PAPER NAME

Capstone_Porject.docx

WORD COUNT

8410 Words

CHARACTER COUNT

56599 Characters

PAGE COUNT

71 Pages

FILE SIZE

3.7MB

SUBMISSION DATE

Apr 4, 2024 6:54 PM GMT+5:30

REPORT DATE

Apr 4, 2024 6:55 PM GMT+5:30

20% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 12% Internet database
- 11% Publications database
- Crossref database
- Crossref Posted Content database
- 12% Submitted Works database

Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less then 10 words)

Home Spice Bliss

Project Report submitted in the partial fulfilment

of

Bachelor of Technology (B.Tech)

in

Computer Science and Business Systems (CSBS)

by

Aditya Nair (E049)

Bhavya Barellia (E072)

Shubham Pandit (E078)

Dr. Deepti Reddy Patil

Associate Professor, Computer Engineering Department, MPSTME

SVKM's NMIMS University

(Deemed-to-be University)

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING (MPSTME)

Vile Parle (W), Mumbai-56

2024

CERTIFICATE

This is to certify that the project entitled ("HomeSpice Bliss") has been done by Mr. Aditya Nair, Mr. Bhavya Barellia and Mr. Shubham Pandit under my guidance and supervision & has been submitted in partial fulfilment of the degree of Bachelors of Technology in Computer Science and Business Systems of MPSTME, SVKM's NMIMS (Deemed-to-be University), Mumbai, India.

_____ _____

Project Mentor Examiner (name & signature)

Date

Place: Mumbai

_____

HOD (name & signature)

ACKNOWLEDGEMENT

Name	Roll No.	SAP-ID
Aditya Nair	E049	70362019215
Bhavya Barellia	E072	70362019337
Shubham Pandit	E078	70362019395

Abstract:

In today's world, it's no secret that many food options tend to be bland, unhealthy, and expensive. Often, it feels like if something is tasty, it must be unhealthy, and vice versa.

But what if I told you there's a way to break free from this cycle? That's where Homespice Bliss comes in. We're not selling a pitch; we're offering a solution. Our web app provides a variety of delicious, nutritious, and budget-friendly meal options tailored specifically for hostellers and seniors.

It's about providing information and choices that empower you to enjoy tasty meals without compromising on your health or budget.
To cater to the problem of repetitive, uninspiring meals plaguing individuals, especially hostellers and seniors, we have come up with a solution:

Homespice Bliss. Our innovative web app revolutionizes mealtime by providing access to a diverse network of chefs curated based on location and cuisine preferences. Additionally, we've implemented a recommendation system to ease users in ordering, ensuring they discover new culinary delights effortlessly. With Homespice Bliss, users can break free from dining monotony and embrace a world of flavorful and nutritious meals tailored to their tastes.

Table of Contents

List of the Figures

Chapter 1

Introduction

1.1 Background

Hostel residents are increasingly dissatisfied with monotonous, bland meals prepared by in-house staff and the limited, unhealthy, and expensive options provided by restaurants. These limitations have created a demand for a targeted solution. To address this issue, our app is designed to cater specifically to this demographic, offering diverse, flavorful, and nutritious meal alternatives that align with their preferences and budget constraints.

1.2 Motivation & Scope

The restricted, unhealthy, and pricey menu options offered by eateries and the monotonous, boring meals made by on-site workers are driving away hostel inhabitants. A focused solution is now in demand due to these constraints. In order to solve this problem, our app is made especially for this group of users, providing a variety of tasty and nourishing meal options that fit their budgets and tastes.

1.3 Problem Statement

Hostel residents are dissatisfied with dull, unhealthy meals. To address this issue, our web app, HomeSpice, aims to offer diverse, budget-friendly, and delicious meal alternatives tailored to their preferences.

Our mission is to not only solve this culinary problem but also to significantly enhance the overall hostel dining experience. By providing a comprehensive solution that prioritizes both taste and nutrition while accommodating budget constraints, we aim to foster a more satisfying and enjoyable dining environment for hostel residents.

Through HomeSpice, we seek to transform the way hostel residents perceive and experience their daily meals, ensuring they look forward to each dining occasion with enthusiasm and culinary delight.

1.4 Salient Contribution

This project contributes in a number of ways to the overall society:

- Empowerment of Home Chefs – It provides a platform for home chefs to showcase their skills and reach an audience, and helps create new opportunities.
- Sustainable Practices – Promotes homemade meals over mass-produced and unhealthy meals, thus reducing food wastage and a more environmentally conscious audience is raised.
- Better Eating Habits – Improves the overall health and habits of consumers.
- Cultural Diversity & Exploration – Allows users to explore a diverse range of cuisines and discovery of new flavors and cooking styles.
- Scalability & Growth – Allows for the growth of the application, expand to new regions and further contribute to economic growth.

Chapter 2
Literature Survey

1. Important Contributions, Challenges and Future Research Directions:

This paper offers a comprehensive overview of contemporary food recommender systems, emphasizing their vital role in guiding users toward healthier dietary choices. It explores various algorithms, including collaborative filtering techniques such as Latent Dirichlet Allocation (LDA) and Weighted Matrix Factorization (WRMF), drawing on a substantial dataset from allrecipes.com. Among these, LDA and WRMF emerge as the most effective collaborative filtering methods for food recommendations. Additionally, the paper introduces innovative health-related metrics (FSA and WHO) to assess the nutritional disparity between predicted and actual food selections. Yang et al. (2017) also present promising outcomes by incorporating images and embeddings (Deep Neural Networks - DNNs) to grasp user preferences. Despite these advancements, the study underscores existing research gaps, including the underexplored realm of implicit feedback sources in food recommendation and the challenges posed by limited user feedback and the cold-start problem within this domain. It calls for innovative strategies and metrics to advance the promotion of healthy eating through food recommender systems.

2. Food Recommendation: Framework, Existing Solutions and Challenges:

This paper introduces a novel approach, a unified framework for food recommendation within the multimedia domain, aimed at addressing the pressing issue of unhealthy eating habits and their associated health risks. It underlines the relative lag in advancements in food recommendation compared to domains like movies and points of interest. While the paper delves into critical factors such as context, domain knowledge, and food characteristics in recommendation systems, it falls short of explicitly detailing the specific algorithms or techniques within this framework. Moreover, it lacks empirical evaluations or in-depth results for the proposed approach and existing solutions in food recommendation. Nonetheless, the paper serves as a valuable resource for researchers, offering insights and potential directions to enhance the performance and user experience of food recommendation

systems, ultimately contributing to the combat against global overweight and obesity concerns.

3. Interaction Design in a Mobile Food Recommender System:

This paper introduces a user-centric mobile app for food recipe suggestions, focusing on ease of use and personalized recommendations. It successfully conducts real user tests, demonstrating user satisfaction and usability. However, the study primarily emphasizes interaction design and preference elicitation, neglecting critical aspects like evaluation metrics and scalability. Furthermore, it lacks insights into recommendation diversity and novelty. The paper overlooks potential challenges in collecting user preferences and falls short in discussing user engagement and satisfaction as key metrics.

