# A MINI PROJECT REPORT

## On

**A NOVEL TIME-AWARE FOOD RECOMMENDER-SYSTEM BASED ON DEEP LEARNING AND GRAPH CLUSTERING**

*Submitted by,*

## A.SAI KRISHNA 20J41A05C2

## MD.ASIF 20J41A05F7

## M.SRINIDHI 20J41A05F9

## P.VISHNUSAI 20J41A05G5

## E.VENKATASAI 20J41A05D8

*in partial fulfillment of the requirements for the award of the degree of*

# BACHELOR OF TECHNOLOGY

in

# COMPUTER SCIENCE AND ENGINEERING

Under the Guidance of

# Dr. A. RAMASWAMI REDDY

Professor, Computer Science and Engineering

# COMPUTER SCIENCE AND ENGINEERING MALLA REDDY ENGINEERING COLLEGE

(An UGC Autonomous Institution, Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad) Maisammaguda, Secunderabad, Telangana, India 500100

# AUGUST - 2023

**MALLA REDDY ENGINEERING COLLEGE**

Maisammaguda, Secunderabad, Telangana, India 500100



**BONAFIDE CERTIFICATE**

This is to certify that this minor project work entitled “**A NOVEL TIME AWARE FOOD RECOMMENDER SYSTEM BASED ON DEEP LEARNING AND GRAPH CLUSTERING**”, submitted by **A.SAI KRISHNA(20J41A05C2),MD.ASIF(20J41A05F7),M.SRINIDHI(20J41A05F9),G.VISHNUSAI(20J41A05G5), E.VENKATASAI(20J41A05D8)** to Malla Reddy Engineering College affiliated to JNTUH, Hyderabad in partial fulfillment for the award of **Bachelor of Technology** in COMPUTER SCIENCE AND ENGINEERING is a *bonafide* record of project work carried out under my/our supervision during the academic year 2023 – 2024 and that this work has not been submitted elsewhere for a degree.

|  |  |
| --- | --- |
| **SIGNATURE**  **DR.A. RAMASWAMI REDDY SUPERVISOR, PRINCIPAL**  CSE  Malla Reddy Engineering College  Secunderabad, 500 100 | **SIGNATURE**  **DR.P.SRIRAMA CHANDRA MURTHY, HOD**  CSE  Malla Reddy Engineering College Secunderabad, 500 100 |
|  |  |

#### Submitted for Mini Project viva-voce examination held on

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**MALLA REDDY ENGINEERING COLLEGE**

Maisammaguda, Secunderabad, Telangana, India 500100

**ACKNOWLEDGEMENT**

We express our sincere thanks to our **Principal**, **Dr. A. Ramaswami Reddy**, who took keen interest and encouraged us in every effort during the project work.

We express our heartfelt thanks to **Dr. P.Sri Rama Chandra Murthy**, **Professor and Head**, Department of Computer Science and Engineering, for his kind attention and valuable guidance throughout the project work.

We are thankful to our Project Coordinator **Mr. K. Srikanth, Assistant Professor**, Department of Computer Science and Engineering, for his cooperation during the project work.

We are extremely thankful to our Project Guide **Ms. Rasagnya**, **Assistant professor**, for her constant guidance and support to complete the project work.

We also thank all the teaching and non-teaching staff of Department for their cooperation during the project work.

## SAI KRISHNA 20J41A05C2

## ASIF 20J41A05F7

## SRINIDHI 20J41A05F9

## VISHNUSAI 20J41A05G5

## VENKATASAI 20J41A05D8

**ABSTRACT**

Food recommender-systems are considered an effective tool to help users adjust their eating habits and achieve a healthier diet. This paper aims to develop a new hybrid food recommender-system to overcome the shortcomings of previous systems, such as ignoring food ingredients, time factor, cold start users, cold start food items and community aspects. The proposed method involves two phases: food content-based recommendation and user-based recommendation. Graph clustering is used in the first phase, and a deep-learning based approach is used in the second phase to cluster both users and food items. Besides a holistic-like approach is employed to account for time and user-community related issues in a way that improves the quality of the recommendation provided to the user. We compared our model with a set of state-of-the-art recommender-systems using ve distinct performance metrics: Precision, Recall, F1, AUC and NDCG. Experiments using dataset extracted from ``Allrecipes.com'' demonstrated that the developed food recommender-system performed best.

***Keywords*:** Recommender-system, food recommendation, healthcare, deep learning, graph clustering

|  |  |
| --- | --- |
| **ABSTRACT** | **iv** |
| **DESCRIPTION** | **PAGE NO.** |
| **1. INTRODUCTION**   * 1. Drawbacks of Previous Systems   2. Benefits of Food Recommendation System | 1 - 5  2-3  4 |
| **2. LITERATURE SURVEY** | 6 - 7 |
| **3.SYSTEM ANALYSIS**   * 1. Existing System   2. Existing System Disadvantages   3. Proposed System   4. Proposed System Advantages | 7-9  7  8  8  9 |
| **4.SYSTEM STUDY**   * 1. Economic Feasibility   2. Technical Feasibility   3. Social Feasibility | 10  10  10  10 |
| **5.SYSTEM DESIGN**   * 1. System Architecture   2. Data Flow Diagram   3. UML Diagrams      1. Use Case Diagram      2. Class Diagram      3. Sequence Diagram      4. Activity Diagram | 11-19  11  12-13  13-19  14-15  16  17  18-19 |
| **6.IMPLEMENTATION**   * 1. Modules      1. Service Provider Module      2. View and Authorize Module      3. Remote User Module | 20  20  20  20  20 |
| **7. System Requirements**  7.1 Hardware Requirements  7.2 Software Requrements | 23  23  23 |
| **8. Software Enviroments**  8.1 Python  8.2 History of Python  8.3 Python Features  8.4 Python Operators  8.7 Python Functions  8.8 Exceptions  8.9 Database API | 24-53  24  24  24  26-28  29-36  36  36-39  39 |
| **9. System Testing**  9.1 Types of Test  9.1.1 unit Testing  9.1.2 Integration Testing  9.1.3 Acceptence Testing | 39-46 |
| **10. Sample Output** | 47-52 |
| **11. Conclusion** | 53 |
| **12 . References** | 54-55 |

#### 

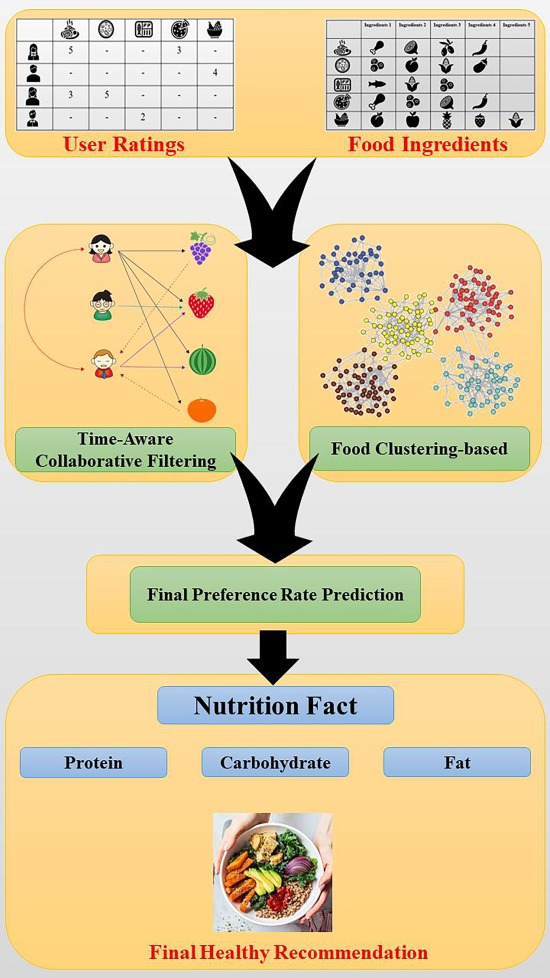
**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO.** | **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 1 | 1.1 | Food Recommendation | 2 |
| 2 | 1.2 | Benefits of Food Recommendation  System | 5 |
| 3 | 5.1 | System Architecture | 11 |
| 4 | 5.2 | Data Flow Diagram | 13 |
| 5 | 5.3 | Use Case Diagram | 15 |
| 6 | 5.4 | Class Diagram | 16 |
| 7 | 5.5 | Sequence Diagram | 17 |
| 8 | 5.6 | Activity Diagram | 18-19 |
| 9 | 8.1 | Python Execution Sequence | 26 |
| 10 | 8.2 | Python Operators | 26 |
| 11 | 10.1 | Home Screen | 47 |
| 12 | 10.2 | User Login Screen | 48 |
| 13 | 10.3 | User Registration Screen | 49 |
| 14 | 10.4 | Service Provider Login Screen | 50 |
| 15 | 10.5 | Prediction Type Screen | 51 |
| 16 | 10.6 | Prediction Output Type Screen | 51 |
| 17 | 10.7 | Output Bar Graph Screen | 52 |
| 18 | 10.8 | Output Piechart Screen | 52 |

# CHAPTER 1 INTRODUCTION

The internet has become an important part of people's daily lives and used in various tasks, ranging from leisure (i.e., chatting with other users, shopping, searching for hotels, travel deals) to professional development (i.e., using a web platform to develop professional services) . The tremendous amount of information from tens of thousands of sources that can be accessed by a user as part of his/her request creates important uncertainty and ambiguity that can easily divert the user from his original request . Although search engines have attempted to address the problem of redundancy of information in recent decades, they have not been very successful in personalizing search results and reducing the amount of noisy information . Many of these systems return the same results even for users with completely different proles and interests. In recent years, researchers have become more interested in recommender-systems as one of the most successful personalization tools on the web . It can be used to help the user identify the right service, reduce the information overload, guide the user towards some personalized behavior, and find user's favorite items within a large amount of information, among others. In a typical recommender-system, users' interests are discovered and items and services are recommended accordingly. In a variety of lifestyle applications and services, food recommendation plays an important role as a tool assisting users to change behavior and adopt healthy lifestyle . Typically, food recommendation attempt to provide the user with a personalized food recommendation in terms of recipes, scale of change and time required to achieve specific objectives that might be associated with diet requirement or any lifestyle demand . Traditionally, research in food recommendation has seen little attention when compared to recommendation in other leisure and entertainment fields (e.g., music, book, shopping recommendation systems), possibly due to cultural barriers and difficulty to predict what people might like to eat. Although, lifestyle and diet related illnesses, such as obesity and diabetes, account for almost 60% of total deaths .

