SMART NUTRITION PLANNER

MAJOR PROJECT – II

(20SSP65)

Submitted by

SANJAY K

20MSS040

Under the Guidance of

Dr. V. Usharani MCA, M.Phil., Ph.D.

Associate Professor,

Department of Software Systems

In partial fulfilment of the requirements for the award of the degree of

MASTER OF SCIENCE IN SOFTWARE SYSTEMS

(Five Years Integrated Course) of

Bharathiar University

DEPARTMENT OF SOFTWARE SYSTEMS

PSG COLLEGE OF ARTS & SCIENCE

An Autonomous college - Affiliated to Bharathiar University

Accredited with 'A++' grade by NAAC (4th Cycle)

College with Potential for Excellence

(Status Awarded by the UGC)

Star College Status Awarded by DBT - MST

An ISO 9001:2015 Certified Institution

Coimbatore - 641 014

APRIL 2025

DEPARTMENT OF SOFTWARE SYSTEMS

PSG COLLEGE OF ARTS & SCIENCE

An Autonomous college - Affiliated to Bharathiar University

Accredited with 'A++' grade by NAAC (4th Cycle)

College with Potential for Excellence

(Status Awarded by the UGC)

Star College Status Awarded by DBT - MST

An ISO 9001:2015 Certified Institution

Coimbatore - 641 014

CERTIFICATE

This is to certify that this project work entitled **"Smart Nutrition Planner"** is a bonafide record of work done by **Mr. Sanjay K (20MSS040)** in partial fulfilment of the requirements for the award of Degree of **Master of Science in Software Systems** (Five years Integrated Course) of Bharathiar University.

Faculty Guide

Head of the Department

Submitted for Viva-Voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **Mr. Sanjay K (20MSS040)**, hereby declare that this project work entitled "**Smart Nutrition Planner**" is submitted to **PSG College of Arts & Science, Coimbatore** in partial fulfilment of the requirements for the award of the degree of **Master of Science in Software Systems**, is a record of original work done by me under the supervision and guidance of **Dr. V. Usharani MCA, M.Phil., Ph.D.,** Associate Professor, Department of Software Systems, PSG College of Arts & Science, Coimbatore.

This report has not been submitted by me for the award of any other Degree/ Diploma/ Associate ship/ Fellowship or any other similar degree to any other university.

Place: Coimbatore **Sanjay K**

Date: **20MSS040**

ACKNOWLEDGEMENT

My venture stands imperfect without dedicating my gratitude to a few people who have contributed a lot towards the victorious completion for my project work.

I would like to thank **Thiru L. Gopalakrishnan, Managing Trustee, PSG & Sons Charities**, for providing me prospect and surroundings that made the work possible.

I take this opportunity to express my deep sense of gratitude to **Dr. T. Kannaian, Secretary** of PSG College of Arts & Science, Coimbatore for permitting and doing the needful towards the successful completion of this project.

I express my deep sense of gratitude and sincere thanks to our Principal **Dr. M. Senguttuvan, M.Sc., M.Phil., B.Ed., Ph.D.,** for his valuable advice and concern on students.

I am very thankful **to Dr. M. Jayanthi M.Com.,MBA.,M.Phil.,Ph.D.,** Vice Principal (Aided), **Dr. M Umarani, MBA, M.Phil., Ph.D.**, Vice Principal (SF) for their support towards my project.

I sincerely thank **Dr. K.V. Rukmani., MCA., M.E., Ph.D.,** Head of the Department, Department of Software Systems for her whole hearted help to complete this project successfully by giving valuable suggestions. I convey my heartiest and passionate sense of thankfulness to my project guide **Dr. V. Usharani MCA, M.Phil., Ph.D., Associate Professor,** Department of Software Systems, for her suggestions which had enabled me to complete the project successfully. I also convey my sincere gratitude to all the faculty members of the Department of Software Systems for their immense support, guidance and suggestions.

This note of acknowledgement will be incomplete without paying my heartful devotion to my parents, my friends and other people, for their blessings, encouragement, financial support and the patience, without which it would have been impossible for me to complete the job.

- Sanjay K (20MSS040)

COMPANY CERTIFICATE

6CONNECT

04/12/2024

TO WHOMSOEVER IT MAY CONCERN

To
The Director
PSG College of Arts & Science
Avinashi Road, Civil Aerodrome Road,
Coimbatore - 641014

Respected Sir,

Sub: Confirmation of Enrollment for Internship Program - Reg

We hereby confirm that the below student studying 5th Year M.Sc(Integrated Course) in your PSG College of Arts & Science in Coimbatore

Name : K SANJAY
Reg. No : 20MSS040

In order to motivate and encourage them based on their academic performance, we have offered them the opportunity to undertake their 3 months internship at our company starting from **07/01/2025 to 28/03/2025.**

For **6CONNECT**,

Rajkumar. S
Managing Partner

@6connectcbe
admin@6connect.in
www.6connect.in
+91 - 9894227313

INTERNSHIP COMPLETION CERTIFICATE

6CONNECT

Connect You To Job

02/04/2025

INTERNSHIP COMPLETION CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Mr. SANJAY K (20MSS040)
M.Sc Software System (Integrated Course)
PSG College of Arts & Science , Coimbatore

has successfully completed an internship program as an Intern with our IT Development team at 6Connect from January 2025 to March 2025.

His performance during the program was very good. We wish him all the very best for a successful and rewarding career.

We wish him continued success in all future endeavors.

For 6CONNECT,

Rajkumar. S
Managing Partner

@6connectcbe
admin@6connect.in
www.6connect.in
+91 - 9894227313

SYNOPSIS

The "Smart Nutrition Planner" project allows users to view the results of the machine learning algorithm from anywhere via the Internet. A subfield of computer science and artificial intelligence (AI) called machine learning focuses on using data and algorithms to mimic human learning processes while progressively increasing their accuracy. An essential part of the expanding field of artificial intelligence is machine learning. Algorithms are trained using statistical techniques to classify or predict data and to find important insights for data mining projects. These insights then inform business and application decision-making, ideally influencing important growth indicators.

Data scientists will become more and more in demand as big data keeps growing. They will have to assist in determining which business questions are most pertinent and what information is needed to address them. Typically, frameworks like TensorFlow and PyTorch that speed up solution development are used to create machine learning algorithms. Additionally, the machine learning algorithm can be posted to a website, giving users easy access to all of the algorithms that are available online. A non-parametric, supervised learning classifier, the k-Nearest Neighbors (KNN) algorithm uses proximity to classify or predict how to group a single data point. It is among the most widely used and straightforward regression and classification classifiers in machine learning today.

For tasks like text analysis or recommendation systems, the k-Nearest Neighbors (kNN) algorithm, when paired with cosine similarity, determines the "k" most similar vectors to a target vector based on the cosine of the angle between them. Cosine similarity quantifies the degree of similarity between two vectors, independent of their magnitude, by calculating the cosine of their angle. Perfect dissimilarity (vectors pointing in opposite directions), no similarity (vectors perpendicular), and perfect similarity (vectors pointing in the same direction) are the three extremes of the cosine similarity value, which goes from -1 to 1. Data points become vectors when kNN and cosine similarity are combined. The query vector and every other vector in the dataset are compared using cosine similarity dataset. The highest cosine similarity scores are used to determine the k-nearest neighbors.

.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 COMPANY PROFILE

Based in Coimbatore, Tamil Nadu, India, 6Connect is a middleware company that uses the BOT model to assist clients in achieving their goals. They offer the best solution to achieve your business objectives because they have more than ten years of experience in the digital and development industry. In order to provide the ideal customized solution to meet your needs, they pay close attention to the needs of each customer and comprehend the potential needs of the company. They We mentor everyone until they succeed in addition to educating them. They use technology to automate business processes, which saves time and generates revenue.

They are a group of exceptionally skilled people who handle everything from basic planning to international marketing. They use technology to automate business processes, which saves time and generates revenue. They assist businesses in reaching the intended outcome through a methodical process. They have superior, infallible methods to follow a flawless flow with strong foundational knowledge. Whether it's a big eCommerce store or a basic business website. Using their BOT model, their team brings your hypothetical website to life. We connect everyone and assign the appropriate individuals to the company.

COMPANY INFO

ADDRESS: 329 Db Road near Ganesh store, Opp old Kennedy theatre, RS Puram, Coimbatore - 641002.

EMAIL: admin@6connect.in

CONTACT: +91 9894227313

1.2 PROJECT OVERVIEW

This project's main goal is to comprehend, collect, and analyse user metrics in order to suggest a healthy diet for leading a fit and healthy life.

A supervised machine learning technique called the K-nearest neighbors algorithm uses proximity to classify or predict how to group a single data point. KNN is a lazy, non-parametric learning algorithm that only does calculations during classification and stores the complete training dataset. This implies that KNN generates predictions by directly comparing fresh data points to the stored training data, rather than creating a model during the training phase. The algorithm's performance depends on the distance metric used to measure similarity and the choice of K (the number of nearest neighbors taken into consideration). It is a flexible algorithm that can be used for both classification and regression tasks.

The idea of distance or similarity in feature space forms the basis of KNN. In a multi-dimensional space, each dimension represents a feature, and each data point is represented as a vector. In this feature space, two data points' similarity is inversely proportional to their distance from one another; the closer two points are to one another, the more similar they are regarded to be.

Through the use of labelled datasets, supervised learning trains algorithms to identify patterns or predict outcomes, allowing them to accurately classify or predict new, unseen data. Using this labelled data, the algorithm is trained to understand the connection between the output labels and the input features. The objective is to develop a model that, using the relationships it has learned, can reliably forecast the results for fresh, unseen data.

Python3, Streamlit, and the Fast API are used to create and host the machine learning model online. The user must first open the webpage made with Streamlit in their preferred browser. There, they will find the project's overview and description. After reading the details, they will see a sidebar that allows them to run the Recommendation System by entering the necessary information and viewing the output in their browser along with the graphs and diagrams to better understand the recommendations they have been given.

1.3 MODULE DESCRIPTION

1.3.1 COLLECTION OF DATASETS

Gathering datasets from various sources, including Kaggle, Tableau, Google Research, Google Cloud, papers with code, etc., is the first module's task. The csv format is used to collect the dataset because it is easy to edit, human readable, compact, and faster to handle, making it simple to analyze and proceed to the next stage without any issues.

1.3.2 DATA EXPLORATION

Since different sources contain different types of data—that is, different types of fields are present in different datasets—the datasets that have been gathered from different sources are examined separately. Either manually or with the use of Python commands, those values are examined and listed. After the investigations are completed and the datasets are combined, we can move on to the following phase.

1.3.3 DATA PREPROCESSING

In order to train our machine learning model to generate results with a higher accuracy ratio, the datasets are combined into a single, sizable dataset in this step. Once the datasets have been combined, we have to look for any gaps, like null values, NaN (not a number values), incorrect values, uncoordinated values, etc. Once these potential errors have been checked, we must determine how many fields are available and which fields are necessary in our dataset in order to construct our model. To construct our machine learning model, we must identify the parallels or relationships between values or fields.

1.3.4 SPLITTING THE DATA INTO TRAINING AND TESTING DATA

This step involves splitting the datasets we have gathered and produced into two sections: training data and testing data. To do this, we use train_test_split and standard scalar from scikit-learn, and we allocate 80% of the dataset for training and 20% for testing.

1.3.5 MACHINE LEARNING MODEL BUILDING

We must select the machine learning algorithm that best fits our problem in this step. In this instance, we have decided to create our machine learning model using the k-Nearest Neighbors (kNN) algorithm. To construct our model, we are utilizing Sklearn's Nearest Neighbors and cosine similarity. Cosine similarity, which is helpful in situations like text analysis or recommendation systems, determines the "k" most similar data points to a query point by calculating the cosine of the angle between their vectors.

1.3.6 SHOWCASE OUR FINDINGS

This step will present our results and the machine learning model's (kNN) operation, as well as the graphs and diagrams produced during the analysis of user-provided data. In order to display the diet recommendations on the web using the Streamlit framework and the required graphs to show the composition of the diet recommendation, the Recommendation System leverages pipeline and function transfer from Sklearn.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Users cannot customize the diet recommendations in the current system, nor can they receive recommendations based on the ingredients they prefer. In order to prevent the recommendation system from suggesting a diet that contains ingredients that users should avoid, the current system does not permit users to add their medical conditions. Users are unable to know what they are eating because the current system only offers recommendations, which means it only provides diet names and not recipes or nutrient constituents.

