

Development Of AI Driven Web Interface For Diet Plans Based on Diseases

Nikit M Sambrekar^{1*}, Pallavi P Gaonkar¹, Rakshita M Patil¹
Sahil Ron², H. P. Rajani³, Salma S Shahapur⁴

¹Jain College of Engineering, VTU,
Belagavi, Karnataka, India
nikitasambrekar9@gmail.com
pallavimp10@gmail.com
prakshita06@gmail.com
sahilron9@gmail.com

Abstract. For those looking for dietary advice related to certain symptoms or medical problems, the "Symptoms and Disease Diet Plan Web Interface" attempts to offer a thorough and user-friendly platform. Accessible and trustworthy nutritional information is more important than ever due to the rising incidence of lifestyle-related illnesses and the growing interest in holistic health care. With the help of an intuitive user interface, users will be able to input their symptoms or choose a particular ailment from a carefully curated list using this web application. The interface will use this information to create customized diet programs that follow medical professionals' recommendations and evidence-based dietary guidelines. The meal plans will focus on nutrient-dense foods, appropriate dietary limitations, and doable meal ideas to promote general health and manage symptoms. The interface fills a vital gap by providing easily accessible, individualized food recommendations based on specific medical concerns. In the current healthcare environment, dietary modifications that are in line with particular symptoms or diagnoses are frequently necessary for the efficient management of lifestyle-related diseases including diabetes, cardiovascular problems, and gastrointestinal disorders, which are becoming more and more common. With the use of evidence-based dietary recommendations, this web interface seeks to empower users to make decisions that promote holistic health and symptom management. The software creates customized diet regimens using algorithms that are based on medical studies and the need for nutrition. Users can choose from a pre-defined list of ailments or provide their own symptoms. Plans include suggestions for foods high in nutrients, appropriate portion sizes, and dietary adjustments designed to reduce symptoms and promote general health.

Keywords: Interface, Nutritional health, Symptoms Management, Disease

1. Introduction

The goal of this initiative is to democratize the availability of trustworthy health information. It aims to close the knowledge gap between professional practice and everyday use by providing users with a complete resource base in addition to a meal planning tool. The app seeks to promote a culture of educated decision-making and proactive health management through instructional modules, interactive features, and community support. An equilibrium in nutrition plays a major part of leading a healthy lifestyle. Furthermore, the combination of proper diet and enough physical activities can help anyone to maintain a healthy weight. Unfortunately, today people are losing control in their whole well-being and risk of chronic diseases are increasing rapidly. Considering to the recent studies, around 30% of world's population is overweight or obese [1].

In an effort to overcome these obstacles, the "Symptoms and Disease Diet Plan Web Interface" offers a thorough solution that gives users access to individualized dietary advice. The app attempts to close the gap between medical knowledge and common eating behaviors by utilizing technological breakthroughs and incorporating strong nutritional research. The application aims to enable consumers to make knowledgeable food selection that can enhance their quality of life and health outcomes by providing them with educational

materials, customized meal plans, and easy interfaces. The scalability and adaptiveness of conversational AI allow an individualized learning support for all learners combined with collaboration opportunities and thus more equality in education [2]. A set of questionnaires and different data analysis methods were used in this study to measure participant's self perceived maintenance degree of a healthy diet [3].

Objectives of project

1. Create an intuitive web application interface that allows users to easily input symptoms or select specific diseases.
2. Ensure the interface is accessible across various devices and user-friendly for individuals with different levels of technological proficiency.
3. Utilize algorithms based on evidence-based nutritional guidelines to create customized eating schedules that are suited to individual symptoms or diseases.
4. Include recommendations for nutrient-dense foods, portion sizes, and dietary modifications to address specific health concerns.
5. Offer detailed information on essential nutrients and their roles in managing symptoms or conditions.

2. Literature Survey

Personalized Nutrition: Principles and applications by Lynnette R. ferguson deal into the science of individualized diet, examining how individual food requirements and genetic applications and genetic profiles influences health outcomes.

