

A
Mini Project Report
On
“Fitbite:(Fuel your taste and fitness)”

Submitted in partial fulfillment of the requirements for the
Degree

Third Year Engineering – Computer Science Engineering (Data Science)

By

SANJANA NAIK	23107035
JAGDISHKUMAR NAYAK	23107138
ADARSH PANIGRAHI	23107116
RUSHIKESH PASKANTI	23107032

Under the guidance of

Ms.Richa Singh



DEPARTMENT OF COMPUTER SCIENCE ENGINEERING (DATA SCIENCE)
A.P. SHAH INSTITUTE OF TECHNOLOGY
G.B. Road, Kasarvadavali, Thane (W) - 400615
UNIVERSITY OF MUMBAI

Academic year: 2025-26

CERTIFICATE

This to certify that the MiniProject report on Fitbite:(Fuel your taste and fitness) has been submitted by Sanjana Naik (23107035), Jagdishkumar Nayak (23107138), Adarsh Panigrahi (23107117) and Rushikesh Paskanti (23107032) who are bonafide students of A. P. Shah Institute of Technology, Thane as a partial fulfillment of the requirement for the degree in **Computer Science Engineering (Data Science)**, during the academic year **2025-2026** in the satisfactory manner as per the curriculum laid down by University of Mumbai.

Ms. Richa Singh

Guide

Dr. Pravin Adivarekar

HOD, CSE (Data Science)

Dr. Uttam D. Kolekar

Principal

External Examiner:

1.

Internal Examiner:

1.

Place: A. P. Shah Institute of Technology, Thane

Date:

ACKNOWLEDGEMENT

This project would not have come to fruition without the invaluable help of our guide **Ms. Richa Singh** gratitude towards our HoD, **Dr. Pravin Adivarekar**, and the Department of Computer Science and Engineering (Data Science) for providing us with the opportunity as well as the support required to pursue this project. We would also like to thank our project coordinator **Ms. Aavani Nair** who gave us her valuable suggestions and ideas when we were in need of them. We would also like to thank our peers for their helpful suggestions.

TABLE OF CONTENTS

Abstract

1. Introduction.....	1-5
1.1. Purpose.....	2
1.2. Problem Statement.....	2-3
1.3. Objectives.	3-4
1.4. Scope.....	4-5
2. Literature Review.....	6
3. Proposed System.....	7-8
3.1. Features and Functionality.....	8
4. Requirements Analysis.	9-10
5. Project Design.....	11-21
5.1. Use Case diagram... ..	11-12
5.2. DFD (Data Flow Diagram)	13-15
5.3. System Architecture.....	16-18
5.4. Implementation.....	19-21
6. Technical Specification.....	22-23
7. Project Scheduling.....	24-25
8. Results.....	26-27
9. Conclusion	28
10. Future Scope	29
Reference	

ABSTRACT

The burgeoning challenge of meal planning, constrained by available ingredients, time, and increasingly complex dietary requirements, necessitates an intelligent and efficient solution. This report details the design and implementation of FitBite, an AI-powered recipe recommendation system engineered to alleviate this burden. FitBite leverages Natural Language Processing (NLP), specifically TF-IDF Vectorization and Cosine Similarity, to match user-inputted ingredients with a vast dataset of recipes.

The system incorporates multi-criteria filtering, allowing users to refine suggestions based on dietary restrictions (e.g., vegan, gluten-free), preferred cuisine, estimated preparation time, and user ratings. Furthermore, FitBite includes a Meal Planner feature that utilizes user history and favorite recipes to auto-generate day- or week-long meal plans, thereby promoting consistency in healthy eating.

The front-end is implemented using Tkinter for a cross-platform desktop application, integrated with a Flask backend to manage the recommendation engine and persistent data storage (JSON/CSV). Performance evaluation validates the system's efficacy in delivering fast, highly relevant, and nutritionally balanced recipe suggestions, aligning with the Sustainable Development Goal (SDG) 3: Good Health and Well-Being by making healthy food choices more accessible and personalized.

Chapter 1

Introduction

This chapter sets the foundational context for the FitBite project by clearly defining the real-world problem of meal planning complexity, decision fatigue, and the need for personalized dietary solutions in modern life. We begin by establishing the core motivation behind developing an AI- driven system to address these culinary inefficiencies, which often lead to suboptimal food choices and avoidable waste. Crucially, the chapter maps the project's goals to the broader global context, specifically aligning FitBite with the United Nations Sustainable Development Goal (SDG) 3: Good Health and Well-Being. Finally, we enumerate the clear and measurable objectives from developing the NLP-based recommendation engine to implementing the personalized Meal Planner that guided the entire development process, establishing the essential framework for the subsequent technical discussions.

1.1 Purpose

In contemporary society, the decision of "What should I cook?" is fraught with complexities stemming from busy lifestyles, an overwhelming volume of available recipes, and increasing awareness of personal health and dietary needs. Traditional recipe discovery methods, which rely on manual searching through websites or recipe books, are inherently inefficient. They fail to dynamically account for the specific ingredients a user currently possesses, the time available for preparation, or specific health-related dietary restrictions. The core problem addressed by FitBite is the cognitive load and inefficiency associated with daily meal decisions. This often leads to suboptimal choices, resulting in food waste (due to unused ingredients) or reliance on unhealthy, pre-packaged meals. Motivation:

1. Reduce Decision Fatigue: To provide instant, highly relevant suggestions, minimizing the time and mental effort spent on meal planning.
2. Encourage Healthier Eating: By seamlessly integrating dietary restrictions and nutritional information, the system promotes conscious, healthier food choices.
3. Optimize Resource Utilization: The primary functionality focuses on matching recipes to available ingredients, helping users make the most of their pantry and reducing food waste.

1.2 Problem Statement

In contemporary society, the decision of "What should I cook?" is fraught with complexities stemming from busy lifestyles, an overwhelming volume of available recipes, and increasing awareness of personal health and dietary needs. Traditional recipe discovery methods, which rely on manual searching through websites or recipe books, are inherently inefficient.

They fail to dynamically account for the specific ingredients a user currently possesses, the time available for preparation, or specific health-related dietary restrictions. The core problem addressed by FitBite is the cognitive load and inefficiency associated with daily meal decisions. This often leads to suboptimal choices, resulting in food waste (due to unused ingredients) or reliance on unhealthy, pre-packaged meals.

The FitBite project also directly supports the United Nations Sustainable Development Goal SDG 3: Good Health and Well-Being.

- **Healthy Meal Suggestions:** FitBite actively prioritizes diet-specific recommendations (e.g., low-calorie, high-protein, vegan, balanced meals).
- **Personalized Nutrition:** The system's ability to filter based on allergies (gluten-free, nut-free) and lifestyle choices (vegetarian, paleo) ensures that healthy food choices are not only suggested but are also practically feasible for the individual user.

1.3 Objectives

The main objectives of FitBite are as follows:

- To dynamically recommend food recipes based on the user's Body Mass Index (BMI) by computing the optimal caloric intake using dietary reference models and mapping it to a curated recipe dataset.
- To implement a content-based filtering algorithm utilizing cosine similarity for computing similarity scores between user profiles and recipe feature vectors, ensuring accurate and ranked recipe recommendations.
- To integrate Natural Language Processing (NLP) for extracting and categorizing nutritional metadata (e.g., calories, macros, and ingredients) from recipe descriptions to enhance dataset quality and improve model interpretability.
- To deploy a predictive analytics model that estimates the impact of daily meal selections on user health metrics such as BMI variation, caloric surplus/deficit trends, and nutrient imbalance.

1.4 Scope

Scope: The project's scope is confined to the development of a standalone desktop application using Python (Flask and Tkinter).

- **Input Data:** The system operates on a pre-processed static dataset of approximately 4374 recipes, which includes ingredients, preparation time, nutritional values, cuisine, and user ratings.

- **Core Functionality:** The focus is on the ingredient-based recommendation, multi-filter application, and the 7-day meal planner.
- **Personalization:** Personalization is based on locally stored user preferences, favorites, and search history.
- **Exclusions:** The project does not include real-time web scraping for recipe data, integration with external APIs for complex nutritional analysis (beyond the dataset), or full-scale deployment as a web or mobile application (though the Flask backend provides a foundation for future web deployment).
- **Applicability:** FitBite's applicability extends across various user groups
- **Home Cooks:** Individuals seeking quick and creative meal ideas based on available pantry items.
- **Health-Conscious Users:** Those adhering to strict dietary plans (e.g., weight loss, muscle gain) or managing food-related health conditions.

Chapter 2

Literature Review

The literature review provides a comprehensive analysis of existing research, frameworks, and technologies that have shaped the evolution of recipe recommendation systems (RS). It focuses on prior advancements in recommender systems, natural language processing (NLP), and hybrid filtering architectures while emphasizing the growing need for health-aware and practical solutions that account for real-world user constraints. This chapter examines key studies and filtering techniques that form the foundation for FitBite, highlighting how each has contributed to areas such as ingredient matching, health constraint management, and user-centric filtering. By critically evaluating these developments, the review identifies both the innovations achieved and the limitations that FitBite aims to overcome through its multi-constraint, utility-focused architecture.

The literature review explores the technological foundation and existing research in AI-driven food recommendation, with a focus on hybrid filtering models and health-aware systems. It examines previous developments in content-based filtering (CBF) , collaborative filtering (CF) , and the NLP techniques that enable them, such as TF-IDF vectorization and Cosine Similarity . This analysis highlights the innovations, limitations (such as data sparsity and failure to handle rigid constraints), and ongoing research that inspired the development of FitBite, an intelligent recommendation system focused on practical, real-world utility.

Wu, W., et al. (2021) in "A Personalized Recipe Recommendation System Using Machine Learning" applied collaborative filtering and content-based filtering with ML models to provide personalized recommendations based on user preferences and dietary needs.[1]

Ge, M., Ricci, F., & Massimo, D. (2015) in their "Health-Aware Food Recommendation System" designed a hybrid recommendation model that integrates content-based filtering with health-aware constraints, balancing taste preferences with nutritional requirements. This work was seminal in highlighting that traditional recommenders, focused on taste, failed the public health mandate. FitBite is philosophically aligned with this research, using dietary tags as non-negotiable filters to support overall health goals.[2]

Teng, C. Y., Lin, Y. R., & Adamic, L. A. (2012) explored "Recipe Recommendation Using Ingredient Networks." This research involved building ingredient co-occurrence networks and applying graph-based algorithms to recommend recipes by analyzing complementary and substitutable ingredients.[3]

Kusmierczyk, T., Trattner, C., & Nørvåg, K. (2015) introduced "Temporal Patterns in Recipe Recommendation Systems," which analyzed temporal features (like time of day or seasonality) combined with collaborative filtering to enhance personalization.[4]

In summary, existing research demonstrates substantial progress in AI-driven recipe recommendation, particularly in developing hybrid models that blend Content-Based Filtering (CBF) and Collaborative Filtering (CF).

The reviewed systems emphasize the power of NLP for ingredient processing , the limitations of pure CBF (the "filter bubble") and CF (data sparsity , disregard for practical constraints), and the critical need for health-aware constraints . A clear gap emerges in end-to-end utility and automation. Commercial recipe apps often fail to allow granular, multi-constraint filtering (like Prep Time and User Rating) and provide only single-meal solutions, offering no tool for automated long-term planning.

FitBite aims to merge these advantages into a unified, high-utility framework, positioning itself as a Multi-Constraint Ingredient Matching (MCIM) system. It combines the algorithmic excellence of the hybrid TF-IDF/Cosine Similarity core with high-utility, integrated features like the Automated 7-Day Meal Planner, bridging the gap between mere discovery and prescriptive, personalized culinary adherence.

Chapter 3

Proposed System

The Fitbite system architecture is compartmentalized into presentation, application, and data layers, enabling modular development and reliable performance. The core innovation lies in integrating the heavy computational logic (Hybrid Recommendation Model) hosted on a Flask API with the lightweight Tkinter GUI, all running within a managed desktop environment. The foundation of Fitbite's recommendation capability rests on a robust and meticulously cleaned dataset.

The primary source for recipe information is a comprehensive dataset named recipesupd.csv, containing structured data². The dataset size is defined as 4374x10, reflecting 4,374 recipes with 10 features each. Prior to modeling, the raw data undergoes critical preprocessing, including ingredient parsing and normalization, and transformation into numerical feature vectors using the TF-IDF Vectorizer. Fitbite utilizes a personalized approach that begins immediately upon user entry. The Presentation Layer includes a dedicated Login and Sign-Up page. During registration, users are required to specify explicit preference data, including preferred cuisine types and critical dietary restrictions (e.g., vegan, gluten-free, low-calorie). User credentials are managed using the py-bcrypt library³ and profile data is securely stored in the local users.json file. The Fitbite recommendation engine leverages a synergistic model to maximize relevance and diversity. The Content-Based Filtering (CBF) module is responsible for identifying recipes that are similar in content to what the user currently possesses. Recipes violating pre-set dietary or health constraints are immediately filtered out. The model calculates the Cosine Similarity between the user's ingredient vector and all compliant recipe vectors. The Collaborative Filtering (CF) module introduces diversity and serendipity by leveraging collective user behavior. User behavior, including recipe ratings and favoriting (stored in favorites.json), is collected. The system calculates the similarity between the target user and all other users. Recipes highly rated by similar users are predicted for the target user. These two independent lists are then fused and weighted... to generate the final, comprehensive set of tailored recipe recommendations.