2.2 PROPOSED SYSTEM

Users can customize the diet recommendations in the suggested system by entering the ingredients they have on hand. This is a great advantage because it adds authenticity to the recommendations rather than merely relying on a random selection of suggestions. In order to receive accurate recommendations and avoid major accidents, the proposed system enables users to enter their medical conditions, such as severe allergies or other issues. The suggested system lets users track their dietary intake and health conditions, provides information to help them consult with their doctors for a healthy lifestyle, and displays the nutritional compositions of the recommended diet so that users are aware of what they are consuming. Users can enter their own nutritional component composition into the suggested system to receive recommendations. In contrast to other recommendation systems that rely on user data from the past to generate recommendations, it also avoids the cold start issue by using a machine learning algorithm to analyze the input data and produce recommendations that are extremely relevant to the user.

CHAPTER 3

SYSTEM CONFIGURATION

3.1 HARDWARE SPECIFICATION

COMPONENTS		REQUIREMENTS
PROCESSOR	:	INTEL CORE i5
RAM	:	4 GB RAM
KEY BOARD	:	STANDARD KEYBOARD

3.2 SOFTWARE SPECIFICATION

COMPONENTS		REQUIREMENTS
DEVELOPMENT ENVIRONMENT	:	WINDOWS 7 OR ABOVE
EDITOR	:	VISUAL STUDIO CODE
FRONT END	:	STREAMLIT FRAMEWORK
BACK END	:	PYTHON, FAST API
RUNNING PLATFORM	:	DOCKER

CHAPTER 4

SOFTWARE DESCRIPTION

4.1 FRONT END

4.1.1 STREAMLIT

A free and open-source framework called Streamlit makes it easy to create and distribute stunning web applications for data science and machine learning. This Python-based library was created especially for engineers working in machine learning. Since they are not web developers, data scientists and machine learning engineers have no interest in spending weeks learning how to create web applications using these frameworks. As long as it can show data and gather the parameters required for modeling, they prefer a tool that is simpler to understand and operate.

With just a few lines of code and Streamlit, you can create an application that looks amazing. The best thing about Streamlit is that you can start using it and make your first web application without even knowing the fundamentals of web development. Therefore, Streamlit is a good choice if you're interested in data science and want to quickly, simply, and with just a few lines of code deploy your models.

Providing an application with an efficient and user-friendly interface is a crucial component of its success. Building an efficient user interface fast without requiring complex steps is a challenge for many of the data-heavy apps of today. With the help of the promising open-source Python library Streamlit, programmers can quickly create visually appealing user interfaces.

- The simplest method for adding code to a web application, particularly for those without any front-end experience, is Streamlit:
- No prior knowledge or experience with front-end (html, js, or css) is necessary.
- It only takes a few hours or even minutes to create a stunning machine learning or data science application; days or months are not necessary to develop a web application.
- The vast majority of Python libraries, including pandas, matplotlib, seaborn, plotly, Keras, PyTorch, and SymPy(latex), are compatible with it.
- Less code is needed to create amazing web apps.

FEATURES OF STREANLIT

Streamlit framework's salient features include:

- Simple to use: Streamlit can be used to create web apps with a few lines of code and doesn't require any prior front-end knowledge.
- Python-based: Data scientists and machine learning engineers can easily use Streamlit because it is built on top of Python.
- Customizable: Streamlit offers a variety of components that can be altered, such as text inputs, sliders, and buttons.
- Interactive: Streamlit applications have the capability to be interactive, enabling users to enter data and view the outcomes instantly.
- Deployable: Streamlit apps are compatible with a number of platforms, such as GitHub Pages, Heroku, and Snowflake.
- Community-driven: Streamlit boasts a sizable and vibrant user and developer community that actively contributes to and supports the framework.

The following are some advantages of using Streamlit:

- Quick development: Data scientists and machine learning engineers can easily and quickly create web apps with Streamlit.
- Simple sharing: Sharing Streamlit apps with others facilitates collaboration and results communication.
- Customizable: Streamlit offers a number of components that can be easily altered to meet particular requirements.
- All things considered, Streamlit is a strong and adaptable framework that facilitates the creation and sharing of data apps.

4.2 BACK END

4.2.1 PYTHON

Python is an object-oriented, high-level, general-purpose programming language. Python is a dynamic, high-level, interpreted, general-purpose programming language. It facilitates the development of applications using the object-oriented programming approach. It offers many high-level data structures and is straightforward and simple to learn. Python is a popular scripting language for application development because it is simple to learn but also strong and adaptable. Python is the perfect language for scripting and quick application development because of its interpreted nature, dynamic typing, and syntax. Multiple programming patterns, such as imperative, functional, procedural, and object-oriented programming styles, are supported by Python. Python isn't designed to be used in specific fields, like web programming. Because it can be used with web, enterprise, 3D CAD, and other applications, it is a multipurpose programming language. Because it is dynamically typed, we can assign an integer value to an integer variable by writing a=10 instead of using data types to declare it. Python's edit-test-debug cycle is incredibly quick, and the lack of a compilation step speeds up development and debugging.

FEATURES OF PYTHON

Python boasts a thriving community, open-source projects, and a wealth of web-based resources. Developers can easily learn the language, collaborate on projects, and contribute to the Python ecosystem. Python's simple linguistic structure makes it simpler to comprehend and write code in. Because of this, it's an excellent programming language for beginners. It also helps experienced programmers write code that is clearer and free of errors. Python is a programming language that is free and open-source. As a result, it is used in many different fields and disciplines. Readability and maintainability of code are crucial in Python. This makes it simple for another developer to understand and modify the code, even if it was created by someone else. To facilitate its functionality, Python offers a plethora of third-party libraries. These libraries cover a wide range of topics, including data analysis, scientific computing, web development, and more.

4.2.2 FAST API

- Based on common Python type hints, FastAPI is a contemporary, fast (high-performance) web framework for creating Python APIs.
  The salient characteristics are:
- Quick: Thanks to Starlette and Pydantic, it performs incredibly well, comparable to NodeJS and Go. Among the quickest Python frameworks on the market.
- Fast to code: Approximately 200% to 300% faster feature development
- Fewer bugs: Cut down on errors caused by developers by roughly 40%.
- Instinctive: Excellent editor assistance. Everywhere there is completion. Debugging takes less time.
- Simple: Made to be simple to use and understand. Read documents less often.
- In a nutshell: Reduce code duplication. several features from every declaration of a parameter. Bugs are reduced.
- Robust: Obtain code that is ready for production. with interactive documentation that is automatically generated.
- Standards-based: JSON Schema and OpenAPI, formerly known as Swagger, are the open standards for APIs that it is based on and completely compatible with.

FEATURES OF FAST API

You get the following from FastAPI:
Using open standards

- OpenAPI for creating APIs, which includes security, request bodies, parameters, path operations, and more.
- JSON Schema-based automated data model documentation (since OpenAPI is built on JSON Schema).
- Developed following a thorough analysis with these standards in mind. • This also makes it possible to use automatic client code generation in a variety of languages, rather than an afterthought layer on top.

Automatic docs

Web user interfaces for exploration and interactive API documentation. Two options are included by default because the framework is based on OpenAPI.

- Swinge UI allows you to test and call your API straight from the browser and offers interactive exploration.
- ReDoc provides alternative API documentation.

Just Modern Python

Thanks to Pydantic, it is all based on standard Python type declarations. No need to learn any new syntax. Just regular, contemporary Python. Check out this quick tutorial on Python types if you need a two-minute refresher on the subject, even if you don't use FastAPI.

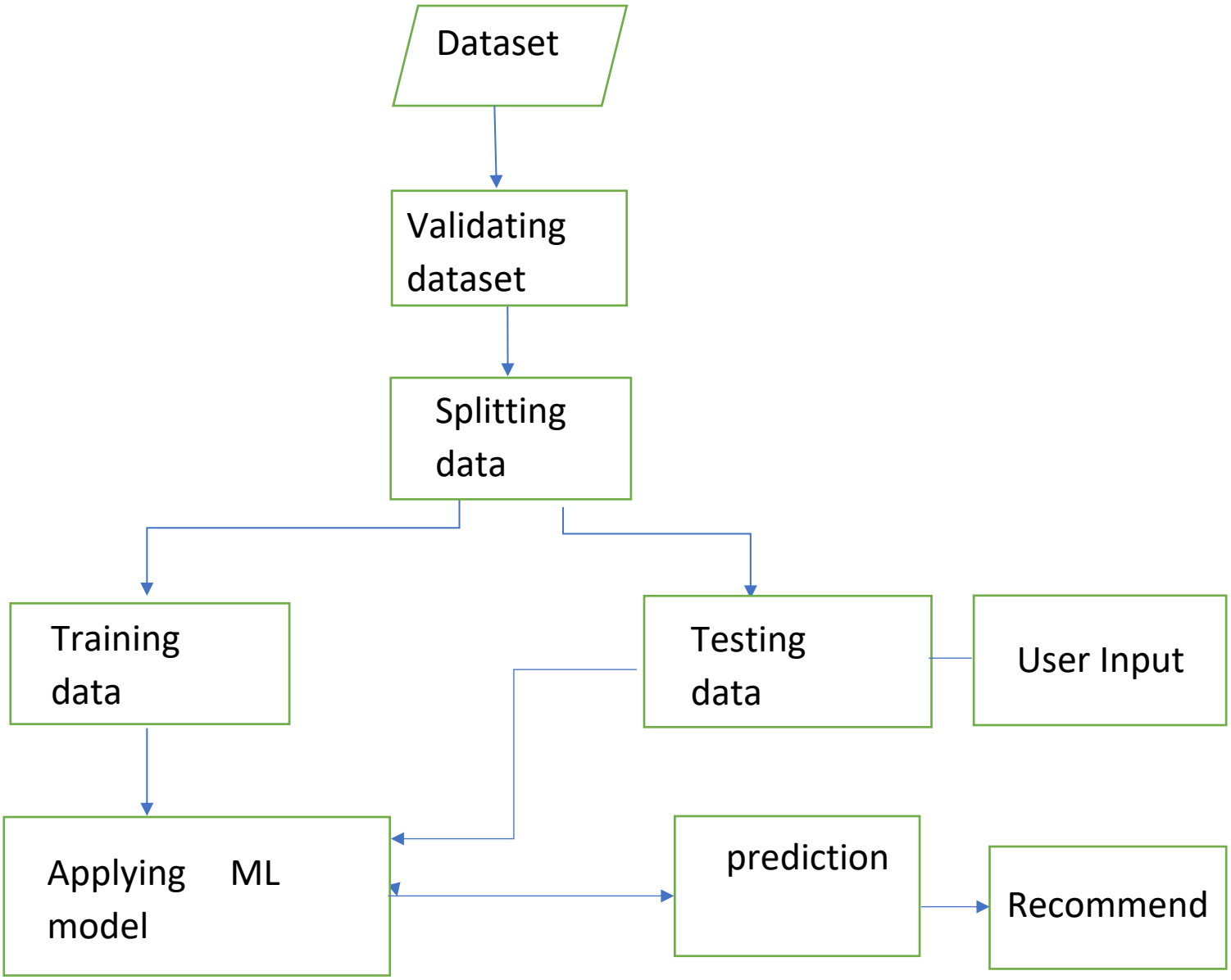
Editor support

To guarantee the greatest development experience, the entire framework was made to be simple and easy to use, and every choice was tested on several editors even before development began. It is evident from the Python developer surveys that "autocompletion" is one of the most popular features. That is the foundation of the entire FastAPI framework. Autocompletion is universal. Seldom will you have to return to the documents.

