0 

PRODUCT RECOMMENDATION SYSTEM USING MACHINE LEARNING

## A MINI PROJECT REPORT

***Submitted by***

# THENDRAL. A [REGISTER NO:211422104518]

# SHINY PRISCILLA. J [REGISTER NO: 211422104461]

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

# BACHELOR OF ENGINEERING

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

# ANNA UNIVERSITY: CHENNAI 600 025

**OCTOBER 2024**

# BONAFIDE CERTIFICATE

Certified that this project report “PRODUCT RECOMMENDATION SYSTEM USING MACHINE LEARNING WITH PYTHON” is the bonafide work of “ THENDRAL. A [211422104518] AND SHINY PRISCILLA. J [211422104461]” who carried out the project work under my supervision.

## SIGNATURE SIGNATURE

### Dr. L. JABASHEELA., M.E., Ph.D., Dr. V SATHYA PREIYA.,M.E., Ph.D.,

### HEAD OF THE DEPARTMENT SUPERVISOR

**PROFESSOR ASSOCIATE PROFESSOR**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE, NASARATHPETTAI, NASARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

# DECLARATION BY THE STUDENT

We, **THENDRAL A.** (Register No. 211422104518) and **SHINY PRISCILLA J.** (Register No. 211422104461), hereby declare that this project report titled **"Product Recommendation System Using Machine Learning"**, completed under the guidance of **Dr.V SATHYA PREIYA** ,**M.E.,Ph.D.**,is our original work. We confirm that this work has not been plagiarized or submitted for any other degree or certification in any university by us.

**ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr. P. CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our directors **Tmt. C. VIJAYARAJESWARI**, **Dr. C. SAKTHI KUMAR, M.E., Ph.D.** and **Dr. SARANYASREE SAKTHI KUMAR B.E., M.B.A., Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr. K. MANI, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L. JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank our **Project Guide Dr.V.SATHYA PREIYA , M.E.,Ph.d.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

THENDRAL A [211422104518]  
SHINY PRISCILLA J [211422104461]

**ABSTRACT**

Product recommendation systems play a crucial role in enhancing the customer shopping experience by providing personalized suggestions, improving sales, and increasing customer satisfaction. Machine learning techniques have gained popularity for their ability to analyze vast amounts of customer transaction data and provide accurate recommendations. This project explores the application of the **Apriori algorithm** for generating product recommendations. Apriori is an association rule learning algorithm that identifies relationships between frequently purchased products, making it well-suited for analyzing shopping patterns. In this study, historical transaction data, such as product purchases from different customers, are collected and pre processed. These data are then used to train the Apriori model, which generates recommendations based on the identified associations between products. This system can suggest additional products that customers are likely to buy, thus improving the overall shopping experience.

.

**LIST OF TABLES**

|  |  |
| --- | --- |
| **Figure** | **Caption** |
| 7.1.1 | Test case table |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| **Figure** | **Caption** |
| 4.1.1 | Data Flow Diagram |
| 4.2.1 | Activity Diagram |
| 4.3.1 | Use Case Diagram |
| 5.1.1 | System architecture Diagram |
| 7.1.1 | Test case 1 |
| 7.1.2 | Test case 2 |
| 7.1.3 | Test case 3 |
| 7.1.4 | Test case 4 |
| 7.1.5 | Test case 5 |

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | iii |
|  | **LIST OF TABLES** | iv |
|  | **LIST OF FIGURES** | v |
| **1.** | **INTRODUCTION** | 1 |
|  | 1.1 Problem Definition | 1 |
| **2.** | **LITERATURE SURVEY** | 2 |
| **3.** | **SYSTEM ANALYSIS** | 4 |
|  | 3.1 Existing System | 4 |
|  | 3.2 Proposed system | 5 |
|  | 3.3 Feasibility Study | 7 |
|  | 3.4 Hardware Environment | 8 |
|  | 3.5 Software Environment | 9 |
| **4.** | **SYSTEM DESIGN** | 10 |
|  | 4.1 Data Flow Diagram | 10 |
|  | 4.2 UML Diagrams | 11 |
| **5.** | **SYSTEM ARCHITECTURE** | 12 |
|  | 5.1 Architecture Overview | 12 |
|  | 5.2 Module Design Specification | 13 |
|  | 5.3 Algorithms | 15 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| **6.** | **SYSTEM IMPLEMENTATION** | 18 |
|  | 6.1 Coding | 18 |
| **7.** | **PERFORMANCE ANALYSIS** | 28 |
|  | 7.1 Accuracy & Comparison | 28 |
| **8.** | **CONCLUSION** | 30 |
|  | 8.1 Conclusion and Future Enhancements | 30 |
| **A.** | **APPENDICES** | 31 |
|  | A.1 Sample Screens | 31 |
|  | **REFERENCES** | 35 |

## INTRODUCTION

### Problem Definition

* **Personalized Product Recommendations:** Develop a machine learning model using Python that can suggest relevant products to customers based on their current and past purchases. The system should analyze shopping patterns and offer personalized recommendations that improve the overall shopping experience.
* **Frequent Itemset Prediction:** Create a machine learning system in Python capable of identifying frequently bought-together products using association rule mining techniques. This system will help suggest additional items that complement the customer's current selections.
* **Dynamic Product Bundling**: Develop a dynamic bundling system that recommends combinations of products, enabling better cross-selling opportunities. This system will analyze customer preferences and offer product bundles based on commonly associated items in previous transactions.
* **Real-time Purchase Recommendations**: Create a recommendation model that provides real-time suggestions as customers add products to their cart. The system should update the recommended items list based on the evolving contents of the customer’s cart to ensure relevant and timely recommendations.

## LITERATURE SURVEY

Improving the accuracy of machine learning techniques for product recommendation systems has been a primary focus for researchers in recent years. Various studies have explored different approaches to enhancing recommendation accuracy and personalizing the shopping experience. Some of the related studies are discussed here.

Bhardwaj and Duhoon [1] applied soft computing techniques for recommendation systems, highlighting the role of artificial intelligence in enhancing customer experiences by providing relevant suggestions.

Haghbin et al. [2] explored the use of association rule mining for predicting customer purchase patterns in e-commerce, emphasizing how frequent itemsets can be effectively utilized to improve product recommendations.

Jayasingh et al. [3] implemented a hybrid approach combining machine learning models and collaborative filtering techniques to improve the precision of recommendations in online retail systems.

Khajure and Mohod [4] used soft computing techniques for analyzing purchasing behavior and predicting future customer preferences, which could be applied to recommendation systems in various retail platforms.

Litta et al. [5] developed an artificial neural network model to predict customer buying patterns, demonstrating the power of neural networks in accurately forecasting customer needs based on past transactions.

Lee and Lee [6] constructed an efficient recommendation model using big data analysis, providing insights into how large-scale transactional data can help fine-tune recommendation engines for better results.

Singh et al. [7] utilized machine learning algorithms like Random Forest and k-Nearest Neighbors (k-NN) to develop a recommendation system that predicts products based on historical shopping data, demonstrating the effectiveness of machine learning techniques in product recommendation.