4. Market2Dish: Health-aware Food Recommendation:

This paper talks about the need for food recommendations that consider a person's health. Most research only looks at finding recipes, what people like, and nutrition, but they don't focus on personalized health. The paper's method has three parts: finding recipes, figuring out a person's health from their social media, and giving food recommendations based on health. They use a special computer system for this. They tested their method a lot, and it seemed to work well, but they didn't provide specific numbers to show how accurate it is compared to other methods. In conclusion, the paper introduces a way to recommend healthy food based on what ingredients you have. It seems to do a good job, but more detailed comparisons with other methods are needed.

40 5. Using Tags and Latent Factors in a Food Recommender System:

This paper introduces an innovative approach to food recommendation by incorporating user ratings and tags to enhance personalized suggestions. The inclusion of tags significantly improves prediction accuracy, outperforming state-of-the-art algorithms. However, it lacks exploration of negative tags, online evaluation, and real-world deployment. Additionally, it focuses on tags and latent factors but overlooks factors like dietary restrictions and cultural preferences. A comprehensive comparison

with existing systems is missing. Scalability and potential biases in user data are unaddressed, and the system's adaptability to diverse contexts remains unexplored, presenting avenues for future research in this domain.

6. Food Recommendation System Based on Content Based and Collaborative Filtering Techniques:

The Python programming language and K-nearest neighbor techniques are used in the paper to create a food recommendation system. Using content-based filtering, the system suggests food based on the food's name, food ID, cuisine type, and diet type (vegetarian or non-vegetarian). By comparing linguistic properties of food products using the string-matching algorithm's cosine similarity distance, the system's capacity to recognize textual similarities is improved. For every food item, a feature vector is created using the outcomes of string-matching algorithms and numerical properties. To ensure a declining order of choice, the recommendations are sorted based on food names and average ratings. The performance of the proposed model is evaluated using error values like root mean square error (RMSE) and mean absolute error (MAE). The authors plan to make improvements based on the obtained error values and quantify the accuracy of the recommendations

The paper utilizes content-based filtering and collaborative filtering techniques for food recommendation. Content-based filtering involves using textual information associated with food items to identify similarities and make recommendations based on these similarities. Cosine similarity is used to calculate the similarity between the preferences or characteristics of different food items or user profiles. String matching algorithm cosine similarity distance is applied to compare the textual attributes between food items. Feature vectors are constructed for each food item, incorporating numerical attributes and the results of string-matching algorithms.

7. Hybrid Recommendation System with Graph based and Collaborative Filtering Recommendation Systems

The paper discusses the need for a simple recommendation system solution that combines the predictions of two or more recommendation techniques. It proposes a hybrid recommendation system that combines product-based predictions using a graph-based recommendation system and user-based predictions using collaborative

filtering. The paper also mentions the possibility of implementing and integrating various types of recommender systems to fine-tune recommendations. It refers to a literature survey on research paper recommender systems that reviews different methods, including graph-based methods. The author highlights the use of clustering and classification models for product recommendations and the application of machine learning models, such as random forest and K-NN, in recommendation systems.

The paper briefly explains the collaborative filtering technique, which uses similarities between users and their interactions with products to make recommendations. The proposed hybrid recommendation system combines the predictions from a graph-based recommendation system and a collaborative filtering recommendation system. The graph-based recommendation system uses network analysis or link analysis to identify relationships in a network and create recommendations based on strong connections. The collaborative filtering recommendation system utilizes similarities between users and their interactions with products to make personalized recommendations. By combining the predictions from both systems, the hybrid recommendation system aims to provide more accurate and diverse recommendations to users. The system takes into account both product-based predictions and user-based predictions, leveraging the strengths of both approaches. The hybrid system is demonstrated in the context of a movie recommendation system, using a dataset that contains movie details and ratings from IMDB. The effectiveness of the hybrid recommendation system can be evaluated by comparing its recommendations against human behavior and preferences.

8. User Profile Feature-Based Approach to Address the Cold Start Problem in Collaborative Filtering for Personalized Movie Recommendation

The paper proposes using a feature-based profile approach, collaborative filtering for individualized movie recommendations can overcome the cold start issue. The authors optimize the prediction algorithm's input parameters utilized in the recommender system by using the relationship between user feature-scores acquired from user-item interaction via ratings. a user profile feature-based approach to address the cold start problem in collaborative filtering for personalized movie recommendation. The authors apply the relationship of user feature-scores derived from user-item

interaction via ratings to optimize the prediction algorithm's input parameters used in the recommender system. The proposed approach improves the accuracy of predictions with less past user records and shows an improvement of 8.4% compared to the base collaborative filtering algorithm. The user-feature generation and evaluation of the system are carried out using the 'MovieLens 100k dataset'. The proposed system can be generalized to other domains as well. A user profile feature-based technique is the methodology suggested in the research to solve the cold start issue in collaborative filtering for tailored movie recommendations. The authors optimise the prediction algorithm's input parameters utilised in the recommender system by using the relationship between user feature-scores acquired from user-item interaction via ratings.

The authors optimise the prediction algorithm's input parameters utilised in the recommender system by using the relationship between user feature-scores acquired from user-item interaction via ratings. They use the 'MovieLens 100k dataset' for user-feature generation and evaluation of the system. The authors compare the performance of their approach with the base collaborative filtering algorithm and show an improvement of 8.4% in accuracy. The proposed approach shows an improvement of 8.4% compared to the base collaborative filtering algorithm in terms of accuracy of predictions. The user profile feature-based approach optimizes the prediction algorithm's input parameters, resulting in improved accuracy with less past user records. The evaluation results show that the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics are used to evaluate the accuracy of the recommendation algorithm. The proposed approach outperforms other similar approaches in addressing the cold start problem.

9. Movie Recommendation System using RNN and Cognitive thinking

The paper proposes a movie recommendation system that utilizes an ensemble recommendation system model, incorporating cognitive thinking based on the age of the user to provide personalized recommendations. The system uses Recurrent Neural Networks (RNNs) to effectively process sequential data and capture temporal dependencies in movie genres. The IMDb Bollywood movies dataset is used, which includes features such as genre, director, year of release, average rating, total votes received, duration, and cast.

The paper conducts a comprehensive analysis of existing movie recommendation algorithms and highlights the use of RNNs in processing natural language and speech recognition. The Singular Value Decomposition (SVD) technique is used to predict user-item evaluations and identify latent features in the dataset.

The system combines multiple models in parallel using ensemble learning to provide better recommendations based on user preferences and likes/dislikes. The movie recommendation system utilizes an ensemble recommendation system model that combines the results of four different individual learners to provide near-perfect recommendations to users.

The system incorporates cognitive thinking by considering the age of the user and recommending genres based on age group psychology. Recurrent Neural Networks (RNNs) are used to effectively process sequential data and capture temporal dependencies in movie genres.

The system uses the IMDb Bollywood movies dataset, which includes features such as genre, director, year of release, average rating, total votes received, duration, and cast. The evaluation metrics of the RNN model, which takes age as a parameter, show the effectiveness of considering age while recommending movies, with lower Mean Absolute Error (MAE) and Mean Squared Error (MSE) compared to the standard RNN model. The system provides recommendations for the top 10 films with the highest average rating based on the anticipated ratings of the model. The popularity-based recommendation system, which recommends movies based on overall popularity, is simple to implement and can provide good recommendations for new or infrequent users.