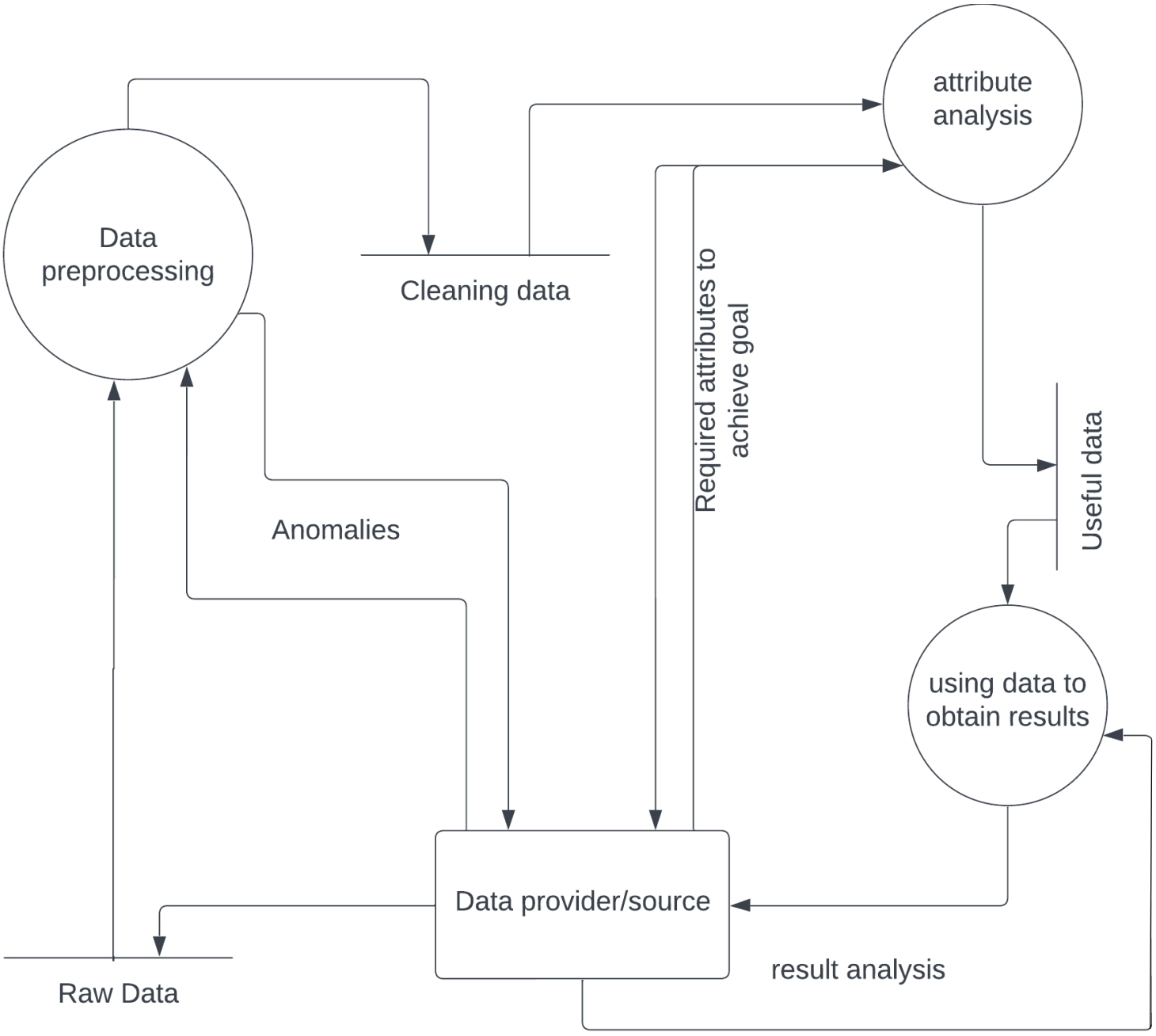
CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM FLOW DIAGRAM



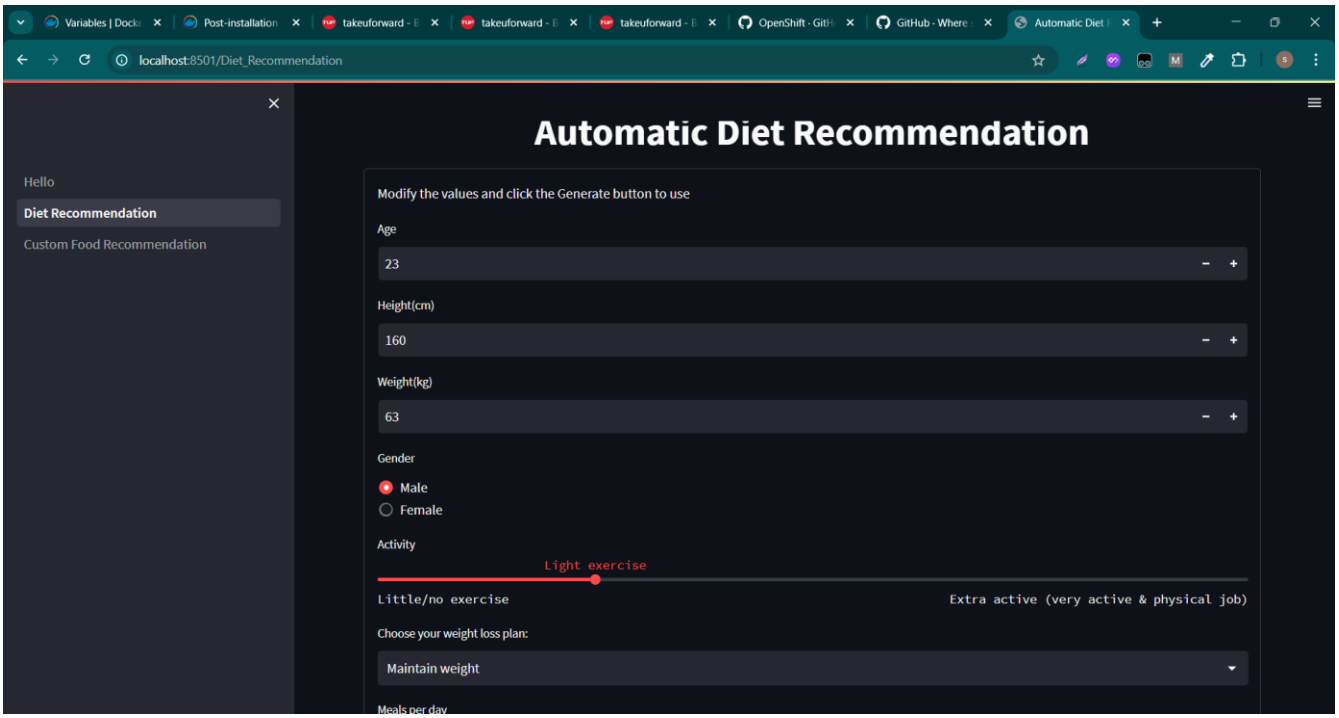
5.2 PROCESS DIAGRAM



5.4 INPUT DESIGN

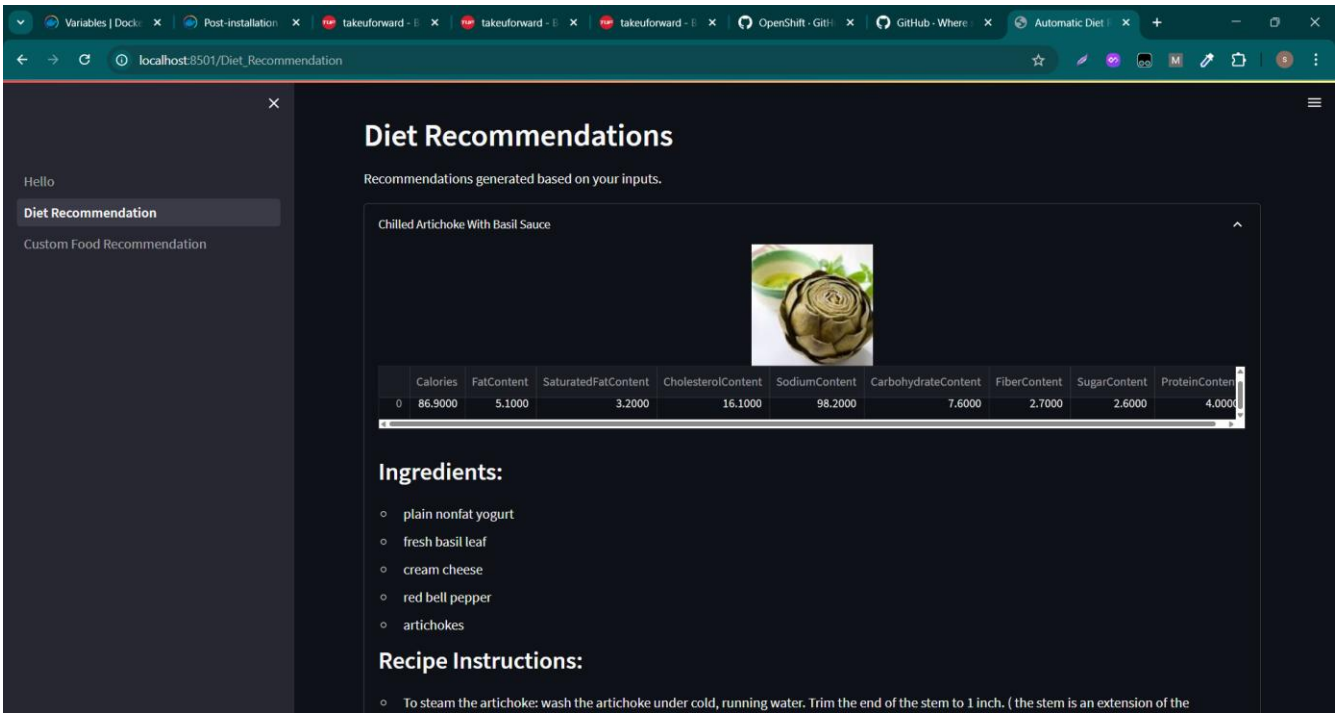
The interface between the user and the information system is input design. It is the aspect of the overall system design that needs the utmost care. The most costly component of the system is frequently the input data collection process. The processing and output will amplify errors if the data entering the system is inaccurate. The primary goals of input design are as follows.

- Getting the most accurate results and making recommendations that are extremely pertinent; making sure the user understands and accepts the input.
- In this project, the user must manually enter the necessary dataset into the program; the remaining steps are carried out in accordance with the backends' coded functions.
- Care must be taken when inserting the data because it is crucial. The input must be acceptable and comprehensible to the user in order to produce the right output and attain high accuracy.



5.5 OUTPUT DESIGN

An essential part of a system is its output design. Getting the right output requires focus because a system is only considered correct if it produces the right output based on the input. The suggestions are essential to this project. Based on the user's input on the webpage, it presents extremely pertinent dietary recommendations.



CHAPTER 6

SYSTEM IMPLEMENTATION AND TESTING

6.1 SYSTEM IMPLEMENTATION

The project's implementation phase is when the theoretical design is transformed into a functional system. The implementation phase is a stand-alone system project. It entails thorough planning, analysis of the current system and implementation limitations, development of strategies to accomplish the changeover, staff training on the process, and assessment of the changeover approach. Selecting the techniques and timeline to be used is the first step in the implementation process. After planning is finished, the main task is to make sure that the system's programs are operating correctly. After staff members have received training, the entire system—computer and user—can be operated efficiently.

As a result, activities have well-defined plans. The process of turning a new or updated system design into an operational one is known as implementation.

6.2 SOFTWARE TESTING

Errors in the program are discovered through the testing process. The primary quality metric used in software development is this one. The program is run under a series of conditions called test cases during testing, and the results are assessed to see if the program is operating as intended.

Software testing is the process of running software to verify its correctness and functionality. It involves running a program with the goal of identifying errors. A test case with a high likelihood of discovering an error that hasn't been found yet is considered good. A test that finds an error that hasn't been found yet is considered successful. There are two main reasons why software is tested.

- Defect detection
- Reliability estimation

6.2.1 TESTING OBJECTIVES

- A good test case is one that has a high likelihood of discovering an as-yet-undiscovered error.
- Testing is the process of running a program with the goal of identifying an error.
- A test that finds an error that hasn't been found yet is considered successful.

6.2.2 TESTING PRINCIPLES

- Every test ought to be able to be linked to the needs of the client.
- Before testing starts, extensive planning should be done for the tests.
- Testing ought to start "in the small" and work its way up to "in the large."

6.2.3 TYPES OF TESTING

The various levels of testing techniques that are used at various stages of software development to ensure that the system is error-free are:

Unit Testing:

As each module is finished and made executable, unit testing is carried out on it. It is limited to what the designer wants. The two approaches listed below can be used to test each module:

Black Box Testing:

Some test cases are created using this approach as input conditions that completely carry out all of the program's functional requirements. Only the output is examined for accuracy in this testing. The data's logical flow is not examined.

Errors in the following categories have been discovered using this testing:

a) Missing or inaccurate functions
b) Inaccurate interfaces
c) Data structure or external database access errors
d) Performance issues
e) startup and shutdown issues

White Box testing:

Logical decisions are tested on all cases, and test cases are created based on the logic of each module by creating flow graphs of that module. In the following situations, it has been used to create test cases:

a) Ensure that every independent path has been followed
b) Apply both the true and false sides of all logical decisions.
c) Complete every loop within its operational parameters and at its boundaries.
d) Run internal data structures to verify their accuracy.

Integrating Testing

Software and subsystems are tested for integration to make sure they function as a whole. To ensure that the modules function correctly when combined, it tests each module's interface.

System Testing

involves testing the entire system internally before delivering it to the user. Its goal is to ensure that the user is satisfied by the system's compliance with all client specifications.

Acceptance Testing

Pre-delivery testing involves testing the entire system on real-world data at the client's location in order to identify any errors.

Validation Testing

Since the system has undergone successful testing and implementation, all of the requirements specified in the software requirements specification have been fully met. When incorrect input is entered, the appropriate error messages are shown.