The process of generating a food recommendation is often viewed as a machine learning task . Therefore,it is crucial to understand user's food preferences accurately to build an effective food recommendation.Even for building health-oriented food services,the user can only be encouraged to pursue a recommendation if the recommended food matches his taste preferences.



**Fig 1.1:** Food Recommendation

### DRAWBACKS OF PREVIOUS SYSTEMS

In recent decades, many recommender-systems have been developed to predict person' preferences and/or guide his choice according to some predefined objectives. Although previous food recommender-systems have shown good performance in learning persons' preferences by mapping user's historical interactions with food items and recipes, these systems still suffer from the following drawbacks

1) Ingredients of foods: Most previous food recommender-systems [29], [30] rely primarily on historical ratings of users to draw upon food recommendations through a collaborative filtering approach that ignores food ingredients. This is due to the observation that a given food is usually preferred by an individual

because it contains ingredients, he/she may like to eat. This may overlook some important aspects in the recommendation. For example, foods containing chicken wings may be a person's favorite food, while he/she may be allergic to some types of spices that can be used during the food preparation. Therefore, collaborative filtering recommender-systems may not be enough to account for such user's preferences and constraints.

2) Time factor: Traditional recommender-systems [19], [26][28] are based on the premise that users with similar preferences in the past will have similar tastes in the future. Accordingly, these recommender-systems use static data and ignore potential changes in user's food preferences, diet or life style that can occur over

time in realistic scenarios.

3) Cold start users and cold start foods: Due to the fact that users often rate just a few foods, traditional collaborative filtering-based food recommender systems

have difficulty recognizing active user neighbors or similar foods. Accordingly, collaborative filtering-based food recommendation are only able to suggest foods to users who have rated enough foods. Cold start users, who have rated only few food items, are thereby ignored. Similarly, new food items (food cold start) that have not attracted yet enough ratings from users are ignored as well by such a collaborative filtering-based approach.

4) Users' community: Another issue, which is again ignored in existing recommender-systems, is the user's neighborhood or community aspect. Intuitively, community aspect can be utilized to predict the rating of unseen food item and the success likelihood of a given diet, extrapolating from active users' activities in the neighborhood. Typically, community aspect can be handled using clustering-based models. Nevertheless, it has been shown that such an approach also suffers from several other difficulties as well, which are somehow inherent to clustering techniques employed (e.g., optimal number of clusters, efficiency of similarity measures employed).

**1.2 BENEFITS OF FOOD RECOMMENDATION SYSTEMS**

To address these drawbacks, this paper develops a novel time-aware and food ingredient-based recommendation model that simultaneously tackles all the aforesaid challenges. Our system particularly considers the health factor and the ingredient content of foods. The model is named the Healthy and Time-Aware Food Recommendation System (HTFRS). In short, HTFRS suggests desired and healthy foods to the active user. To predict user-based rates, we first consider the historical ratings of users. Overall, the novelty of our developed food recommendation system can be summarized as below:

**.Healthy Recommendation:** In contrast with many previous food recommendation models, this paper incorporates health and nutrition factors into the food recommendation model so that the system guides users to a healthy eating style. In other words, the purpose of this study is to present the general framework for developing an efficient food recommendation model based on user preference and nutrition factors.

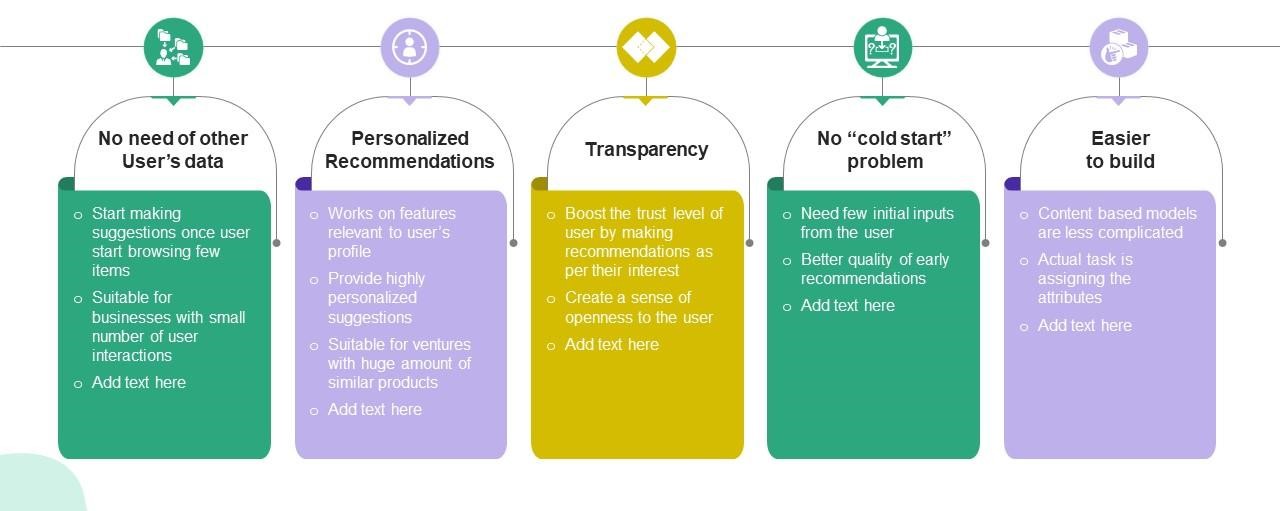
**.Ingredients-Aware:** Unlike classic food recommendation systems that ignore food contents in their recommendations, our model employs historical user ratings as well as food ingredients in the data-processing pipeline of our recommender system.

**.Time-Aware:** In contrast to previous works ([Gao et al., 2019](https://www.sciencedirect.com/science/article/pii/S0957417423002208#b0100), [Gao et al., 2022](https://www.sciencedirect.com/science/article/pii/S0957417423002208#b0105), [Meng et al., 2020](https://www.sciencedirect.com/science/article/pii/S0957417423002208#b0190), [Toledo et al., 2019](https://www.sciencedirect.com/science/article/pii/S0957417423002208#b0275)) that have ignored the time factor, our model introduces a novel time-aware similarity function to capture the temporal information of ratings, which are then explicitly accounted for in calculating the user-similarity scores. In this model, we designed a weighting mechanism to weigh the importance of the time factor for different ratings, where old ratings have a lower importance than new ones. Armed with this time-aware function, we can incorporate the dynamic nature of users’ preferences into the recommendation process.

**.Food Similarity-Based Recommendation:** To our knowledge, this is the first study where recipe information in the food recommendation is represented with an attributed graph. This developed food content-based [similarity matrix](https://www.sciencedirect.com/topics/computer-science/similarity-matrix) enabled us to efficiently overcome the cold-start foods problem of classic food recommendation systems. This matrix could also be used to predict the rating of unrated foods in the system since there could be a high correlation between a food’s category and its ingredients.

**.Food Group-Aware:** Contrary to previous works that have ignored food categories, the developed system in this study explicitly accounts for this aspect. This new system systematically approaches the optimum number of food categories. To the best of our knowledge, this is the first work that accommodates sparse datasets by utilizing a graph-based representation where edge weights are calculated based on food ingredients-based similarities.

**.Controllable Recommendation:** This study develops, for the first time, a novel controllable food recommender system that enables users to participate actively in the recommendation process and to strike a balance between their own preferences and the health factor of the food.



**Fig 1.2:** Benefits of Food Recommendation Systems

Based on this investigation, the following observations can be drawn from the reviewed food recommender-systems:

* Time factor in user rating is ignored.
* With the exception to, user communities are ignored, and the analysis of user-user interaction is rather light.
* Trust relationships among users, when available, are neglected.
* Only few works attempted to integrate user ratings and food content, which is a vital component of any food recommender-system.

# CHAPTER 2

# LITERATURE SURVEY

1. \*"Matrix Factorization Techniques for Recommender Systems"\* by Koren, Y., Bell, R., & Volinsky, C. (2009)

This paper focuses on matrix factorization techniques, which are widely used in collaborative filtering-based recommender systems. Collaborative filtering relies on user-item interaction data to make recommendations. Matrix factorization involves decomposing the user-item interaction matrix into lower-dimensional matrices that represent latent features. This allows the system to capture hidden patterns and similarities between users and items, enabling accurate recommendations.

2. \*"A Survey of Restaurant Recommendation Techniques"\* by Lu, Y., Wang, S., & Zhai, C. (2013)

This survey paper provides an in-depth overview of various techniques used in restaurant recommendation systems. It covers content-based approaches (where recommendations are based on item attributes), collaborative filtering techniques (which use user-item interactions), and hybrid methods (which combine multiple approaches). The paper discusses the strengths and weaknesses of each technique and provides insights into their applications in the restaurant domain.

3. \*"TasteWeights: A Visual Interactive Hybrid Recommender System"\* by Ricci, F., Rokach, L., & Shapira, B. (2011)

TasteWeights is introduced as a hybrid recommendation system that combines collaborative filtering and content-based methods. Collaborative filtering leverages user preferences and similarities, while content-based filtering considers item characteristics. The paper proposes an interactive visualization approach that allows users to adjust the importance of different recommendation aspects, such as popularity or novelty. This personalized weighting enhances the system's recommendations based on individual user preferences.

4. \*"Yelp Help: Language Modeling for Restaurant Review Spam Detection"\* by Rayana, S., Akoglu, L., & Faloutsos, C. (2015)

This paper addresses a critical aspect of recommendation systems in the food domain: fake reviews. Inaccurate reviews can significantly affect the quality of recommendations. The paper discusses techniques for detecting spam and fake reviews using language modeling and text analysis. By identifying and filtering out unreliable reviews, recommendation systems can provide more trustworthy suggestions to users.

5. \*"Improving Recommendation Lists Through Topic Diversification"\* by Vig, J., Sen, S., & Riedl, J. (2012)

This paper delves into the concept of topic diversification in recommendation systems. While not food-specific, the concept is relevant to food recommendation systems aiming to offer users a variety of culinary choices. The paper introduces methods to diversify recommendation lists by considering item characteristics and user preferences. Diversified recommendations help prevent users from being exposed to the same types of items repeatedly.