10. Coffee Shop Recommendation System Using an Item-Based Collaborative Filtering Approach:

The paper proposes a coffee shop recommendation system to help users choose the right coffee shop based on their preferences. The recommendation system uses item-based collaborative filtering, which compares the ratings of different coffee shops to generate a list of recommendations. The system calculates the similarity between coffee shops using the adjusted cosine similarity algorithm.

The evaluation of the recommendation system can be done using metrics like Mean Absolute Error (MAE), which measures the difference between predicted and actual ratings. The research follows the ADDIE model, consisting of analysis, design, development, implementation, and evaluation stages.

Data collection for the research includes literature review, interviews, and questionnaires with customers and coffee shop owners in Padang city. The system development stage involves designing a conceptual and system design, followed by implementation and testing.

The evaluation stage measures the effectiveness of the system and user performance, ensuring it meets user requirements. The paper provides a list of coffee shops used in the research for calculating similarity values.

11. Food Recommendation System Based on Collaborative Filtering and Taste Profiling

The paper presents a recommendation system for ordering food at restaurants, consisting of three options: personalized recommendations based on collaborative filtering and targeted models, alternative recommendations based on similar ingredients and preparation methods, and recommendations for increasing revenue per order based on association rule mining. The system utilizes a database of food recipes from Food.com, which provides details about individual dishes, ingredients, preparation methods, nutritional values, and user reviews. The recommendation engine combines collaborative filtering and targeted models to suggest menu items based on a user's previous orders and detailed comparison of reviews. It also generates similar item recommendations by analyzing the characteristics of liked dishes, such as ingredients and preparation methods. The paper acknowledges the limitations of using publicly available datasets and suggests leveraging additional parameters, such as dish price and time of day, to enhance the recommender system. Overall, the paper proposes a comprehensive food recommendation system that considers user preferences, dish characteristics, and association rules to facilitate personalized and alternative food choices at restaurants.

12. Intelligent Movie Recommendation System Based on Hybrid Recommendation Algorithms

A hybrid recommendation algorithm-based movie recommendation system is presented in this paper. By suggesting films that suit their tastes, the system hopes to cut down on the amount of time users spend looking for information and improve the effectiveness of their searches. The 9 HomeSpice Bliss 2024 system's hybrid recommendation algorithm outperformed the conventional CB, Item-Based CF, and User-Based CF algorithms with an accuracy rate of 81%. The recommendation system combines the recommendation results of three recommendation algorithms and filters out unwanted recommendation items. It then adjusts the ranking of the recommendation results based on the user's usage scenario to provide personalized recommendations. The collaborative filtering (CF) algorithm is one of the main recommendation algorithms used in the system. It predicts the behavior trajectory of target users based on associated neighbors and provides personalized recommendations based on users' historical behavior data. The CF algorithm only requires the correlation matrix between the item and the user, instead of extracting features from the content of the recommended item. For example, when recommending a movie, the CF algorithm only needs the rating data of the movie by the user. The overall system architecture of the recommendation system includes a front-end display layer, recommendation business layer, model training layer, and data processing layer. The front-end display layer is responsible for displaying the front-end page of the recommendation system, while the recommendation business layer focuses on the recommendation process.

13. A systematic review on food recommender systems

The paper provides a systematic review of food recommendation systems, focusing on the techniques used, evaluation methods employed, and limitations identified in the primary studies. The study selection process involved a significant reduction in the number of collected papers, indicating challenges in finding relevant and high-quality research in this domain. The evaluation of recommendations emerged as a common limitation, with some studies lacking any evaluation or using few metrics. The authors

highlight the need for more comprehensive evaluation strategies in food recommendation research.

Furthermore, the paper discusses the increasing trend in research efforts in food recommendation systems, with a slight decrease in publications in 2019 and 2020. Various databases, including ISI Web of Science, SCOPUS, ACM, and IEEE, were used to retrieve relevant papers. The techniques used in food recommendation systems are explored, shedding light on the different approaches and methodologies employed in this field.

Overall, the paper provides valuable insights into the current state of food recommendation systems, highlighting the challenges, trends, and opportunities for future research in this area. The authors emphasize the importance of robust evaluation methods and the need for continued advancements in food recommendation technology to enhance user experience and satisfaction.

14. A Food Recommender System Considering Nutritional Information and User Preferences

The paper "A Food Recommender System Considering Nutritional Information and User Preferences" presents a comprehensive framework for a food recommender system that takes into account both nutritional information and user preferences. The research addresses the challenges in the development of food recommendation technologies, emphasizing the importance of incorporating nutritional concepts, user preferences, and personalized nutrition into computational models. The study highlights the limited depth of nutritional incorporation in existing models and the scarcity of works focusing on processing user preferences in personalized nutrition scenarios.

Key contributions of the research include:

Creation of a meal recommendation model that incorporates data on consumer preferences and nutrition. Integration of nutritional data with Multi-Criteria Decision Making (MCDM) sorting methods in the food recommendation sector. use of user

profiling techniques based on feedback in the field of meal suggestion. The paper is organized into sections covering background information on recommender systems, previous works in food recommendation, and an overview of the proposed food recommendation process. It introduces an architecture for food recommendation comprising information gathering, user profiling, an intelligent system layer, and an end-user interface. The framework includes an AHPSort-based pre-filtering stage and an optimization-based approach for menu generation, focusing on maximizing user preferences while ensuring nutritional requirements are met.

In comparison to traditional optimization-based menu generation approaches and typical recommender system methods, the proposed framework stands out for its emphasis on user preferences and nutritional considerations. The research aims to generate personalized meal plans for users based on their nutritional needs and food preferences, filling a gap in existing computational tools for food intake advice. Overall, the paper contributes to the advancement of food recommender systems by providing a unique method of individualized meal recommendations by combining preference-based and nutritional data into a single recommendation model.

15. A Review of Academic Recommendation Systems Based on Intelligent Recommendation Algorithms

The explosive growth of academic information on the Internet has made it challenging for researchers to efficiently retrieve relevant academic materials. Recommendation systems play a crucial role in extracting user interest characteristics from raw data and providing personalized recommendations. The integration of deep learning with recommender systems has emerged as a new trend in enhancing recommendation algorithms.

Brief Description of Recommendation Algorithms:

Traditional recommendation algorithms are categorized into content-based, collaborative filtering, and hybrid recommendation algorithms. Content-based algorithms analyze item features to recommend similar items, while collaborative filtering algorithms make recommendations based on user behavior and preferences.

The paper discusses the calculation of keyword frequency, inverse document frequency, and cosine similarity in the context of recommendation algorithms.

The document highlights existing academic recommendation systems such as Academia.edu, ResearchGate, Semantic Scholar, and PubMed, which utilize traditional content-based recommendation algorithms.

Researchers like Kazunari Sugiyama have developed systems that recommend academic papers based on user preferences and classification into different researcher categories. Collaborative filtering recommendation algorithms have been employed to address data sparsity issues and improve recommendation accuracy in academic research.

Recent advancements in academic recommendation systems include the integration of convolutional text and heterogeneous information networks to enhance recommendation diversity. Models like the implicit feedback-based convolutional depth semantic matching paper recommendation model have been proposed to learn semantic expressions and improve recommendation accuracy.