CHAPTER 7

CONCLUSION

It is therefore determined that the website and the integrated machine learning model operate effectively, offering accurate, personalized, and context-aware dietary recommendations. The system architecture is robust and modular, allowing for easy maintenance and future scalability. The website includes all essential features required for a seamless user experience, such as user authentication, profile management, dynamic input forms, and real-time feedback. The front end is built with responsive design principles, ensuring compatibility across devices and browsers. On the backend, the FastAPI framework provides high-performance API endpoints, ensuring low latency in model predictions and data retrieval.

The machine learning model has been trained on a comprehensive and diverse nutritional dataset, using preprocessing techniques such as normalization, feature selection, and class balancing to enhance performance. Evaluation metrics like precision, recall, F1-score, and accuracy have been used to validate the model's effectiveness. Additionally, the application has been rigorously tested through unit, integration, and usability testing to ensure it is fully debugged and production ready.

Deployment has been handled using Docker containers, ensuring consistency across development and production environments. A CI/CD pipeline is integrated to streamline updates and minimize downtime. With its scalable backend and user-friendly front end, the platform is well-suited for real-world application in the field of digital health and nutrition.

.

CHAPTER 8

FUTURE ENHANCEMENTS

Every application comes with its own set of advantages and limitations, and this project is no exception. While the current implementation satisfies nearly all functional and non-functional requirements, there remains ample scope for future enhancements. The codebase is modular and well-structured, adhering to best practices such as separation of concerns and reusable components, which makes it easier to scale and maintain. Future improvements can be efficiently implemented by introducing new modules, refining existing ones, or integrating advanced algorithms.

The current version supports a single machine learning model, but future iterations may include ensemble learning or model comparison capabilities, allowing for dynamic model selection based on user input or context. Improvements to the user interface (UI) and user experience (UX) can include interactive visualizations, voice input capabilities, and real-time health tracking integration through APIs or wearable device support.

Security and performance optimization are other areas that can be enhanced. Incorporating authentication protocols like OAuth 2.0, rate-limiting, and database indexing will contribute to a more secure and efficient platform. Integration with external services such as nutrition databases or AI-powered recommendation engines could further enrich the application's utility. Overall, the application is built with extensibility in mind and is well-prepared for continuous evolution.

9. BIBLIOGRAPHY

The following books and websites were referred during the analysis and execution phase of the project.

REFERENCE BOOKS:

1. François Voron, *Building Data Science Applications with FastAPI*, Packt Publishing, Kindle Edition
2. Jake Vander Plas, *Python Data Science Handbook: Essential Tools for Working With Data*, O'Reilly Media, Inc.
3. Erico Andrei, *FastAPI Cookbook*, Independently Published, Latest Edition
4. Tyler Richards, *Streamlit for Data Science - Second Edition*, Packt Publishing, Second Edition
5. Tyler Richards, *Getting Started with Streamlit for Data Science*, Packt Publishing, First Edition
6. Nigel Poulton, *Docker Deep Dive*, Independently Published, Latest Edition
7. Jeff Nickoloff and Stephen Kuenzli, *Docker in Action*, Manning Publications, Second Edition

REFERENCE WEBSITES:

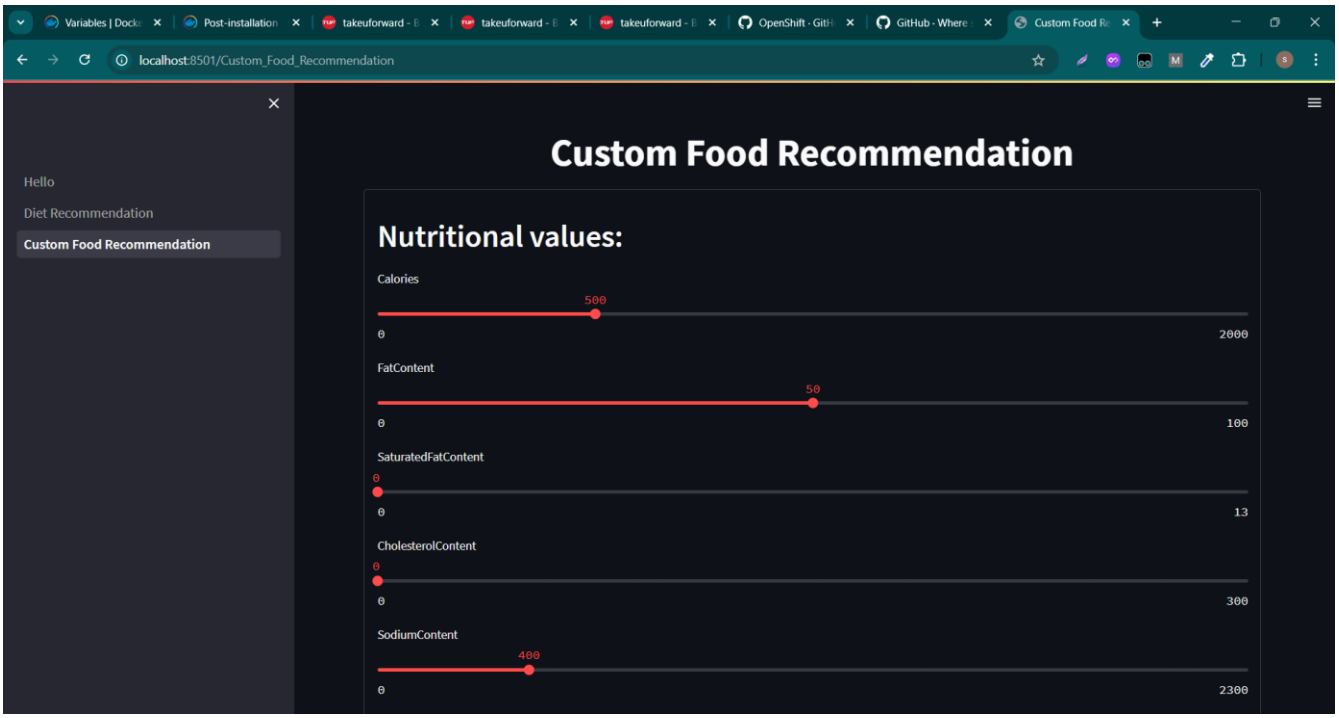
1. https://stackoverflow.com/
2. https://www.python.org
3. https://scikit-learn.org/stable/modules/neighbors.html
4. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/
5. https://fastapi.tiangolo.com/
6. https://www.w3schools.com/python/python_ml_knn.asp