These papers collectively cover a range of techniques and challenges in the field of food recommendation systems. Remember that my information is based on knowledge up until September 2021, so I recommend searching for more recent research to stay updated on the latest advancements in this area.

# CHAPTER 3 SYSTEM ANALYSIS

### EXISTING SYSTEM

1) *Ingredients of foods:* Most previous food recommender-systems rely primarily on historical ratings of users to draw upon food recommendations through a collaborative filtering approach that ignores food ingredients. This is due to the observation that a given food is usually preferred by an individual because it contains ingredients, he/she may like to eat. This may overlook some important aspects in the recommendation. For example, foods containing chicken wings may be a person's favorite food, while he/she may be allergic to some types of spices that can be used during the food preparation. Therefore, collaborative filtering recommender-systems may not be enough to account for such user's preferences and constraints.

2) *Time factor:* Traditional recommender-systems are based on the premise that users with similar preferences in the past will have similar tastes in the future. Accordingly, these recommender-systems use static data and ignore potential changes in user's food preferences, diet or life style that can occur over time in realistic scenarios.

3) *Cold start users and cold start foods:* Due to the fact that users often rate just a few foods, traditional collaborative filtering-based food recommender systems have difficulty recognizing active user neighbors or similar foods. Accordingly, collaborative Filtering-based food recommendation are only able to suggest foods to users who have rated enough foods. Cold start users, who have rated only few food items, are thereby ignored. Similarly, new food items (food cold start) that have not attracted yet enough ratings from users are ignored as well by such a collaborative

filtering-based approach.

4) *Users' community:* Another issue, which is again ignored in existing recommender-systems, is the user's neighborhood or community aspect. Intuitively, community aspect can be utilized to predict the rating of unseen food item and the success likelihood of a given diet, extrapolating from active users' activities in the neighborhood. Typically, community aspect can be handled using clustering-based models. Nevertheless, it has been shown that such an approach also suffers from several other difficulties as well, which are somehow inherent to clustering techniques employed (e.g., optimal number of clusters, efficiency of similarity measures employed).

### EXISTING SYSTEM DISADVANTAGES

* ­­\_Time factor in user rating is ignored.
* \_With the exception to, user communities are ignored, and the analysis of user-user interaction is rather light.
* \_Trust relationships among users, when available, are neglected.
* \_Only few works attempted to integrate user ratings and food content, which is a vital component of any food recommender-system.

### PROPOSED SYSTEM

1) *Ingredients-aware food recommender-system:* Unlike traditional collaborative-based food recommender systems, our model integrates both collaborative filtering-based model (user-based phase) and content-based model (food-based phase). As a result, a set of foods that both suit the user's preferences and utilize his/her previous ratings are recommended.

2) *Time-aware food recommender-system:* A novel time-aware similarity measure that takes into account changes in food preferences or diet over time is developed in this paper. This makes the proposal suitable to handle cases where users change his/her rating / preferences over time.

3) *Trust-aware food recommender-system:* A trust-aware food recommender-system is developed to overcome the cold start user and cold start foods problems of the traditional collaborative filtering-based food recommender-systems. Our proposed model builds a trust network of users based on trust (follower following)

statements to predict user ratings efficiently.

The trust network generation plays an important role in addressing the neighbor selection problem. Trust statements can be used to predict the rating of unseen items

in food recommender-systems since there is a high correlation between users' trust and user ratings-based similarity measure. The user's trust network and the user ratings-based similarity are integrated in this study to address the data sparsity problem utilizing knowledge that is stored outside of the user's local neighborhood

of similarity.

4) *Community-aware food recommender-system:* Contrary to previous works where users' communities are not considered in the food recommendation process, our model explicitly accounts for such aspects where the optimal number of users' clusters is determined automatically. Moreover, using a graphical like representation

where edge weights are calculated according to user ratings-based similarity and trust network, the proposed method accommodates sparse datasets.

### PROPOSED SYSTEM ADVANTAGES

* The purpose Time-aware food recommender-system based on Deep and Learning and Graph Clustering (TDLGC) and Machine Learning Classifiers.
* User based rating prediction.
* Food-based rating prediction.
* In the first phase,

1. by utilizing both the user rating and the follower-following network, the user- user similarity matrix as well as the users' trust network are generated.Then,
2. based on the user similarities and the trust network, the given user set is mapped onto a weighted graph.

* In the next step,

1. a novel time-aware graph clustering algorithm is proposed to cluster the users into different groups accordingly. Finally,
2. utilizing users' clusters from previous step, user similarity and historical ratings, new user-based ratings are predicted.

# CHAPTER 4 SYSTEM STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

### ECONOMIC FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system is well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

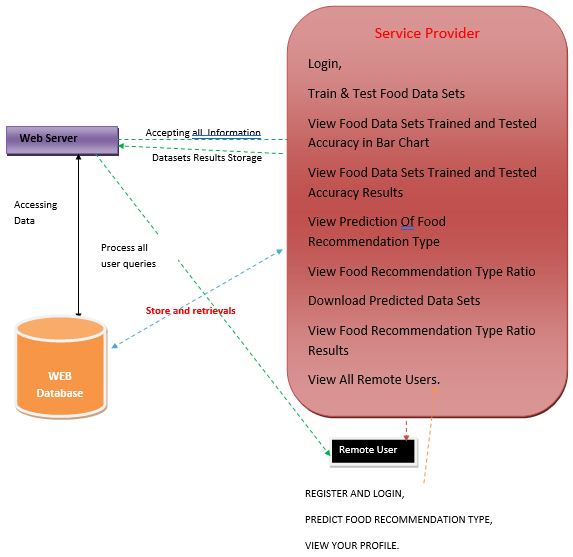
This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand for the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### SOCIAL FEASIBILITY

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# CHAPTER 5 SYSTEM DESIGN

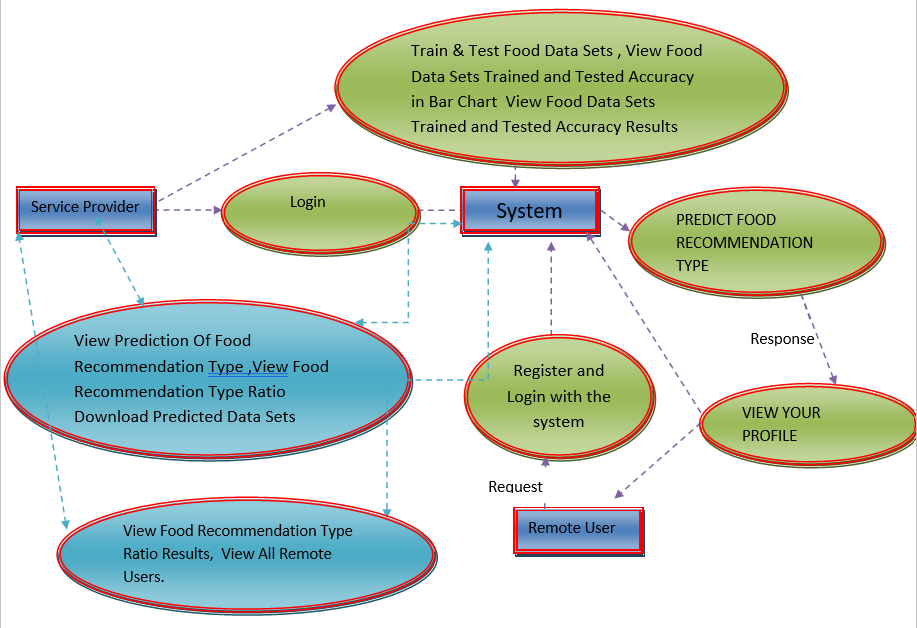
### SYSTEM ARCHITECTURE

****

**Fig 5.1:** System Architecture

### DATA FLOW DIAGRAM

* + 1. The Data Flow Diagram (DFD) is also called a bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data generated by this system.
    2. The DFD is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
    3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
    4. DFD is also known as a bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



**Fig 5.2:** Data Flow Diagram

### UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general- purpose UML modeling language in the field of object-oriented software engineering. The standard is managed and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object- oriented computer software. In its current form, UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing, and documenting the artifacts of a software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

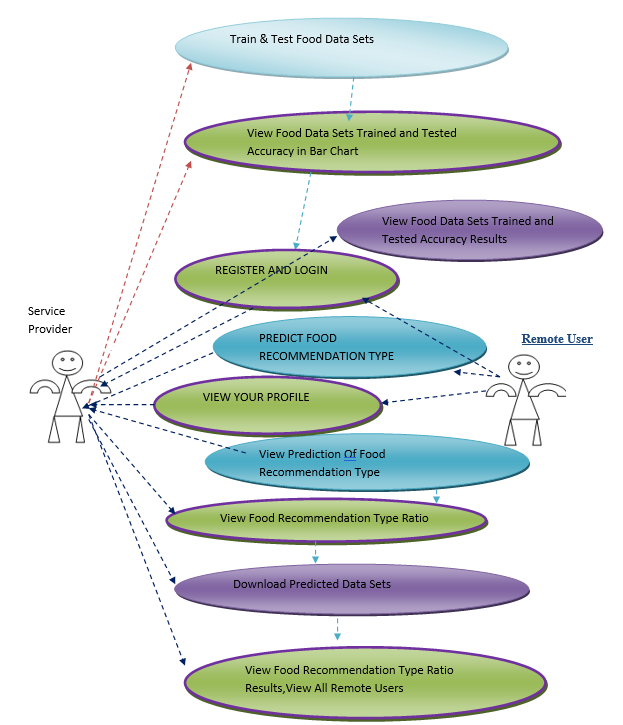
UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

The Primary goals in the design of the UML are as follows:

* + 1. Provide users with a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
    2. Provide extensibility and specialization mechanisms to extend the core concepts.
    3. Be independent of particular programming languages and development processes.
    4. Provide a formal basis for understanding the modeling language.
    5. Encourage the growth of the OO tools market.
    6. Support higher-level development concepts such as collaborations, frameworks, patterns, and components.
    7. Integrate best practices.