Schultz et al. [8] compared the performance of deep learning models with traditional recommendation algorithms, showcasing how deep learning can outperform conventional approaches in certain product recommendation tasks.

Sharma and Agarwal [9] used a neural network-based recommendation system that incorporated customer behavior and preferences, offering highly personalized recommendations to users.

Sofian et al. [10] employed artificial neural networks to enhance recommendation accuracy in e-commerce systems, illustrating the adaptability of neural networks for making predictions on customer purchasing patterns.

Vathsala and Koolagudi [11] applied neuro-fuzzy models to develop a recommendation system that accounts for uncertainty in customer preferences, combining the strengths of fuzzy logic and neural networks to improve recommendation outcomes.

## SYSTEM ANALYSIS

### Existing System

The existing systems for product recommendation, particularly in e-commerce and retail, have certain limitations that need to be addressed. One major drawback is the over-reliance on simple collaborative filtering techniques, which often fail to provide personalized recommendations when sufficient user data is unavailable (the "cold start" problem). These systems also tend to recommend popular products, which may not always align with a user's specific preferences or needs, leading to less relevant suggestions. Additionally, existing systems may lack the ability to capture the dynamic nature of a customer’s behavior, making it difficult to adjust recommendations in real-time based on evolving customer interests.

Another significant limitation is the scalability of the recommendation algorithms, especially when dealing with large datasets. As the number of users and products increases, traditional recommendation systems can struggle with efficiency and accuracy. Furthermore, many existing systems fail to effectively utilize association rule mining or machine learning techniques, resulting in less accurate predictions of what customers might want to purchase next.

These drawbacks underscore the need for more advanced machine learning models capable of providing better, more personalized product recommendations by analyzing purchasing patterns and customer behavior more effectively.

*Disadvantages of Existing system*

1. Difficulties in providing personalized recommendations for users with limited transaction history.

2. Lack of real-time adaptation to changes in customer behavior or preferences.

### Proposed System

The proposed system employs various machine learning techniques and algorithms to develop an accurate product recommendation system. It predicts relevant products based on historical transaction data, enabling users to receive personalized product suggestions that enhance their shopping experience. The system leverages association rule mining and machine learning algorithms to analyze data, identify patterns, and provide tailored recommendations. Here's an overview of the system:

***Data Collection****:*

Collect transaction data from users, including items purchased, frequency of purchases, and other relevant buying patterns.

***Algorithm Selection****:*

Implement association rule mining techniques, such as the **Apriori algorithm**, to uncover frequent itemsets and identify strong associations between products. This will be combined with machine learning algorithms to enhance the precision of recommendations.

***Real-Time Recommendations****:*

Enable users to input their current product choices into the system. Based on these choices, the system will generate real-time product recommendations, suggesting complementary or frequently bought-together items.

***Multi-Level Recommendations for High Accuracy****:*

Implement a multi-level recommendation approach to improve the accuracy of product suggestions. The **Apriori algorithm** will ensure that frequently associated items are recommended, while machine learning models fine-tune these suggestions to match the user's individual preferences.

***Data Integration and Standardization****:*

Standardize and integrate transaction data to ensure consistency and accuracy in recommendations. This will help in creating a seamless and reliable recommendation system that offers accurate and relevant suggestions to the users.

### *Advantages of the Proposed System:*

### *Multiple Levels of Recommendations for High Accuracy:*

The use of a combination of association rule mining and machine learning algorithms ensures accurate and relevant product recommendations, even when dealing with diverse customer preferences and large datasets.

### *Real-Time Product Recommendations:*

Users receive real-time product suggestions as they add items to their shopping cart, enhancing their shopping experience with timely and relevant recommendations.

***Personalized Shopping Experience****:*

The proposed system tailors recommendations based on individual user behavior, offering a personalized experience that improves customer satisfaction and increases the likelihood of repeat purchases.

***Improved Cross-Selling Opportunities****:*

By analyzing transaction patterns and frequently bought-together items, the system enables better cross-selling opportunities, helping businesses boost sales and enhance product discovery.

### Feasibility System

Conducting a comprehensive feasibility study for the "Product Recommendation System Using Machine Learning" project is essential to evaluate its practicality and likelihood of success. The technical feasibility will assess whether the required machine learning expertise and infrastructure, including algorithms like Apriori for association rule mining, and the necessary hardware and software resources, are available or can be easily acquired. This ensures that the system can handle large datasets of user transactions and generate real-time product recommendations effectively.Financial feasibility involves estimating the costs associated with data collection, model development, web application implementation, and ongoing system maintenance. These costs must be balanced against the expected benefits, such as increased sales, enhanced customer satisfaction, and a positive return on investment for businesses utilizing the recommendation system. Operational feasibility will examine how well the proposed system integrates with existing e-commerce platforms or business environments. It will also focus on the user experience, ensuring that both businesses and customers can easily adopt and interact with the system for seamless, real-time product recommendations.Finally, scheduling and timeline feasibility will help define realistic project milestones and ensure that the system can be developed and deployed within a reasonable timeframe. This includes phases for data preprocessing, algorithm development, integration, and thorough testing. This feasibility study will provide a solid foundation for informed decision-making, ensuring the successful execution and deployment of the product recommendation system.

### Hardware Environment

* Processor - Core i3/i5/i7
* RAM - 2 GB/4GB
* SSD – 256 GB

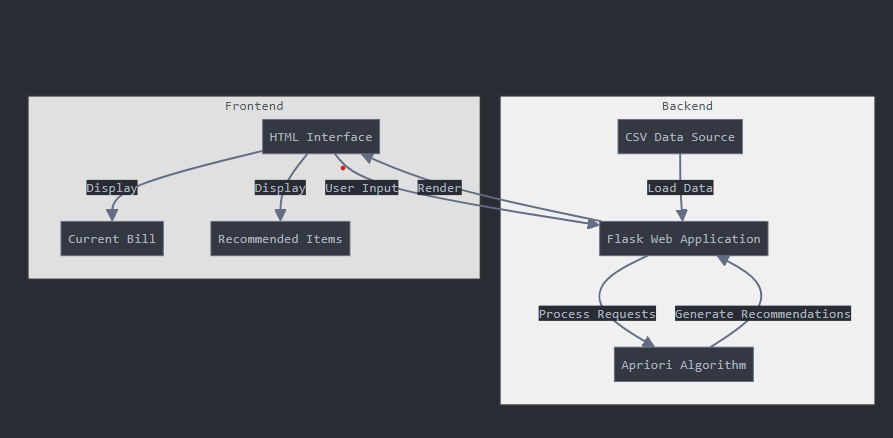
### Software Environment

* **Platform**: Windows 7/10/11
* **Programming Language**: Python
* **Data Processing Libraries**: NumPy and Pandas
* **Machine Learning Libraries**: Scikit-Learn and Apyori (for association rule mining)
* **Graphical User Interface (GUI)**: Flask (for web-based interface)
* **Data Visualization**: Matplotlib (for displaying product recommendation trends and user behavior)

## SYSTEM DESIGN

### Data Flow Diagram

### A Data Flow Diagram (DFD) is a graphical representation used to visualize how data moves within a system. It helps in understanding the flow of information from input to output, showing the processes, data stores, data sources, and data destinations.