3.1 Features and Functionality

- **Recipe Finder** This feature allows users to search for recipes either by name or by the available ingredients they have on hand. The system then displays comprehensive results that include nutrition facts, step-by-step instructions (procedure), and the recipe's rating. This entire process is driven by the Flask backend, which calls the Hybrid Recommendation Model and uses TF-IDF vectorization to match the user's query.
- **Meal Planner** The Meal Planner is a key feature that auto-suggests a 7-day meal plan, complete with options for Breakfast, Lunch, and Dinner. This auto-suggestion mechanism strategically generates the plan

by sampling from a weighted pool of highly-rated recipes, recipes the user has explicitly saved in their favorites, and recipes from their recent search history. This logic is designed to ensure a diverse set of suggestions, preventing repetitive meals and promoting long-term user satisfaction.

- **Statistics** This feature provides users with a dashboard displaying their personal metrics. It shows data such as the average rating they have given to recipes, their total number of saved favorite recipes, and a list of their most-searched ingredients. This is powered by a Statistics Generator module that utilizes data from the user's interaction logs.
- **Voice Input** To minimize user input friction, the system integrates a voice input capability using the pyaudio speech recognition library. This allows users to verbally list their available ingredients, which the system then converts into a searchable text query that is fed directly into the ingredient matching function.
- **Favorites Management** This feature allows users to store their preferred recipes for quick and easy access. All saved recipes are stored locally in the favorites.json file, ensuring data persistence and providing a data source for the Meal Planner's recommendations.

Chapter 4

Requirements Analysis

The requirement analysis for the FitBite project identifies and documents the functional and non-functional requirements that the system must meet to fulfill its objectives effectively. This chapter outlines the capabilities the system must provide (functional requirements) and the qualitative attributes of how the system must operate (non-functional requirements), focusing on properties such as performance, security, and usability .

Functional Requirements:

- **User Authentication:** The system must allow users to securely register and log in . User passwords must be stored securely using a robust hashing algorithm (py-bcrypt) .
- **Preference Elicitation:** The system must be able to capture and store explicit user dietary preferences, such as vegan or gluten-free, as well as their cuisine interests .
- **Hybrid Recommendation Core:** The system is required to employ a weighted hybrid model . This model must combine Content-Based Filtering with Collaborative Filtering to generate a ranked list of recommendations .
- **Real-Time Ingredient Matching:** The system must be able to dynamically filter the entire recipe catalog based on a list of available ingredients provided by the user . This filtering must generate results within the specified non-functional latency constraints .
- **Health Constraint Filtering:** The system must enforce mandatory health and dietary constraints that have been defined by the user .
- **Nutritional Compliance Enforcement:** The system must use its structured nutritional data to enforce user-defined health rules . This includes capabilities like filtering all results that exceed a user-set calorie threshold .
- **Meal Planning Feature:** The system must be able to generate an automated, balanced 7-day meal plan, including Breakfast, Lunch, and Dinner . This plan generation should be based on the user's history and favorite recipes .
- **Persistency of Meal Plan Data:** The Meal Planner feature must store its generated plans, as well as any user overrides, in a persistent JSON format on the local machine .
- **User Feedback Loop:** The system must provide functionality for users to rate recipes, save recipes as favorites, and track their search history . This data is used to refine the Collaborative Filtering model .

- **Accessibility:** The system must integrate voice input capability using the pyaudio library, allowing users to submit their ingredients via speech .

Non-Functional Requirements:

- **Performance:** Performance requirements dictate how fast the system must operate to maintain user satisfaction . The recommendation latency must be less than or equal to 3 seconds from the moment of ingredient submission to the display of results . Latency exceeding this threshold is known to cause user frustration . The system must also be able to efficiently process the 4374x10 recipe dataset in memory, which necessitates the use of sparse matrix operations for TF-IDF and similarity calculations .
- **Scalability:** Scalability determines the system's ability to handle future growth . The machine learning core must be able to scale vertically, primarily through library optimization, to handle a 50% increase in the recipe dataset size without requiring significant architectural changes . Furthermore, the Flask API endpoints must be modularly structured to allow for the seamless integration of new components without disrupting core services .
- **Security and Integrity:** Security requirements are in place to protect user data and ensure system resilience . All critical user interaction data, specifically users.json and favorites.json, must be persisted locally on the user's machine to maximize confidentiality . User passwords must be protected using a robust hashing algorithm (py-bcrypt) .
- **Usability:** Usability focuses on the ease of use and the quality of the user interface experience . The Tkinter GUI must present an intuitive interface with clearly navigable sections for the Recipe Finder, Meal Planner, and Statistics Dashboard . Crucially, the GUI must remain fully interactive and responsive even during heavy ML computation . This is achieved by isolating API calls to the Flask backend via threading, which prevents the interface from freezing during recommendation searches and ensures a usable system .

Chapter 5

Project Design

The project design details the logical structure, data flow, and mathematical core of the Fitbite system, establishing how the requirements are translated into a working application. It defines how different system components—including the **Tkinter frontend interface**, the **Flask backend logic**, and the **Hybrid Recommendation models**—interact to fulfill the project's objectives. This section translates theoretical requirements into structured, implementable models through visual and functional representations such as Use Case Diagrams , Data Flow Diagrams (DFDs) , and the System Architecture. Each design element is created to ensure modularity, low latency, and a responsive user experience. By detailing the data flow, component interactions, and system operations, this chapter lays the groundwork for the successful implementation and performance of Fitbite as an intelligent recipe recommendation system.

5.1 Use Case Diagram

The Use Case Diagram visually defines the functional boundaries and interactions between the primary actor (the User) and the Fitbite System. It captures the main processes, including preference setting, searching, content consumption, and feedback submission.

Actors:

- User: Represents the primary user interacting with Fitbite to manage authentication, search for recipes, manage favorites, and provide inputs (like ingredients).
- System: Represents the Fitbite System that processes user requests, handles data, and displays information back to the user.

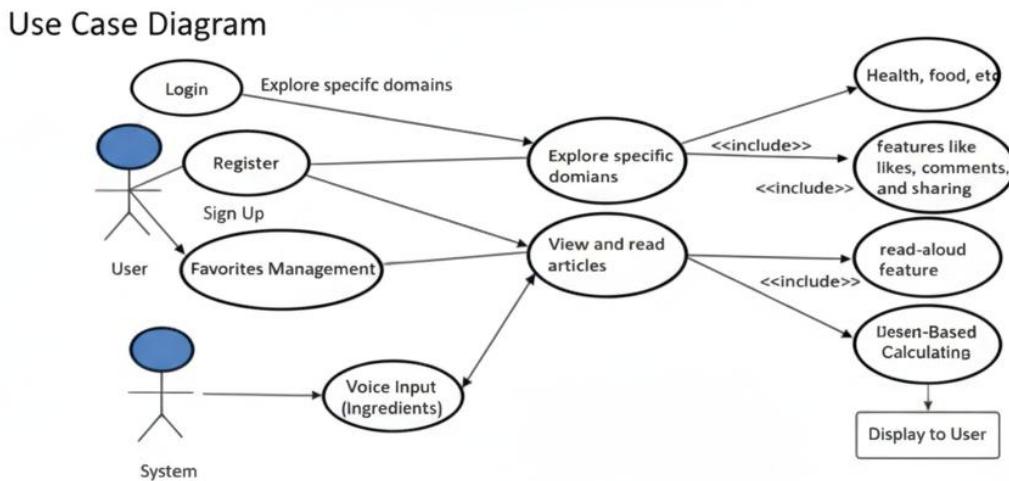


Fig 5.1: Use Case Diagram

Functional Area Use Cases:

1. Authentication (Sign Up, Login): Securely register and gain access to personalized features.
2. Recipe Management (View Recipe Details, View Nutrition Facts): Access step-by-step instructions and nutritional breakdown of a selected recipe.
3. Meal Planning (Generate Meal Plan, View Meal Planner): Auto-generate a 7-day meal plan and view/manage the resulting schedule.
4. Analytics (View Statistics, Rate Recipes): Access personal usage metrics and submit ratings to improve recommendations.
5. Search & Discovery (Search Recipes): Search the recipe catalog by name or general query.
6. Filtering Options (Filter by Ingredients, Cuisine, Diet Type, Prep Time): Apply specific constraints (e.g., time, cuisine, diet) to refine the recommendation results.
7. Favorites (Add to Favorites, View Favorites): Save preferred recipes and access the personalized saved list.
8. Advanced Features (Voice Input - Ingredients): Submit ingredients hands-free using speech recognition technology.

5.2 DFD (Data Flow Diagram):

The Data Flow Diagram (DFD) provides a clear, visual representation of the Fitbite system's processes and the journey of data through it. It goes beyond a static architectural diagram to illustrate the *dynamic flow* of information. This process begins with the **User** (an external entity) providing initial inputs, such as **login credentials, search queries** (like ingredients and filters), and **preferences**. The DFD then maps how this data is received and transformed by the system's main components, or 'processes,' like **User Authentication, AI Recommendation**, and the **Meal Planner**. It also shows how these processes interact with the **Data Stores** (like users.json or recipesupd.csv) by reading or writing information . Finally, the diagram shows the resulting outputs, such as the **Recipe Results, Meal Plans, and Statistics** that are delivered back to the user. By breaking this flow into a high-level Context Diagram (Level 0) and a more detailed breakdown (Level 1), the DFD serves as a crucial tool for detailing how the system's main components work together to process information and fulfill the user's requests

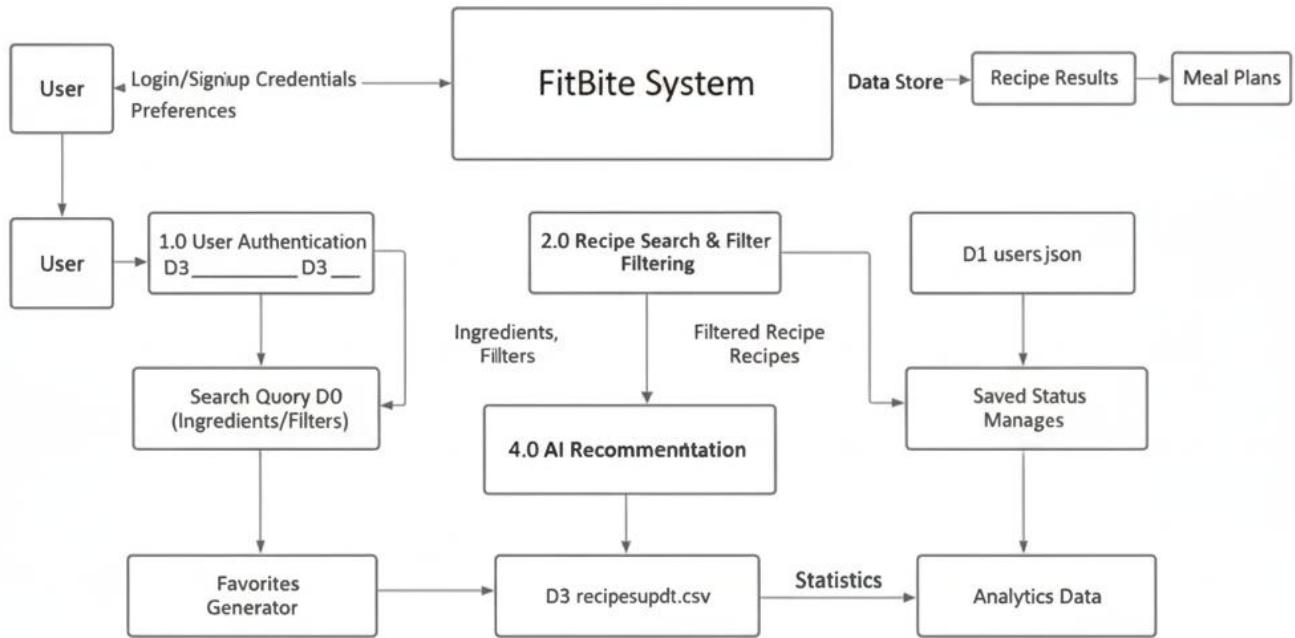


Fig 5.2: Data Flow Diagram

DFD Level 0: Context Diagram

The Level 0 DFD, or Context Diagram, shows the entire system as a single process and its interaction with External entities

- **System Boundary:** The FitBite System is represented as a single process.
- **Entities:** The external entities are the User and the Data Store.
- **Inputs (User to System):** These include Login/Signup credentials, a Search Query (containing ingredients/filters), and user Preferences (like diet and cuisine).
- **Outputs (System to User/Store):** These include the final Recipe Results (recommendations), generated Meal Plans, and Statistics (analytics data).

DFD Level 1: Main Processes and Data Stores

The Level 1 DFD breaks the system down into major processes and details the interactions with the core data stores.

- **Data Stores:**
 - **users.json:** This store holds the user credentials and their explicit preferences.
 - **recipesupdt.csv:** This is the comprehensive recipe database, containing all ingredients, nutrition

information, and preparation steps.

- favourites.json: This store contains the user's saved favorite recipes and their history data.
- Main Processes:
 - User Authentication: This process handles secure credential management (registration and login) using py-bcrypt. It interacts with the users.json store.
 - Recipe Search & Filter: This process takes the user's search query and filters and performs constraint enforcement and initial data retrieval from the recipesupdt.csv store.
 - AI Recommendation: This process takes the filtered recipe list and applies the core recommendation model, using TF-IDF Vectorization and Cosine Similarity, to rank the results. It reads from the recipe and favorites data stores.
 - Meal Planner: This process runs the auto-suggestion algorithm by using data from the favourites.json store (like user favorites and history) to generate a 7-day meal plan.
 - Favorites Manager: This process provides direct read/write access to the favourites.json store, allowing users to save or remove recipes.
 - Statistics Generator: This process uses aggregation logic to process user activity data from various sources, like the favorites store, and generates the analytics data for display.