Applications in Nutrition: Review existing AI techniques in nutrition and dietetics

Focusing on how AI is utilized to analyze dietary data and provide recommendations.

Diet Plan: It provides effective means for restoring health, even before pathology has set in, and the individual has fallen sick

It lists several levels of malfunctioning in its classification of dosha imbalance prior to pathologic symptoms manifesting and disease occurring.

A set of questionnaires and different data analysis methods were used in this study to measure participant's self perceived maintenance degree of a healthy diet [3].

3. Methodology

The below fig 1. shows the detail block diagram of WEB APPLICATION Login/Register Page makes user authentication. Manages user login, registration, and session management. Generates and manages customized eating regimens. Stores consumer data and authentication credentials. Database stores user profile information and preferences. External service like Food Database API provides nutritional information for various foods.

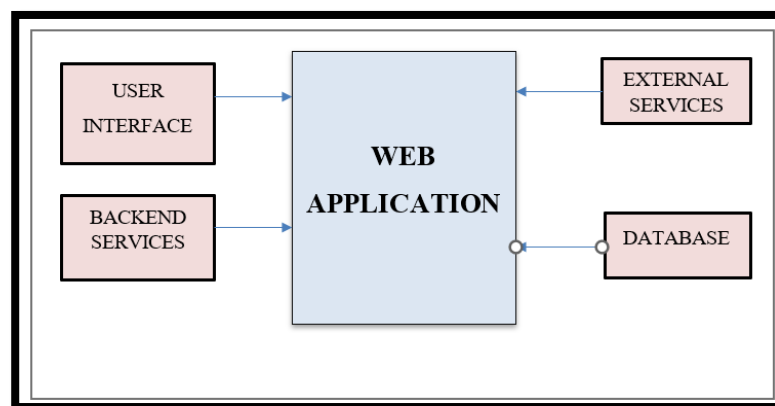


Fig. 1. Block diagram

➤ **User interface**

The user interface is the part of the web application with which users interact. It includes all the elements that users can interact with, such as buttons, forms, menus, and other visual components.

Work: The UI sends client inputs to the web application and shows reactions from the web application to the client.

➤ **Web_application_architecture**

The web application is the core component that processes requests from the user interface and backend services. It contains the business logic that governs how data is processed and how the application behaves shown in fig 2

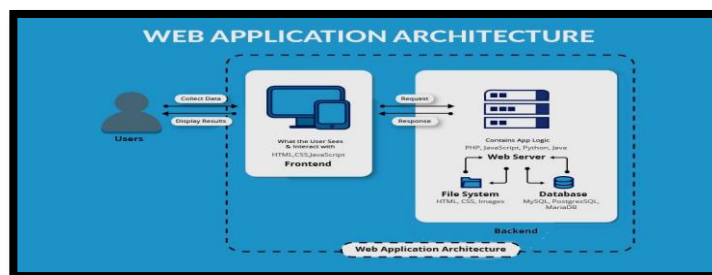


Fig. 2. Web Application Architecture

The architecture of a web application designed for symptom tracking and personalized diet plans consists of several key components, including the client-side (frontend), server-side (backend), database, and external services.

➤ **Client-Side (Frontend)**

The client-side is the interface through which users interact with the application. This is typically accessed via a web browser. The frontend includes several key pages and components:

- **Login/Register Page:** This page allows users to authenticate themselves by either logging in with existing credentials or registering as new users. It handles user authentication and ensures that only authorized users can access their personalized data.
- **Symptom Tracker:** This feature allows users to input their symptoms, view their symptom history, and analyze trends over time. Users can log various symptoms they experience, and the system tracks these logs to help identify patterns or triggers.
- **Diet Plan:** Based on the user's health data and preferences, this component provides personalized diet recommendations. Users can view and edit their diet plans, ensuring the recommendations are tailored to their needs and preferences.
- **Profile Management:** Users are able to modify their personal details and preferences through the profile management section. This includes updating contact information, changing password, and setting dietary preferences or health goals.