****

**Fig.4.1.1 Data flow diagram of Product recommendation system**

This diagram represents the architecture of a **Product Recommendation System** divided into two main sections: **Frontend** and **Backend**.

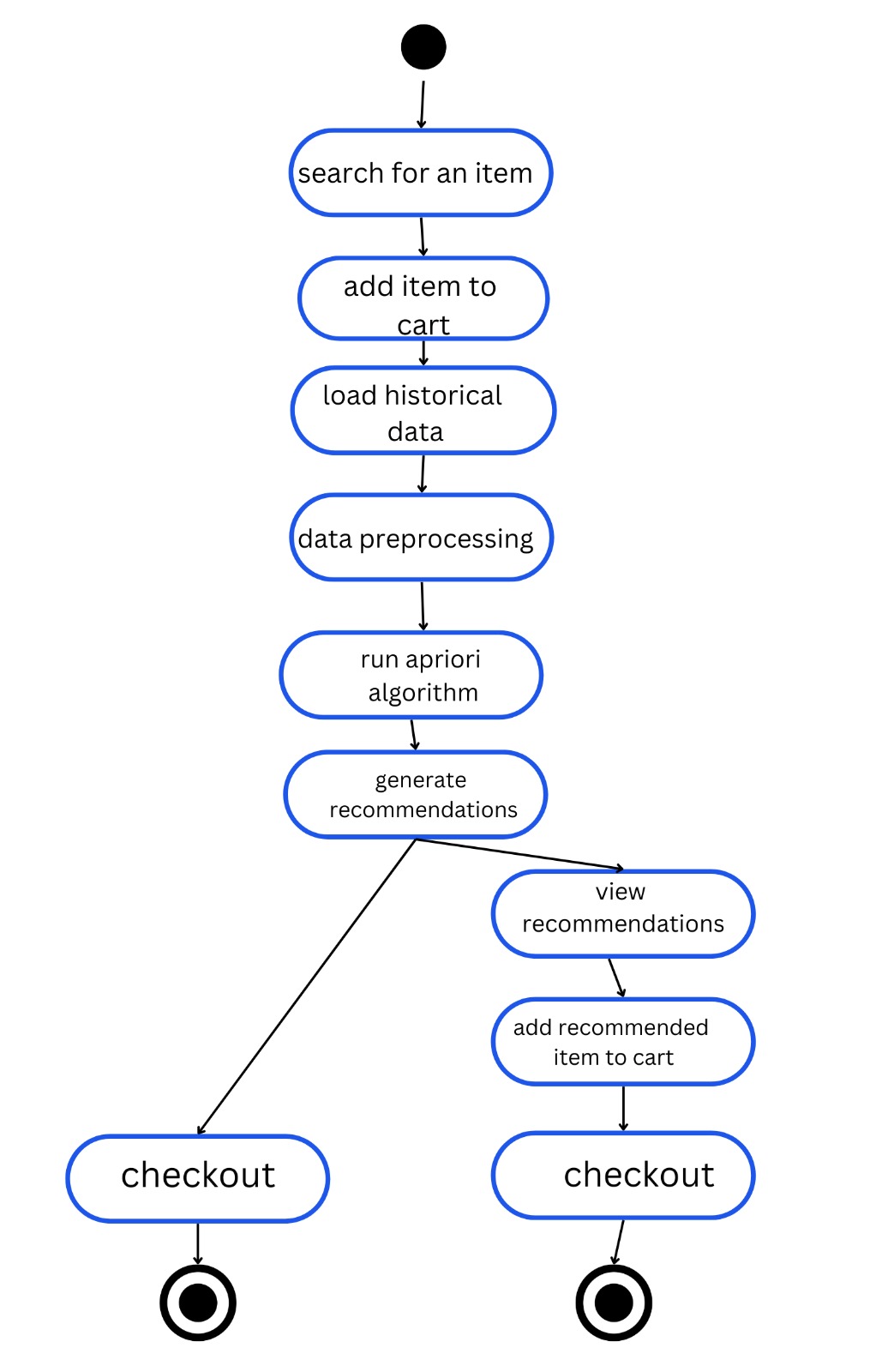
**Frontend:**

* The **HTML Interface** serves as the user interface that handles interaction with users. It captures **User Input** and then renders the required information back to the user.
* The interface displays two key elements:
  1. **Current Bill**: Shows the user's current shopping bill.
  2. **Recommended Items**: Displays personalized product recommendations based on the user's input.

**Backend:**

* **CSV Data Source**: A file-based data source containing user purchase history and other relevant product data.
* **Flask Web Application**: A backend framework (Flask) loads the data from the CSV file, processes user requests, and generates recommendations.
  + It receives user input from the frontend, processes requests, and runs the **Apriori Algorithm** to generate **Recommended Items**.
  + The **Apriori Algorithm** identifies frequent itemsets from the user's purchase history and other data, allowing the system to generate personalized recommendations.
  1. **Activity Diagram**

An Activity Diagram is a behavioral UML (Unified Modeling Language) diagram that represents the flow of activities within a system, showing the sequence of steps from the start to the end of a process.