16. A Review of Healthcare Recommendation Systems Using Several Categories of Filtering and Machine Learning-Based Methods

The paper emphasizes the importance of healthcare recommendation systems in providing better health services to patients and assisting healthcare professionals in decision-making. It highlights the unique needs and challenges of implementing recommendation systems in the medical healthcare domain compared to other industries like e-learning and e-commerce.

Various classification and recommendation models are presented for healthcare systems, aiming to enhance the quality of recommendations for users. The development of healthcare recommendation systems involves evaluating and implementing models using deep learning (DL) and machine learning (ML) methods.

The document includes a comparison analysis of different healthcare recommendation systems based on categories such as Content-Based (CB), Collaborative Filtering (CF), Knowledge-Based (KB), Context-Aware (CA), Fuzzy Linguistic (FL), and Text

Classification (TC). Specific healthcare applications like chronic disease diagnosis, dietary recommendation services, doctor selection, and clinical decision support are discussed. The paper reviews the outcomes of existing proposed healthcare recommendation systems and provides insights into the future scope of research in this field. It outlines the phases involved in developing recommendation systems, including information collection, learning, and prediction/recommendation phases.

17. A Study on Product Recommendation System based on Deep Learning and Collaborative Filtering

The paper analyzes and contrasts the literature on AI-based product recommendation systems, highlighting the fundamental ideas, experimental techniques, and performance assessment criteria used by different researchers The authors have developed a system that suggests products to users based on ML algorithms, establishing a link between people and products The paper presents a flowchart of the recommended method, which combines AI with machine learning, NLP, and natural language processing to simplify utility and enhance user satisfaction The suggested system includes a GUI with a login page for users to access the product recommendation system The paper concludes by emphasizing the importance of comparative analysis of different deep learning and collaborative filtering-based product recommendation systems and highlights the successful implementation of their suggested approach.

18. An Intelligent Recommendation System for Performance Equipment Operation and Maintenance via Deep Neural Network and Attention Mechanism

The paper proposes an intelligent recommendation system for performance equipment operation and maintenance based on deep learning network and attention mechanism. The system uses a deep neural network for feature extraction of performing arts equipment and integrates an attention mechanism to recommend equipment that maintenance personnel may be interested in. The system calculates the cosine similarity between the calculated features to generate recommended results. The paper

utilizes the Adam optimizer to optimize the parameters, sets the learning rate to 0.0005, and uses a dropout layer to prevent overfitting. The feature data is processed by transforming category fields into numbers and using them as indices for the embedding layer. The embedding layer transforms the index value into a dense vector with a fixed size. The deep neural network (DNN) used in the system has fully connected layers and employs forward propagation and backpropagation algorithms for calculating outputs and updating parameters, respectively.

19. Collaborative Filtering Recommendation with Fluctuations of User' Preference

The paper introduces a method that incorporates personal preference fluctuations into traditional collaborative filtering systems. It quantifies user's preference change and recommends items that dissimilar users have rated to the pointed user. The experiment conducted on MovieLens datasets shows that the method including changes of personal preference performs better than other methods. The performance of the proposed method is evaluated using MAE, NMAE, and F1-measure. The results show that the method outperforms other methods in terms of accuracy and recommendation quality. The paper discusses the concept of fluctuation in recommender systems and how it can be combined with traditional collaborative filtering approaches. The paper introduces a method that incorporates personal preference fluctuations into traditional collaborative filtering systems.The process involves quantifying user's preference change and recommending items that dissimilar users have rated to the pointed user Experimental results on MovieLens datasets show that the method including changes of personal preference performs better than other methods. The method is evaluated using MAE, NMAE, and F1-measure, and it outperforms other methods in terms of accuracy and recommendation quality. The paper discusses the concept of fluctuation in recommender systems and how it can be combined with traditional collaborative filtering approaches. The conclusion of the paper is that incorporating personal preference fluctuations improves the performance of collaborative filtering recommendation systems

20. Content-Based Recommendation Using Machine Learning

The paper proposes a three-step profiling method for accurately capturing users' profiles in a content-based recommendation system. The method includes purchase item prediction using Logistic Regression, purchase category prediction using Support Vector Machine (SVM), and user's rating prediction using Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). The proposed method outperformed the baseline model on a user dataset collected from Amazon, demonstrating its ability to provide reasonable recommendations for online purchases. The paper also mentions the potential use of video signal processing techniques to capture users' facial expressions for better recommendation in the future. The paper provides a recommender system trained on 200,000 pieces of data collected from Amazon, including purchased item, item category, price, and user ratings. The paper presents a reasonable approach to capturing users' profiles, enabling personalized recommendations. The proposed method outperforms current popular recommender system models.

21. E-Commerce Intelligent Recommendation System Based on Deep Learning

The paper proposes an e-commerce intelligent recommendation system (IRS) based on deep learning, which aims to improve the recommendation accuracy in the field of e-commerce. The overall design, functional modules, and system architecture of the e-commerce IRS are discussed. The paper focuses on the recommendation algorithm in the e-commerce IRS and optimizes it based on convolutional neural network (CNN). The performance of three popular recommendation algorithms is compared and analyzed using the Alibaba dataset. The experimental results show that the proposed CNN recommendation algorithm based on deep learning outperforms the other two algorithms. The evaluation of the recommendation performance is done using recall rate and NDCG indicators. Recall rate measures the percentage of successfully recommended products, while NDCG is used to evaluate the ranking quality of information retrieval. The paper acknowledges that although the proposed algorithm improves the recommendation results compared to traditional algorithms, the accuracy is still low. Further research is needed to adjust the algorithm structure and improve the accuracy of the results.

22. Food Recommendation using Ontology and Heuristics

The study extends the Hermes framework with meal recommendation functionality by proposing a food personalization framework based on adaptive hypermedia. It integrates a conventional food database, healthful heuristics, and the TF-IDF term extraction approach with the cosine similarity measure into the knowledge-base. The framework uses a domain ontology extracted from the United States Department of Agriculture (USDA) food database and allows users to select relevant concepts. Semantic recommender systems, like the proposed one, outperform traditional recommender systems in terms of accuracy, precision, recall, and F-measure. The framework includes steps such as preprocessing user input, applying nutrition heuristics, matching descriptions to the knowledge base, and constructing user profiles using TF-IDF values. The similarity between food items is calculated using semantic measures and the ontology. TF and IDF values are computed to determine the relevance of food items. Future research directions include extending the querying language, exploring alternative weighting schemes and similarity functions, and investigating other types of food recommendation services.