➤ **Backend services**

Backend services are services that the web operation interacts with to perform various tasks. These could include authentication services, payment gateways, data processing services, and more. Backend services support the web operation's functionality by furnishing necessary data, performing calculations, or easing relations with other systems. The garçon- side, or backend, is responsible for recycling the sense and managing the data of the operation. It handles requests from the frontend and communicates with the database and external services. The backend includes several services

- **Authentication Service** This service manages stoner login, enrollment, and session management. It ensures that stoner data is secure and that only authenticated drug-gies can pierce the operation.

- **Symptom Management Service** This service handles the storehouse, reclamation, and analysis of symptom data. It allows druggies to log symptoms, recoup their symptom history, and perform analyses to identify trends.
- **Diet Plan Service** This service generates and manages substantiated diet plans grounded on stoner data and preferences. It uses algorithms to recommend suitable diets and allows users to customize their plans.
- **Profile Management Service** This service manages stoner profile data and preferences. It allows druggies to modernize their particular information and acclimate their settings according to their requirements.

Database

- The database is the storehouse subcaste where all the operation data is persisted. It includes several crucial datasets
 - **Storage-**This dataset stores stoner information, including authentication credentials and particular details.
 - **Symptom Data** This dataset stores the symptoms logged by druggies, along with affiliated metadata similar as timestamps and inflexibility situations.
 - **Diet Plan-** This dataset stores the substantiated diet plans for each stoner, including the specifics of the recommended foods and salutary schedules.
 - **Profile** -This dataset stores the profile information and preferences of each stoner.
- Different stages used in this work

3.1.1 Different stages used in this work

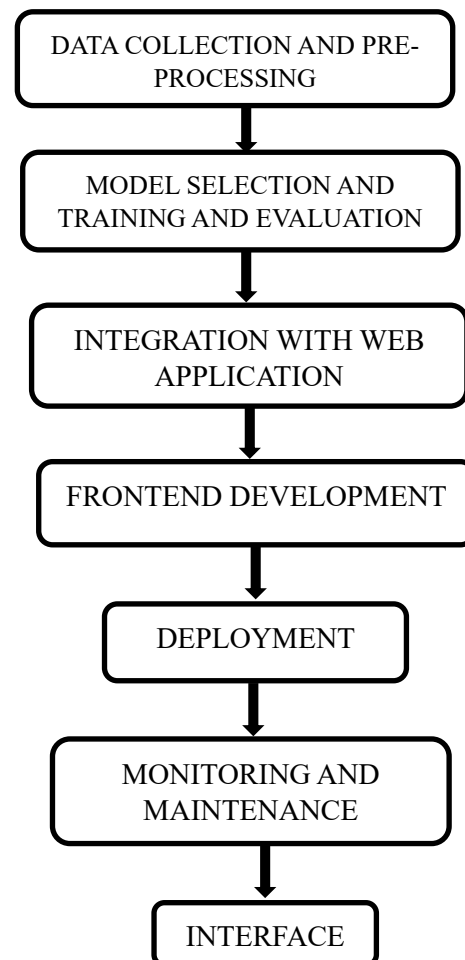


Fig. 2. Flow chart of proposed work

3.1.1 Data collection and preprocessing

In this stage, colorful websites are surveyed to collect the needed data set for the probing work. The data set attained has an aggregate. The collected data undergoes several preprocessing ways to ameliorate the delicacy of the models. These ways include feature selection, data cleaning, data balancing, removing NaN values, etc.