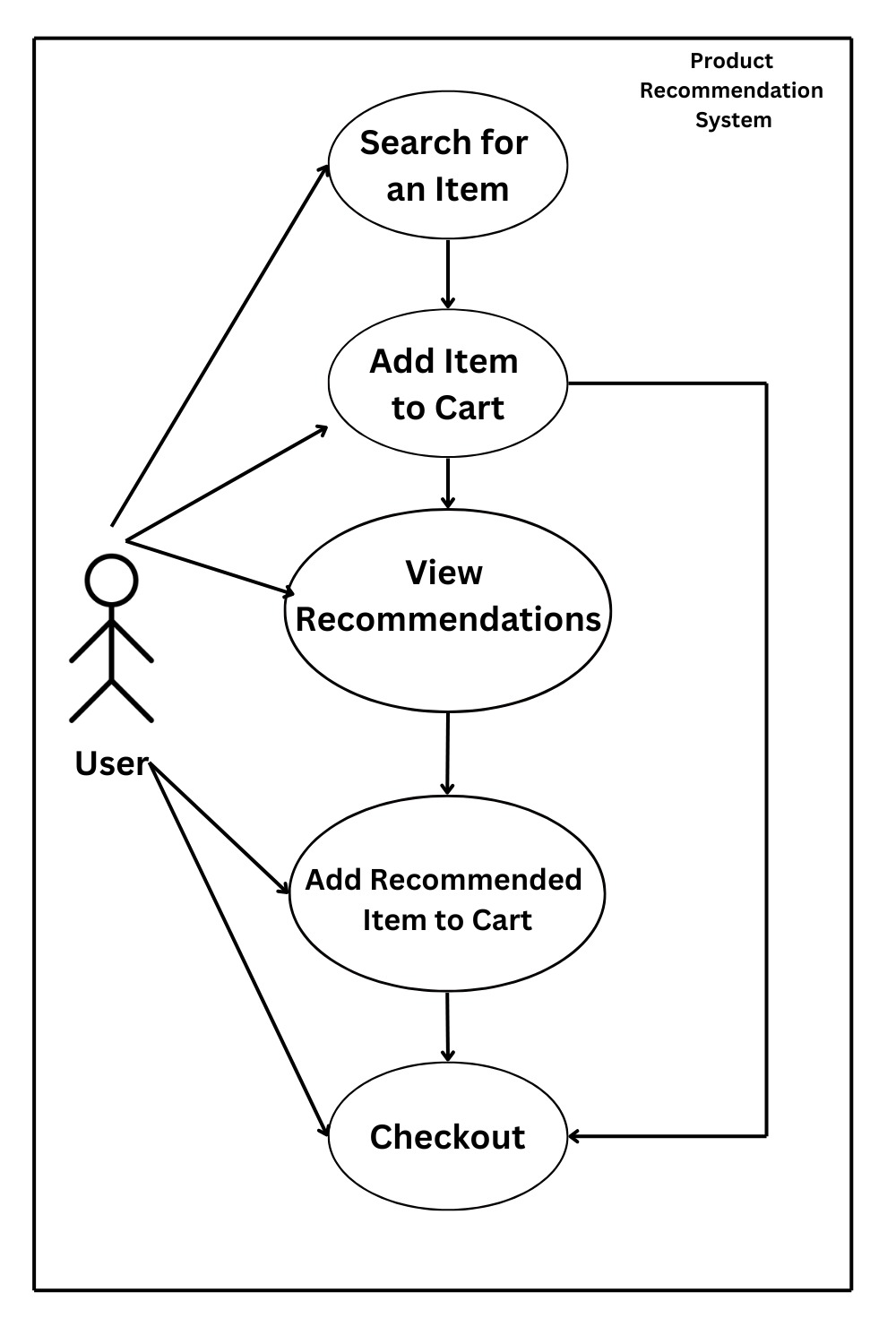


**Fig.4.2.1 Activity Diagram for Product Recommendation System**

The activity diagram for the product recommendation system begins with the user initiating the system by inputting items or data. This input, which could be product details or user preferences, triggers the next step, which is data processing. In this stage, the system processes the input to prepare it for use in training a machine learning (ML) model.Once the model is trained, it becomes ready to make recommendations. Based on the trained model, the system predicts and generates product recommendations for the user. Finally, the process ends once the recommendations are provided, marking the conclusion of the workflow.

* 1. **Use Case Diagram**

A Use Case Diagram is a type of behavioral diagram in UML (Unified Modeling Language) that visually represents the interactions between users (called actors) and a system. It illustrates the functional requirements of a system by showing the various ways the system is used to achieve specific goals**.**



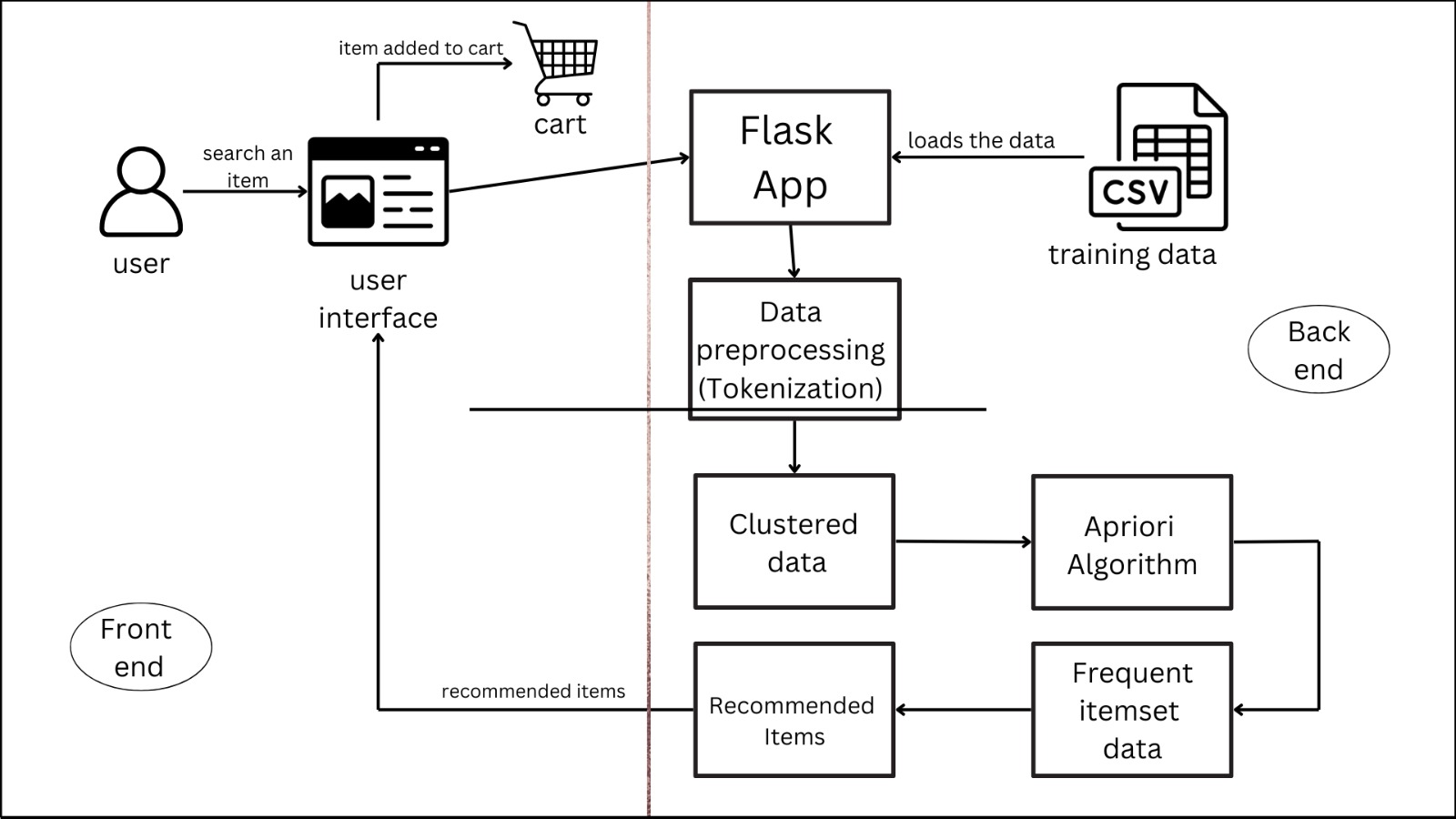
**Fig.4.3.1 Use Case Diagram of Product Recommendation System**

The **Use Case Diagram** for the product recommendation system describes how a user interacts with the system to search, select, and checkout items. It begins with the **User** searching for an item, which triggers the process. The user can then add an item to the shopping cart. After this, the system shows the user product recommendations based on the selected items. The user can choose to add a recommended item to the cart. The final step in the process is the **Checkout**, where the user proceeds to complete the purchase. This diagram illustrates the primary interactions between the user and the system during the shopping and recommendation process.