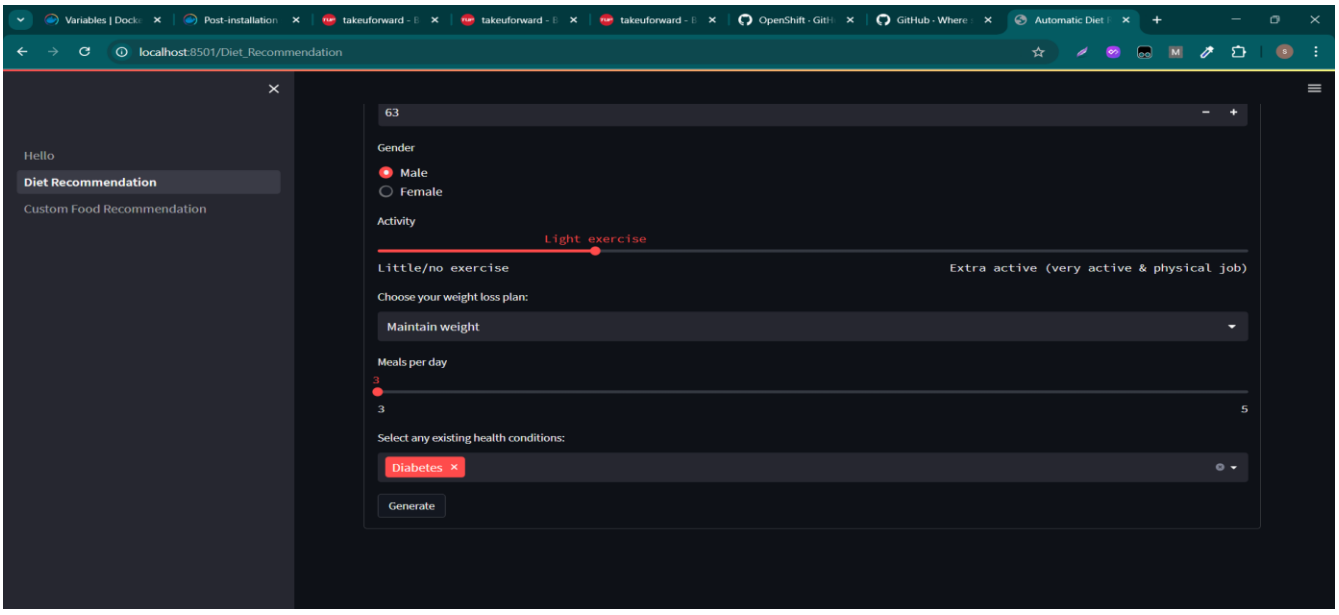
CHAPTER 10

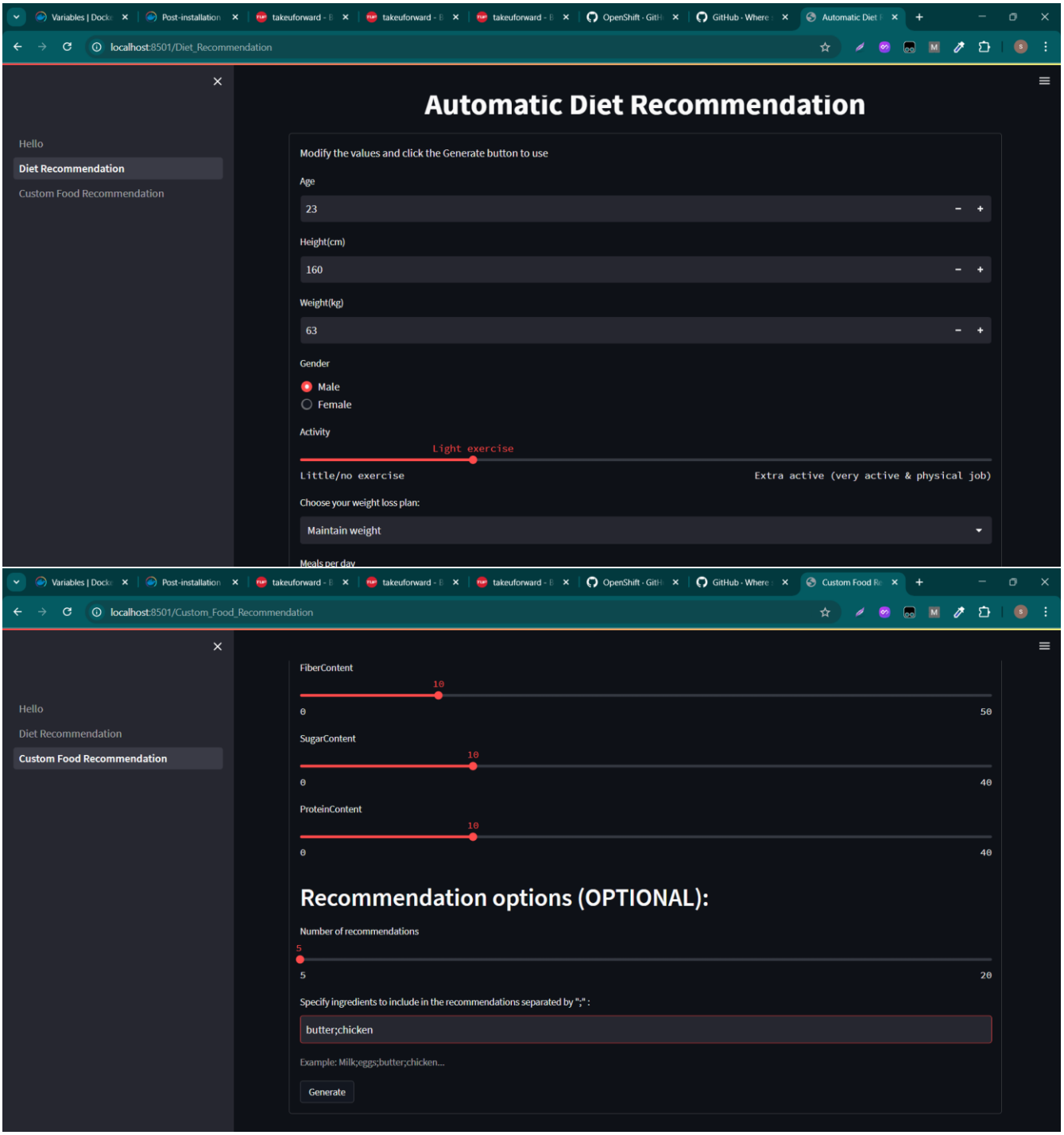
10. APPENDIX

10.1 SCREENSHOTS

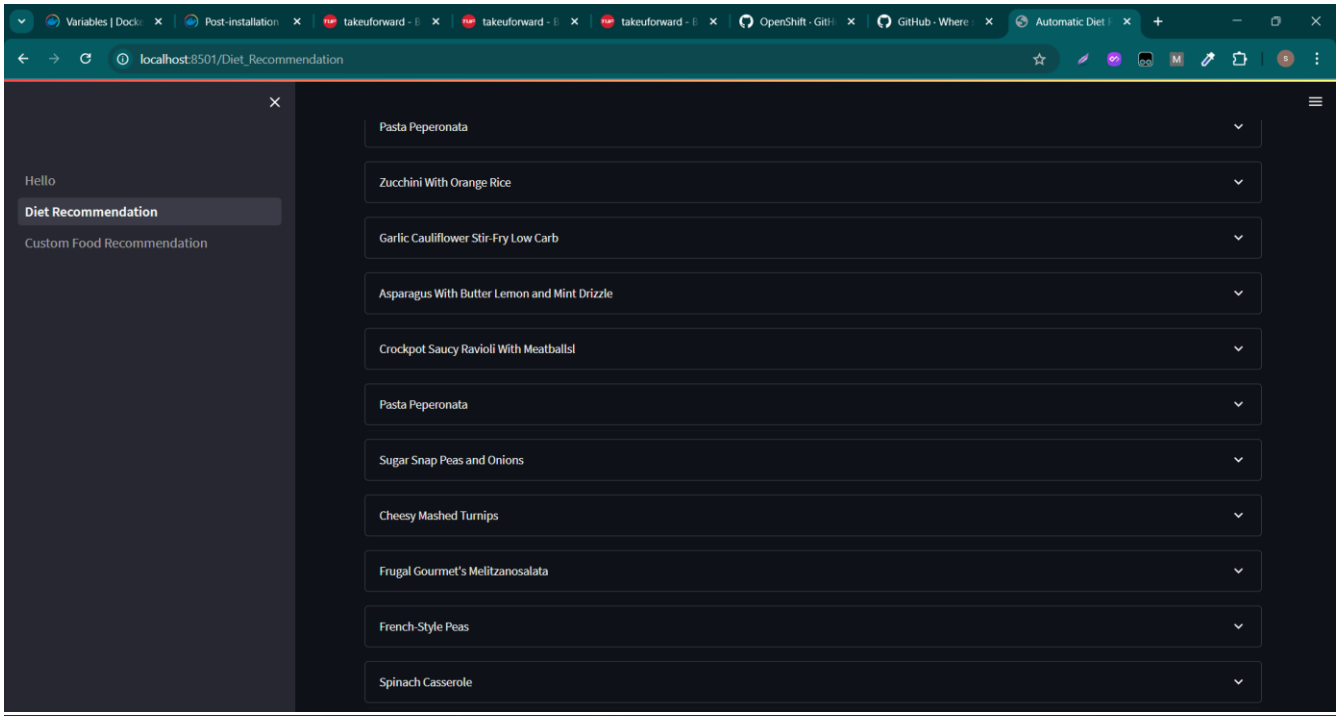
HOME PAGE

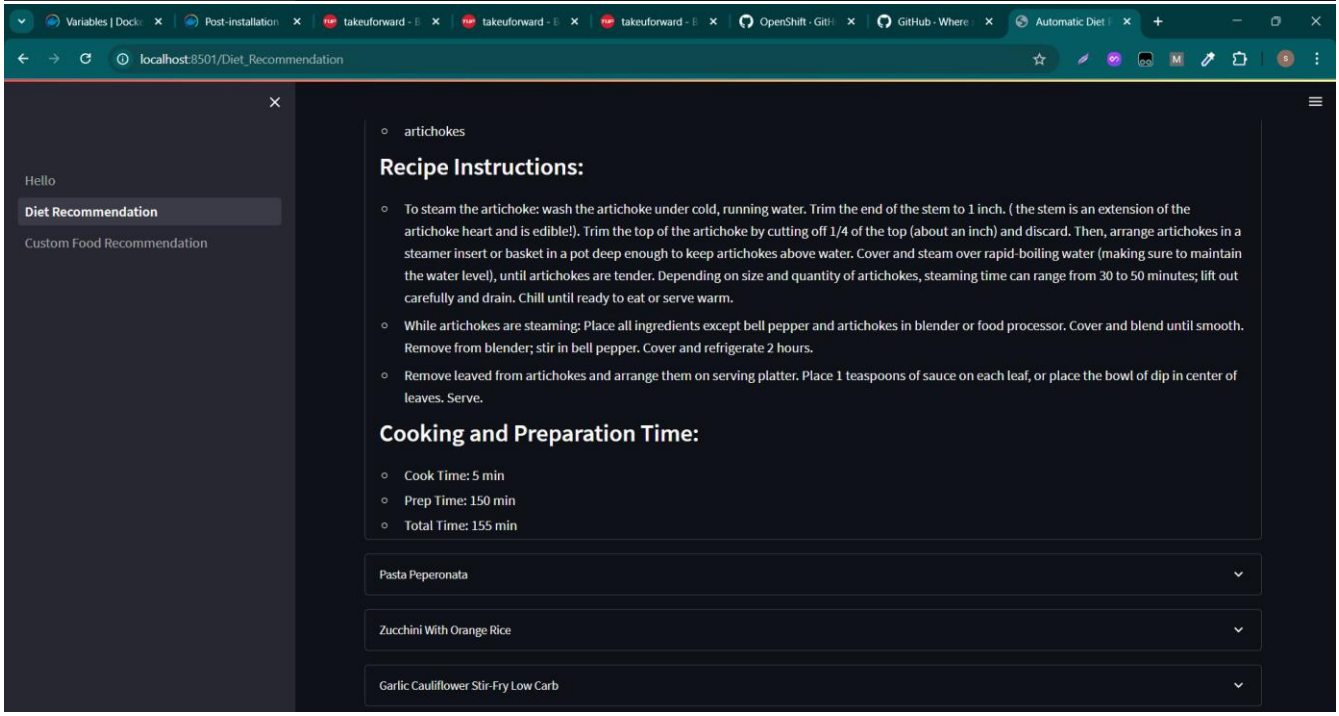


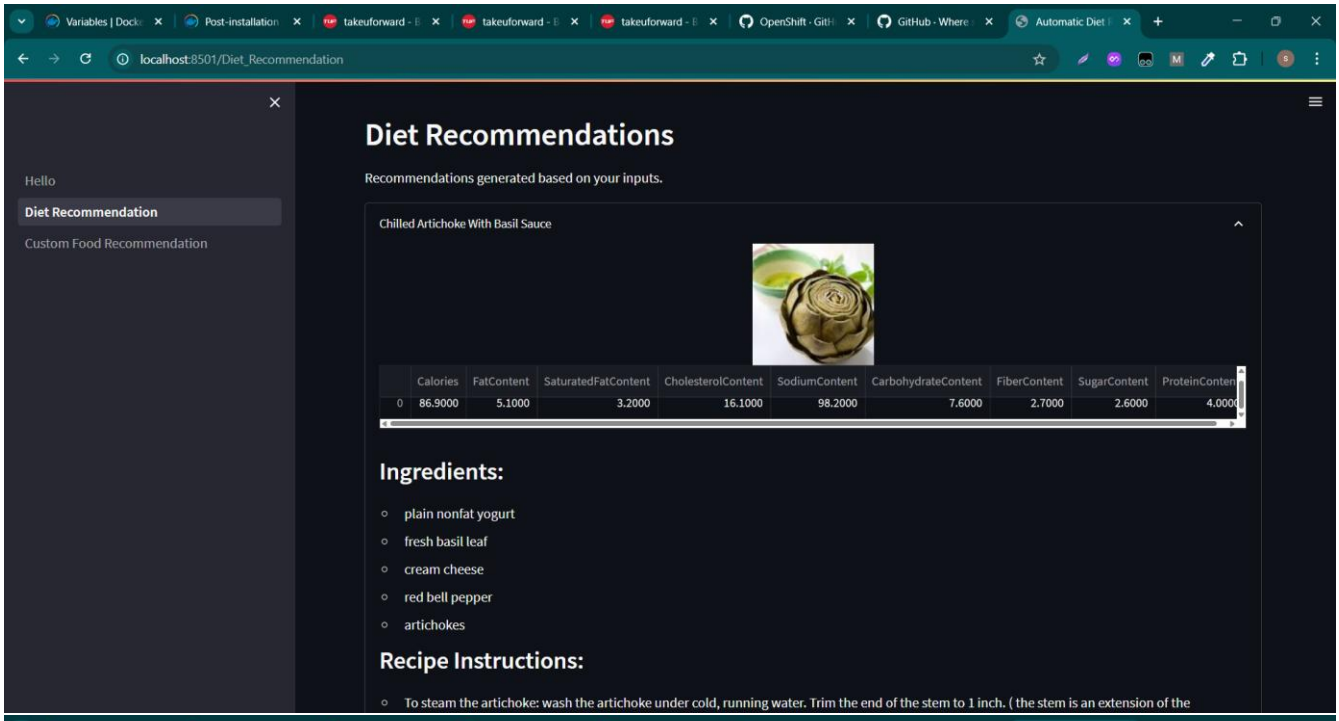


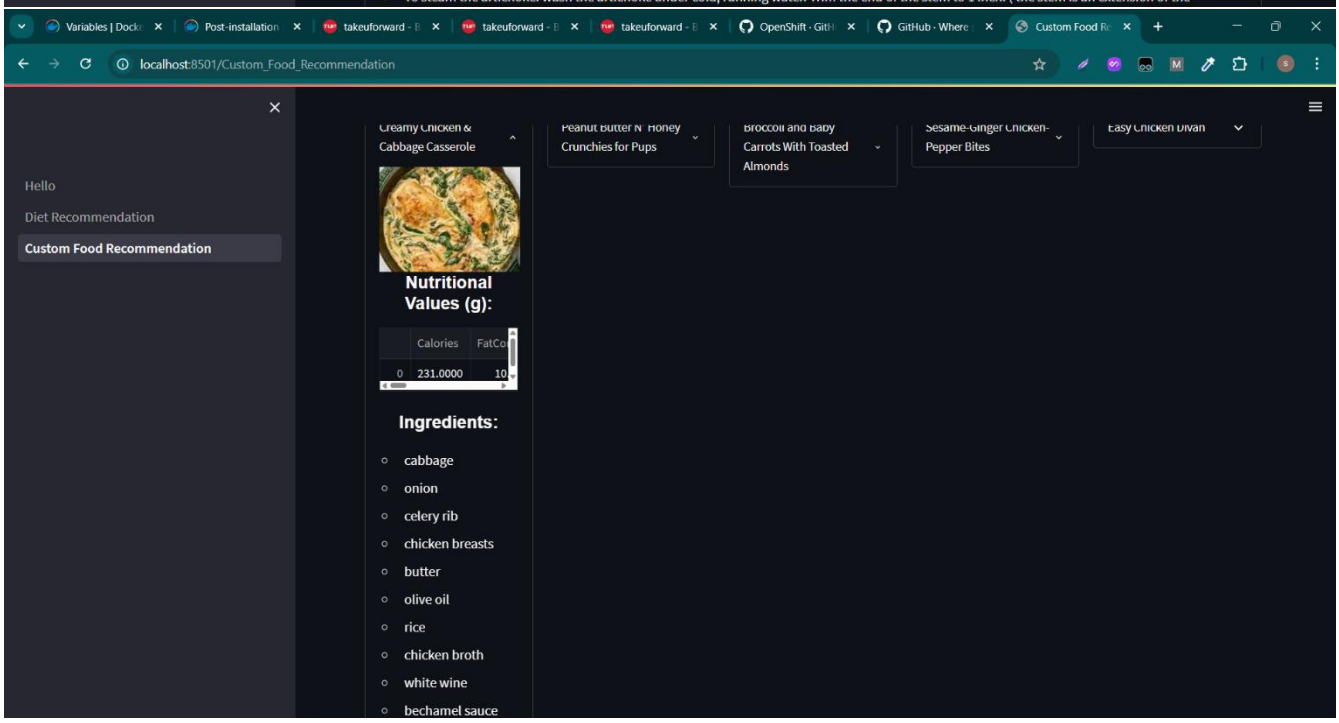

localhost:8501/Diet_Recommendation
Automatic Diet Recommendation
Hello
Diet Recommendation
Custom Food Recommendation
Modify the values and click the Generate button to use
Age
23
Height(cm)
160
Weight(kg)
63
Gender
Male
Female
Activity
Light exercise
Little/no exercise
Extra active (very active & physical job)
Choose your weight loss plan:
Maintain weight
Meals per day
localhost:8501/Custom_Food_Recommendation
Hello
Diet Recommendation
Custom Food Recommendation
FiberContent
10
0
50
SugarContent
10
0
40
ProteinContent
10
0
40
Recommendation options (OPTIONAL):
Number of recommendations
5
5
20
Specify ingredients to include in the recommendations separated by ";" :
butter;chicken
Example: Milk;eggs;butter;chicken...
Generate