### USE CASE DIAGRAM

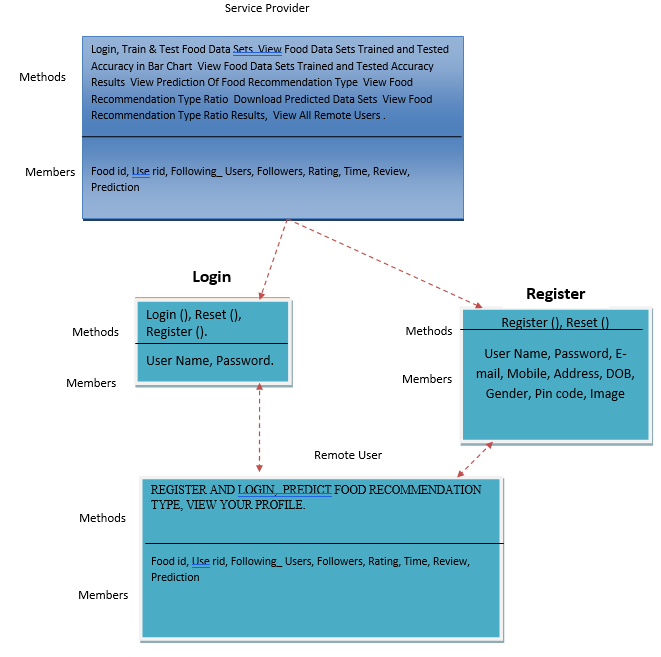
A use-case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. The roles of the actors in the system can be depicted.



**Fig 5.3:** Use Case Diagram

### CLASS DIAGRAM

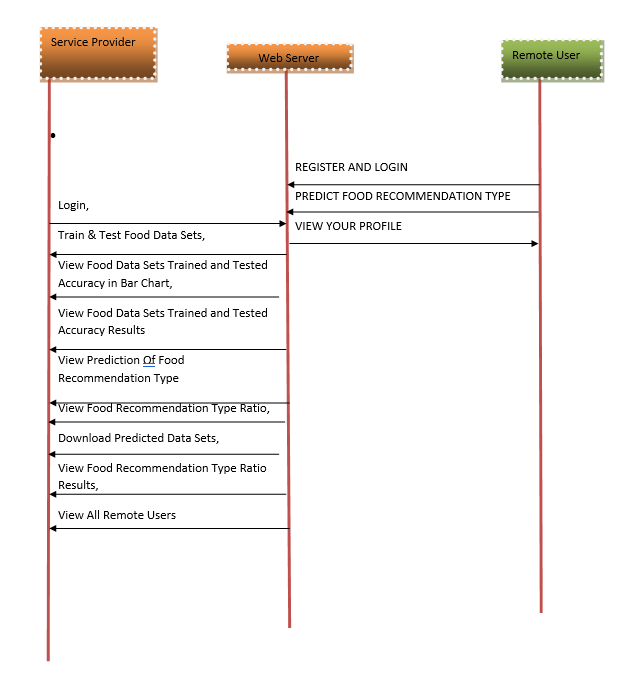
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**Fig 5.4:** Class Diagram

### SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



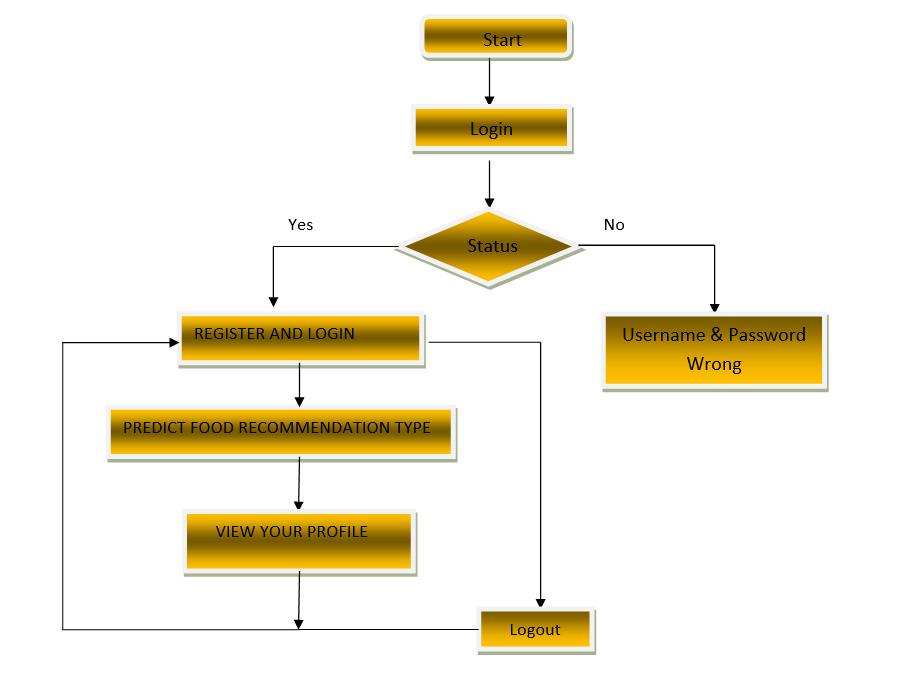
**Fig 5.5:** Sequence Diagram

### ACTIVITY DIAGRAM

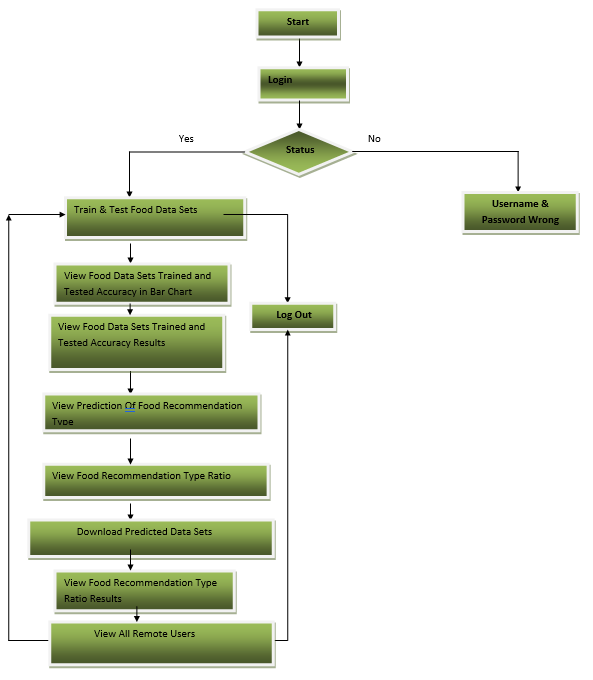
Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by- step workflows of components in a system. An activity diagram shows the overall flow of

control.

* **Flow Chart : Remote User**



* **Flow Chart :** **Service Provider**



**Fig 5.6:** Activity Diagram

### CHAPTER 6

### IMPLEMENTATION

### 6.1 MODULES

* + - Service Provider
    - View and Authorize Users
    - Remote User

**6.1.1 Service Provider**

In this module, the Service Provider has to login by using valid user name and password. After login successful he can do some operations such as

Login, Train & Test Food Data Sets, View Food Data Sets Trained and Tested Accuracy in Bar Chart, View Food Data Sets Trained and Tested Accuracy Results , View Prediction Of Food Recommendation ,Type View Food Recommendation Type Ratio, Download Predicted Data Sets, View Food Recommendation Type Ratio Results, View All Remote Users.

**6.1.2 View and Authorize Users**

In this module, the admin can view the list of users who all registered. In this, the admin can view the user’s details such as, user name, email, address and admin authorizes the users.

**6.1.3Remote User**

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like REGISTER AND LOGIN, PREDICT FOOD RECOMMENDATION TYPE,VIEW YOUR PROFILE.

**ALGORITHMS**

Decision tree classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability

of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from

training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes

C1, C2, …, Ck is as follows:

Step 1. If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf

labeled with this class

Step 2. Otherwise, let T be some test with possible outcomes O1, O2,…, On. Each object in S has one outcome

for T so the test partitions S into subsets S1, S2,… Sn where each object in Si has outcome Oi for T. T becomes

the root of the decision tree and for each outcome Oi we build a subsidiary decision tree by invoking the same

procedure recursively on the set Si

Decision tree classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability

of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from

training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes

C1, C2, …, Ck is as follows:

Step 1. If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf

labeled with this class

Step 2. Otherwise, let T be some test with possible outcomes O1, O2,…, On. Each object in S has one outcome

for T so the test partitions S into subsets S1, S2,… Sn where each object in Si has outcome Oi for T. T becomes

the root of the decision tree and for each outcome Oi we build a subsidiary decision tree by invoking the same

procedure recursively on the set Si

**Decision tree classifiers**

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C1, C2, …, Ck is as follows: Step 1. If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf labeled with this class Step 2. Otherwise, let T be some test with possible outcomes O1, O2,…, On. Each object in S has one outcome for T so the test partitions S into subsets S1, S2,… Sn where each object in Si has outcome Oi for T. T becomes the root of the decision tree and for each outcome Oi we build a subsidiary decision tree by invoking the same procedure recursively on the set Si.

**Gradient boosting**

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest.A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.grandient boosting plays an important key role in the machine learning. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.grandient boosting plays an important key role in the machine learning.

**K-Nearest Neighbors (KNN)**

Simple, but a very powerful classification algorithm

* Classifies based on a similarity measure
* Non-parametric
* Lazy learning
* Does not “learn” until the test example is given
* Whenever we have a new data to classify, we find its K-nearest neighbors from the training data Example
* Training dataset consists of k-closest examples in feature space
* Feature space means, space with categorization variables (non-metric variables)
* Learning based on instances, and thus also works lazily because instance close to the input vector for test or prediction may take time to occur in the training dataset.

**Logistic regression Classifiers**

Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name logistic regression is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name multinomial logistic regression is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar. Logistic regression competes with discriminant analysis as a method for analyzing categorical-response variables. Many statisticians feel that logistic regression is more versatile and better suited for modeling most situations than is discriminant analysis. This is because logistic regression does not assume that the independent variables are normally distributed, as discriminant analysis does. This program computes binary logistic regression and multinomial logistic regression on both numeric and categorical independent variables. It reports on the regression equation as well as the goodness of fit, odds ratios, confidence limits, likelihood, and deviance. It performs a comprehensive residual analysis including diagnostic residual reports and plots. It can perform an independent variable subset selection search, looking for the best regression model with the fewest independent variables.

Naïve Bayes

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes

that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other

feature .

Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning

techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based

on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis,

logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the

parameters of the classifier (the learning bias).