3.1.2 Model selection

In this work 5 models are been used

- i. Logistic Regression: A system for Discrete Probability Modelling events with multiple input variables. Combination of the maximum notorious double issues. A logistic regression model is either real or double false, yes or no, etc. Popular models for double problems logistic retrogression bracket. It helps to prognosticate the probability control will fall into one of two groups. Real conversion characteristic returns with a probability ranging from 0 and 1 is a central element retrogression.
- ii. Random Forest: Leo Breiman and Adele Cutler are the trademark holders of this widely used machine literacy algorithm, which summations the affair of several decision trees to generate a single result. Considering that it can handle both retrogression and classification problems, its inflexibility and ease of use have increased its relinquishment. Ensemble literacy approaches that are robust and flexible are machine literacy ways like Random Forest.
- iii. Decision Tree: In a decision tree, every internal knot represents a “test” on an attribute (similar as whether a coinflips heads or tails), every branch represents the test’s result, and every splint knot represents a class marker (the choice made after calculating all attributes). The form resembles a flowchart. Class guidelines are represented by the paths from the root to the splint. Decision trees represent a group of machine literacy ways that are constantly applied to both retrogression and bracket applications. It's a tree- suchlike graphic illustration of a decision- making process, where each internal knot represents a choice or test on a specific point, each branch represents the test’s outgrowth, and every splint knot represents the veritably last selection or vaticination. Decision trees are useful for numerous tasks, similar as diagnosing medical conditions and soothsaying client waste. likewise, they're fluently interpreted.
- iv. Kneighbors Classifiers: Simple but effective supervised machine learning K-neighbors (K-NN) algorithm for regression and issues with classification. It is a non-differential data classification method depending on the number of adjacent neighbors in the feature space. In K-NN, “K” indicates the number of data points to consider; so far hyperparameters are to be modified depending on the particular problem at hand. Algorithms of ten use interval measures for Euclidean interval to Determine the proximity of the data point.
- v. Gradient Boosting: It is a powerful machine-learning technique for applications involving both regression and classification. It works by gradually combining inadequate predictive models, usually decision trees, to establish a strong ensemble model. The basic idea behind gradient boosting is to minimize the errors of the previous model in the ensemble by fitting each new model to the residual errors of the combined ensemble. In order to accomplish these weights of each weak model are changed, and the best fit is then found through gradient descent optimization. Popular gradient boosting implementations include the algorithms AdaBoost, XGBoost, LightGBM, and CatBoost, each has unique benefits and variations.

3.1.3 Model training and evaluation

The information is split into a group of training and testing set. Utilizing the training set allows to training set is used to train the models, and the testing set is used for their performance. Comparison between the results of the various algorithms employed in the testing and training process based on both accuracy & time performance.

3.1.4 Integration with web application

Embedding the trained model into a web application. This involves setting up a server (e.g., using Flask or Django) that can handle requests and run the model to provide predictions or insights depending on user input.

3.1.5 Frontend development

Creating the user interface of the web application. This involves designing and implementing the layout, forms, buttons, and visual elements using HTML, CSS, JavaScript, and frontend frameworks like React or Angular.

3.1.6 Deployment

Making the web application available to users. This involves setting up a hosting environment, configuring servers, deploying the code, and ensuring the application is accessible over the internet.

3.1.7 Monitoring and maintenance

Continuously tracking the performance of the application, including its uptime, response times, and error rates.

Regularly updating the application to fix bugs, improve performance, and add new features. This also includes retraining the model with new data if necessary.

3.1.8 Interface

Ensuring that the user interface is user-friendly and provides a seamless experience for the end-users. This may involve user testing, gathering feedback, and making iterative improvements to the interface.