23. User Profile-Based Recommendation Engine Mitigating the Cold-Start Problem

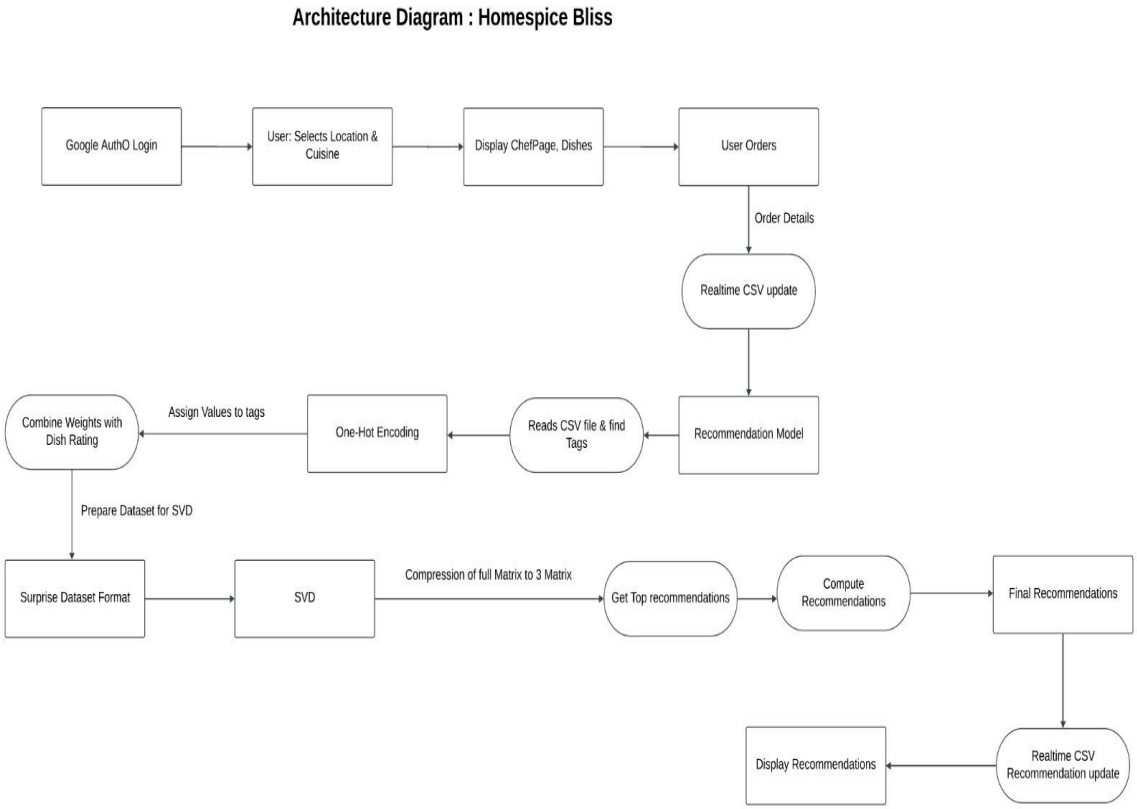
The paper proposes a recommendation engine that addresses the cold-start problem in recommendation systems, where no information about new or infrequent users is available. The engine generates restaurant and category recommendations for restaurant visitors using clustering techniques, collaborative filtering methods, and spatial information. It mitigates the cold-start problem by using matrix factorization and spatial information for users with limited restaurant visits. Recommendations are evaluated and adapted based on user behavior to improve results. The engine provides recommendations through an API, combining location and user-based recommendations to cater to users' needs. Clustering-based Recommendations, Feature-Matrix-based Recommendations, Location-based Recommendations. The recommendation engine implements various methods for creating recommendations, including clustering-based, feature-matrix-based, and location-based techniques. It tackles the cold-start problem by providing personalized recommendations with

limited user data. Future work includes incorporating trustworthiness information about restaurants and leveraging user opinions to reduce the cold-start problem. The paper presents a microservice-based architecture that addresses the cold-start problem by combining spatial data and user profiles. Spatial information is used to recommend restaurants within a certain radius of the visited location. User-based recommendations are created using persisted clustering to save computation time. Future work includes considering trustworthiness information and user opinions to further reduce the cold-start problem. The recommendation engine's results can be queried via an API, providing recommended locations and categories for a given user.