While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners

which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program

and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy

is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model

easy to interpret and deploy, they does not understand the interest of such a technique.

Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to

understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical

aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare

the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic

regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This

largely explains the good performance of the method in comparison to others. In the second part, we use various

tools on the same dataset (Weka 3.6.0, R 2.9.2, Knime 2.1.1, Orange 2.0b and RapidMiner 4.6.0). We try above

all to understand the obtained results.

**Naïve Bayes**

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature . Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).

While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model easy to interpret and deploy, they does not understand the interest of such a technique.

Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This largely explains the good performance of the method in comparison to others. In the second part, we use various tools on the same dataset (Weka 3.6.0, R 2.9.2, Knime 2.1.1, Orange 2.0b and RapidMiner 4.6.0). We try above all to understand the obtained results.

**SVM**

In classification tasks a discriminant machine learning technique aims at finding, based on an independent and identically distributed (iid) training dataset, a discriminant function that can correctly predict labels for newly acquired instances. Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point x and assigns it to one of the different classes that are a part of the classification task. Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed. From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space.

# CHAPTER 7

# SYSTEM REQUIREMENTS

### HARDWARE REQUIREMENTS

➢ Processor - Pentium –IV

➢ RAM - 4 GB (min)

➢ Hard Disk - 20 GB

➢ Key Board - Standard Windows Keyboard

➢ Mouse - Two or Three Button Mouse

➢ Monitor - SVGA

### SOFTWARE REQUIREMENTS

* **Operating system :** MicrosoftWindows 7 Ultimate.
* **Coding Language :** Python.
* **Front-End :** Python.
* **Back-End :** Django-ORM
* **Designing :** Html, css, javascript.
* **Data Base :** MySQL (WAMP Server).

Decision tree classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability

of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from

training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes

C1, C2, …, Ck is as follows:

Step 1. If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf

labeled with this class

Step 2. Otherwise, let T be some test with possible outcomes O1, O2,…, On. Each object in S has one outcome

for T so the test partitions S into subsets S1, S2,… Sn where each object in Si has outcome Oi for T. T becomes

the root of the decision tree and for each outcome Oi we build a subsidiary decision tree by invoking the same

procedure recursively on the set Si

# CHAPTER 8 SOFTWARE ENVIRONMENT

**8.1 PYTHON**

Python is a **high-level, interpreted**, **interactive** and **object-oriented scripting** **language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## **8.2 History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

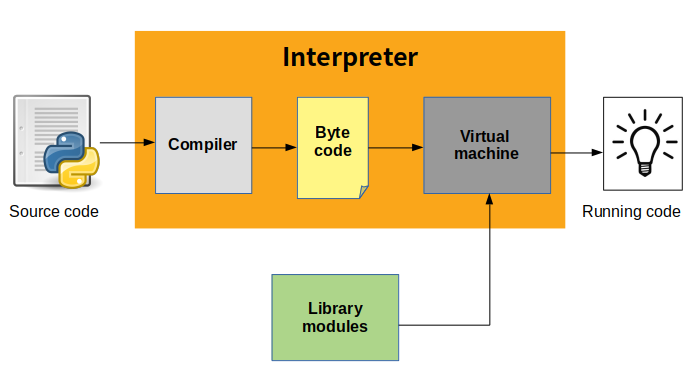
## **8.3 Python Features**

Python's features include:

* **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
* **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases:** Python provides interfaces to all major commercial databases.
* **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable:** Python provides a better structure and support for large programs than shell scripting.

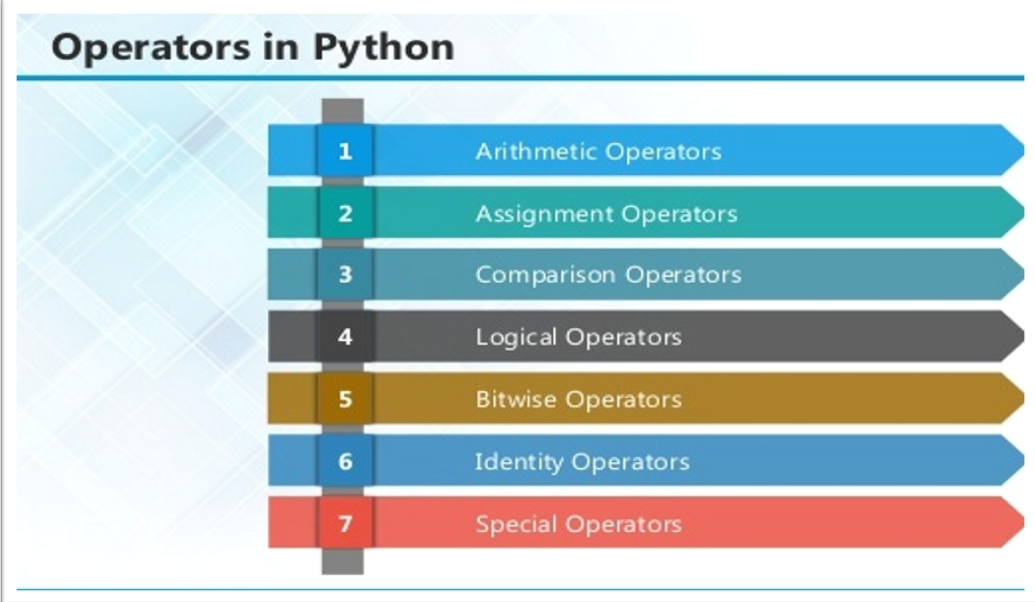
Python has a big list of good features:

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.



**Fig:8.1.** Python Execution Sequence

**8.4 Python Operators**



**Fig:8.2.** Python Operators

**HTML:**

Html is a language which is used to create web pages with html marking up a pageto indicate its format, telling the web browser where you want a new line to begin or how you want text or images aligned and more are possible.

**Table:**

Tables are so popular with web page authors is that they let you arrange the elements ofa web page in such a way that the browser won’t rearrange them web page authors frequently use tables to structure web pages.

**TR:**

TR is used to create a row in a table encloses <TH> and <TD> elements. <TR> contain many attributes. Some of them are,

ALIGN: specifies the horizontal alignment of the text in the table row.

BGCOLOR: Specifies the background color for the row.

BORDERCOLOR: Sets the external border color for the row.

VALIGN: Sets the vertical alignment of the data in this row.

**TH:**

TH is used to create table heading.

* ALIGN: Setsthe horizontal alignment ofthe content in the table cell. Sets LEFT, RIGHT, CENTER.
* BACKGROUND: Species the back ground image for the table cell.
* BGCOLOR: Specifies the background color of the table cell
* VALIGN: Sets the vertical alignment of the data. Sets to TOP, MIDDLE, BOTTOM or BASELINE.
* WIDTH: Specifies the width of the cell. Set to a pixel width or a percentage of the display area.

**TD:**

TD is used to create table data that appears in the cells of a table.

* ALIGN: Species the horizontal alignment of content in the table cell. Sets to LEFT, CENTER, RIGHT.
* BGCOLOR: Specifies the background image for the table cell.
* BGCOLOR: sets the background color of the table cells.

**Frames:**

Frames are used for either run offthe page or display only small slices of what are supposed to be shown and to configure the frame we can use <FRAMESET>There are two important points to consider when working with <FRAMESET>.

* <FRAMESET> element actually takes the place of the <BODY> element in a document.
*  Specifying actual pixel dimensions for frames .

<FRAME> Elements are used to create actual frames.

From the frameset point of view dividing the browser into tow vertical frames means creating two columns using the <FRAMESET> elements COLS attribute.

The syntax for vertical fragmentation is,

<FRAMESET COLS =”50%, 50%”> </FRAMESET>

Similarly, if we replace COLS with ROWS then we get horizontal fragmentation. The syntax for horizontal fragmentation is, <FRAMESET ROWS=”50%, 50%”> </FRAMESET>

**Form:**

The purpose of FORM is to create an HTML form; used to enclose HTML controls, like buttonsand text fields.

**Controls in HTML:**

<INPUT TYPE =BUTTON>:

Creates an html button in a form.

**ATTRIBUTES:**

* NAME: gives the element a name. Set to alphanumeric characters.
* SIZE: sets the size.
* VALUE: sets the caption of the element.

<INPUT TYPE = PASSWORD>:

Creates a password text field, which makes typed input.

ATTRIBUTES:

* NAME: gives the element a name, set to alphanumeric characters.
* VALUE: sets the default content of the element.

<INPUT TYPE=RADIO>:

Creates a radio button in a form.

ATTRIBUTE:

* NAME: Gives the element a name. Set to alphanumeric character.
* VALUE: Sets the default content of the element.

<INPUT TYPE=SUBMIT>:

Creates a submit buttonthat the user can click to send data in the form back to the web server.

ATTRIBUTES:

* NAME: Gives the element a name. Set to alphanumeric characters.
* VALUE: Gives this button another label besides the default, Submit Query. Set to alphanumeric characters.

<INPUT TYPE=TEXT>:

Creates atext field that the user can enter or edit text in.

ATTRIBUTES:

* NAME: Gives the element a name. Set to alphanumeric characters.
* VALUE: Holds the initial text in the text field. Set to alphanumeric characters.

## **User-Defined Functions**

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing. Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called *user-defined functions.*

## **8.5 Defining a Function**

Simple rules to define a function in Python.

* Function blocks begin with the keyword def followed by the function name and parentheses ( ( ) ).
* Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
* The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
* The code block within every function starts with a colon (:) and is indented.
* The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

def functionname( parameters ):

"function\_docstring"

function\_suite

return [expression]

**Calling a Function**

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme() function −

# Function definition is here

def printme( str ):

"This prints a passed string into this function"

print str

return;

# Now you can call printme function

printme("I'm first call to user defined function!")

printme("Again second call to the same function")

When the above code is executed, it produces the following result −

I'm first call to user defined function!

Again second call to the same function

## **Function Arguments**

You can call a function by using the following types of formal arguments:

* Required arguments
* Keyword arguments
* Default arguments
* Variable-length arguments

## **Packages in Python**

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-sub packages.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code −

def Pots():

print "I'm Pots Phone"

Similar way, we have another two files having different functions with the same name as above −

* *Phone/Isdn.py* file having function Isdn()
* *Phone/G3.py* file having function G3()

Now, create one more file \_\_init\_\_.py in *Phone* directory −

* Phone/\_\_init\_\_.py

To make all of your functions available when you've imported Phone,to put explicit import statements in \_\_init\_\_.py as follows −

from Pots import Pots

from Isdn import Isdn

from G3 import G3

After you add these lines to \_\_init\_\_.py, you have all of these classes available when you import the Phone package.