3.2 Algorithm

- **Data Collection and Preprocessing**
 - **Collect Data:** Gather relevant datasets from various sources.
 - **Clean Data:** Perform data cleaning to handle missing values and remove outliers.
 - **Normalize Data:** Normalize the data to ensure consistency and comparability.
 - **Feature Extraction:** Extract and select relevant features for the model.
- **Model Selection and Training**
 - **Select Models:** Choose appropriate machine learning models based on the problem.
 - **Validate Models:** Validate the models to ensure they generalize well to unseen data.
 - **Evaluate Performance:** Use performance metrics to evaluate the trained models.
 - **Select Best Model:** Choose the model with the best performance for deployment.
- **Embed Model:** Integrate the selected model into a web application framework.
- **Design UI:** Develop the user interface for the web application.
- **Implement Client-Side Functionality:** Create the client-side functionality to facilitate user interactions.
- **Deploy Application:** Deploy the web application, ensuring proper configuration of all components.
- **Monitor Application:** Continuously monitor the application for performance issues and errors.
- **Maintain Application:** Regularly update and maintain the application to keep it functional and secure.
- **Facilitate User Interaction:** Ensure the user interface is intuitive and user-friendly.
- **Collect Feedback:** Gather user feedback and performance data.
- **Refine Application:** Use the collected feedback to refine and improve the application over time. This algorithm outlines the process from data collection to user interaction, ensuring a structured approach to developing and maintaining a machine learning-powered

4 RESULT

LOGISTIC REGRESSION

```
LOGISTIC REGRESSION

[ ] classifier=LogisticRegression()

[ ] classifier.fit(x_train, y_train)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
LogisticRegression
LogisticRegression()

[ ] predicted=classifier.predict(x_test)

[ ] predicted
array([3, 4, 4, ..., 1, 5, 1])

[ ] from sklearn.metrics import accuracy_score

[ ] accuracy_score(y_test, predicted)
0.6
```

Fig. 3 Logistic Regression Accuracy

The above fig 3 refers to Logistic regression is a statistical technique employed for binary classification tasks, where the dependent variable has two possible outcomes, often represented as 0 and 1. The model calculates the likelihood that a given input point falls into one of these two categories

RANDOM FOREST

```
RANDOM FOREST

[ ] from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=40)
model.fit(x_train, y_train)

RandomForestClassifier
RandomForestClassifier(n_estimators=40)

[ ] model.score(x_test, y_test)
1.0

[ ] y_predicted=model.predict(x_test)
print(y_predicted)
[4 1 4 ... 1 5 1]

from sklearn.metrics import pair_confusion_matrix
cm=confusion_matrix(y_test, y_predicted)
cm
array([[ 45,   0,   0,   0,   0,   0],
       [  0, 378,   0,   0,   0,   0],
       [  0,   0, 221,   0,   0,   0],
       [  0,   0,   0, 219,   0,   0],
       [  0,   0,   0,   0, 227,   0],
       [  0,   0,   0,   0,   0, 230]])

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(10,5))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
Text(95.72222222222221, 0.5, 'Truth')
```

Fig. 4 Random Forest accuracy

The above fig 4 shows that to ensure the model generalizes effectively, it's crucial to test it on a separate dataset. Also, verify that there is no data leakage and that the dataset isn't overly simplistic, as these can also lead to perfect accuracy. Random forests are robust because of their ensemble structure, randomization, and use of bagging, which typically enhance accuracy and stability.

DECISION TREE

```
DECISION TREE

[ ] from sklearn.tree import DecisionTreeClassifier
    tree = DecisionTreeClassifier(random_state=2023)

[ ] X_train, X_test, y_train, y_test = train_test_split(data.drop(columns=['Target']), data['Target'],
    test_size=0.2, random_state=1984)

[ ] tree.fit(X_train, y_train)

+ DecisionTreeClassifier
DecisionTreeClassifier(random_state=2023)

[ ] accuracy_decision_tree = tree.score(X_test,y_test)

[ ] print("the accuracy of decision tree is equal to: " + str(accuracy_decision_tree))
the accuracy of decision tree is equal to: 1.0

[ ] tree.score(X_train,y_train)
1.0
```

Fig. 5 Decision Tree accuracy

The above fig 5 shows machine learning. It divides data into subsets based on feature values, forming a tree structure of decisions. The process starts with a root node that represents the entire dataset and splits into branches that reflect different feature values. These branches lead to internal nodes and, ultimately, to leaf nodes where final predictions are made.