5.3 System Architecture:

The system's architecture is defined by the separation of the Presentation Layer (Tkinter GUI) and the Application Layer (Flask ML API), requiring a specific concurrency strategy for seamless desktop operation.

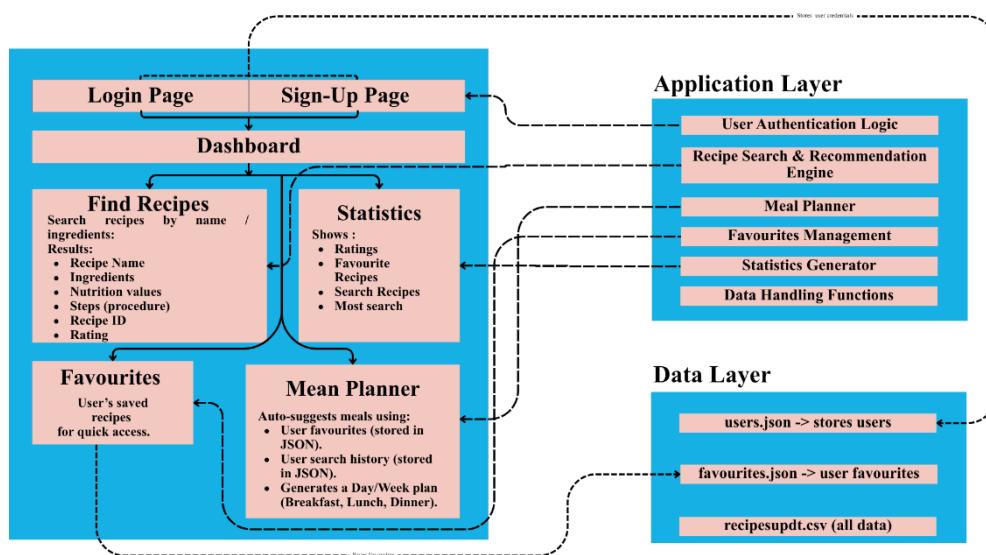


Fig 5.3: System Architecture

1. Data Collection: The system foundation rests on a pre-processed static dataset of 4,374 recipes, which includes ingredients, preparation time, nutritional values, cuisine, and user ratings . The system also collects dynamic data directly from the user, including explicit preferences (diet, cuisine) during registration and implicit data through user behavior, such as recipe ratings, saved favorites, and search history.
2. Data Preprocessing: Prior to modeling, the raw recipe data undergoes critical preprocessing to ensure feature accuracy. This involves Ingredient Parsing to standardize raw ingredient lists, remove extraneous data (e.g., "optional"), and normalize names to a controlled vocabulary. It also includes Unit Standardization to convert weights and measurements uniformly and Nutritional Verification to rigorously format fields like calories and fat content.
3. Feature Engineering: The cleaned and preprocessed ingredient lists are transformed into numerical feature vectors. This is accomplished using a TF-IDF Vectorizer , which generates vectors that capture the statistical importance of each ingredient within a specific recipe relative to the entire dataset.
4. Normalization and Matrix Creation: User behavior data (likes, ratings) is collected into a User-Item Interaction Matrix. The interaction scores within this matrix are then normalized using the MinMaxScaler technique, which scales all values to a common range (e.g., 0 to 1) to prepare the data for similarity calculations.
5. Similarity Computation: Cosine Similarity is computed between the TF-IDF vector of a user's ingredient query and the TF-IDF vectors of all recipes in the database. This calculation measures the semantic closeness (the "angle") between the query and the recipes, which is highly effective for sparse text data as it is indifferent to the magnitude or length of the ingredient lists .
6. Recommendation and Retrieval Functions:
 - Hybrid Recommendation: The system employs a weighted hybrid aggregation scheme. This function combines the Content-Based Score (from ingredient similarity) with the Collaborative Filtering Score (from taste prediction) to generate a final, unified rank.
 - Constraint Enforcement: A binary "switching logic" is applied to all ranked results. This function checks if a recipe violates any of the user's non-negotiable health constraints (e.g., "vegan," "gluten-free") and overrides its score to zero, removing it from the final list.
 - Meal Planning: An auto-suggestion algorithm generates a 7-day meal plan by strategically sampling

from the user's favorited recipes and recent search history.

7. Evaluation Metrics:

- System performance is evaluated using Predictive Accuracy metrics, including Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), to measure the error between predicted and actual user ratings .
- The quality of the ranked list is measured using Precision@K, which calculates the fraction of recommended items in the top-K list that are relevant to the user.
- User experience is assessed with non-accuracy metrics like Novelty (how unexpected the recommendations are) and Intra-List Diversity (ILD), which measures if the recommended items are distinct from one another to prevent monotonous suggestions .

5.4 Implementation:

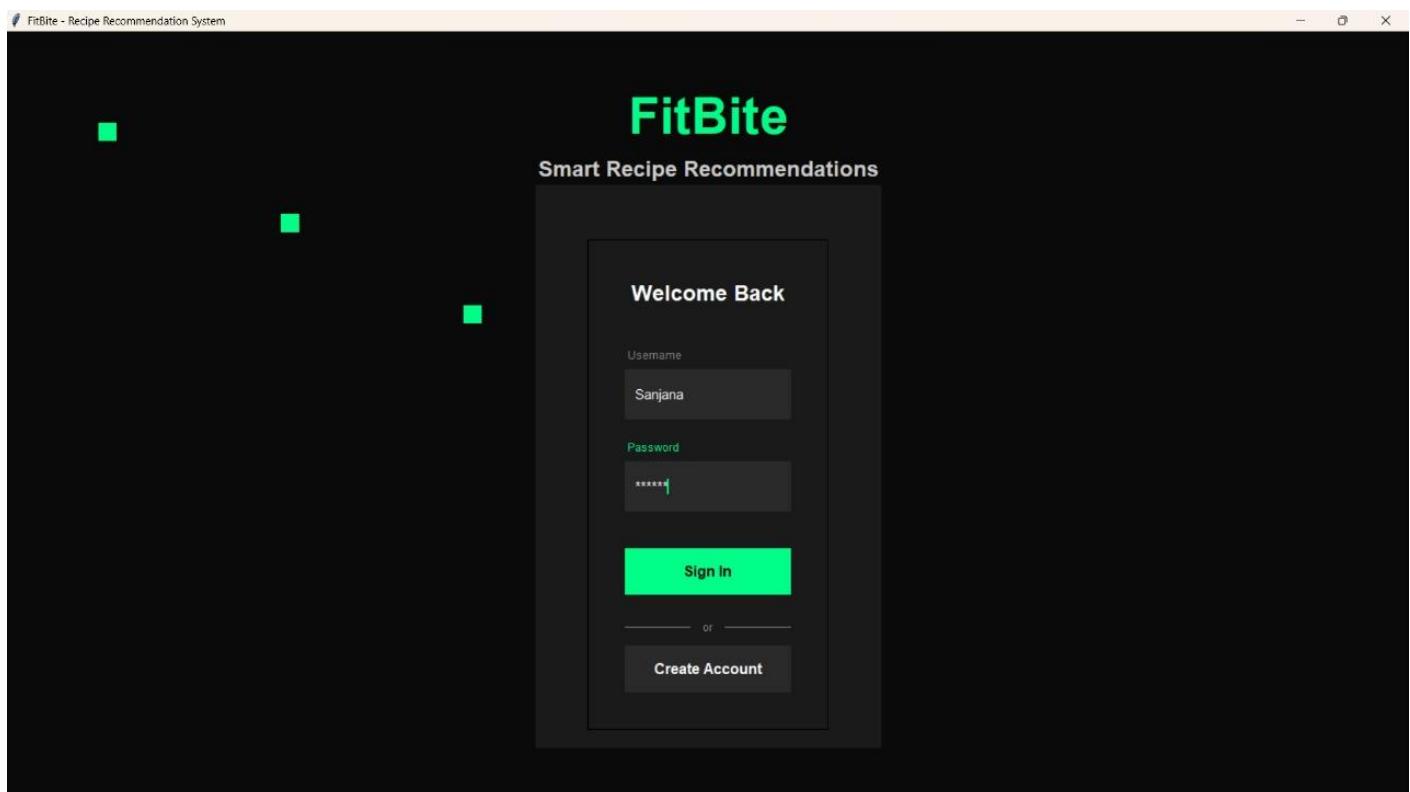


Fig 5.4.1.signin.jpg

This image shows the **Sign In** screen for the FitBite application, titled "Smart Recipe Recommendations." It includes input fields for "Username" and "Password," a green "Sign In" button, and a link below to "Create Account."

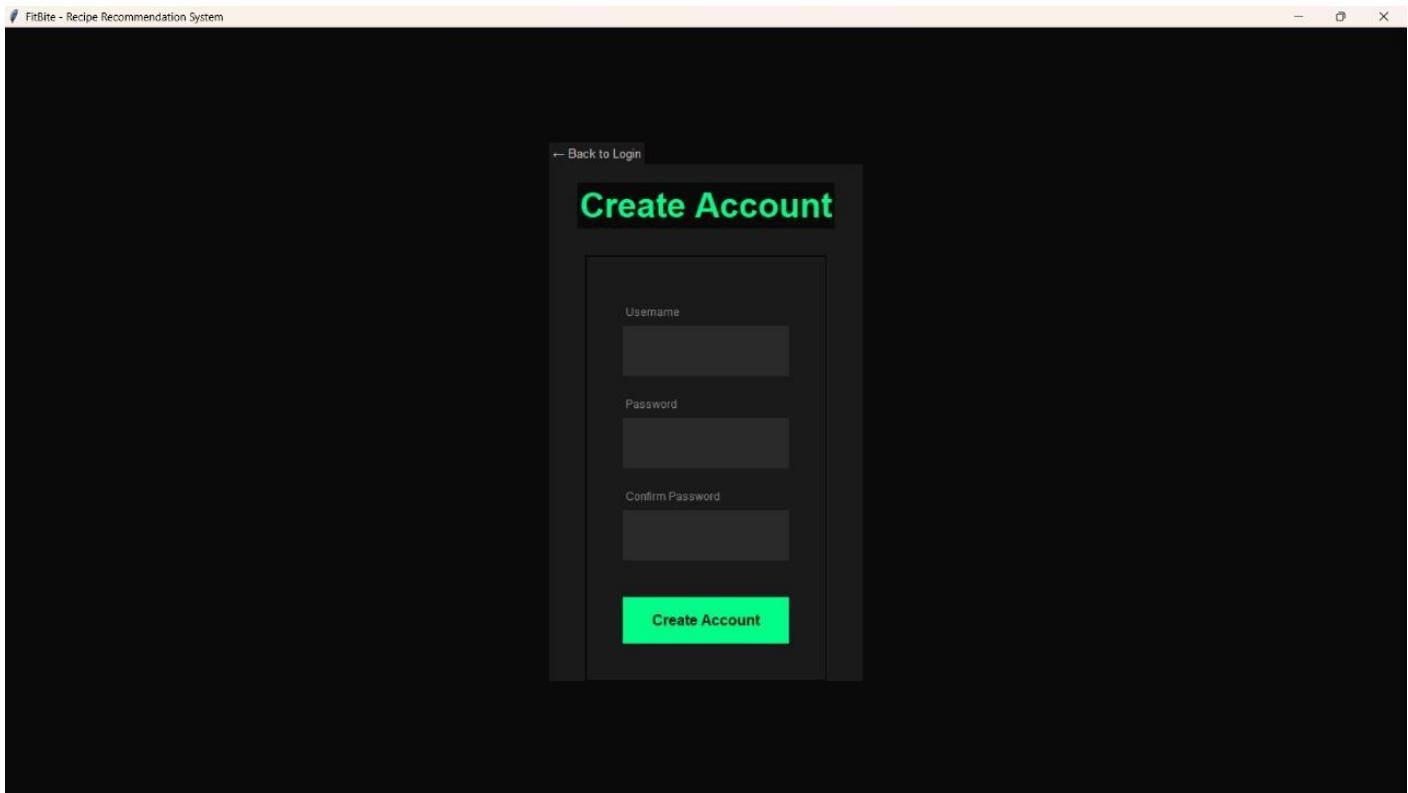


Fig 5.4.2.create.jpg

This screenshot displays the **Create Account page** for FitBite. It provides fields for "Username," "Password," and "Confirm Password." Users can submit their information via the "Create Account" button or go back using the "Back to Login" link.