## SYSTEM ARCHITECTURE

### Architecture Overview

An Architecture Diagram is a visual representation that outlines the structure of a system, depicting its components and their relationships. It shows how different parts of the system, such as software modules, hardware components, databases, and user interfaces, interact to achieve the desired functionality.



**Fig.5.1.1 System architecture Diagram of Product Recommendation System**

The architecture diagram illustrates the interaction between the front-end (user interface) and back-end (machine learning and recommendation engine) of a product recommendation system. Here's a breakdown of the components:

1. User: The process begins with the user interacting with the system by searching for an item via the user interface.
2. User Interface: This is the front-end where the user searches for products and adds them to the shopping cart. When the user adds an item to the cart, this action is forwarded to the Flask App.
3. Flask App: This back-end web application handles the requests from the user interface. It takes the data from the cart and interacts with the recommendation system to provide item recommendations. The Flask App also loads the training data for further processing.
4. Data Preprocessing (Tokenization): The loaded data undergoes preprocessing, such as tokenization, to prepare it for analysis. This step ensures that the data is in the right format for the next stages of the recommendation process.
5. Clustered Data: After preprocessing, the data is clustered into groups of similar items, allowing the system to better understand relationships between products.
6. Apriori Algorithm: This algorithm is employed to identify frequent itemsets from the clustered data. The Apriori Algorithm finds items that are often bought together, forming the basis of recommendations.
7. Frequent Itemset Data: The frequent patterns discovered by the Apriori algorithm are stored as frequent itemset data, which is then used to generate recommendations.
8. Recommended Items: Based on the frequent itemsets, the system provides the user with recommended items that they might be interested in, which are displayed in the user interface.
9. Cart: The user adds the recommended item to the cart, completing the cycle from searching for an item to viewing recommendations and finalizing choices.

### Module Design Specification

* + 1. **Data Pre-Processing Module**

The quality of the data used in any machine learning model significantly affects its performance and accuracy. Ensuring that the data is properly cleaned, structured, and prepared is essential for building an effective recommendation system. The pre-processing module will focus on handling missing data, removing outliers, and transforming raw transactional data into a format suitable for analysis. Exploratory data analysis will help in gaining insights into the data to ensure its quality and readiness for model training.

* **Exploratory Analysis**:
* Perform data exploration to gain insights into the transactional data. Address issues such as missing data, outliers, and inconsistencies to enhance the quality of the input data for generating accurate product recommendations.

### Training Module

Once the data has been pre-processed, the next phase involves training the machine learning model. This module focuses on selecting the appropriate algorithm (such as Apriori for association rule mining or Random Forest for recommendation). The data is split into two parts: the training set and the test set. The training set is used to develop the model, while the test set is held back to evaluate the model's performance.

* **Data Splitting**:
* Divide the cleaned transactional data into training and test datasets. The training dataset is used to build the recommendation model, while the test dataset is reserved for evaluation to ensure the model's accuracy and generalization.

### Prediction Module

In the prediction module, users are prompted to input the products they have already selected or are interested in purchasing. Based on these inputs, the system interactively collects and analyzes the user's data, leveraging the trained machine learning model to generate product recommendations. The system will use association rule mining (Apriori) and other machine learning algorithms to predict additional products the user might be interested in, based on patterns found in the dataset.

* **Product Selection**:

Allow users to specify the products they have added to their bill or are considering. Interactively collect input on selected items, and use this data to predict and recommend additional products using the trained recommendation model.

### Algorithms

The **Apriori Algorithm** is a popular data mining technique used for association rule learning, which is especially effective in market basket analysis and product recommendation systems. It helps identify frequent itemsets and generate rules that highlight relationships between products based on user purchasing behavior. Below is an overview of the key concepts and techniques used in the Apriori Algorithm for this project:

* **Frequent Itemsets**:

The core idea of Apriori is to find itemsets (groups of products) that frequently occur together in transactions. For example, if customers often buy "bread" and "butter" together, Apriori identifies this as a frequent itemset.

* **Support**:  
  Support measures how frequently an itemset appears in the dataset. It is calculated as the proportion of transactions that contain the itemset. The algorithm filters out itemsets that don’t meet a minimum support threshold, ensuring that only relevant product combinations are considered.
* **Confidence**:  
  Confidence refers to the likelihood that when a customer buys one product (A), they will also buy another product (B). It is calculated as the ratio of the number of transactions containing both A and B to the number of transactions containing A. High confidence indicates a strong association between products.
* **Association Rules**:

Apriori generates association rules based on frequent itemsets. These rules describe relationships between products, such as “If a customer buys Product A, they are likely to buy Product B.” These rules are used to make product recommendations by suggesting related products to users.

* **Lift**:  
  Lift is a measure of how much more likely two products are bought together than if they were randomly selected. A lift value greater than 1 indicates a positive association between products, making it a valuable metric in generating high-quality recommendations.
* **Pruning**:  
  Apriori uses a technique called pruning to reduce the number of itemsets it has to analyze. It relies on the principle that if an itemset is infrequent, any larger itemset containing that itemset will also be infrequent. This step ensures that the algorithm runs efficiently, even on large datasets.
* **Candidate Generation**:

The algorithm iteratively generates candidate itemsets (product combinations) by combining frequent itemsets of smaller sizes. For instance, if “milk” and “bread” are frequent individual items, the algorithm may combine them to check if “milk and bread” is a frequent itemset.

* **Iterative Process:**

Apriori starts by identifying frequent individual items and gradually builds up larger itemsets by combining smaller ones. The process repeats until no more frequent itemsets can be found, meaning all significant product associations have been identified.

* **Thresholds**:  
  The performance of the Apriori Algorithm depends on setting appropriate thresholds for support and confidence. Higher thresholds generate fewer but stronger rules, while lower thresholds may generate a broader range of rules, including weaker associations.
* **Applications in Product Recommendation**:

In the context of product recommendation, the Apriori Algorithm helps by analyzing historical purchasing patterns to suggest products that are frequently bought together. When a customer adds an item to their cart, the system can recommend complementary products based on association rules generated by Apriori.