GRADIENT BOOSTING

```
GRADIENT BOOSTING

▶ from sklearn.ensemble import GradientBoostingClassifier
  gbtree = GradientBoostingClassifier(learning_rate=0.1,
    max_depth=1, random_state=1984)

  gbtree.fit(X_train, y_train)
  gbtree_score_test=gbtree.score(X_test,y_test)
  gbtree_score_train=gbtree.score(X_train,y_train)

  print("score of gntree in training data is " + str(gbtree_score_train))
  print("score of gbtree in testing data is " + str(gbtree_score_test))

↩ score of gntree in training data is 0.675
  score of gbtree in testing data is 0.6454545454545455
```

Fig. 6 Gradient Boosting Accuracy

The above fig 6 Gradient boosting is an ensemble learning approach that strengthens prediction accuracy by integrating several weak models. It develops these models one after another, with each new model aimed at correcting the mistakes made by the previous ones.

K-Neighbors Classifier

A screenshot of a Jupyter Notebook titled "KNEIGHBORS CLASSIFIER". It contains five code cells. The first cell imports KNeighborsClassifier from sklearn.neighbors and creates an instance KMN with n_neighbors=1. The second cell calls KMN.fit(X_train, y_train). The third cell displays the output of the fit method, which is a KNeighborsClassifier object. The fourth cell imports warnings and filters them to ignore. The fifth cell calls KMN.score(X_test, y_test), which returns 1.0.

```
KNEIGHBORS CLASSIFIER

[60] from sklearn.neighbors import KNeighborsClassifier # KNeighborsClassifier is the KNN machine learning algorithm
      KMN = KNeighborsClassifier(n_neighbors=1)

[61] KMN.fit(X_train, y_train)

KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)

import warnings
warnings.filterwarnings('ignore')

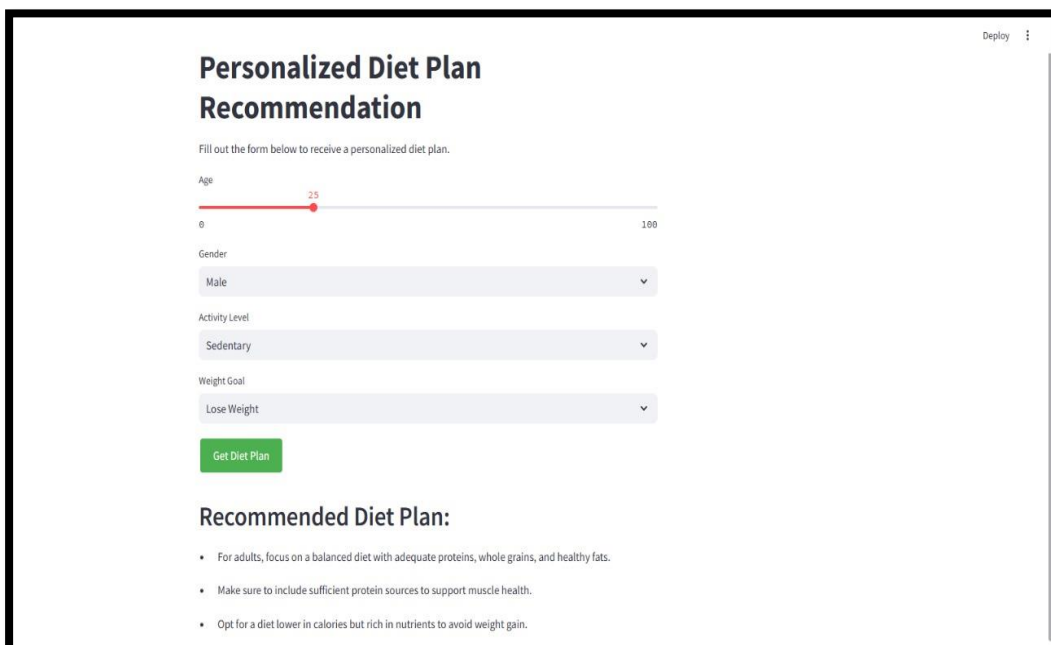
[65] KMN.score(X_test, y_test)

1.0
```

Fig. 7 K – Neighbors Classifiers Accuracy

The above fig 7. k-nearest neighbors (KNN) classifier is a straightforward algorithm that assigns a class to a data point based on the majority class of its k nearest neighbors in the feature space. It works by evaluating the proximity of data points and making predictions based on the classes of the closest examples.