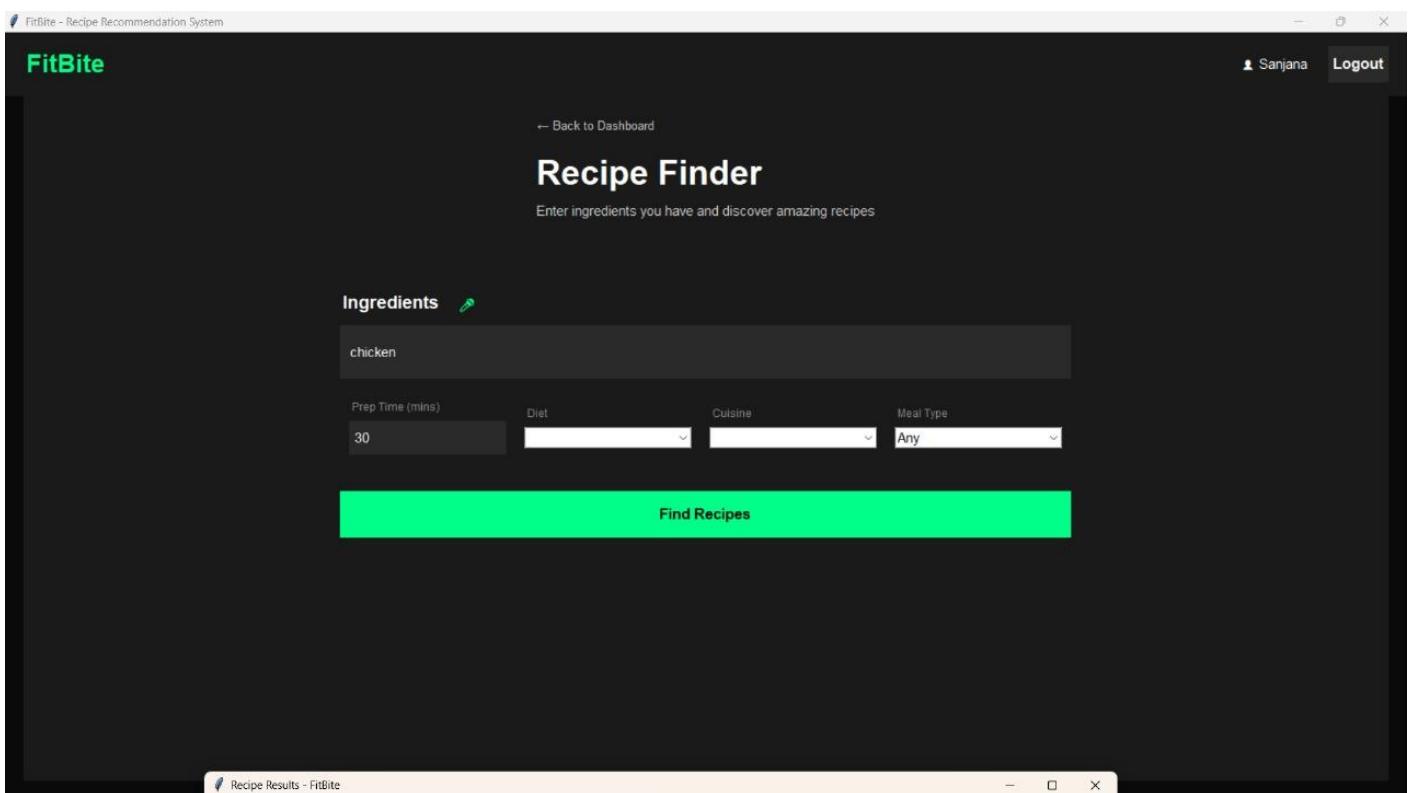


Fig 5.4.3.search.jpg

This image captures the **Recipe Finder interface**, the main search tool in FitBite. It prompts the user to "Enter ingredients you have" and shows "chicken" already entered. Below the ingredient input, there are filter options for "Prep time (mins)," "Diet," "Cuisine," and "Meal Type," followed by the "Find Recipes" button.

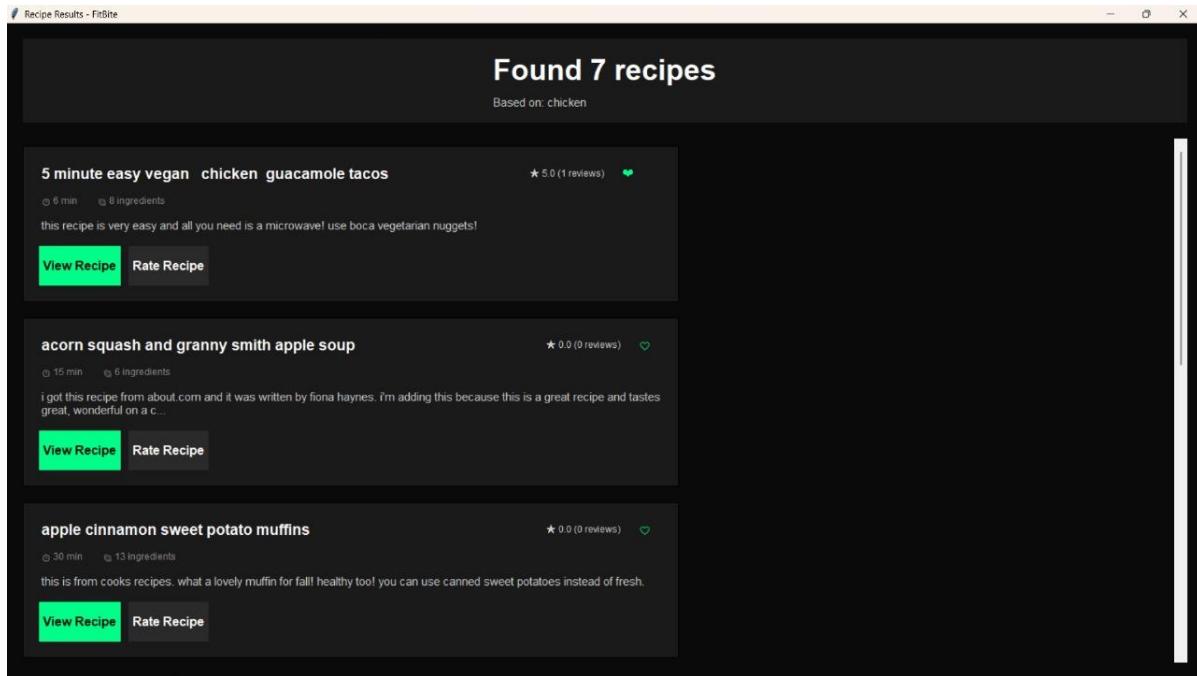
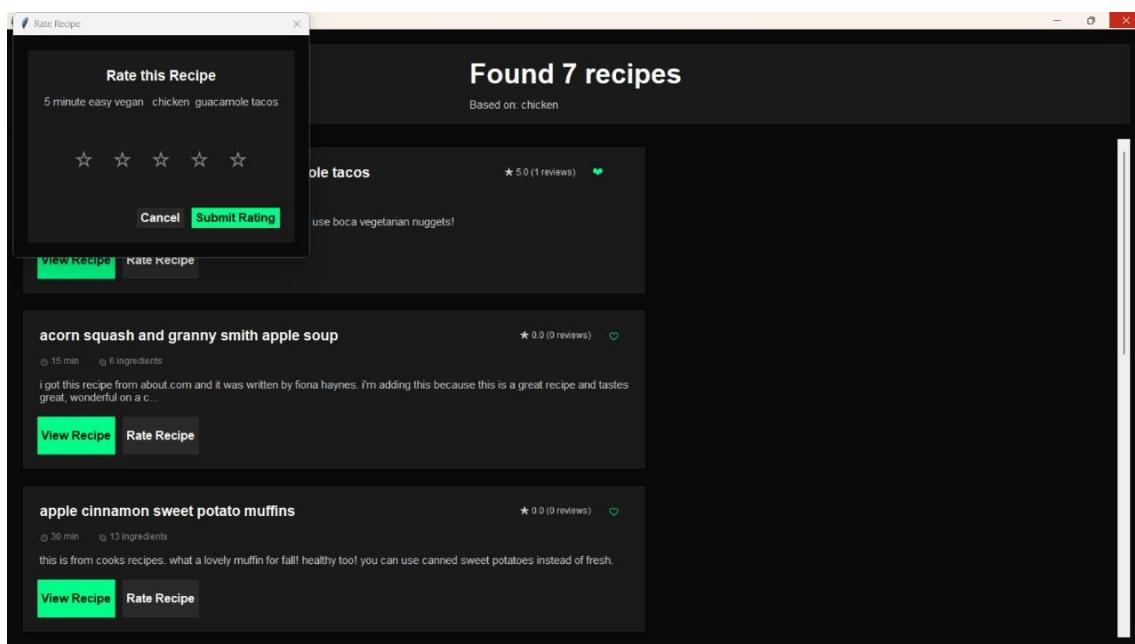
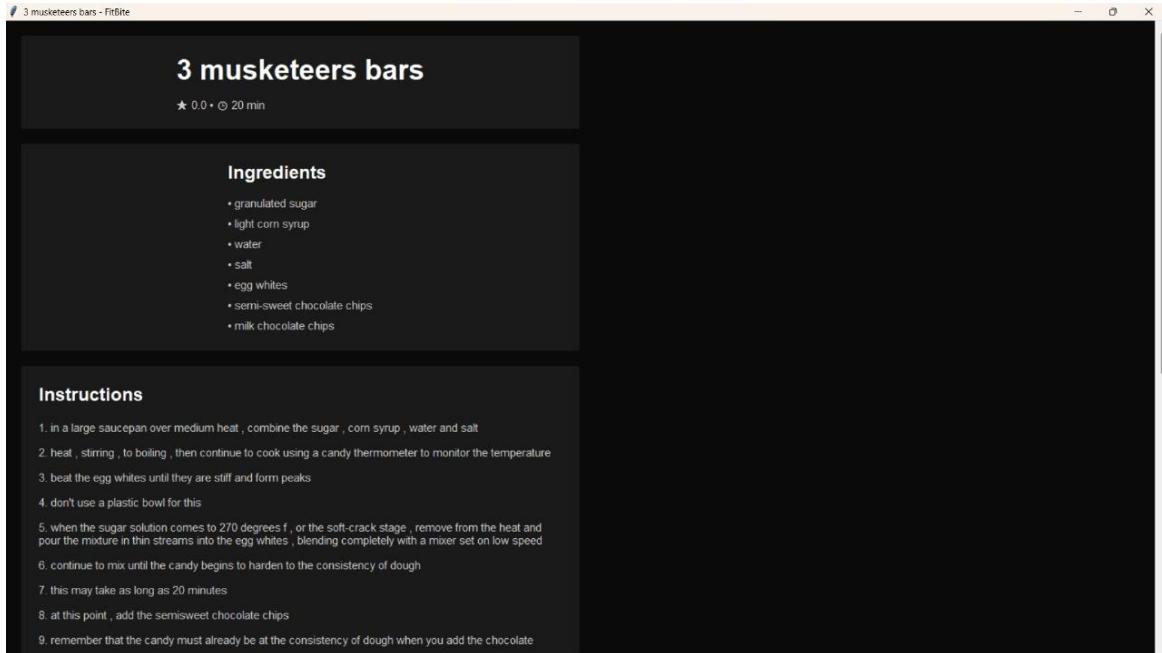


Fig 5.4.4.search result.jpg

This screenshot displays the **search results page** after a query. It indicates "Found 7 recipes" based on the ingredient "chicken." Several recipe cards are listed, such as "5 minute easy vegan chicken guacamole tacos" and "apple cinnamon sweet potato muffins," each with options to "View Recipe" and "Rate Recipe."



Fig



5.4.5.rating.jpg

This image shows the **search results page** again, but this time focusing on the user rating feature. A "**Rate this Recipe**" pop-up window is overlaid on the results, displaying the name of the recipe and five stars for rating selection, along with "Cancel" and "Submit Rating" buttons.

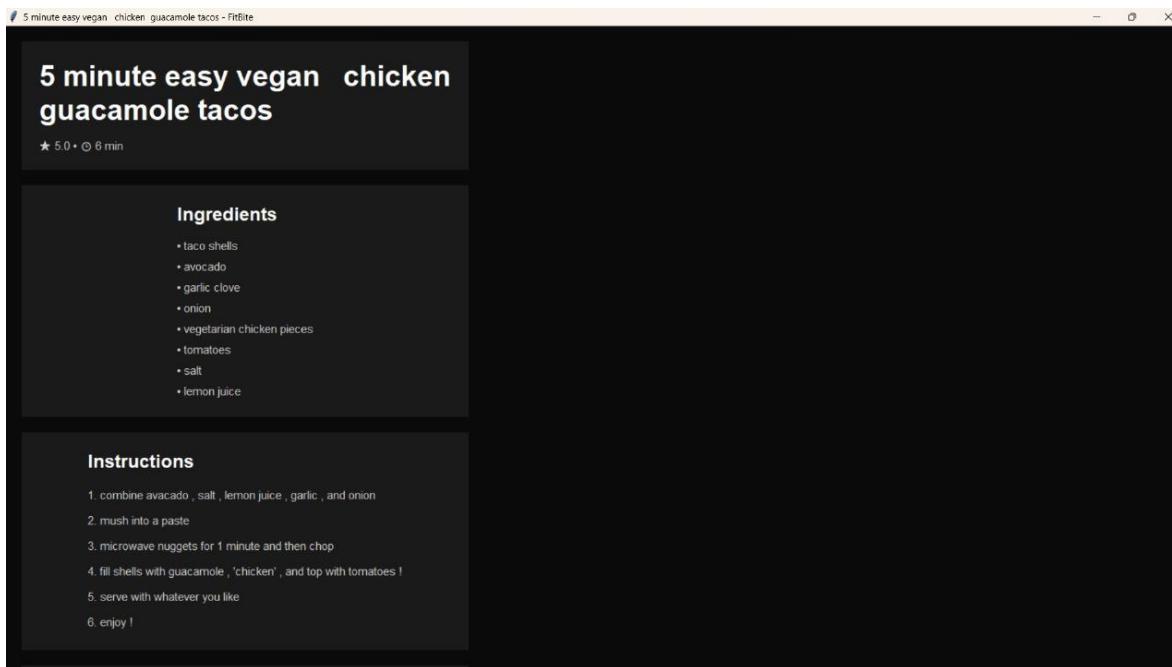


Fig 5.4.6.recipe.jpg

This screenshot presents the **detailed view for a specific recipe**: "5 minute easy vegan chicken guacamole tacos." The page is clearly divided into sections for "Ingredients" (listing items like taco shells, avocado, etc.) and "Instructions" (showing the step-by-step cooking process).