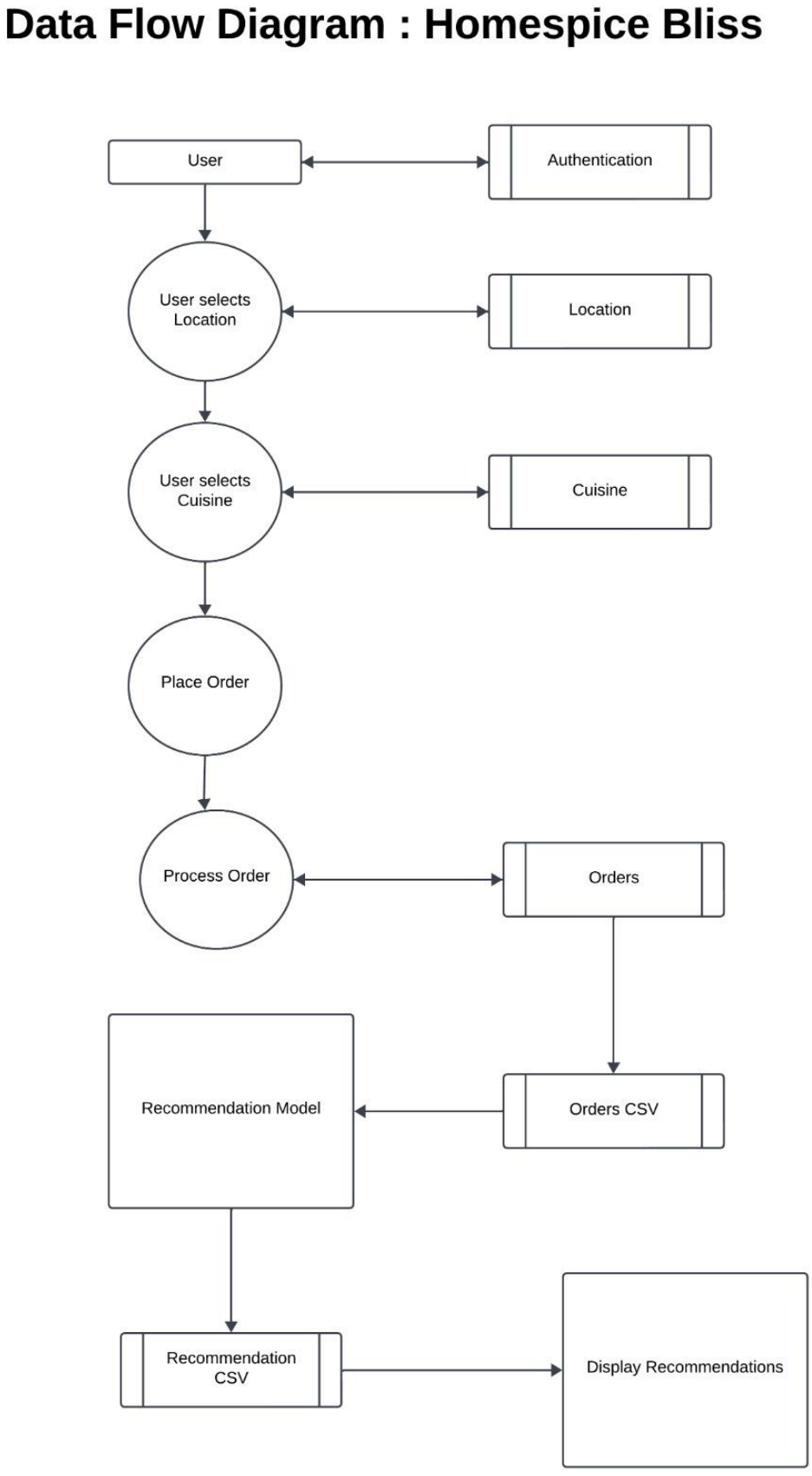
Chapter 3

Methodology & Implementation

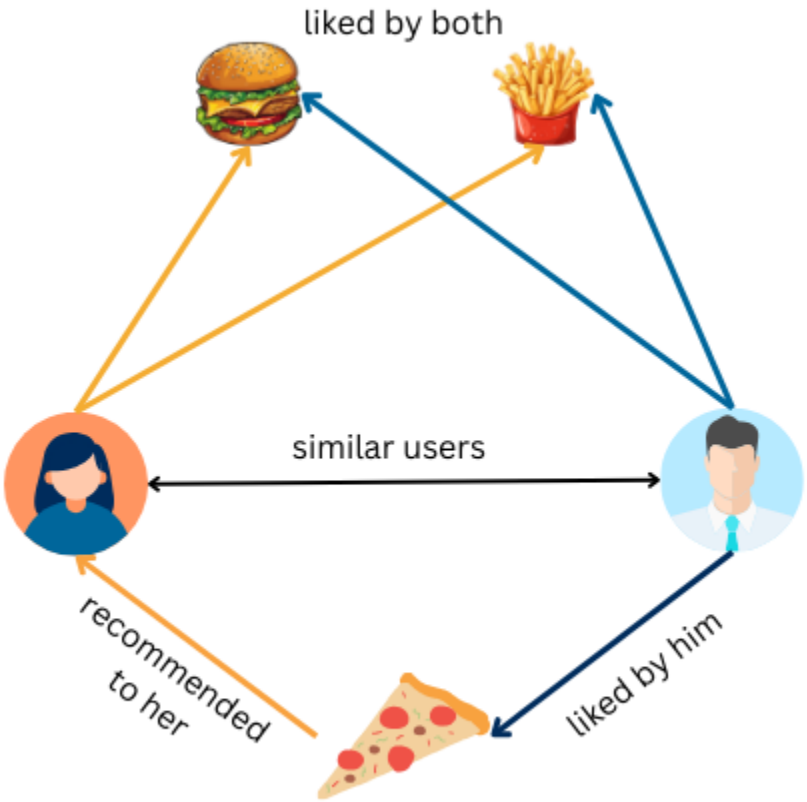
3.1 Block Diagram



Architecture Diagram



Data Flow Diagram



Collaborative Filtering

3.2 Hardware Description

3.3 Software Description

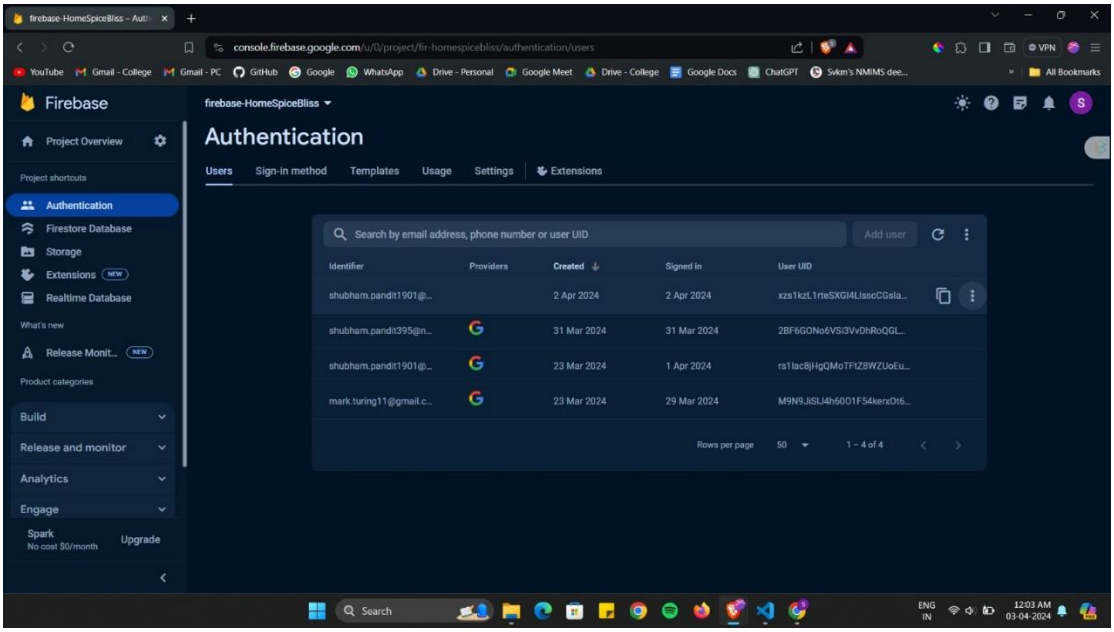
The algorithm used in the application is Tag based Collaborative Filtering System:

It is a combination of using tags and latent factors to identify the various dishes in various categories and using those tags to identify and recommend dishes which are similar to the selected/ordered dish.