## SYSTEM IMPLEMENTATION

### Coding

**App.py**

from flask import Flask, render\_template, request

import pandas as pd

from apyori import apriori

app = Flask(\_\_name\_\_)

bill = []

def load\_data():

    dataset = pd.read\_csv('Market\_Basket\_Optimisation.csv', header=None)

    transactions = []

    for i in range(0, 7501):

        transactions.append([str(dataset.values[i, j]) for j in range(0, 20) if str(dataset.values[i, j]) != 'nan'])

    return transactions

def get\_recommendations(item):

    transactions = load\_data()

    rules = apriori(transactions, min\_support=0.002, min\_confidence=0.1, min\_lift=1, min\_length=2, max\_length=2)

    results = list(rules)

    lhs = []

    rhs = []

    for result in results:

        for relation\_record in result.ordered\_statistics:

            lhs.append(tuple(relation\_record.items\_base))

            rhs.append(tuple(relation\_record.items\_add))

    recommendations = []

    for left, right in zip(lhs, rhs):

        if item in left:

            recommendations.extend([x for x in right if x != item])

    return list(set(recommendations))

@app.route('/', methods=['GET', 'POST'])

def index():

    global bill

    recommendations = []

    if request.method == 'POST':

        item = request.form.get('item')

        if item:

            bill.append(item)

            recommendations = get\_recommendations(item)

            print(f"Item received: {item}")  # Debug statement

            print(f"Recommendations: {recommendations}")  # Debug statement

    return render\_template('index.html', bill=bill, recommendations=recommendations)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Product Generator System</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

    <script>

        function fillInput(item) {

            document.getElementById('item').value = item;

            document.getElementById('productForm').submit();

        }

    </script>

</head>

<body>

    <div class="container">

        <h1 class="mt-5">Product Recommendations System</h1>

        <form method="post" id="productForm">

            <div class="form-group">

                <label for="item">Enter an Item:</label>

                <input type="text" class="form-control" id="item" name="item" placeholder="Enter an item" required>

            </div>

            <button type="submit" class="btn btn-primary">Add to Bill & Get Recommendations</button>

        </form>

        <h2 class="mt-5">Current Bill:</h2>

        <ul class="list-group" id="bill">

            {% for product in bill %}

            <li class="list-group-item">{{ product }}</li>

            {% endfor %}

        </ul>

        {% if recommendations %}

        <h2 class="mt-5">Recommended Items:</h2>

        <div>

            {% for recommendation in recommendations %}

            <button type="button" class="btn btn-secondary m-1" onclick="fillInput('{{ recommendation }}')">{{ recommendation }}</button>

            {% endfor %}

        </div>

        {% endif %}

    </div>

</body>

</html>

**Sample dataset**

* **Product Recommendation System.csv**

1. burgers,meatballs,eggs
2. chutney
3. turkey,avocado
4. mineral water,milk,energy bar,whole wheat rice,green tea
5. low fat yogurt
6. whole wheat pasta,french fries
7. soup,light cream,shallot
8. frozen vegetables,spaghetti,green tea
9. french fries
10. eggs,pet food
11. cookies
12. turkey,burgers,mineral water,eggs,cooking oil
13. spaghetti,champagne,cookies
14. mineral water,salmon
15. mineral water,turkey,eggs
16. mineral water,salmon
17. mineral water
18. shrimp,chocolate,chicken
19. honey,oil,cooking oil,low fat yogurt
20. turkey,eggs
21. turkey,fresh tuna,tomatoes
22. spaghetti,mineral water,black tea,salmon,eggs
23. chicken,extra dark chocolate
24. meatballs,milk,honey,french fries,protein bar
25. red wine,shrimp,pasta,pepper,eggs,chocolate,shampoo
26. rice,sparkling water
27. spaghetti,mineral water,ham,body spray,pancakes,green tea
28. burgers,grated cheese,shrimp,pasta,avocado,honey
29. white wine,toothpaste
30. eggs
31. parmesan cheese,spaghetti,soup,avocado,milk,fresh bread
32. ground beef,spaghetti,mineral water
33. milk,energy bar,black tea,salmon,frozen smoothie
34. sparkling water
35. mineral water,eggs,chicken,chocolate,french fries
36. frozen vegetables,spaghetti,yams
37. mineral water,herb, pepper,tomato sauce,light cream
38. mineral water,chocolate,avocado,eggs
39. turkey,french fries,strawberries
40. frozen vegetables,strong cheese,chocolate
41. cookies
42. pickles,spaghetti,salmon,escalope
43. energy bar,french fries
44. red wine,ground beef,mineral water
45. mineral water,cake,cottage cheese
46. pickles,champagne,green tea
47. spaghetti
48. fresh tuna,frozen vegetables,spaghetti,
49. mineral water,honey,whole wheat rice
50. frozen smoothie,escalope
51. spaghetti
52. soup,meatballs,hot dogs,sparkling water
53. escalope
54. soup,avocado,french fries
55. mineral water,chicken,cereals,clothes accessories
56. mineral water,bug spray
57. avocado,muffins
58. burgers,black tea,green tea
59. red wine,ground beef,mineral water
60. mineral water,cake,cottage cheese
61. mineral water,bug spray
62. avocado,muffins
63. burgers,black tea,green tea
64. spaghetti,chocolate,brownies,white wine,green tea
65. fresh tuna,mineral water,eggs
66. spaghetti,muffins
67. spaghetti,chocolate
68. french fries,escalope,champagne
69. tomato sauce,light mayo
70. red wine,ground beef,mineral water
71. honey,oil,cooking oil,low fat yogurt
72. turkey,eggs
73. turkey,fresh tuna,tomatoes
74. spaghetti,mineral water,black tea,salmon,eggs
75. chicken,extra dark chocolate