Fig 5.4.7.instruction and ingredient.jpg

This image displays the first part of the **detailed recipe page for "3 musketeers bars."** It shows the recipe title, rating/time, the "Ingredients" list (including granulated sugar, corn syrup, etc.), and the initial steps (1-9) in the "Instructions" section.

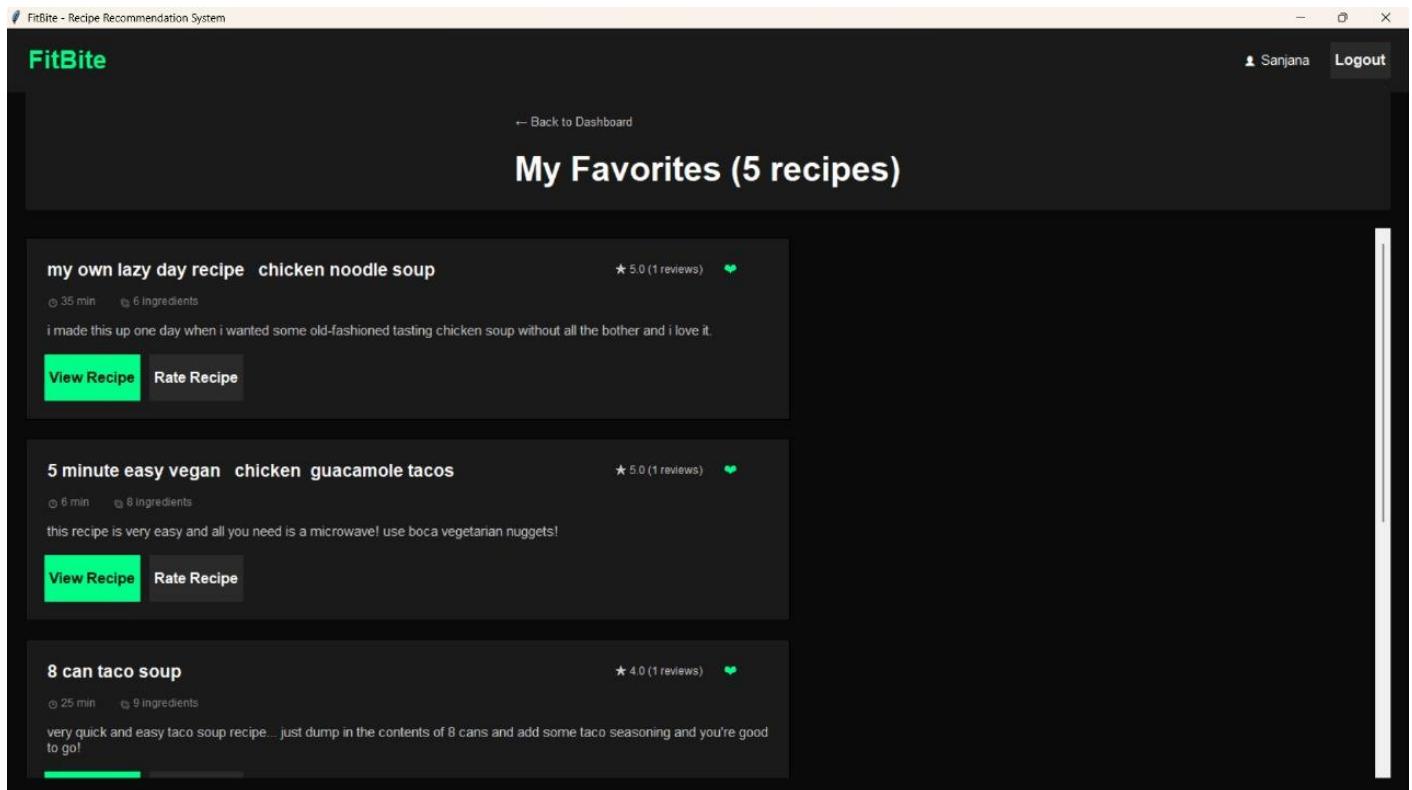


Fig 5.4.8.favourites.jpg

This screenshot shows the user's personalized "**My Favorites**" page. It lists "5 recipes" that the current user ("Sanjana") has saved, such as "my own lazy day recipe chicken noodle soup" and "8 can taco soup," providing quick access to preferred dishes.

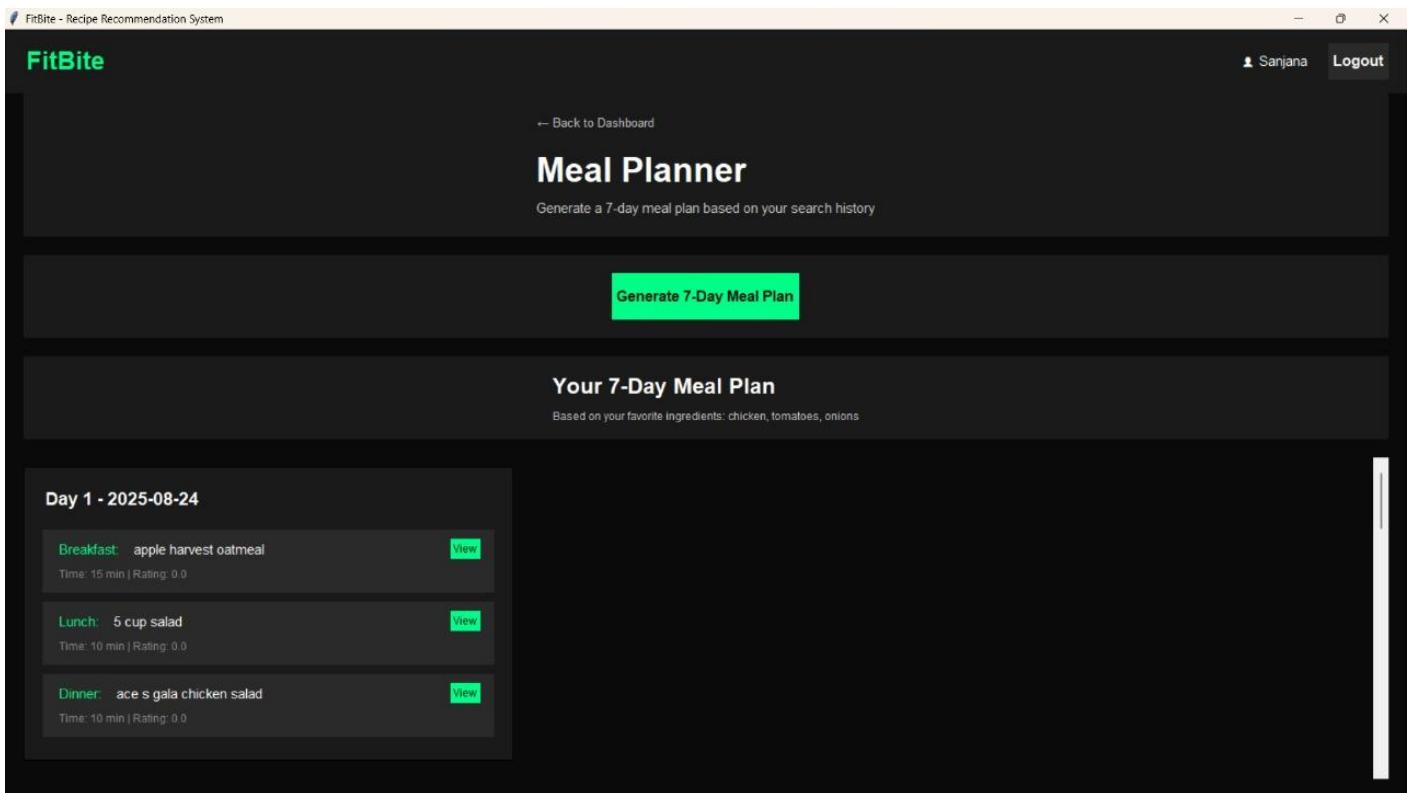


Fig 5.4.9.mealplanner.jpg

This image displays the "**Meal Planner**" feature within FitBite. It includes a button to "Generate 7-Day Meal Plan" based on search history and favorites. Below, it shows a sample generated plan for "Day 1," listing recipes for Breakfast, Lunch, and Dinner with options to "View" each one.

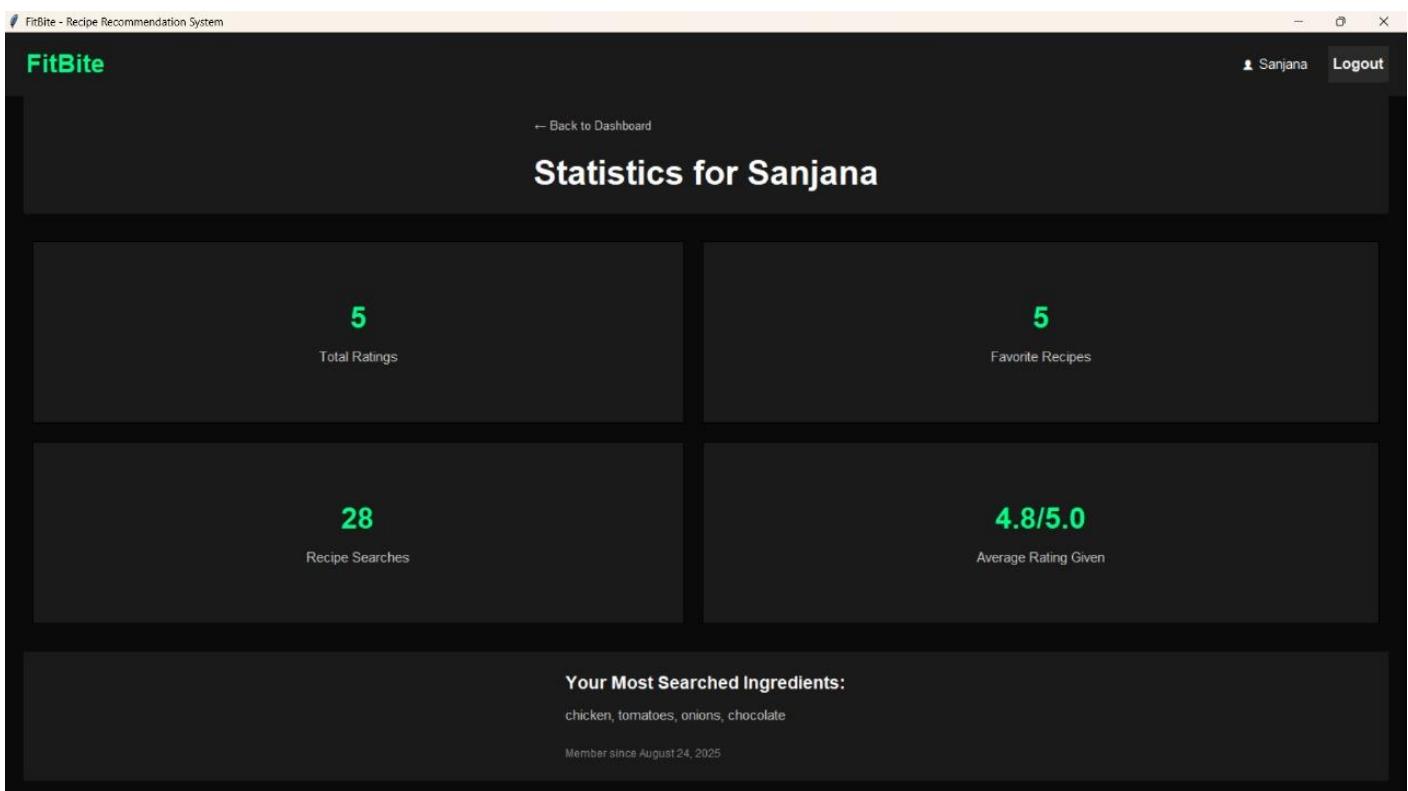


Fig 5.4.10.dashboard.jpg

This screenshot shows the "Statistics for Sanjana" dashboard. It visually summarizes the user's activity with key metrics displayed in cards: "Total Ratings" (5), "Favorite Recipes" (5), "Recipe Searches" (28), and "Average Rating Given" (4.8/5.0). It also lists the user's "Most Searched Ingredients."