OUTPUT PAGE

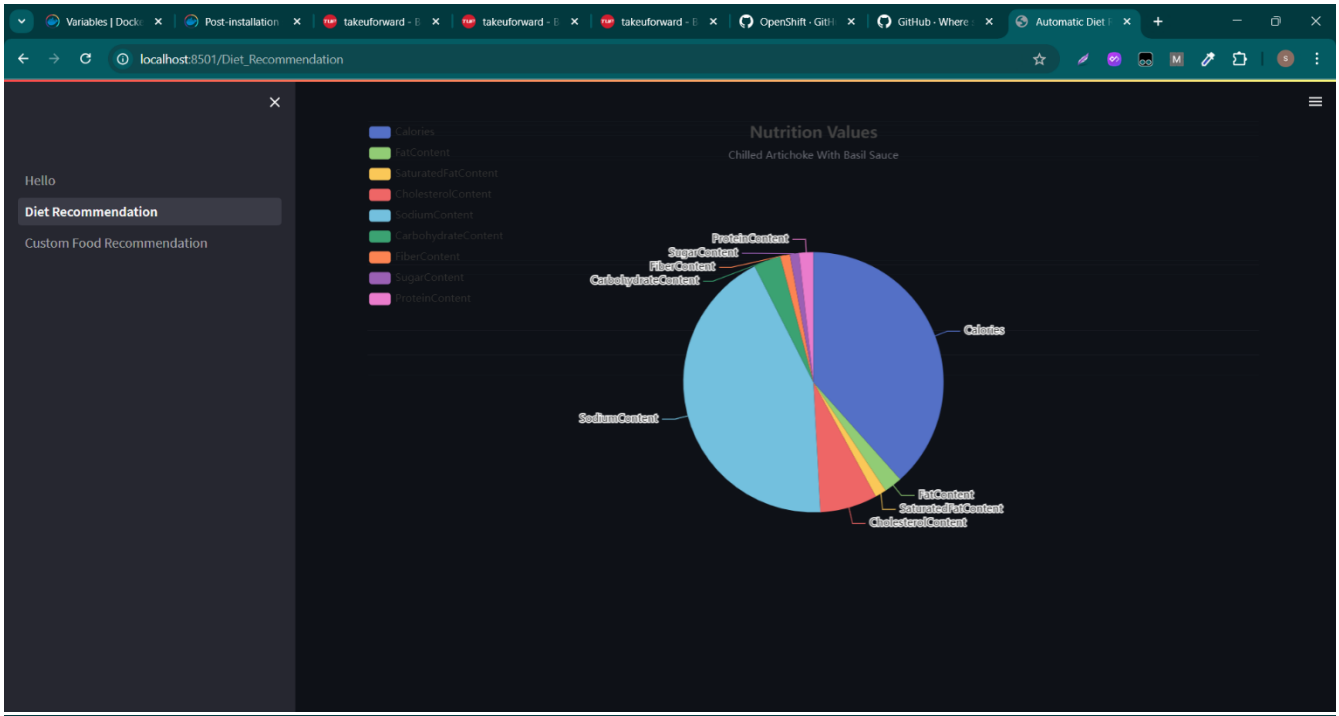

localhost:8501/Diet_Recommendation
Hello
Diet Recommendation
Custom Food Recommendation
Pasta Peperonata
Zucchini With Orange Rice
Garlic Cauliflower Stir-Fry Low Carb
Asparagus With Butter Lemon and Mint Drizzle
Crockpot Saucy Ravioli With Meatballsl
Pasta Peperonata
Sugar Snap Peas and Onions
Cheesy Mashed Turnips
Frugal Gourmet's Melitzanosalata
French-Style Peas
Spinach Casserole

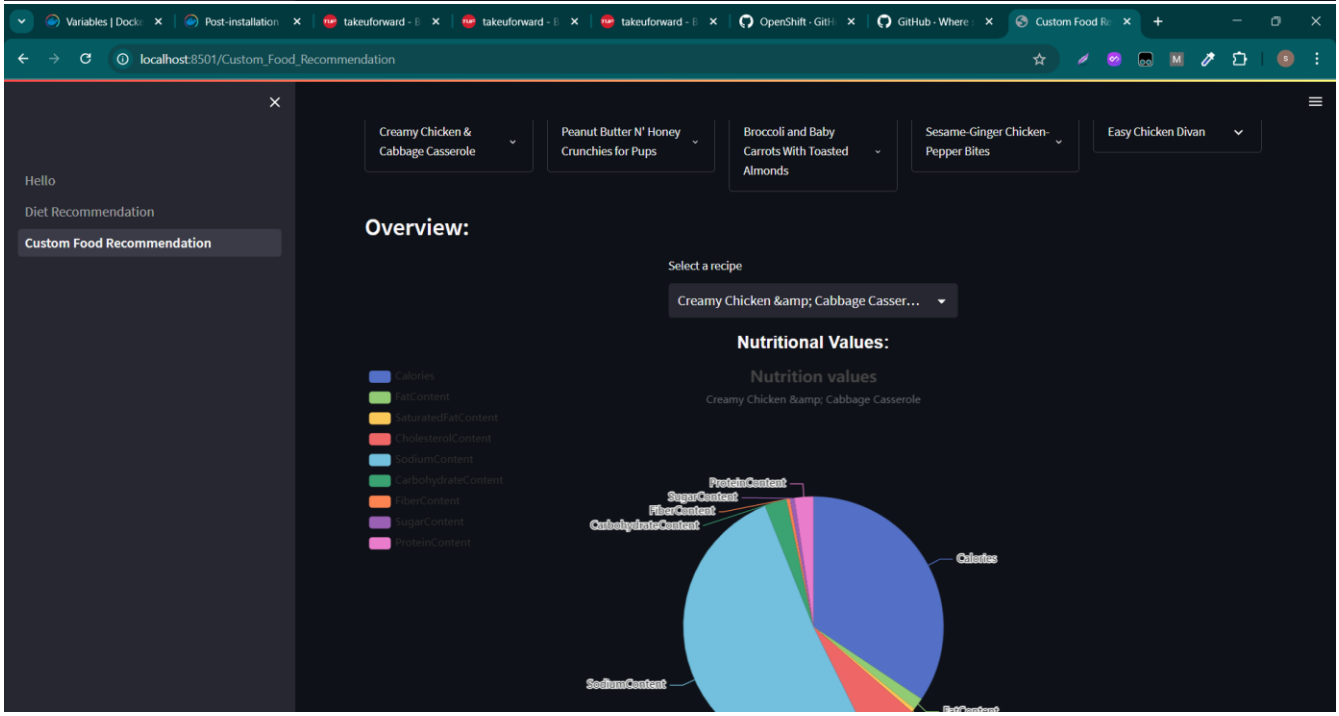

localhost:8501/Diet_Recommendation
Hello
Diet Recommendation
Custom Food Recommendation
artichokes
Recipe Instructions:
To steam the artichoke: wash the artichoke under cold, running water. Trim the end of the stem to 1 inch. (the stem is an extension of the artichoke heart and is edible!). Trim the top of the artichoke by cutting off 1/4 of the top (about an inch) and discard. Then, arrange artichokes in a steamer insert or basket in a pot deep enough to keep artichokes above water. Cover and steam over rapid-boiling water (making sure to maintain the water level), until artichokes are tender. Depending on size and quantity of artichokes, steaming time can range from 30 to 50 minutes; lift out carefully and drain. Chill until ready to eat or serve warm.
While artichokes are steaming: Place all ingredients except bell pepper and artichokes in blender or food processor. Cover and blend until smooth. Remove from blender; stir in bell pepper. Cover and refrigerate 2 hours.
Remove leaved from artichokes and arrange them on serving platter. Place 1 teaspoons of sauce on each leaf, or place the bowl of dip in center of leaves. Serve.
Cooking and Preparation Time:
Cook Time: 5 min
Prep Time: 150 min
Total Time: 155 min
Pasta Peperonata
Zucchini With Orange Rice
Garlic Cauliflower Stir-Fry Low Carb


localhost:8501/Diet_Recommendation
Hello
Diet Recommendation
Custom Food Recommendation
Diet Recommendations
Recommendations generated based on your inputs.
Chilled Artichoke With Basil Sauce
Calories FatContent SaturatedFatContent CholesterolContent SodiumContent CarbohydrateContent FiberContent SugarContent ProteinConten
0 86.9000 5.1000 3.2000 16.1000 98.2000 7.6000 2.7000 2.6000 4.000
Ingredients:
plain nonfat yogurt
fresh basil leaf
cream cheese
red bell pepper
artichokes
Recipe Instructions:
To steam the artichoke: wash the artichoke under cold, running water. Trim the end of the stem to 1 inch. (the stem is an extension of the

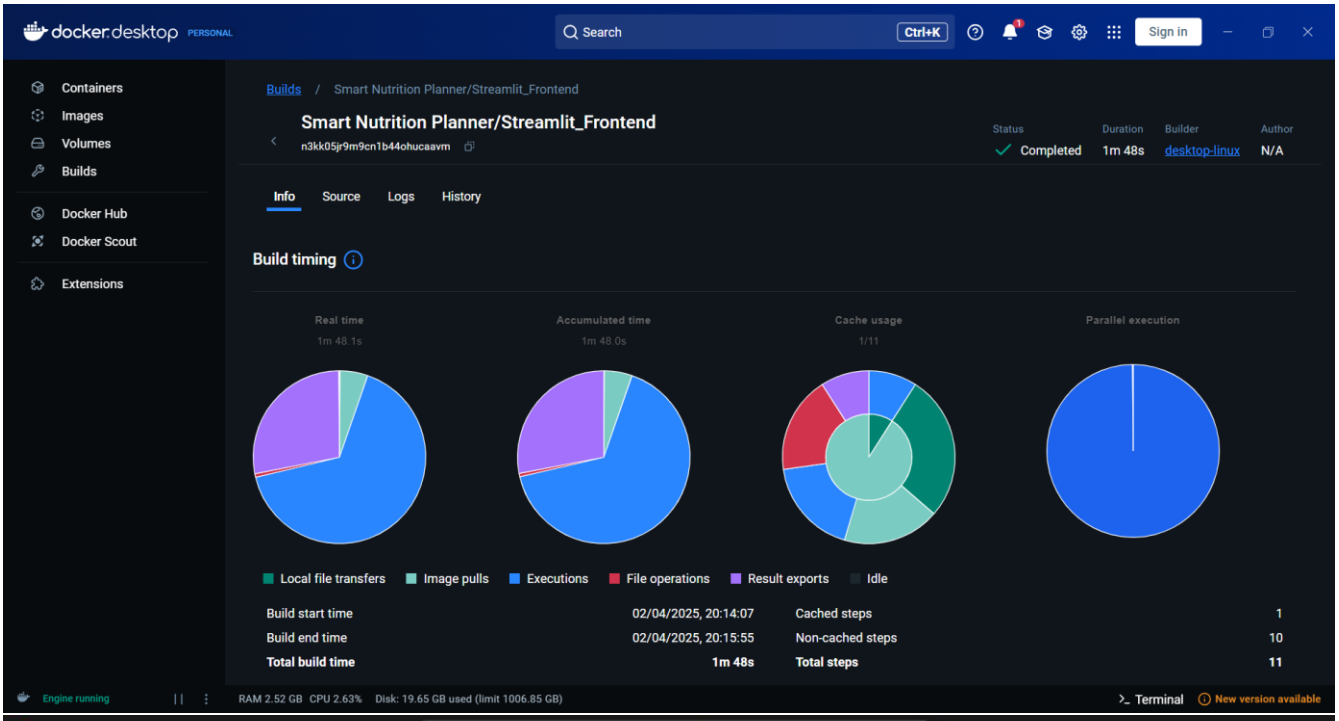

localhost:8501/Custom_Food_Recommendation
Hello
Diet Recommendation
Custom Food Recommendation
Creamy Chicken & Cabbage Casserole
Peanut Butter N' Honey Crunchies for Pups
Broccoli and Baby Carrots With Toasted Almonds
Sesame-Ginger Chicken-Pepper Bites
Easy Chicken Divan
Nutritional Values (g):
Calories FatCo
0 231.0000 10
Ingredients:
cabbage
onion
celery rib
chicken breasts
butter
olive oil
rice
chicken broth
white wine
bechamel sauce