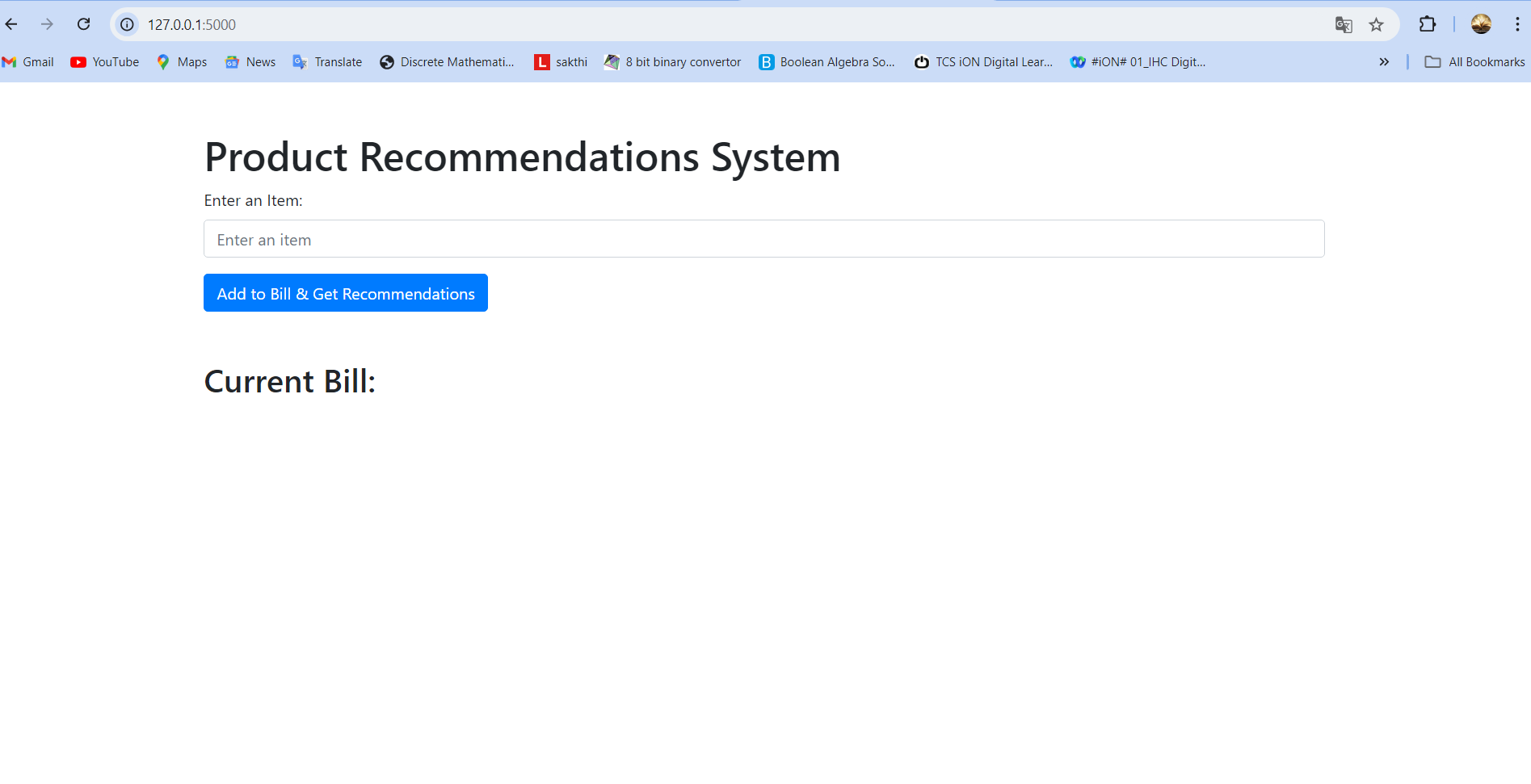
## PERFORMANCE ANALYSIS

## 7.1.1 Test Case Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST**  **CASE ID** | **TESTCASE/**  **ACTION TO BE PERFORMED** | **EXPECTED RESULT** | **ACTUAL RESULT** | **PASS/ FAIL** |
| 1 | Launch the Product recommendation system application | Application opens without errors | Application opens without errors | Pass |
| 2 | Accept the user input data item | Successfully accepts the user input | Successfully accepts the user input | Pass |
| 3 | Add the item to the cart | Item is added to the bill successfully | Item is added to the bill successfully | Pass |
| 4 | Display the recommended items | Frequent items displayed successfully | Frequent items displayed successfully | Pass |
| 5 | Accept the recommended item as input | Input accepted successfully | Input accepted successfully | Pass |

**TEST CASE 1:**

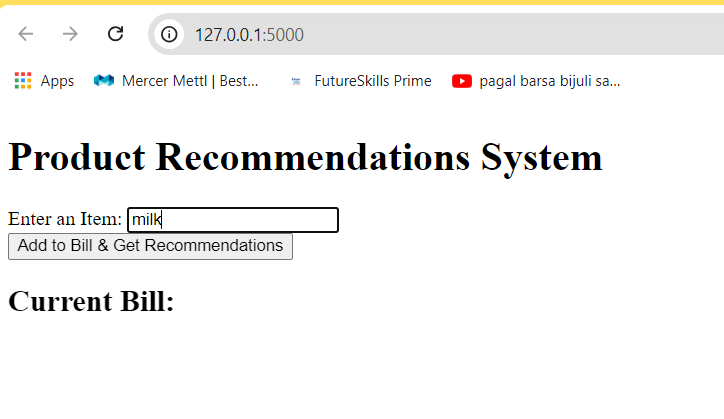
Application opens without errors



**Fig:7.1.1 Test Case 1**

**TEST CASE 2:**

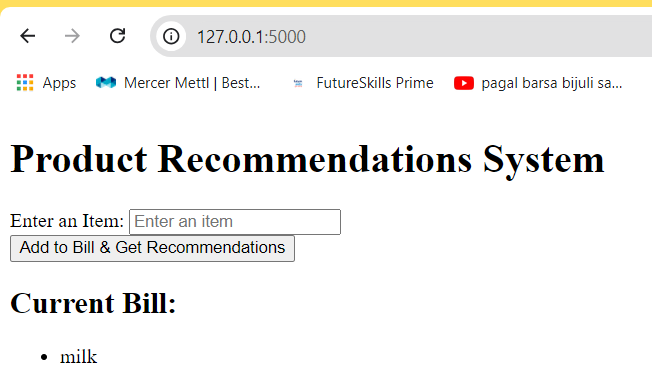
Successfully accepts the user input.



**Fig:7.1.2 Test Case 2**

**TEST CASE 3:**

Item is added to the bill successful.

****

**Fig:7.1.3 Test Case 3**

**TEST CASE 4:**

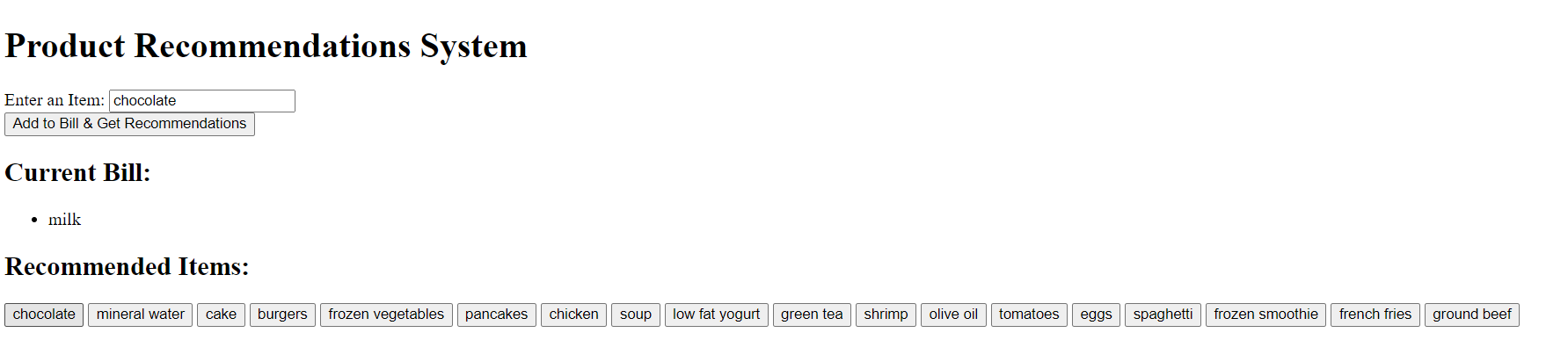
Frequent items displayed successfully.



**Fig:7.1.4 Test Case 4**

**TEST CASE 5:**

Input accepted successfully.