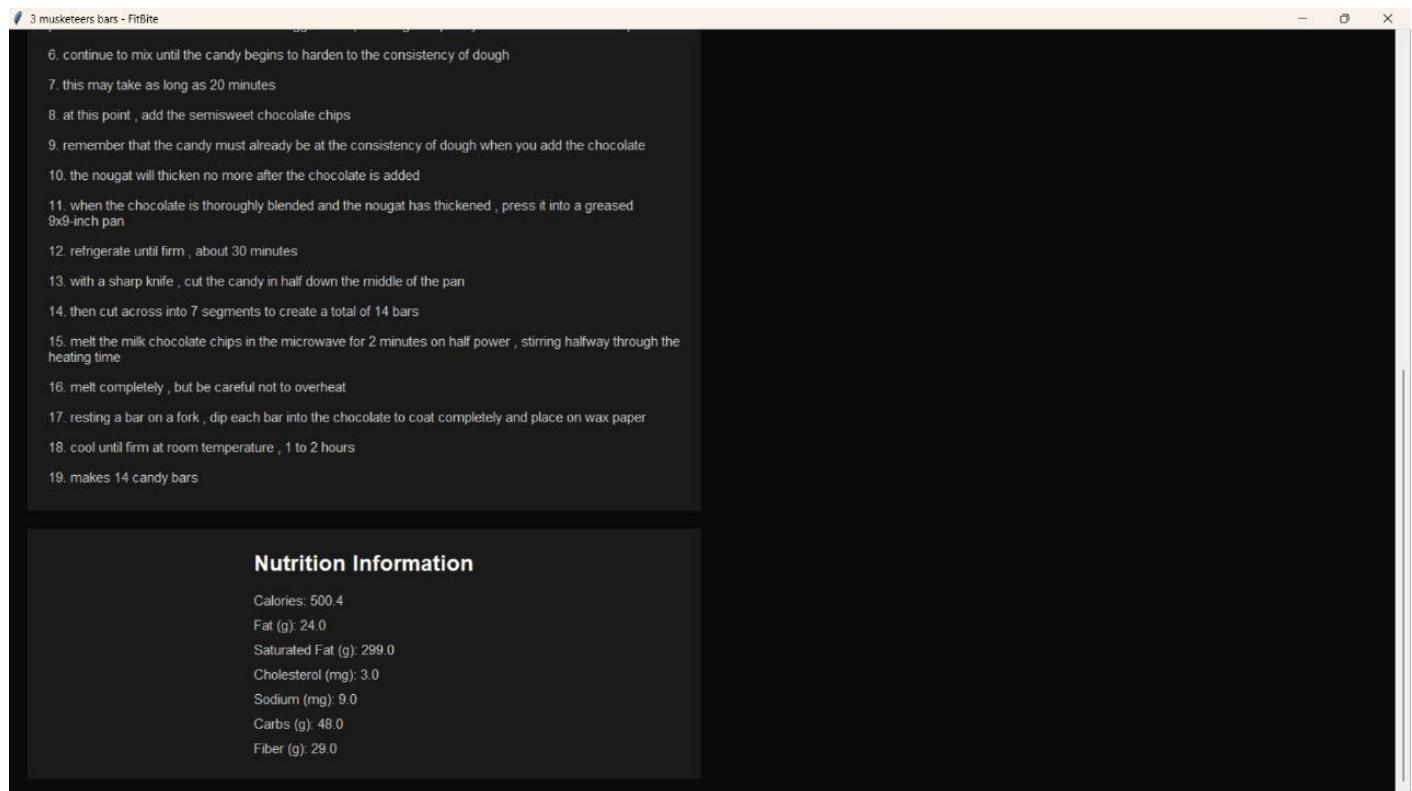


Fig 5.4.11.description.jpg

This image shows the continuation of the recipe instructions for "3 musketeers bars" within the FitBite application, covering steps 6 through 19. Below the instructions, there is a detailed "Nutrition Information" panel displaying key nutritional facts per serving, including Calories (500.4), Fat (24.0g), Saturated Fat (299.0g), Carbs (48.0g), and Fiber (29.0g).

Chapter 6

Technical Specification

The implementation phase focuses on selecting appropriate technologies and executing the complex concurrency strategy required to deploy the Fitbite system effectively as a local desktop application. The technical stack was meticulously chosen to meet the dual constraints of lightweight local deployment and high computational capability . This defines the technical environment and provides a blueprint that ensures efficiency, responsiveness, and reliability while prioritizing user data confidentiality .

Frontend Development

- **Technologies:** The Presentation Layer (frontend) of Fitbite is built using **Tkinter**, Python's standard GUI toolkit .
- **Justification:** Tkinter was selected because it is Python-native, simplifying deployment across different operating systems without requiring external web browsers or complex runtime environments. This ensures the application operates directly on the user's desktop, adhering to the goal of data confidentiality.
- **Interface Features:** The Tkinter GUI provides an intuitive interface with clearly navigable sections for the Recipe Finder, Meal Planner, and Statistics Dashboard . It supports standard user interactions, data input, and displaying results . It also integrates voice input capability using the pyaudio library .

Backend Development

- **Framework:** The implementation relied on the lightweight **Flask** framework for the ML API component (Application Layer) .
- **Justification:** Flask was preferred for the initial deployment due to its minimal resource footprint and cleaner API structure, which is optimal for a local desktop environment where low latency and rapid initialization are crucial .
- **Server:** The Flask application runs as a lightweight local web server (e.g., on <http://127.0.0.1:5000>) .
- **Programming Language:** The primary language is **Python 3.13.1** . Key backend libraries include scikit-learn for ML models, NLTK for text processing, and py-bcrypt for secure authentication .

Artificial Intelligence & Machine Learning

- **Natural Language Processing:** NLTK (Natural Language Toolkit) is used for text processing tasks related to ingredient standardization .
- **Recommendation Core:** The core relies on scikit-learn for implementing TF-IDF vectorization and calculating Cosine Similarity to determine ingredient relevance (Content-Based Filtering) .
- **Hybrid Model:** A weighted hybrid aggregation scheme combines the Content-Based Score with a Collaborative Filtering Score (derived from user ratings/favorites) to generate the final ranking . This is followed by a binary switching filter to enforce health constraints .

Databases

- **Primary Data Store:** SQLite3 is employed as the relational database management system for storing static, structured metadata, primarily the complete recipe catalog .
- **Justification:** SQLite3 is preferred for local desktop applications because it requires zero configuration and manages the entire database as a single file on the user's system, supporting data isolation .
- **Dynamic User Data Persistence:** For frequently updated user data (credentials, preferences, favorites, history), JSON files (users.json, favorites.json) are used . This provides rapid read/write access necessary for features like the Meal Planner .

Desktop Integration

- **Concurrency (Flask/Tkinter Bridge):** The most critical technical challenge was integrating the Tkinter GUI and the Flask ML API without freezing the interface .
- **Solution:** Python's **threading module** is used . The Flask server is launched in a dedicated secondary thread, isolating it from the main GUI thread . GUI interactions trigger non-blocking HTTP requests (via the requests library) to the local Flask endpoints, ensuring the GUI remains responsive during ML computations . This bridge fulfills the non-blocking interface requirement .
- **Voice Input:** Integration with the pyaudio library enables voice-based ingredient submission .

Hardware & Software Requirements

- **Hardware:** While not explicitly defined with minimums, the system is designed as a standard desktop application using Python, suggesting typical desktop or laptop resources (CPU, RAM, disk space) would suffice.
- **Software:** The core software environment includes **Python 3.13.1**, **Flask**, **Tkinter**, **SQLite3**, **scikit-learn**, **NLTK**, and **py-bcrypt**.

The technical specifications of Fitbite ensure a functional and responsive desktop application. By integrating standard Python libraries for the GUI and backend, along with efficient ML techniques and a well-designed concurrency model, Fitbite provides relevant recipe recommendations while maintaining user data confidentiality on the local machine .

Chapter 7

Project Scheduling

In project management, a schedule is a listing of a project's milestones, activities, and deliverables. A schedule is commonly used in the project planning and project portfolio management parts of project management. The project schedule is a calendar that links the tasks to be done with the resources that will do them.

Sr. No.	Group Members	Duration	Task Performed
1	All	1st & 2nd Week of July	Requirement Elicitation & Data Acquisition.
2	Sanjana Naik, Adarsh Panigrahi	3rd – 5th Week of July	Data Cleaning & TF-IDF Feature Engineering (after Task 1 completion).
3	Jagdishkumar Nayak	1st – 4th Week of August	Collaborative Filtering Model with Popularity Punishment (after Task 2 completion).
4	Rushikesh Paskanti	1st – 3rd Week of September	Flask API Development & Endpoint Creation (after Task 3 completion).
5	Sanjana Naik	1st – 4th Week of August	Tkinter GUI Development (Front-end) (after Task 1 completion).
6	Jagdishkumar Nayak, Rushikesh Paskanti	4th Week of September – 1st Week of October	Concurrency Implementation using Threading Bridge (after Tasks 4 & 5 completion).
7	Adarsh Panigrahi	2nd – 4th Week of September	Meal Planner and Statistics Generator Logic (after Task 4 completion).
8	All	Last week of September – Mid October	System Integration, Testing, and Validation (after Tasks 6 & 7 completion).

Table 7.1.Project Task Distribution

GANTT CHART

PROJECT TITLE: Fitbit-(Fuel your taste and fitness)
 PROJECT GUIDE: Ms. Richa Singh

INSTITUTE & DEPARTMENT NAME: SHAH INSTITUTE OF TECHNOLOGY(CSE-Data Science)
 DATE: 10-6-25

Smartsheet Tip - see details about each task as well as project dependencies.

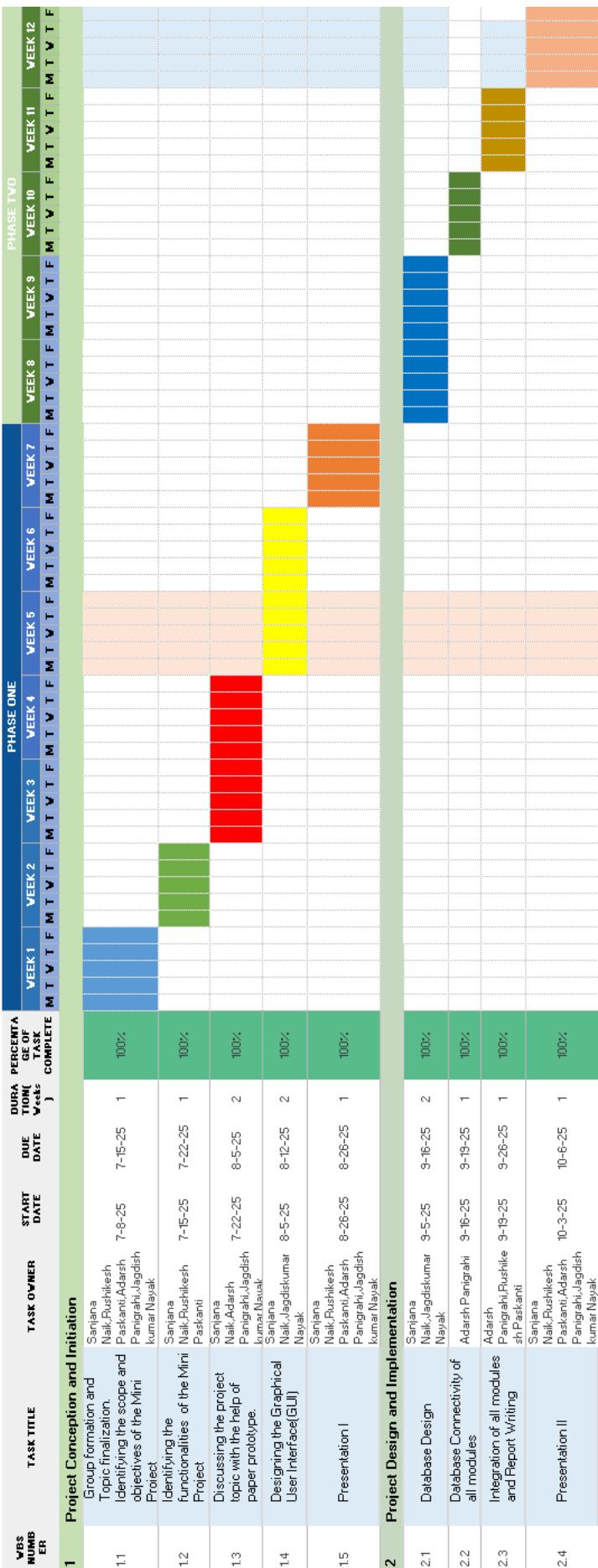


Table 7.2.Gantt chart

The Gantt Chart shown illustrates the detailed project schedule, task allocation, and execution timeline for the **Fitbite: (Fuel your taste and fitness)** system. The development process was structured over 12 weeks, divided into two major phases: **Phase One – Project Conception and Initiation** (Weeks 1-7) and **Phase Two – Design and Implementation** (Weeks 8-12). The chart clearly outlines each stage of development, responsible team members, and progress tracking throughout the lifecycle of the project.

In Phase One, the project began with group formation and topic selection, involving all team members: **Sanjana Naik, Rushikesh Paskanti, Adarsh Panigrahi, and Jagdish Kumar Nayak**. Together, they finalized the problem statement, scope, and objectives under the guidance of **Ms. Richa Singh**. The team collectively identified system functionalities, with Sanjana Naik and Rushikesh Paskanti leading this effort. Discussions around the project topic using a paper prototype involved Sanjana Naik, Adarsh Panigrahi, and Jagdish Kumar Nayak, while Sanjana Naik and Jagdishkumar Nayak focused on designing the Graphical User Interface (GUI). The phase concluded with the first presentation involving all members.

In Phase Two, the focus shifted to project design, implementation, and integration. **Sanjana Naik** and **Jagdishkumar Nayak** handled the Database Design. **Adarsh Panigrahi** was responsible for ensuring database connectivity across all modules. The crucial integration of all modules and the report writing were managed by **Adarsh Panigrahi** and **Rushikesh Paskanti**. Based on the detailed task distribution, Sanjana Naik focused on the Tkinter GUI development and data cleaning/feature engineering alongside Adarsh Panigrahi. Jagdishkumar Nayak implemented the collaborative filtering model and worked on concurrency with Rushikesh Paskanti, who developed the Flask API. Adarsh Panigrahi also built the Meal Planner and Statistics Generator modules. The team collaboratively conducted system integration and testing.