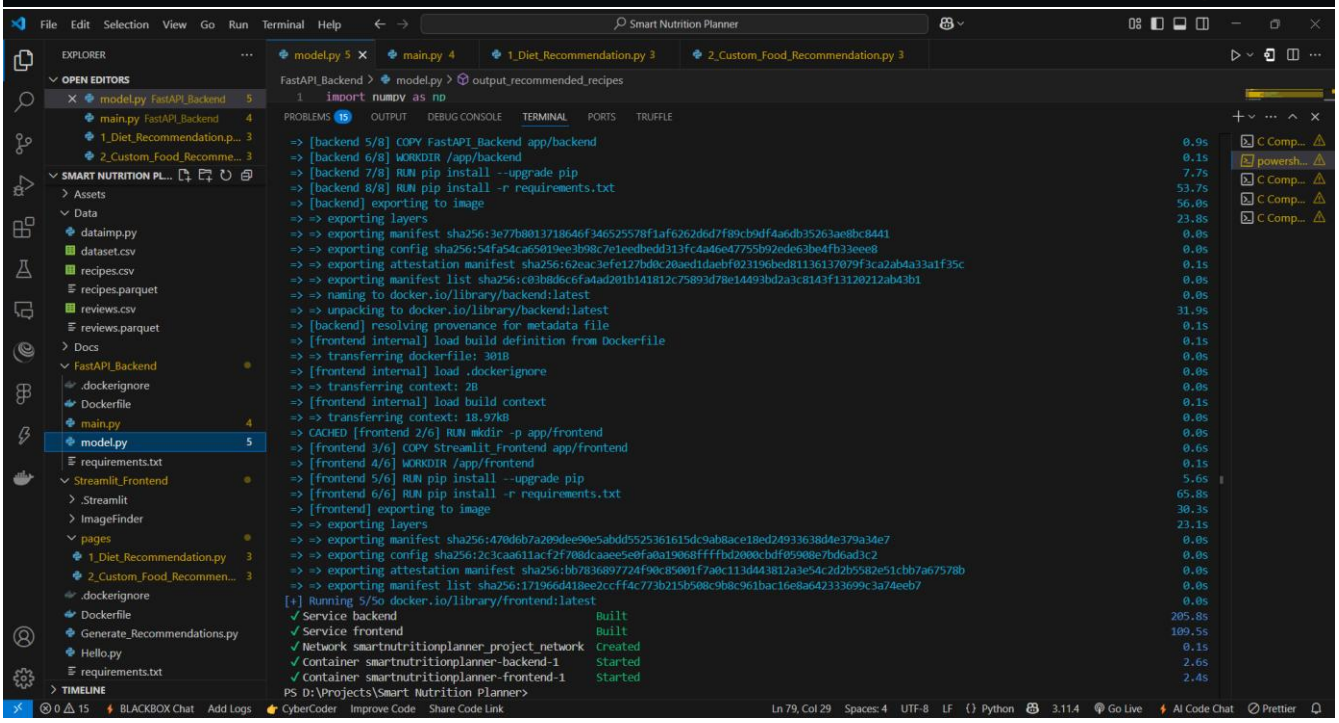
PLOT DIAGRAMS

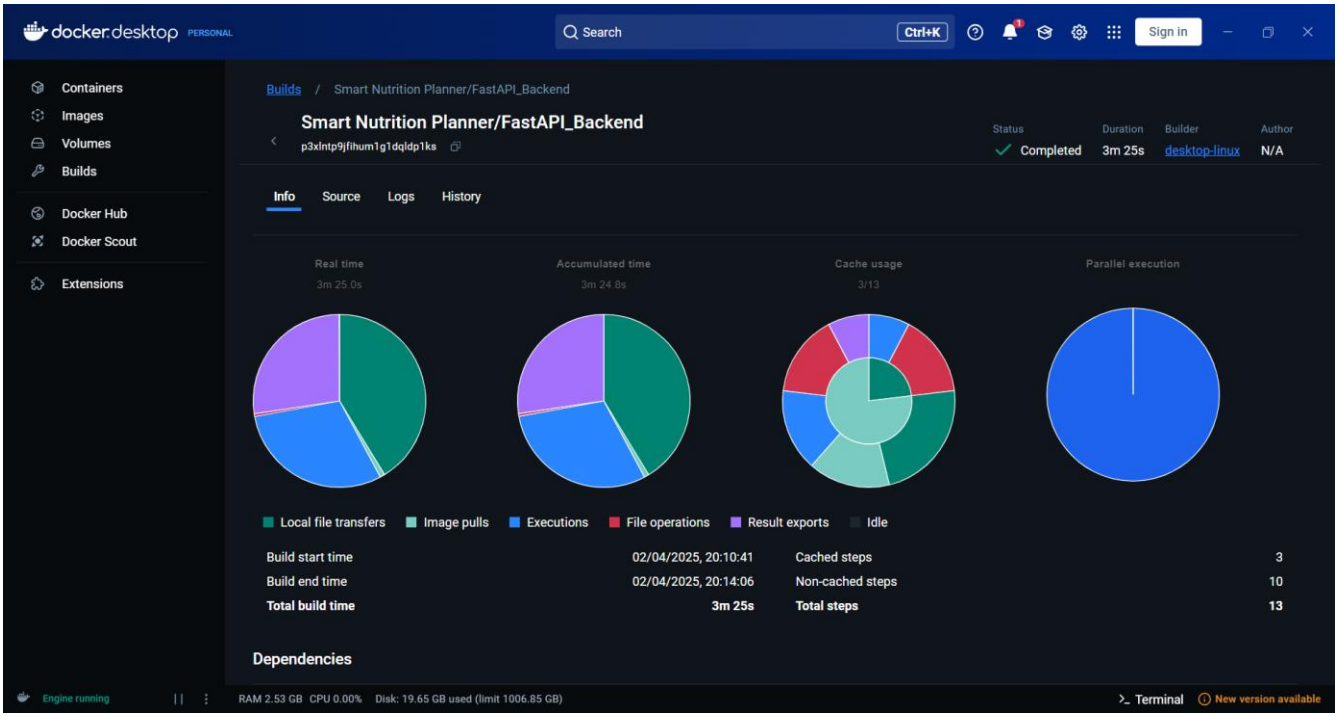




BACKEND **SCREENSHOTS**







10.2 SAMPLE CODE:

Model.py

```
import numpy as np
import re
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import FunctionTransformer

def scaling(dataframe):
    scaler = StandardScaler()
    prep_data = scaler.fit_transform(dataframe.iloc[:, 6:15].to_numpy())
    return prep_data, scaler

def nn_predictor(prep_data):
    neigh = NearestNeighbors(metric='cosine', algorithm='brute')
    neigh.fit(prep_data)
    return neigh

def build_pipeline(neigh, scaler, params):
```

```
    transformer = FunctionTransformer(lambda X: neigh.kneighbors(scaler.transform(X), **params))
    pipeline = Pipeline([('std_scaler', scaler), ('NN', transformer)])
    return pipeline

def extract_data(dataframe, ingredients):
    extracted_data = dataframe.copy()
    extracted_data = extract_ingredient_filtered_data(extracted_data, ingredients)
    return extracted_data

def extract_ingredient_filtered_data(dataframe, ingredients):
    extracted_data = dataframe.copy()

    if ingredients:
        if isinstance(extracted_data['RecipeIngredientParts'].iloc[0], list):
            extracted_data = extracted_data[
                extracted_data['RecipeIngredientParts'].apply(lambda x: all(ing.lower() in map(str.lower, x)
for ing in ingredients))
            ]
        else:
            regex_string = ''.join(map(lambda x: f'(?=.*{x})', ingredients))
            extracted_data = extracted_data[
                extracted_data['RecipeIngredientParts'].str.contains(regex_string, regex=True,
flags=re.IGNORECASE, na=False)
            ]

    print(f"Filtered recipes count: {extracted_data.shape[0]}")  # Debugging

    if extracted_data.shape[0] < 5:
        print("Not enough recipes after filtering, relaxing the filtering criteria.")
        return dataframe  # Return full dataset as fallback

    return extracted_data

def apply_pipeline(pipeline, _input, extracted_data):
    try:
        _input = np.array(_input).reshape(1, -1)
        indices = pipeline.transform(_input)[0]
        print(f"Recommended recipe indices: {indices}")  # Debugging
        return extracted_data.iloc[indices]
    except Exception as e:
        print(f"Error in pipeline transformation: {e}")
        return None
```

```
def recommend(dataframe, _input, ingredients=[], params={'n_neighbors': 5, 'return_distance': False}):
    extracted_data = extract_data(dataframe, ingredients)

    if extracted_data.shape[0] < params['n_neighbors']:
        print("Not enough recipes after filtering, using general recommendations.")
        extracted_data = dataframe  # Fallback to the full dataset

    prep_data, scaler = scaling(extracted_data)
    neigh = nn_predictor(prep_data)
    pipeline = build_pipeline(neigh, scaler, params)
    return apply_pipeline(pipeline, _input, extracted_data)

def extract_quoted_strings(s):
    if isinstance(s, str):
        return re.findall(r'"([^"]*)"', s)
    return []

def output_recommended_recipes(dataframe):
    if dataframe is not None and not dataframe.empty:
        output = dataframe.to_dict("records")
        for recipe in output:
            recipe['RecipeIngredientParts'] = extract_quoted_strings(recipe.get('RecipeIngredientParts', ''))
            recipe['RecipeInstructions'] = extract_quoted_strings(recipe.get('RecipeInstructions', ''))
        return output
    else:
        print("No recommended recipes found.")  # Debugging
        return None
```

Main.py

```
from fastapi import FastAPI
from pydantic import BaseModel, conlist
from typing import List, Optional
import pandas as pd
import os
from model import recommend, output_recommended_recipes

# Load dataset (ensure correct file path)
DATASET_PATH = os.getenv("DATASET_PATH", "../Data/dataset.csv")
dataset = pd.read_csv(DATASET_PATH, compression='gzip')

app = FastAPI()
```

```
# Disease-to-ingredient mapping
disease_restrictions = {
    "diabetes": ["sugar", "honey", "white bread"],
    "hypertension": ["salt", "processed meats", "canned foods"],
    "heart disease": ["red meat", "butter", "fried foods"],
    "kidney disease": ["high sodium foods", "processed cheese", "canned soups"]
}

# Request parameter model
class Params(BaseModel):
    n_neighbors: int = 5
    return_distance: bool = False

class PredictionIn(BaseModel):
    nutrition_input: conlist(float, min_items=9, max_items=9)
    ingredients: List[str] = []
    diseases: List[str] = []  # New field for diseases
    params: Optional[Params] = None  # Ensure default is handled

# Recipe output model
class Recipe(BaseModel):
    Name: str
    CookTime: str
    PrepTime: str
    TotalTime: str
    RecipeIngredientParts: List[str]
    Calories: float
    FatContent: float
    SaturatedFatContent: float
    CholesterolContent: float
    SodiumContent: float
    CarbohydrateContent: float
    FiberContent: float
    SugarContent: float
    ProteinContent: float
    RecipeInstructions: List[str]

class PredictionOut(BaseModel):
    output: List[Recipe] = []  # Default to empty list

@app.get("/")
def home():
    return {"health_check": "OK"}
```

```
@app.post("/predict/", response_model=PredictionOut)
def update_item(prediction_input: PredictionIn):
    # Exclude ingredients based on diseases
    restricted_ingredients = set()
    for disease in prediction_input.diseases:
        if disease in disease_restrictions:
            restricted_ingredients.update(map(str.lower, disease_restrictions[disease]))

    # Remove restricted ingredients from user input (case-insensitive match)
    filtered_ingredients = [ing for ing in prediction_input.ingredients if ing.lower() not in
restricted_ingredients]

    # Ensure params is not None
    params_dict = prediction_input.params.dict() if prediction_input.params else {"n_neighbors": 5,
"return_distance": False}

    # Get recommendations
    recommendation_dataframe = recommend(dataset, prediction_input.nutrition_input,
filtered_ingredients, params_dict)
    output = output_recommended_recipes(recommendation_dataframe)

    return {"output": output if output else []}
```