**Fig:7.1.5 Test Case 5**

### Conclusion and Future Enhancements

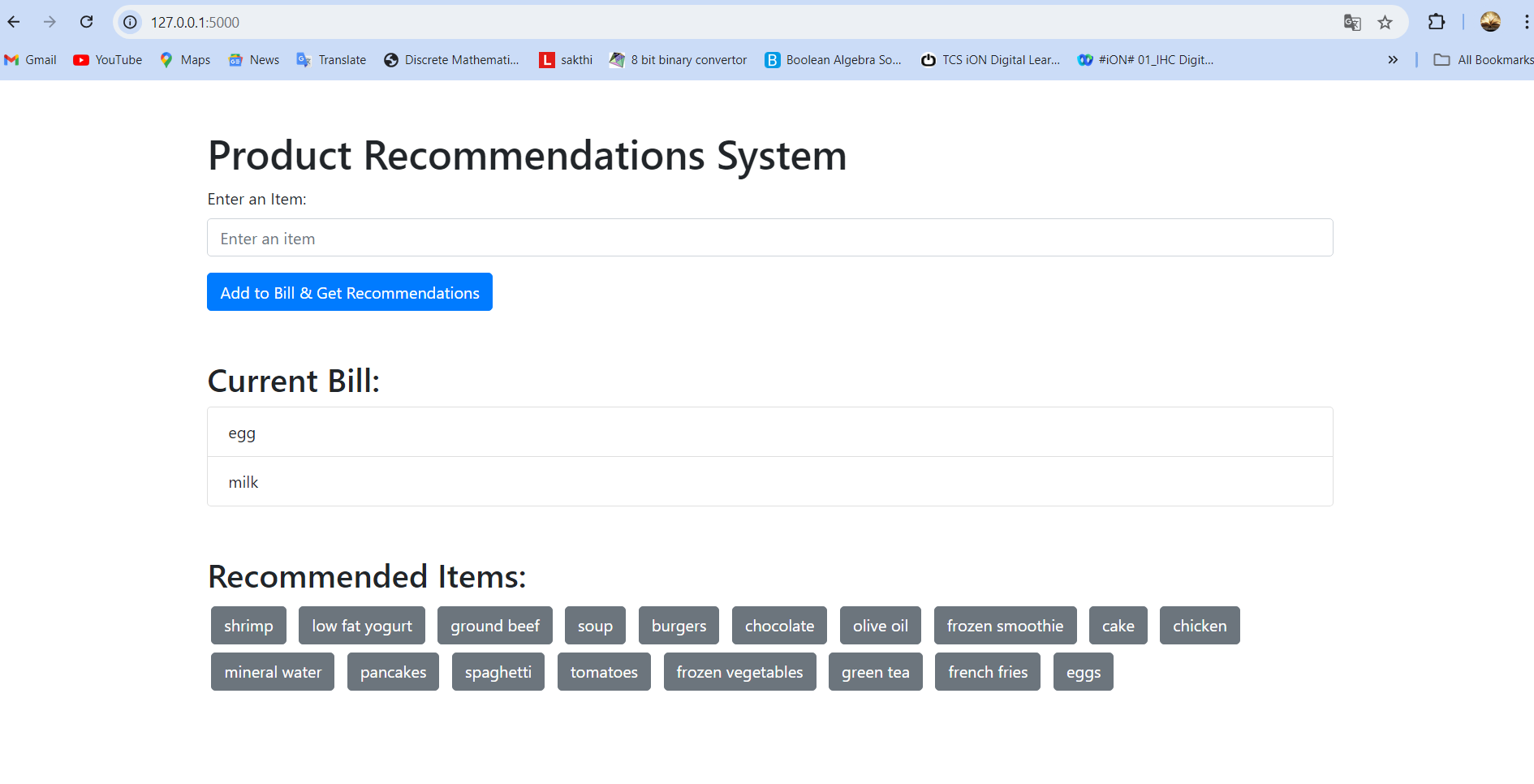
**CONCLUSION:**

In conclusion, this project has successfully developed a product recommendation system using machine learning algorithms and association rule mining, particularly the Apriori algorithm. The system can efficiently recommend products based on user purchasing behavior, helping businesses improve customer engagement and boost sales. By analyzing historical purchase data, the system provides accurate product recommendations, enabling a more personalized shopping experience.

**FUTURE ENHANCEMENT:**

For future enhancements, there are several promising directions. Integrating more advanced recommendation algorithms, such as collaborative filtering or deep learning-based approaches, could further refine the recommendation accuracy. Additionally, expanding the dataset to include user demographics and feedback would enhance personalization. Incorporating real-time data processing could enable dynamic recommendations that respond to customer preferences as they shop. Moreover, adding features like sentiment analysis based on customer reviews can provide deeper insights into product preferences. These improvements will ensure that the recommendation system remains robust, adaptive, and capable of meeting the evolving needs of users and businesses.

**A1.Sample Screen**

****

## REFERENCES

1. Bhardwaj R, Duhoon V (2018) Product recommendation using soft computing techniques. In: International conference on computing, power and communication technologies (GUCON), pp 1111–1115. <https://doi.org/10.1109/GUCON.2018.8675088>
2. Haghbin M, Sharafati A, Motta D (2021) Applications of soft computing models for predicting customer preferences: a comprehensive review and assessment. Prog Earth Planet Sci 8(4). <https://doi.org/10.1186/s40645-020-00400-9>
3. Jayasingh SK, Mantri JK, Pradhan S (2021) Product recommendation using hybrid soft computing models. In: Udgata SK, Sethi S, Srirama SN (eds) Intelligent systems. Lecture notes in networks and systems, vol 185. Springer, Singapore. <https://doi.org/10.1007/978-981-33-6081-5_4>
4. Khajure S, Mohod SW (2019) Future product recommendation using soft computing techniques. Proc Comput Sci 78:402–407. ISSN 1877-0509. <https://doi.org/10.1016/j.procs.2016.02.081>
5. Singh N, Chaturvedi S, Akhter S (2019) Product recommendation using machine learning algorithms. In: International conference on signal processing and communication (ICSC), pp 171–174. <https://doi.org/10.1109/ICSC45622.2019.8938211>
6. Schultz MG, Betancourt C, Gong B, Kleinert F, Langguth M, Leufen LH, Mozaffari A, Stadtler S (2021) Can deep learning beat traditional recommendation systems? Phil Trans R Soc A 379:20200097. <https://doi.org/10.1098/rsta.2020.0097>
7. Sharma A, Agarwal S (2018) Product recommendation using wavelet neural network. Res J Inf Technol 4:22–30. <https://doi.org/10.3923/rjit.2012.22.30>. <https://scialert.net/abstract/?doi=rjit.2012.22.30>
8. Sofian IM, Affandi AK, Iskandar I, Apriani Y (2019) Monthly product demand prediction based on artificial neural networks with backpropagation and radial basis function. Int J Adv Intell Inf 4(2):154–166. ISSN 2548-3161. <https://doi.org/10.26555/ijain.v4i2.208>
9. Vathsala H, Koolagudi SG (2021) Neuro-fuzzy model for product recommendation using data mining and soft computing approaches. IETE J Res. <https://doi.org/10.1080/03772063.2021.1912648>