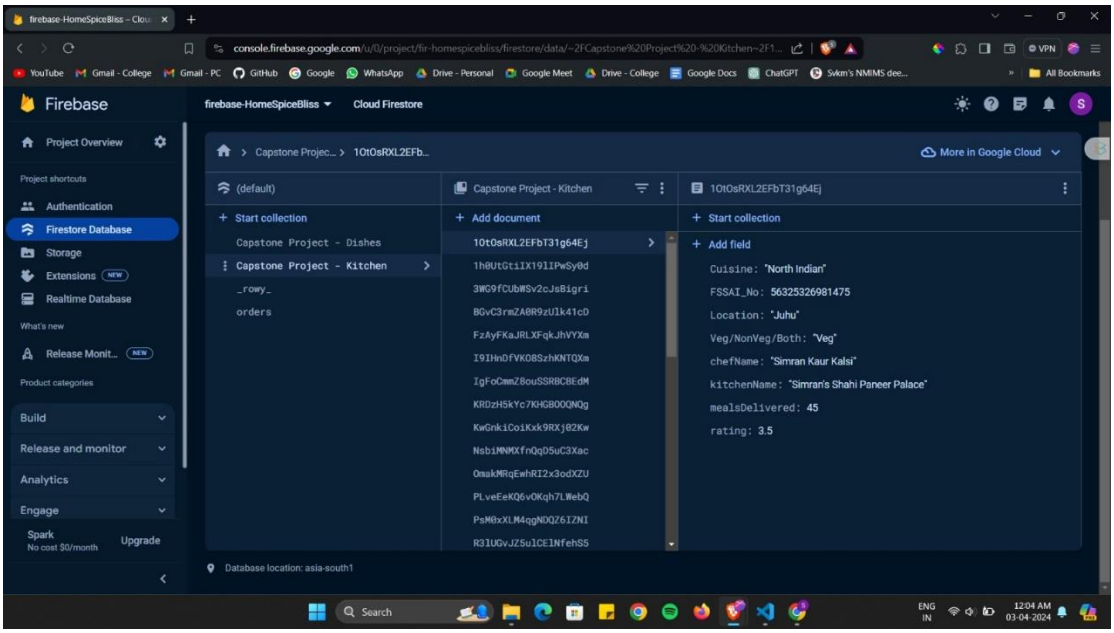
Frontend Technologies: Next.js

Database: Firebase (Firestore and Realtime Firebase), CSV

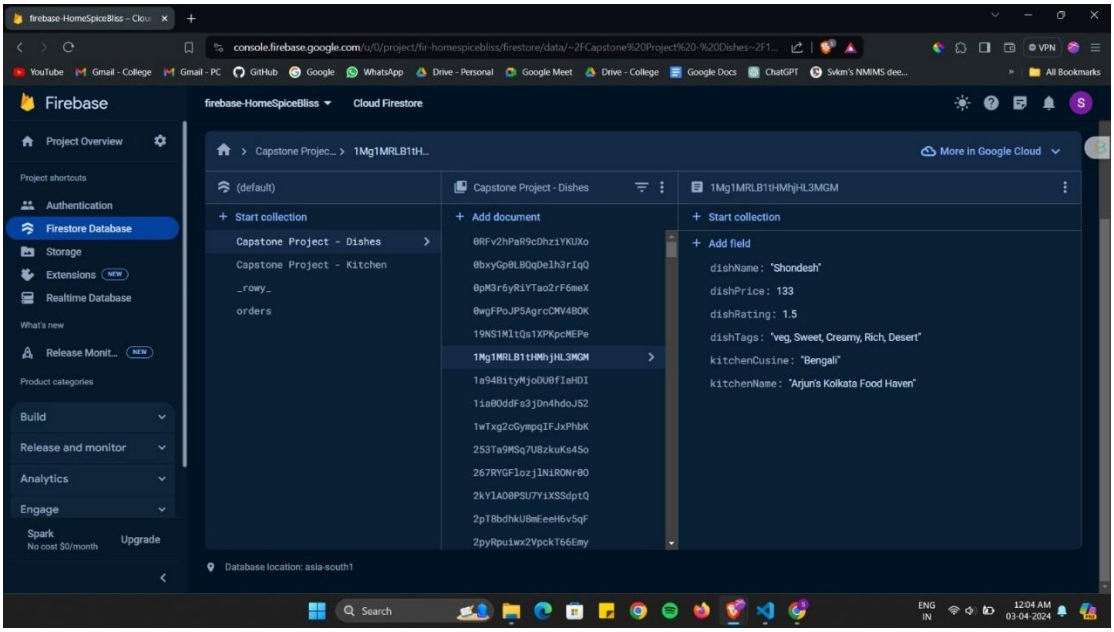
Recommendation System: One-hot encoding, Singular Value Decomposition (SVD), Pandas, Surprise, Itertools