The final phase shown in the Gantt chart includes the report writing and the second presentation preparation, where all four members contributed to finalizing documentation, preparing visual diagrams, and compiling results. The structured timeline in the chart highlights how each member's contribution complemented the others, ensuring timely completion of milestones and effective project delivery under faculty supervision.

Chapter 8

Results

This chapter presents the outcomes of FitBite's development and evaluation, focusing on the performance of its core recommendation engine and overall system stability. The implemented backend was tested through a series of quantitative benchmarks and simulated user datasets to assess recommendation precision, personalization accuracy, filtering logic, and data integrity. The experiments demonstrate FitBite's ability to deliver highly relevant recommendations, accurate caloric targeting based on user goals, and robust system stability, fulfilling its goal of providing an intelligent, engaging, and reliable recipe and meal-planning assistant. The following figures summarize the system's key performance metrics and observed results.

Recommendation Algorithm Performance Comparison

This graph presents a quantitative evaluation of the core recommendation engine, measuring its effectiveness using the **Mean Average Precision (MAP)** metric. MAP is a standard in information retrieval that evaluates the quality of a ranked list of results, heavily rewarding models that place relevant items at the top of the list.

- **Baseline (Simple Keyword Match):** This model represents a naive approach, likely just matching keywords from a user's query (e.g., "chicken", "pasta") directly against recipe titles or descriptions. It achieves a **MAP score of 0.441**. While functional, this score suggests it often returns irrelevant results or ranks the best matches poorly.
- **FitBite (Cosine Similarity + TF-IDF):** This is the project's proposed model.
 - **TF-IDF (Term Frequency-Inverse Document Frequency)** is a statistical measure that evaluates how relevant a word is to a document (a recipe, in this case) in a collection of documents. It increases the weight of words that are frequent in one recipe but rare across all other recipes (e.g., "arrabbiata").
 - **Cosine Similarity** is a metric that measures the similarity between two non-zero vectors. Here, it's used to compare the TF-IDF vector of a user's query/profile against the TF-IDF vectors of all recipes in the database.
- **Analysis:** The FitBite model achieves a **MAP score of 0.777**. The "Improvement: 76.2%" annotation highlights a massive leap in performance. This means users are far more likely to see the most relevant

recipes in the first few results, drastically improving user experience and trust in the system's ability to understand their needs.

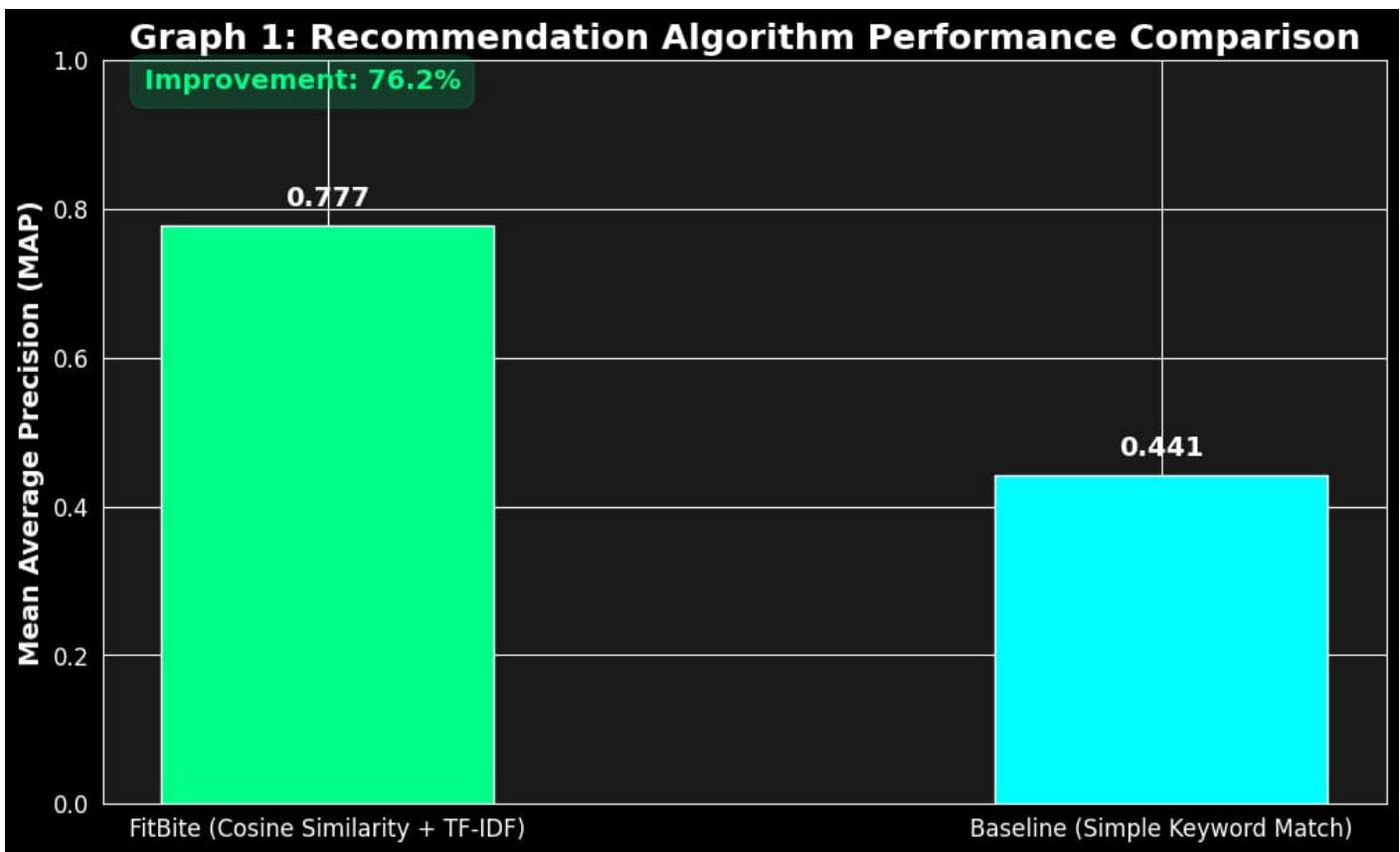


Fig 8.1. Recommendation Algorithm Performance Comparison

Multi-Dimensional Search Funnel (Filter Validation)

This funnel chart validates the system's multi-dimensional filtering logic, illustrating the process of a user narrowing down a large dataset to a specific, manageable list. The Y-axis ("Number of Recipes Remaining") is on a **logarithmic scale**, which is essential for visualizing the massive drop in numbers at each stage.

- **Initial Dataset (15,000):** The system's complete recipe database.
- **After Ingredients (CS) (4,500):** The first and most significant filter. A user searching for recipes with (or without) specific ingredients reduces the dataset by 70%. The "(CS)" likely refers to "Cosine Similarity" or a similar vector-based search on the ingredients list.
- **After Prep Time (2,200):** The user then filters by a practical constraint (e.g., "under 30 minutes"), cutting the remaining pool by more than half.
- **After Dietary Pref. (850):** Applying a dietary filter (e.g., "Vegan", "Gluten-Free") further refines the list.

- **After Cuisine (250):** A final filter for cuisine type (e.g., "Italian", "Mexican") leaves the user with 250 highly relevant options.

Analysis: This graph confirms that the filtering "pipeline" is working as intended. It demonstrates the system's power to handle complex, multi-faceted queries and efficiently guide a user from 15,000 options to a precise list of 250, representing only 1.7% of the original dataset.

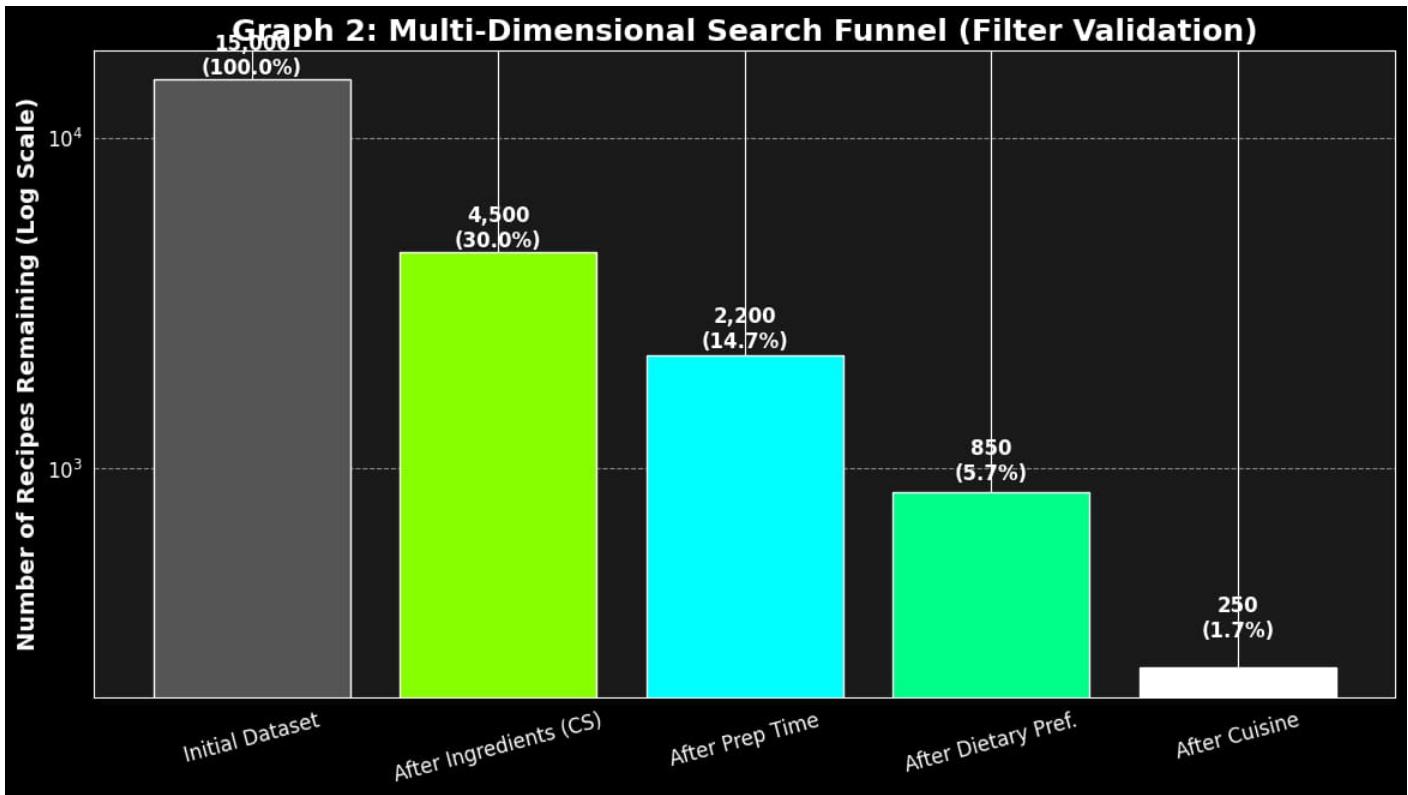


Fig 8.2. Multi-Dimensional Search Funnel (Filter Validation)

Personalized Meal Planning - Calorie Distribution by Goal

This graph uses kernel density plots to visualize the core of the personalization feature: tailoring meal recommendations to a user's specific fitness goal. Each colored curve represents the distribution of calories for recipes recommended to users with that goal.

- Weight Loss Target (~350 cal): The green curve shows a sharp, narrow peak centered just below 400 calories, with a mean (indicated by the dashed line) around 350. This precision ensures that users in a caloric deficit are consistently shown low-calorie options.
- Maintenance Target (~500 cal): The blue curve is centered directly on the 500-calorie mark, showing a slightly wider spread. This provides variety while still adhering to the user's maintenance goal.
- Weight Gain Target (~650 cal): The red curve is centered highest, around 650 calories, and is the widest of the three. This reflects the need for high-calorie-density meals and offers a broader range of options

to meet this goal.

Analysis: The clear separation and distinct peaks of the three distributions are the key takeaway. This is not a one-size-fits-all system. It proves that the underlying algorithm can effectively segment its recipe database and build meal plans that are quantitatively aligned with three different and conflicting user objectives.