PERSONALISED DIET PLAN RECOMMENDATION

A screenshot of a web interface titled "Personalized Diet Plan Recommendation". It features a form with fields for Age (a slider set to 25), Gender (a dropdown menu set to Male), Activity Level (a dropdown menu set to Sedentary), and Weight Goal (a dropdown menu set to Lose Weight). Below the form is a green "Get Diet Plan" button. Underneath the button, the text "Recommended Diet Plan:" is followed by a list of three bullet points: "For adults, focus on a balanced diet with adequate proteins, whole grains, and healthy fats.", "Make sure to include sufficient protein sources to support muscle health.", and "Opt for a diet lower in calories but rich in nutrients to avoid weight gain." A "Deploy" button is visible in the top right corner.

Personalized Diet Plan Recommendation

Fill out the form below to receive a personalized diet plan.

Age: 25 (Slider from 0 to 100)

Gender: Male (Dropdown)

Activity Level: Sedentary (Dropdown)

Weight Goal: Lose Weight (Dropdown)

Get Diet Plan

Recommended Diet Plan:

- For adults, focus on a balanced diet with adequate proteins, whole grains, and healthy fats.
- Make sure to include sufficient protein sources to support muscle health.
- Opt for a diet lower in calories but rich in nutrients to avoid weight gain.

Fig. 8 Personalized Diet Plan Recommendation

The fig 8 shows web interface of a diet plan recommendation page which feature an intuitive layout, allowing users to easily input personal details, dietary preferences, and health goals.