# Now import your Phone Package.

import Phone

Phone.Pots()

Phone.Isdn()

Phone.G3()

RESULT:

I'm Pots Phone

I'm 3G Phone

I'm ISDN Phone

In the above example, we have taken example of a single functions in each file, but you can keep multiple functions in your files. You can also define different Python classes in those files and then you can create your packages out of those classes.

This chapter covers all the basic I/O functions available in Python.

## **Opening and Closing Files**

Until now, you have been reading and writing to the standard input and output. Now, we will see how to use actual data files.

Python provides basic functions and methods necessary to manipulate files by default. You can do most of the file manipulation using a **file** object.

## **The *open* Function**

Before you can read or write a file, you have to open it using Python's built-in *open()* function. This function creates a **file** object, which would be utilized to call other support methods associated with it.

### Syntax

file object = open(file\_name [, access\_mode][, buffering])

Here are parameter details:

* **file\_name:** The file\_name argument is a string value that contains the name of the file that you want to access.
* **access\_mode:** The access\_mode determines the mode in which the file has to be opened, i.e., read, write, append, etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r).
* **buffering:** If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer size. If negative, the buffer size is the system default(default behavior).

## **The *close()* Method**

The close() method of a *file* object flushes any unwritten information and closes the file object, after which no more writing can be done.Python automatically closes a file when the reference object of a file is reassigned to another file. It is a good practice to use the close() method to close a file.

### Syntax

fileObject.close();

### Example

# Open a file

fo = open("foo.txt", "wb")

print "Name of the file: ", fo.name

# Close opend file

fo.close()

Result −

Name of the file: foo.txt

## **File & Directory Related Methods**

There are three important sources, which provide a wide range of utility methods to handle and manipulate files & directories on Windows and Unix operating systems. They are as follows −

* [File Object Methods](https://www.tutorialspoint.com/python/file_methods.htm): The *file* object provides functions to manipulate files.
* [OS Object Methods](https://www.tutorialspoint.com/python/os_file_methods.htm): This provides methods to process files as well as directories.

Python provides two very important features to handle any unexpected error in your Python programs and to add debugging capabilities in them −

* **Exception Handling:** This would be covered in this tutorial. Here is a list standard Exceptions available in Python: [Standard Exceptions](https://www.tutorialspoint.com/python/standard_exceptions.htm).
* **Assertions:** This would be covered in [Assertions in Python](https://www.tutorialspoint.com/python/assertions_in_python.htm)

List of Standard Exceptions −

|  |  |
| --- | --- |
| **EXCEPTION NAME** | **DESCRIPTION** |
| Exception | Base class for all exceptions |
| StopIteration | Raised when the next() method of an iterator does not point to any object. |
| SystemExit | Raised by the sys.exit() function. |
| StandardError | Base class for all built-in exceptions except StopIteration and SystemExit. |
| ArithmeticError | Base class for all errors that occur for numeric calculation. |
| OverflowError | Raised when a calculation exceeds maximum limit for a numeric type. |
| FloatingPointError | Raised when a floating point calculation fails. |
| ZeroDivisionError | Raised when division or modulo by zero takes place for all numeric types. |
| AssertionError | Raised in case of failure of the Assert statement. |
| AttributeError | Raised in case of failure of attribute reference or assignment. |
| EOFError | Raised when there is no input from either the raw\_input() or input() function and the end of file is reached. |
| ImportError | Raised when an import statement fails. |
| KeyboardInterrupt | Raised when the user interrupts program execution, usually by pressing Ctrl+c. |
| LookupError | Base class for all lookup errors. |
| IndexError  KeyError | Raised when an index is not found in a sequence.  Raised when the specified key is not found in the dictionary. |
| NameError | Raised when an identifier is not found in the local or global namespace. |
| UnboundLocalError  EnvironmentError | Raised when trying to access a local variable in a function or method but no value has been assigned to it.  Base class for all exceptions that occur outside the Python environment. |
| IOError  IOError | Raised when an input/ output operation fails, such as the print statement or the open() function when trying to open a file that does not exist.  Raised for operating system-related errors. |
| SyntaxError  IndentationError | Raised when there is an error in Python syntax.  Raised when indentation is not specified properly. |
| SystemError | Raised when the interpreter finds an internal problem, but when this error is encountered the Python interpreter does not exit. |
| SystemExit | Raised when Python interpreter is quit by using the sys.exit() function. If not handled in the code, causes the interpreter to exit. |
| TypeError | Raised when an operation or function is attempted that is invalid for the specified data type. |
| ValueError | Raised when the built-in function for a data type has the valid type of arguments, but the arguments have invalid values specified. |
| RuntimeError | Raised when a generated error does not fall into any category. |
| NotImplementedError | Raised when an abstract method that needs to be implemented in an inherited class is not actually implemented. |

## **8.6 What is Exception?**

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.

When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

## **Handling an exception**

If you have some *suspicious* code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block. After the try: block, include an **except:** statement, followed by a block of code which handles the problem as elegantly as possible.

**8.7 Python Database API:**

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as −

* GadFly
* mSQL
* MySQL
* PostgreSQL
* Microsoft SQL Server 2000
* Informix
* Interbase
* Oracle
* Sybase

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following:

* Importing the API module.
* Acquiring a connection with the database.
* Issuing SQL statements and stored procedures.
* Closing the connection

pip install numpy==1.18.1

pip install matplotlib==3.1.3

pip install pandas==0.25.3

pip install opencv-python==4.2.0.32

pip install keras==2.3.1

pip install sklearn

pip install tensorflow==1.14.0

pip install h5py==2.10.0

pip install pillow==7.0.0

pip install sklearn-genetic==0.2

pip install SwarmPackagePy

pip install sklearn

pip install scikit-learn==0.22.2.post1

Pip install sklearn-extensions==0.0.2

Pip install pyswarms==1.1.0

pip install protobuf==3.20.0

pip install nltk

pip install django==2.1.7

pip install pymysql==0.9.3

pip install matplotlib==3.1.3 (used for display Graph)

pip install pandas==0.25.3 (Used to read dataset)

pip install opencv-python (used for image reading)

pip install keras==2.3.1 (used for neural Network Implementation)

pip install tensorflow==1.14.0 (used for CNN implementation)

pip install tensorflow==1.15.0 --user

pip install keras==2.2.4 --user

pip install h5py==2.10.0

(used for support to tensorflow and keras libraries) -------------str object decoder error

pip install sklearn (used for Machine learning Algorithms implementation like Decision tree,randomforest Tree,etc)

pip install --only-binary :all: mysqlclient --user

pip install mysqlclient --user

sc delete mysql

import pymysql

pymysql.install\_as\_MySQLdb()

predict = model.predict\_classes(test)

pip install --user -U nltk

python

>>> import nltk

>>> nltk.download()

pip install -r requirements.txt

global filename

text.delete('1.0', END)

filename = filedialog.askopenfilename(initialdir="dataset")

dataset = pd.read\_csv(filename)

,on\_delete=models.CASCADE,

python -m pip install –-user -r requirements.txt

3.6.2 python

django==1.11.6

mysqlclient==1.3.12

Index.html

pip install opencv-python==4.6.0.66

Pip install tensorflow==2.9.1 –user

# CHAPTER 9 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**9.1 TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**9.1.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 9.1.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**9.1.3 User Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**SYSTEM TESTING**

### TESTING METHODOLOGIES

The following are the Testing Methodologies:

* **Unit Testing.**
* **Integration Testing.**
* **User Acceptance Testing.**
* **Output Testing.**
* **Validation Testing.**

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

**Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

**The following are the types of Integration Testing:**

**1)Top Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

**2. Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

* The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
* A driver (i.e.) the control program for testing is written to coordinate test case input and output.
* The cluster is tested.
* Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

**User Acceptance Testing**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

**Validation Checking**

Validation checks are performed on the following fields.

**Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

**Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

**Preparation of Test Data**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

**Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

**Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

**TESTING STRATEGY :**

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

**SYSTEM TESTING:**

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

**UNIT TESTING:**

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

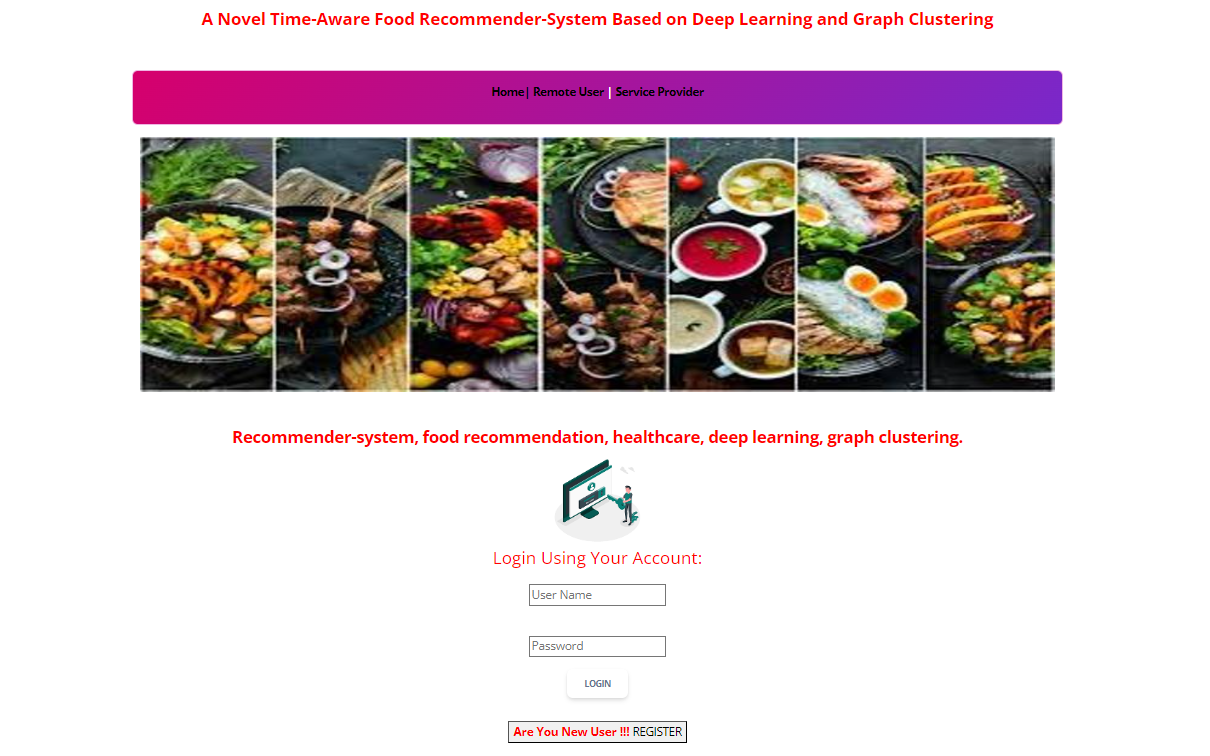
In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this.The future holds a lot to offer to the development and refinement of this project.