Diet_Recommendation.py

```
import streamlit as st
import pandas as pd
from Generate_Recommendations import Generator
from random import uniform as rnd
from ImageFinder.ImageFinder import get_images_links as find_image
from streamlit_echarts import st_echarts

st.set_page_config(page_title="Automatic Diet Recommendation", page_icon="🥗", layout="wide")

disease_options = ["Diabetes", "Hypertension", "Obesity", "Heart Disease", "Anemia", "Osteoporosis"]
nutrition_values = ['Calories', 'FatContent', 'SaturatedFatContent', 'CholesterolContent',
'SodiumContent', 'CarbohydrateContent', 'FiberContent', 'SugarContent', 'ProteinContent']

if 'generated' not in st.session_state:
    st.session_state.generated = False
    st.session_state.recommendations = None
```

```
class Person:
    def __init__(self, age, height, weight, gender, activity, meals_calories_perc, weight_loss, diseases):
        self.age = age
        self.height = height
        self.weight = weight
        self.gender = gender
        self.activity = activity
        self.meals_calories_perc = meals_calories_perc
        self.weight_loss = weight_loss
        self.diseases = diseases

    def calories_calculator(self):
        if self.gender == 'Male':
            return 10 * self.weight + 6.25 * self.height - 5 * self.age + 5
        else:
            return 10 * self.weight + 6.25 * self.height - 5 * self.age - 161

    def generate_recommendations(self):
        total_calories = self.weight_loss * self.calories_calculator()
        recommendations = []

        for meal, percentage in self.meals_calories_perc.items():
            meal_calories = percentage * total_calories
            generator = Generator([meal_calories] + [rnd(10, 30) for _ in range(8)], self.diseases)

            try:
                response = generator.generate()
                if response and response.json().get('output'):
                    recommended_recipes = response.json()['output']
                    for recipe in recommended_recipes:
                        recipe['image_link'] = find_image(recipe['Name'])
                    recommendations.extend(recommended_recipes)
                else:
                    st.warning(f"No recommendations found for {meal}.")
            except Exception as e:
                st.error(f"Error generating recommendations for {meal}: {e}")
                continue

        return recommendations if recommendations else None

st.markdown("<h1 style='text-align: center;'>Automatic Diet Recommendation</h1>",
unsafe_allow_html=True)
```

```
with st.form("recommendation_form"):
    st.write("Modify the values and click the Generate button to use")
    age = st.number_input('Age', min_value=2, max_value=120, step=1)
    height = st.number_input('Height(cm)', min_value=50, max_value=300, step=1)
    weight = st.number_input('Weight(kg)', min_value=10, max_value=300, step=1)
    gender = st.radio('Gender', ('Male', 'Female'))
    activity = st.select_slider('Activity', options=['Little/no exercise', 'Light exercise', 'Moderate exercise
(3-5 days/wk)', 'Very active (6-7 days/wk)', 'Extra active (very active & physical job)'])
    weight_loss_option = st.selectbox('Choose your weight loss plan:', ["Maintain weight", "Mild weight
loss", "Weight loss", "Extreme weight loss"])
    weight_loss = [1, 0.9, 0.8, 0.6][["Maintain weight", "Mild weight loss", "Weight loss", "Extreme
weight loss"].index(weight_loss_option)]

    number_of_meals = st.slider('Meals per day', min_value=3, max_value=5, step=1, value=3)
    if number_of_meals == 3:
        meals_calories_perc = {'breakfast': 0.35, 'lunch': 0.40, 'dinner': 0.25}
    elif number_of_meals == 4:
        meals_calories_perc = {'breakfast': 0.30, 'morning snack': 0.05, 'lunch': 0.40, 'dinner': 0.25}
    else:
        meals_calories_perc = {'breakfast': 0.30, 'morning snack': 0.05, 'lunch': 0.40, 'afternoon snack':
0.05, 'dinner': 0.20}

    diseases = st.multiselect("Select any existing health conditions:", disease_options)
    generated = st.form_submit_button("Generate")

if generated:
    st.session_state.generated = True
    person = Person(age, height, weight, gender, activity, meals_calories_perc, weight_loss, diseases)
    recommendations = person.generate_recommendations()
    st.session_state.recommendations = recommendations

if st.session_state.generated:
    st.header("Diet Recommendations")
    if st.session_state.recommendations:
        st.write("Recommendations generated based on your inputs.")
        for recipe in st.session_state.recommendations:
            expander = st.expander(recipe['Name'])
            expander.markdown(f'<div><center><img src={recipe["image_link"]}
alt="{recipe["Name"]}"></center></div>', unsafe_allow_html=True)
            nutritions_df = pd.DataFrame({value: [recipe[value]] for value in nutrition_values})
            expander.dataframe(nutritions_df)
            expander.markdown("### Ingredients:")
            for ingredient in recipe['RecipeIngredientParts']:
```

```
            expander.markdown(f"- {ingredient}")
        expander.markdown("### Recipe Instructions:")
        for instruction in recipe['RecipeInstructions']:
            expander.markdown(f"- {instruction}")
        expander.markdown("### Cooking and Preparation Time:")
        expander.markdown(f"- Cook Time: {recipe['CookTime']} min\n- Prep Time:
{recipe['PrepTime']} min\n- Total Time: {recipe['TotalTime']} min")

    selected_recipe = st.selectbox("Select a recipe for nutritional overview:", [recipe['Name'] for
recipe in st.session_state.recommendations])
    selected_data = next(recipe for recipe in st.session_state.recommendations if recipe['Name'] ==
selected_recipe)
    options = {
        "title": {"text": "Nutrition Values", "subtext": selected_recipe, "left": "center"},
        "tooltip": {"trigger": "item"},
        "legend": {"orient": "vertical", "left": "left"},
        "series": [{
            "name": "Nutrition values",
            "type": "pie",
            "radius": "50%",
            "data": [{"value": selected_data[nutrient], "name": nutrient} for nutrient in nutrition_values],
        }],
    }
    st_echarts(options=options, height="600px")
else:
    st.warning("No recommendations available.")
```

Custom_Food_Recommendations.py

```
import streamlit as st
from Generate_Recommendations import Generator
from ImageFinder.ImageFinder import get_images_links as find_image
import pandas as pd
from streamlit_echarts import st_echarts

st.set_page_config(page_title="Custom Food Recommendation", page_icon=" 🔍 ",layout="wide")
nutrition_values=['Calories','FatContent','SaturatedFatContent','CholesterolContent','SodiumContent','C
arbohydrateContent','FiberContent','SugarContent','ProteinContent']
if 'generated' not in st.session_state:
    st.session_state.generated = False
    st.session_state.recommendations=None

class Recommendation:
```

```
    def __init__(self,nutrition_list,nb_recommendations,ingredient_txt):
        self.nutrition_list=nutrition_list
        self.nb_recommendations=nb_recommendations
        self.ingredient_txt=ingredient_txt
        pass
    def generate(self,):
        params={'n_neighbors':self.nb_recommendations,'return_distance':False}
        ingredients=self.ingredient_txt.split(';')
        generator=Generator(self.nutrition_list,ingredients,params)
        recommendations=generator.generate()
        recommendations = recommendations.json()['output']
        if recommendations!=None:
            for recipe in recommendations:
                recipe['image_link']=find_image(recipe['Name'])
        return recommendations

class Display:
    def __init__(self):
        self.nutrition_values=nutrition_values

    def display_recommendation(self,recommendations):
        st.subheader('Recommended recipes:')
        if recommendations!=None:
            rows=len(recommendations)//5
            for column,row in zip(st.columns(5),range(5)):
                with column:
                    for recipe in recommendations[rows*row:rows*(row+1)]:
                        recipe_name=recipe['Name']
                        expander = st.expander(recipe_name)
                        recipe_link=recipe['image_link']
                        recipe_img=f'<div><center><img src={recipe_link}
alt={recipe_name}></center></div>'
                        nutritions_df=pd.DataFrame({value:[recipe[value]] for value in nutrition_values})

                        expander.markdown(recipe_img,unsafe_allow_html=True)
                        expander.markdown(f'<h5 style="text-align: center;font-family:sans-serif;">Nutritional
Values (g):</h5>', unsafe_allow_html=True)
                        expander.dataframe(nutritions_df)
                        expander.markdown(f'<h5 style="text-align: center;font-family:sans-
serif;">Ingredients:</h5>', unsafe_allow_html=True)
                        for ingredient in recipe['RecipeIngredientParts']:
                            expander.markdown(f"""
                                    - {ingredient}
```

```
                """)
                expander.markdown(f'<h5 style="text-align: center;font-family:sans-serif;">Recipe Instructions:</h5>', unsafe_allow_html=True)
                for instruction in recipe['RecipeInstructions']:
                    expander.markdown(f"""
                            - {instruction}
                    """)
                expander.markdown(f'<h5 style="text-align: center;font-family:sans-serif;">Cooking and Preparation Time:</h5>', unsafe_allow_html=True)
                expander.markdown(f"""
                    - Cook Time      : {recipe['CookTime']}min
                    - Preparation Time: {recipe['PrepTime']}min
                    - Total Time     : {recipe['TotalTime']}min
                """)
        else:
            st.info('Couldn\'t find any recipes with the specified ingredients', icon="😵")
    def display_overview(self,recommendations):
        if recommendations!=None:
            st.subheader('Overview:')
            col1,col2,col3=st.columns(3)
            with col2:
                selected_recipe_name=st.selectbox('Select a recipe',[recipe['Name'] for recipe in recommendations])
            st.markdown(f'<h5 style="text-align: center;font-family:sans-serif;">Nutritional Values:</h5>', unsafe_allow_html=True)
            for recipe in recommendations:
                if recipe['Name']==selected_recipe_name:
                    selected_recipe=recipe
            options = {
        "title": {"text": "Nutrition values", "subtext": f"{selected_recipe_name}", "left": "center"},
        "tooltip": {"trigger": "item"},
        "legend": {"orient": "vertical", "left": "left",},
        "series": [
            {
                "name": "Nutrition values",
                "type": "pie",
                "radius": "50%",
                "data": [{"value":selected_recipe[nutrition_value],"name":nutrition_value} for nutrition_value in self.nutrition_values],
                "emphasis": {
                    "itemStyle": {
                        "shadowBlur": 10,
                        "shadowOffsetX": 0,
```

```
                "shadowColor": "rgba(0, 0, 0, 0.5)",
            }
        },
    }
    ],
}
    st_echarts(options=options, height="600px",)
    st.caption('You can select/deselect an item (nutrition value) from the legend.')

title="<h1 style='text-align: center;'>Custom Food Recommendation</h1>"
st.markdown(title, unsafe_allow_html=True)
display=Display()
with st.form("recommendation_form"):
    st.header('Nutritional values:')
    Calories = st.slider('Calories', 0, 2000, 500)
    FatContent = st.slider('FatContent', 0, 100, 50)
    SaturatedFatContent = st.slider('SaturatedFatContent', 0, 13, 0)
    CholesterolContent = st.slider('CholesterolContent', 0, 300, 0)
    SodiumContent = st.slider('SodiumContent', 0, 2300, 400)
    CarbohydrateContent = st.slider('CarbohydrateContent', 0, 325, 100)
    FiberContent = st.slider('FiberContent', 0, 50, 10)
    SugarContent = st.slider('SugarContent', 0, 40, 10)
    ProteinContent = st.slider('ProteinContent', 0, 40, 10)
    nutritions_values_list=[Calories,FatContent,SaturatedFatContent,CholesterolContent,SodiumContent
,CarbohydrateContent,FiberContent,SugarContent,ProteinContent]
    st.header('Recommendation options (OPTIONAL):')
    nb_recommendations = st.slider('Number of recommendations', 5, 20,step=5)
    ingredient_txt=st.text_input('Specify ingredients to include in the recommendations separated by ";"
:',placeholder='Ingredient1;Ingredient2;...')
    st.caption('Example: Milk;eggs;butter;chicken...')
    generated = st.form_submit_button("Generate")
if generated:
    with st.spinner('Generating recommendations...'):
        recommendation=Recommendation(nutritions_values_list,nb_recommendations,ingredient_txt)
        recommendations=recommendation.generate()
        st.session_state.recommendations=recommendations
    st.session_state.generated=True

if st.session_state.generated:
    with st.container():
        display.display_recommendation(st.session_state.recommendations)
    with st.container():
        display.display_overview(st.session_state.recommendations
```

SYNOPSIS

The "Smart Nutrition Planner" project allows users to view the results of the machine learning algorithm from anywhere via the Internet. A subfield of computer science and artificial intelligence (AI) called machine learning focuses on using data and algorithms to mimic human learning processes while progressively increasing their accuracy.

A non-parametric, supervised learning classifier, the k-Nearest Neighbors (KNN) algorithm uses proximity to classify or predict how to group a single data point. It is among the most widely used and straightforward regression and classification classifiers in machine learning today.

For tasks like text analysis or recommendation systems, the k-Nearest Neighbors (kNN) algorithm, when paired with cosine similarity, determines the "k" most similar vectors to a target vector based on the cosine of the angle between them. Cosine similarity quantifies the degree of similarity between two vectors, independent of their magnitude, by calculating the cosine of their angle. Perfect dissimilarity (vectors pointing in opposite directions), no similarity (vectors perpendicular), and perfect similarity (vectors pointing in the same direction) are the three extremes of the cosine similarity value, which goes from -1 to 1. Data points become vectors when kNN and cosine similarity are combined.

The query vector's similarity to every other vector in the dataset is ascertained using cosine similarity. The highest cosine similarity scores are used to determine the k-nearest neighbors.

SANJAY K (20MSS040), Fourth year M.Sc Software systems Batch (2020-2025) in PSG college of Arts & science, Coimbatore.