RECOMMENDATION VS DISTRIBUTION

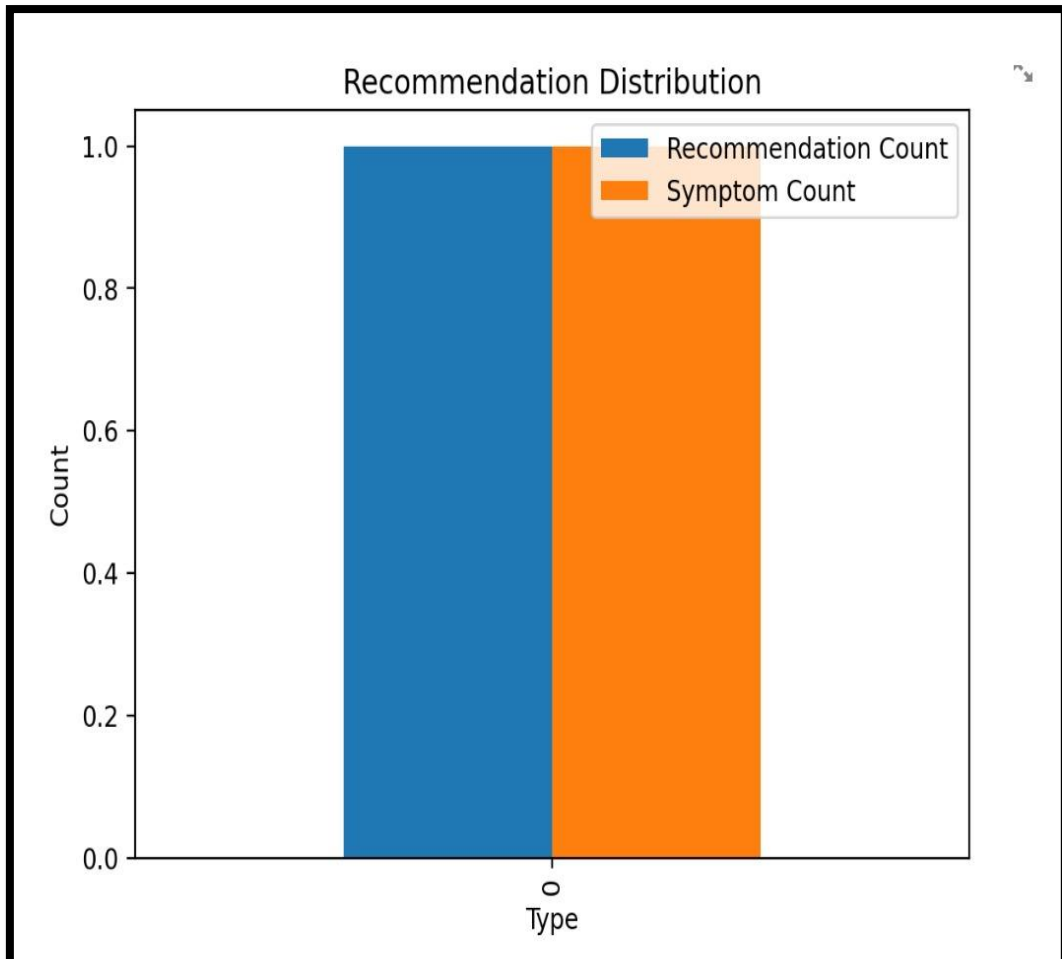


Fig. 9 Type varying Count

The Fig 9. Shows diet plan interface's recommendation section should offer personalized meal suggestions tailored to user input, dietary preferences, and health objectives. A distribution graph can illustrate the nutritional composition of the recommended meals, depicting the ratios of macronutrients such as proteins, carbohydrates, and fats, as well as the calorie distribution. This visual aid helps users comprehend and adhere to their diet plans more effectively.

5 Conclusion

The diet planning app based on dosha types and symptoms offers a unique approach to personalized health management by integrating Ayurvedic principles with modern technology. This app can be a valuable tool for individuals seeking tailored dietary recommendations, preventive healthcare measures, and holistic wellness advice. By focusing on the user's specific symptoms and dosha type, the app provides customized diet plans that aim to improve overall health and well-being

1. Personalized Recommendations: The app provides highly individualized diet plans based on personal health data and dosha types, promoting better health outcomes.
2. Preventive Healthcare: It emphasizes preventive measures, helping users manage and potentially avoid health issues through appropriate dietary choices.
3. Holistic Approach: By combining diet plans with lifestyle and wellness tips, the app offers a comprehensive approach to health management.
4. Wide Applicability: It can be used for various purposes, including chronic disease management, weight management, corporate wellness programs, and more.

6 Reference

- [1] Jones, D. W. (2017). School based prevention and management of childhood overweight and obesity. *Journal of Obesity & Weight Loss Therapy*, 07(06). <https://doi.org/10.4172/2165-7904-c1-051>
- [2] Sonderegger, S., & Seufert, S. (2022). Chatbot-mediated Learning: Conceptual Framework for the Design of Chatbot Use Cases in Education. *Proceedings of the 14th International Conference on Computer Supported Education*. <https://doi.org/10.5220/0010999200003182>
- [3] Nutrition Information for Healthy Weight Management. [dataset]. In *PsycEXTRA Dataset*. American Psychological Association (APA). <https://doi.org/10.1037/e314442004-002>
- [4] Shahapur, S. S., Chitti, P., Patil, S., Nerurkar, C. A., Shivannagol, V. S., Rayanaikar, V. C., ... & Betageri, V. (2024). Decoding Minds: Estimation of Stress Level in Students using Machine Learning. *Indian Journal of Science and Technology* <https://doi.org/10.17485/IJST/v17i19.2951>.