# CHAPTER 10

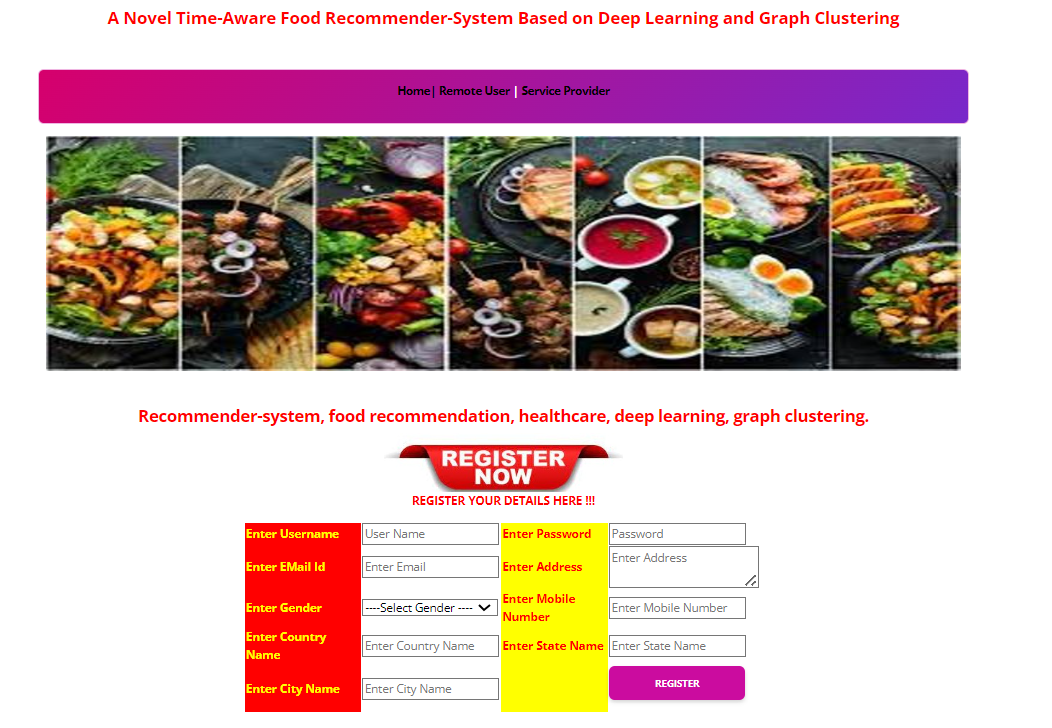
# SAMPLE OUTPUT

# 

**Fig:10.1.** Home Screen



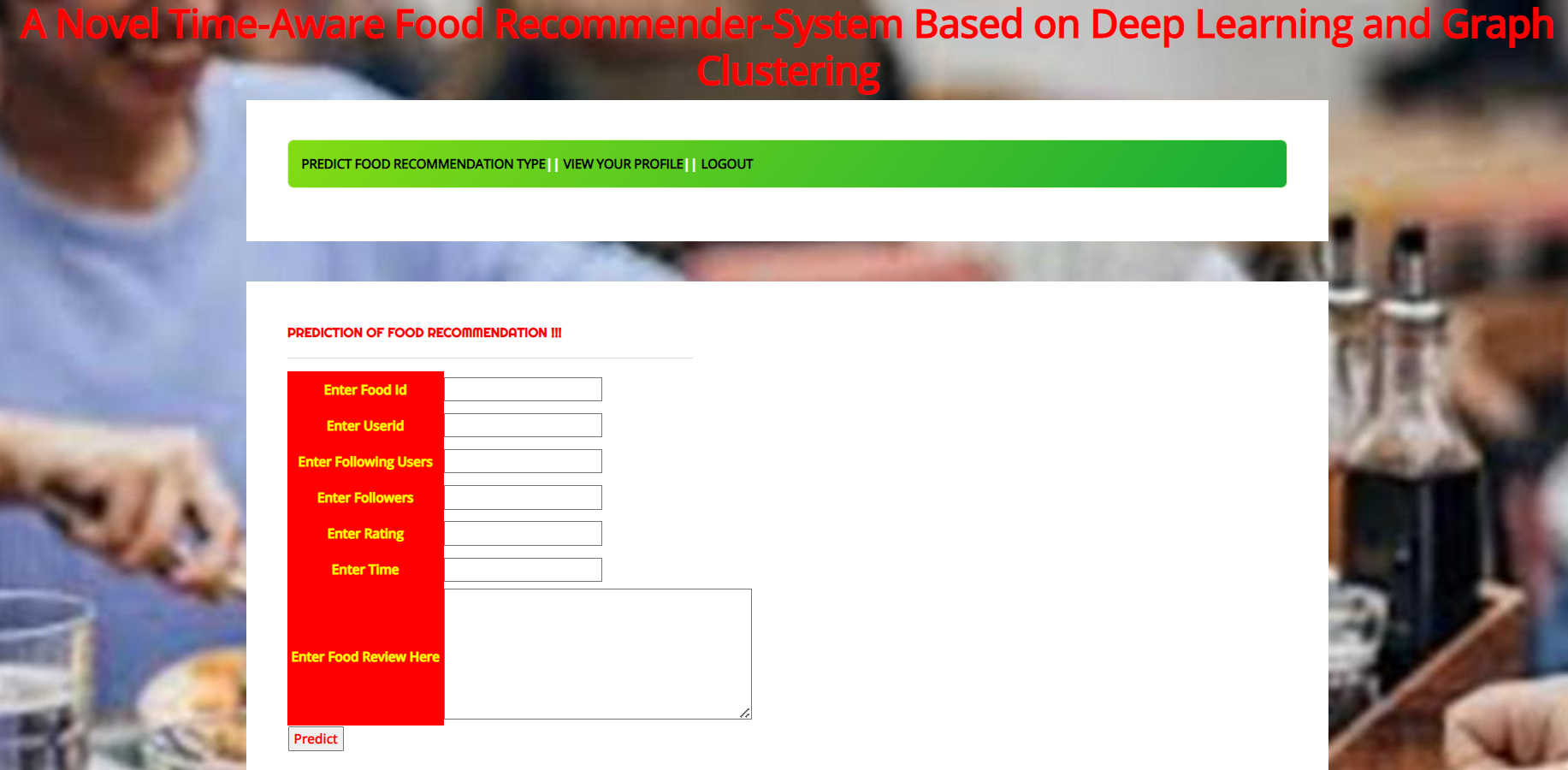
**Fig:10.2.** User Login Screen



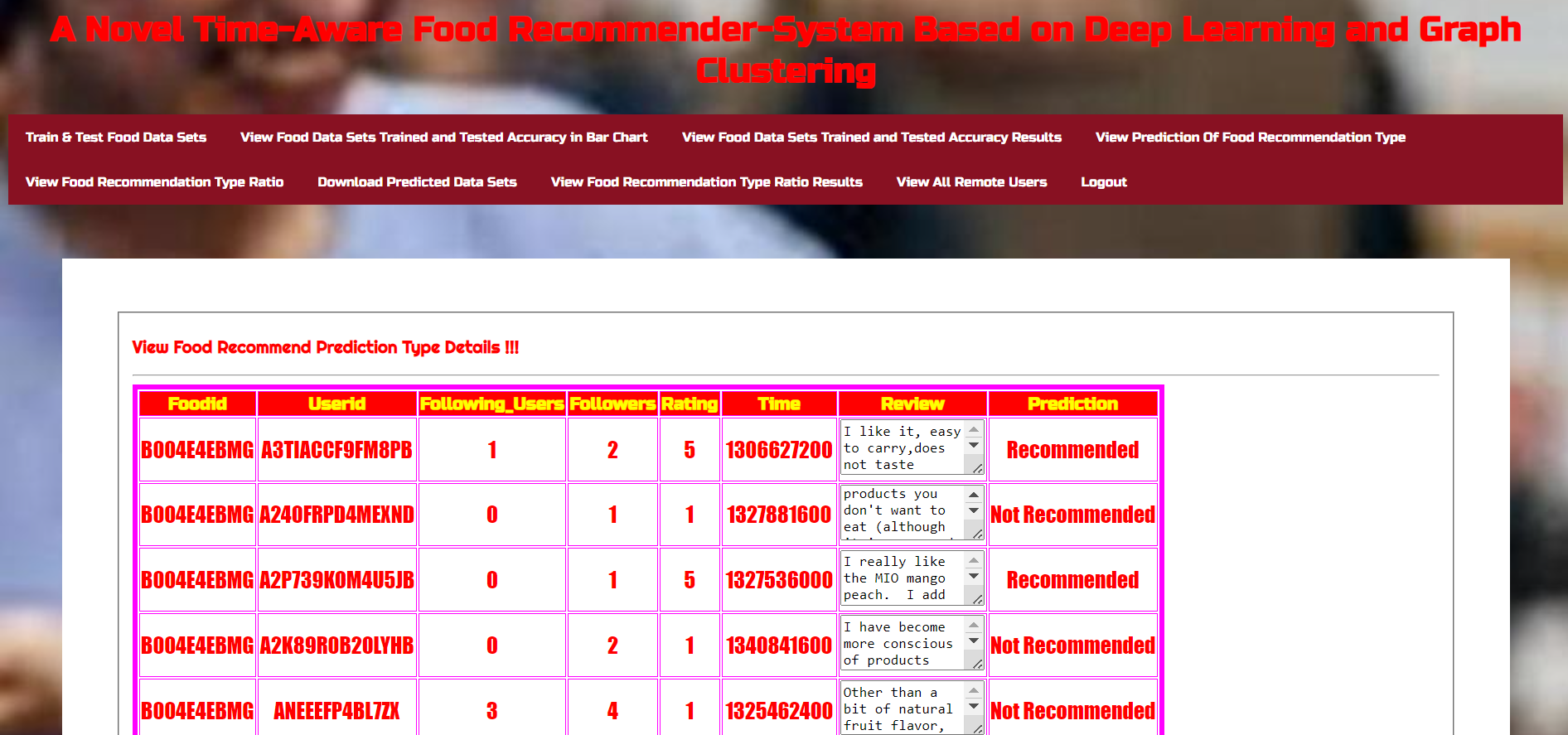
**Fig:10.3.** User Registration



**Fig:10.4.** Service Provider Login Page



**Fig:10.5.** Prediction Type Screen



**Fig:10.6.** Prediction Output Type

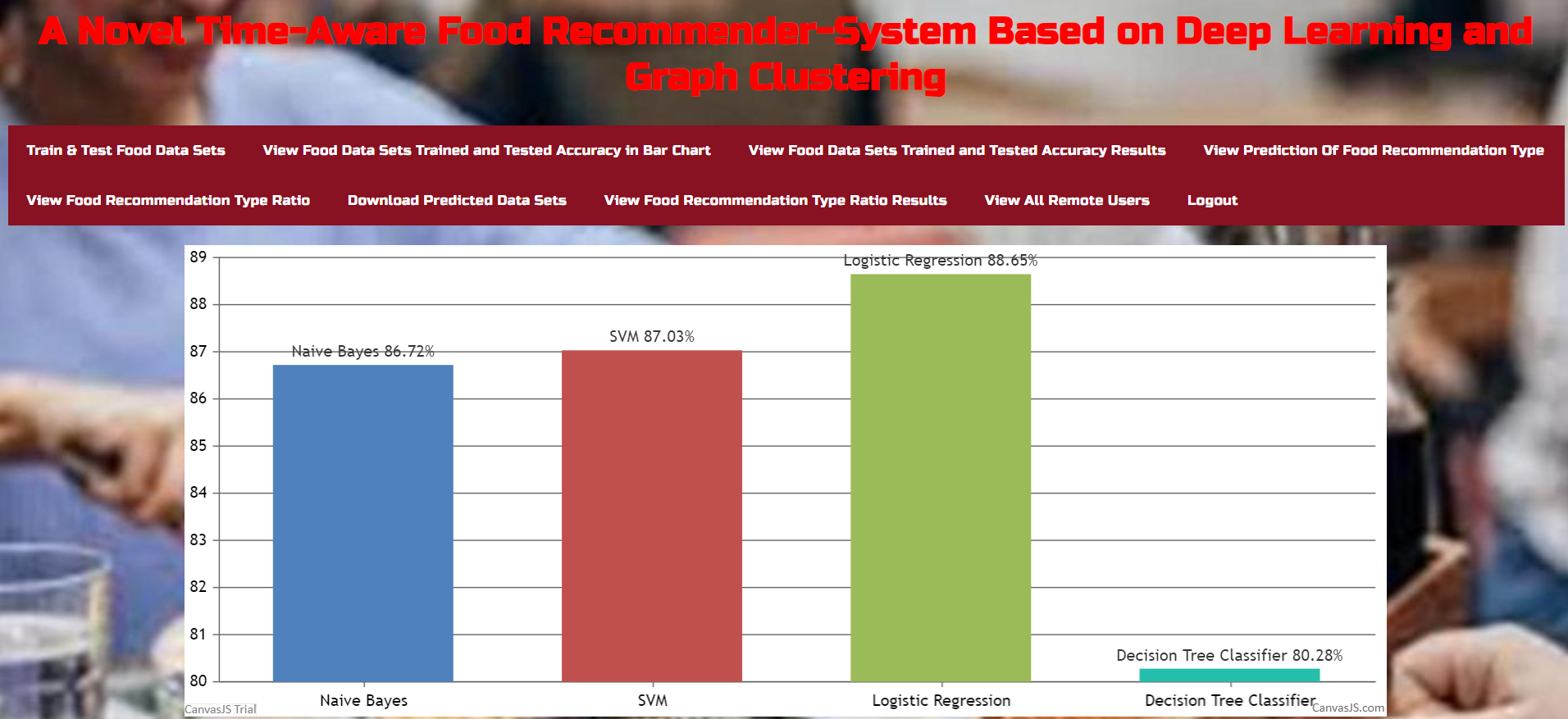


Fig:10.7. Output Bar Graph



**Fig:10.8.** Output Pie Chart Screen

# CHAPTER 11

# CONCLUSION

# 

With the development and increasing popularity of the Internet and the growing number of web users, recommender systems that select items that are reasonably appropriate to the needs of users are gradually becoming more widespread. A variety of lifestyle applications rely on food recommender systems, which are integral parts of many lifestyle services. A novel hybrid food recommender-system is developed in this paper to overcome the shortcomings of previous food recommender-systems, such as ignoring food ingredients, time stamp, cold start users and cold start foods and user community. Using user-based and content-based models as well as using time information, trust network, and user communities, the proposed method addresses all four issues simultaneously and aims to improve the final accuracy of the recommender-system. The proposed method involves two phases: food content-based recommendation and user-based recommendation. Graph clustering is used in the first phase, and a deep-learning based approach is used in the second phase to cluster both users and food items. The model has been compared to the newest proposed food recommender-system including LDA, HAFR and FGCN methods with respect to five different metrics: Precision, Recall, F1,AUCand NDCG.The experimental results indicated that the developed food recommender-system achieved the best performance and outperforms the state-of-the-art food recommender-systems by a noticeable margin. We aim to incorporate the side information of users (e.g., gender, age, weight, height, location, and culture) into the food recommendation framework in the future works to further improve the final performance of the food recommendation.

In addition, a proper eating habit can lessen the severity of symptoms associated with non-infectious diseases. In future works, we aim to use nutritional characteristics of each food as additional information and recommend foods according to each person's health status and diseases.

# REFERENCES

[1] S. Wang, L. Cao, Y. Wang, Q. Z. Sheng, M. A. Orgun, and D. Lian,

``A survey on session-based recommender systems,'' ACM Comput. Surv.,

vol. 54, no. 7, pp. 138, Sep. 2022.

[2] P. Wang, Y. Wang, L. Y. Zhang, and H. Zhu, ``An effective and efcient

fuzzy approach for managing natural noise in recommender systems,'' Inf.

Sci., vol. 570, pp. 623637, Sep. 2021.

[3] A. D. Viniski, J. P. Barddal, A. D. S. Britto, Jr., F. Enembreck, and

