount(amount, from_unit_id,
```

```
                            common_unit_id, unit_conversions),
            common_unit_id, to_unit_id, unit_conversions
        )

    return None
```