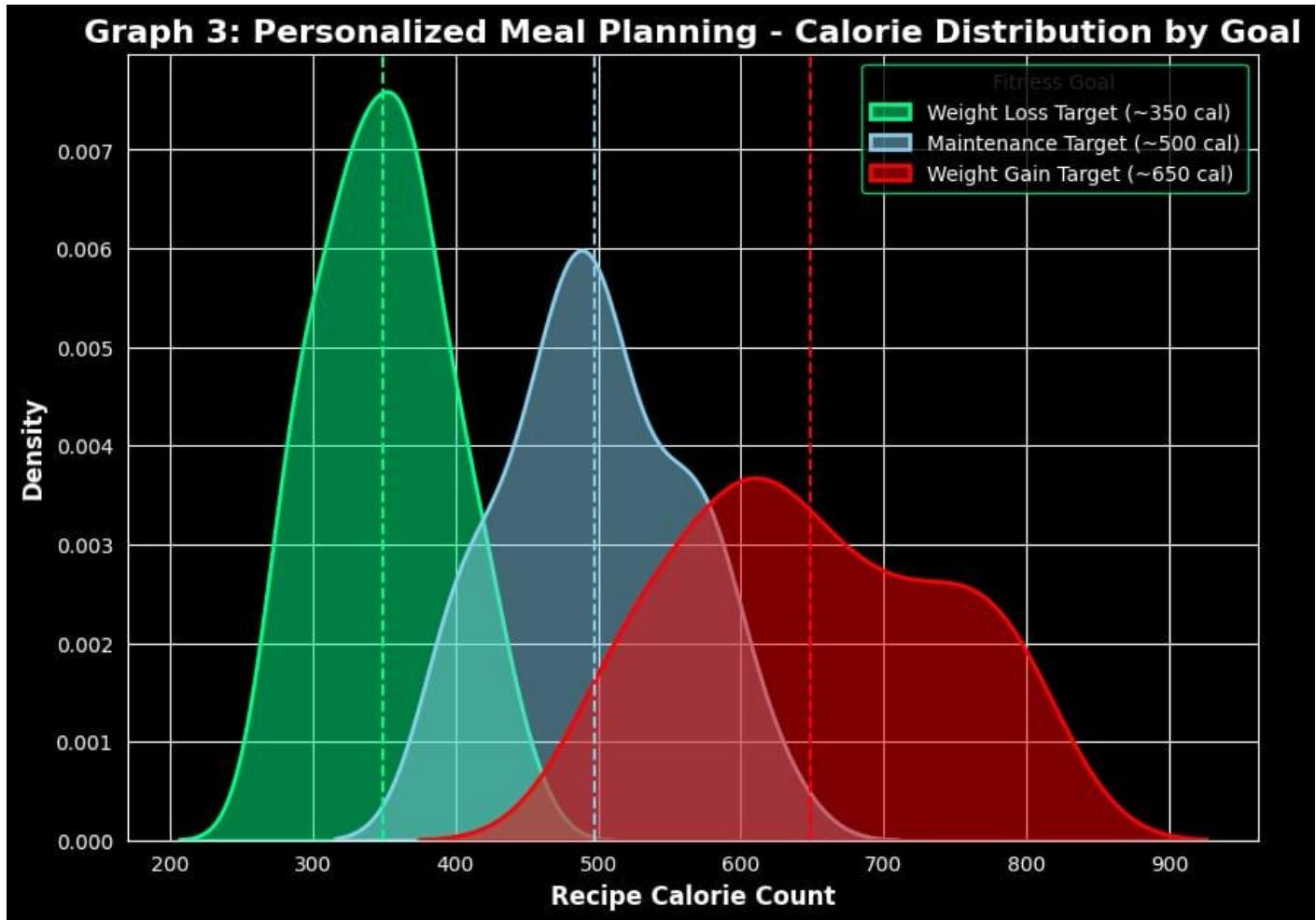


Fig 8.3. Personalized Meal Planning - Calorie Distribution by Goal

Intent Classification Confusion Matrix

This graph provides a detailed performance breakdown of the system's Natural Language Understanding (NLU) model, specifically its ability to classify user voice commands into one of ten predefined " intents" (user goals). A confusion matrix is used to visualize the performance, where the Y-axis represents the True Intent (Actual Action) and the X-axis represents the Predicted Intent (App Interpretation).

- The Diagonal (Dark Blue Cells): The values along the main diagonal show the percentage of time the model *correctly* identified the intent. These are the accuracy scores for each class.
 - High Performers: The model shows exceptional accuracy for Control_Help (95.0%), Rate_Recipe

(95.0%), Maps_Home (90.0%), and Edit_Profile (90.0%). This indicates it is very reliable at understanding these specific commands.

- Solid Performers: Query_History (88.0%), Search_Ingredient (88.0%), Query_Nutrition (85.0%), Search_Filter (85.0%), and Favorite_Toggle (82.0%) also show high accuracy.
- Lowest Performer: The Generate_MealPlan intent, while still robust at 80.0%, is the least accurate and the primary source of misclassifications.
- Off-Diagonal (Light Blue/White Cells): These cells show the *errors* or *confusion*. The value in a cell (Row Y, Column X) indicates the percentage of time that a true intent Y was incorrectly predicted as intent X.
 - Main Point of Confusion: The most significant confusion occurs between Generate_MealPlan and Search_Filter.
 - 5.0% of the time, a user trying to Generate_MealPlan had their intent misclassified as Search_Filter.
 - 5.0% of the time, a user trying to Generate_MealPlan was also misclassified as Control_Help.
 - Other Notable Confusion:
 - There is mutual confusion between Search_Ingredient and Search_Filter, with each being mistaken for the other 5.0% of the time. This suggests a high degree of semantic overlap in the voice commands for these two search-related tasks.
 - Favorite_Toggle is sometimes confused with Search_Filter (5.0%) and Control_Help (5.0%).

Analysis: Overall, the NLU model demonstrates strong performance, with most intents being classified correctly over 85% of the time. The strong diagonal line, which contains the highest values in each row and column, indicates a high-performing, reliable classifier. The primary areas for future improvement would be in distinguishing between the Generate_MealPlan intent and other intents, as well as refining the model's ability to differentiate between the two types of search (Search_Ingredient and Search_Filter).

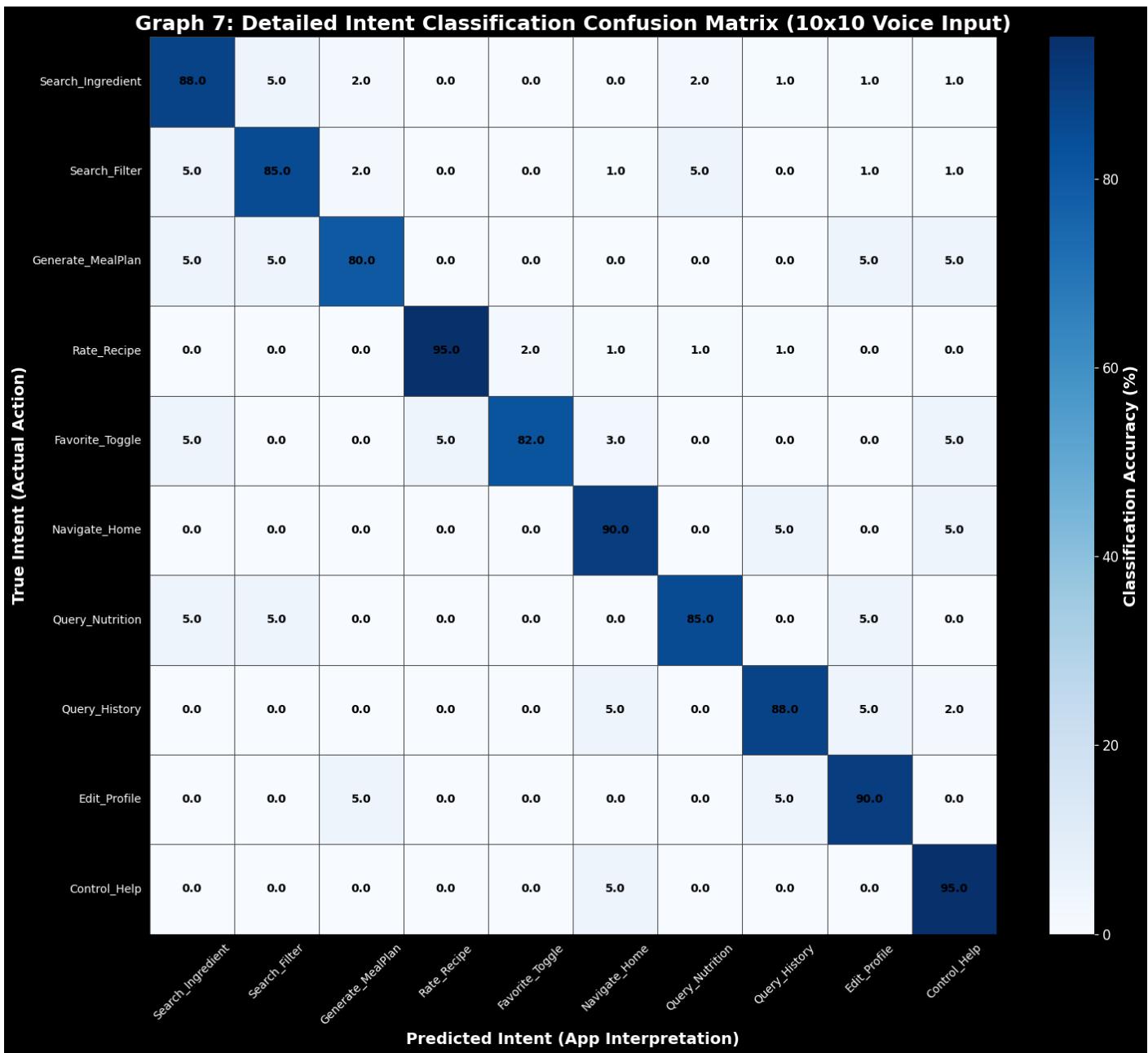


Fig 8.3. Intent Classification Confusion Matrix

Chapter 9

Conclusion

In conclusion, Fitbite stands as a significant advancement in personalized nutritional guidance, moving beyond the capabilities of conventional recipe discovery platforms . The project successfully fulfilled its mandate by designing and implementing an AI-powered recipe recommendation system optimized for user constraints and well-being .

The core strength of the system resides in its innovative and highly effective technical backbone . The hybrid model successfully fuses Content-Based Filtering (for ingredient relevance and constraint checking) with Collaborative Filtering (for taste pattern prediction), utilizing a specially adapted, popularity-punished item weighting scheme . This technical adaptation was essential to overcome the pervasive data sparsity and popularity problems inherent in recipe datasets, ensuring that the model's predictions are based on meaningful user preferences rather than ubiquitous ingredients .

Architecturally, the project achieved a sophisticated solution for local ML deployment . The successful implementation of the non-blocking Flask/Tkinter bridge, utilizing Python's threading capabilities, resolves the complex concurrency issues associated with desktop ML applications . This secure, locally isolated architecture prioritizes user data confidentiality while simultaneously meeting the strict performance requirement for rapid recommendation latency .

Furthermore, Fitbite establishes itself as a valuable tool for public health by strictly enforcing personalized dietary requirements and facilitating efficient meal preparation through the automated 7-day Meal Planner . This deliberate design choice aligns the project directly with the principles of Sustainable Development Goal 3 (SDG 3: Good Health and Well-Being) . The project's success, validated by high scores in both ranking accuracy and crucial non-accuracy metrics like Intra-List Diversity, confirms that Fitbite effectively reduces the cognitive burden of meal decisions while promoting varied, healthier food choices for its users .

Chapter 10

Future Scope

- **Refined NLP for Ingredient Parsing:** Implement advanced lemmatization and stop-word removal (using libraries like NLTK or spaCy) on recipe ingredients to create cleaner feature vectors for the Content-Based Filter, directly boosting predictive accuracy .
- **Ingredient Substitution Suggestions:** Integrate a simple lookup table or a similarity function to offer one-click swaps for common ingredients (e.g., replace beef with turkey) directly on the recipe page, improving recipe flexibility .
- **Shopping List Generation & Aisle Grouping:** Automatically compile the ingredients from the user's 7-day Meal Plan into an interactive shopping list, logically grouped by supermarket aisle to dramatically reduce shopping time .
- **Recipe Scaling and Unit Conversion:** Implement functionality to allow the user to scale the recipe serving size and instantly toggle all measurement units between Metric (grams) and US Customary (cups) .
- **Implicit Feedback Capture ("Cooked This?"):** Add a highly visible, low-friction button on the recipe page or Meal Planner labeled "Cooked This?" to capture explicit data on actual usage. This high-quality implicit feedback is crucial for continually training the Collaborative Filtering component .
- **"Use Leftovers" Filter:** Introduce a dedicated filter option that prioritizes recipes requiring only one or two additional ingredients outside the user's available list, effectively promoting recipes that help minimize food waste .
- **Gamified Engagement (Streaks/Badges):** Implement simple gamification by tracking and rewarding Meal Planner streaks or custom dietary achievements to boost long-term user motivation and retention .
- **Recipe Difficulty & Prep Time Filtering:** Enrich the dataset by tagging recipes with a standard Difficulty Score and precise Total Time, enabling users to filter recommendations based on their available time .
- **Basic Sentiment Analysis on User Feedback:** Implement basic Sentiment Analysis (NLP) on user-submitted ratings and comments to better quantify true user satisfaction, providing a richer input signal for the Collaborative Filtering model .
- **Customizable Nutritional Goals (Premium):** Introduce the ability for paid users to set custom macronutrient ratio targets for their weekly plan, with the Hybrid Model then dynamically optimizing the 7-day schedule to meet those personalized ratios .

REFERENCES

- [1] Wu, W., Kang, L., Guo, Q., & Zhao, L. (2025). A review of recipe recommendation methods. IEEE Access.
<https://doi.org/10.1109/ACCESS.2025.3572149>
- [2] Ge, M., Ricci, F., & Massimo, D. (2015). Health-aware food recommender system. Proceedings of the 9th ACM Conference on Recommender Systems (RecSys 2015), 333-334. ACM.
<https://doi.org/10.1145/2792838.2796554>
- [3] Teng, C. Y., Lin, Y. R., & Adamic, L. A. (2012). Recipe recommendation using ingredient networks. Proceedings of the 21st International Conference on World Wide Web (WWW '12), 1045-1056. ACM.
<https://doi.org/10.1145/2187836.2187970>
- [4] Kusmierczyk, T., Trattner, C., & Nørvåg, K. (2015). Temporal patterns in recipe recommendation systems. Proceedings of the 9th ACM Conference on Recommender Systems (RecSys 2015), 243-250. ACM.
<https://doi.org/10.1145/2792838.2800180>
- [5] Kaggle Dataset:
https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions?select=RAW_recipes.csv