H. V. A. D. Campos, ``Acase study of batch and incremental recommender

systems in supermarket data under concept drifts and cold start,'' Expert

Syst. Appl., vol. 176, Aug. 2021, Art. no. 114890.

[4] X. Yu, Y. Chu, F. Jiang, Y. Guo, and D. Gong, ``SVMs classication based

two-side cross domain collaborative ltering by inferring intrinsic user and

item features,'' Knowl.-Based Syst., vol. 141, pp. 8091, Feb. 2018.

[5] N. Hazrati and F. Ricci, ``Recommender systems effect on the evolution

of users' choices distribution,'' Inf. Process. Manage., vol. 59, no. 1,

Jan. 2022, Art. no. 102766.

[6] L. Xie, Z. Hu, X. Cai, W. Zhang, and J. Chen, ``Explainable recommendation

based on knowledge graph and multi-objective optimization,''

Complex Intell. Syst., vol. 7, no. 3, pp. 12411252, Jun. 2021.

[7] M. Wasid and R. Ali, ``A frequency count approach to multi-criteria

recommender system based on criteria weighting using particle swarm

optimization,'' Appl. Soft Comput., vol. 112, Nov. 2021, Art. no. 107782.

[8] S. Forouzandeh, M. Rostami, and K. Berahmand, ``A hybrid method for

recommendation systems based on tourism with an evolutionary algorithm

and topsis model,'' Fuzzy Inf. Eng., vol. 14, no. 1, pp. 2650, 2022.

[9] S. Forouzandeh, M. Rostami, and K. Berahmand, ``Presentation a trust

Walker for rating prediction in recommender system with biased random

walk: Effects of H-index centrality, similarity in items and friends,'' Eng.

Appl. Artif. Intell., vol. 104, Sep. 2021, Art. no. 104325.

[10] T. N. T. Tran, A. Felfernig, and N. Tintarev, ``Humanized recommender

systems: State-of-the-art and research issues,'' ACM Trans. Interact. Intell.

Syst., vol. 11, no. 2, pp. 141, Jul. 2021.

[11] M. Slokom, A. Hanjalic, and M. Larson, ``Towards user-oriented privacy

for recommender system data: A personalization-based approach to gender

obfuscation for user proles,'' Inf. Process. Manage., vol. 58, no. 6,

Nov. 2021, Art. no. 102722.

[12] X.Yu, F. Jiang, J. Du, and D. Gong, ``Across-domain collaborative ltering

algorithm with expanding user and item features via the latent factor space

of auxiliary domains,'' Pattern Recognit., vol. 94, pp. 96109, Oct. 2019.

[13] M. Ge, F. Ricci, and D. Massimo, ``Health-aware food recommender system,''

in Proc. 9th ACMConf. Recommender Syst., Sep. 2015, pp. 333334.

[14] D. Bianchini, V. De Antonellis, N. De Franceschi, and M. Melchiori,

``PREFer: A prescription-based food recommender system,'' Comput.

Standards Interfaces, vol. 54, pp. 6475, Nov. 2017.

[15] M. B. Vivek, N. Manju, and M. N. Vijay, ``Machine learning based

food recipe recommendation system,'' in Proc. Int. Conf. Cogn. Recognit.

Singapore: Springer, 2018, pp. 1119.

[16] T. N. T. Tran, A. Felfernig, C. Trattner, and A. Holzinger, ``Recommender

systems in the healthcare domain: State-of-the-art and research issues,''

J. Intell. Inf. Syst., vol. 57, no. 1, pp. 171201, Aug. 2021.