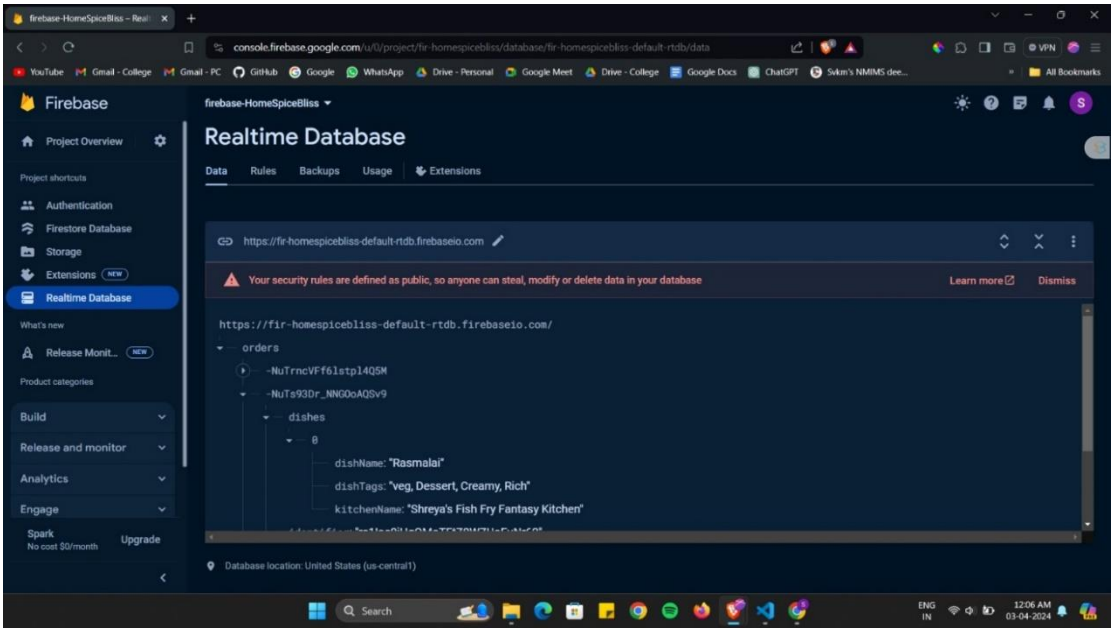
Authentication



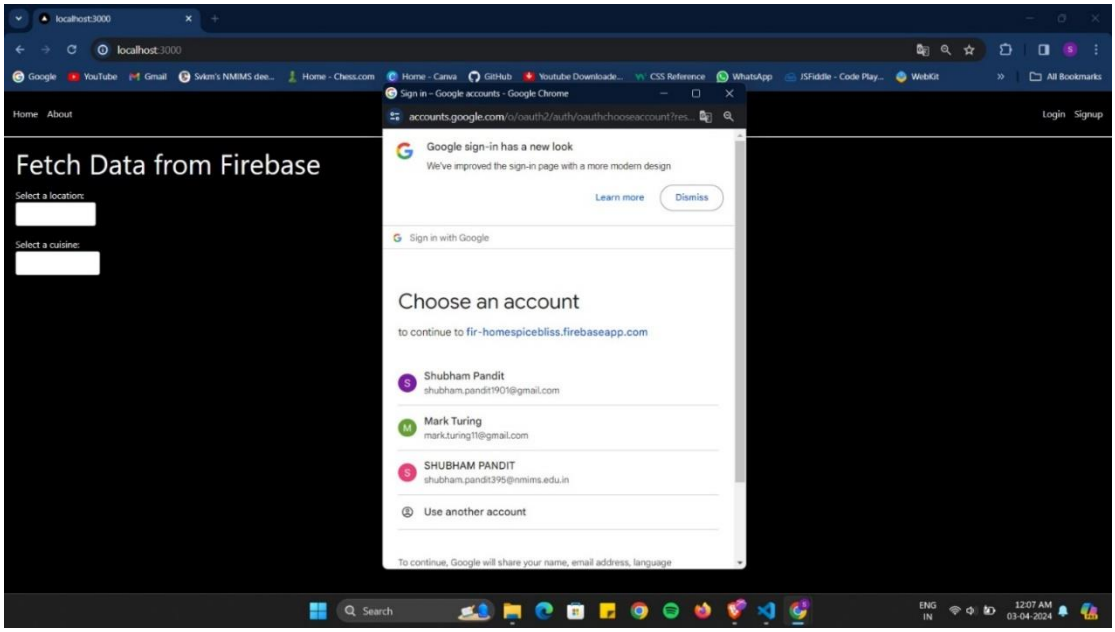
Kitchens Data



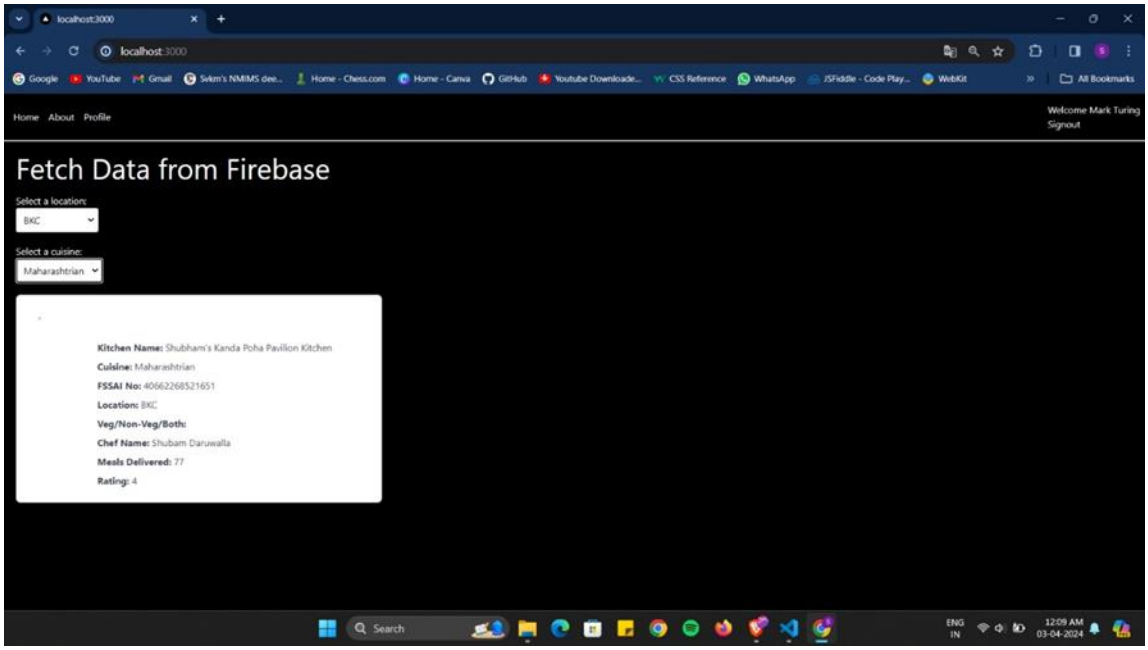
Dishes Data



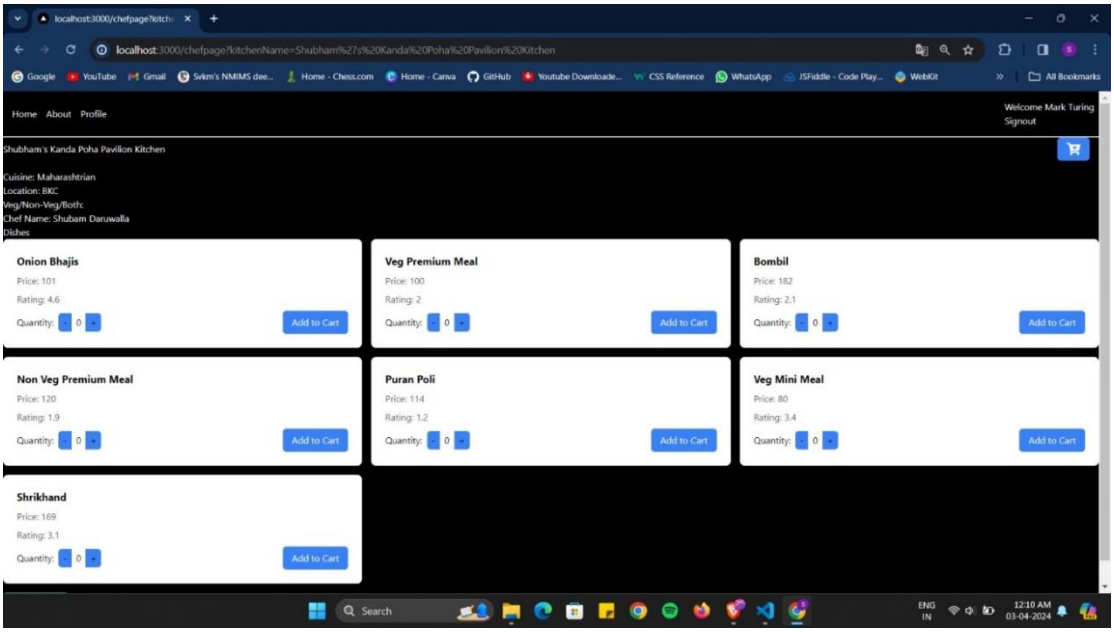
Order Placed-Details : Stored in Realtime Database in firebase



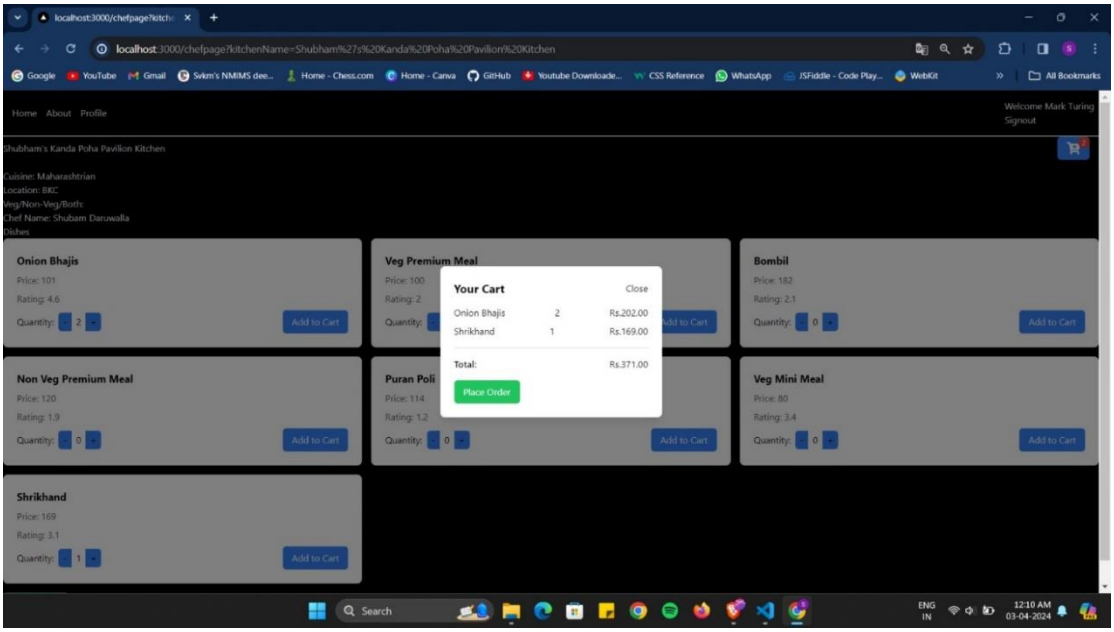
Login with Google AuthO



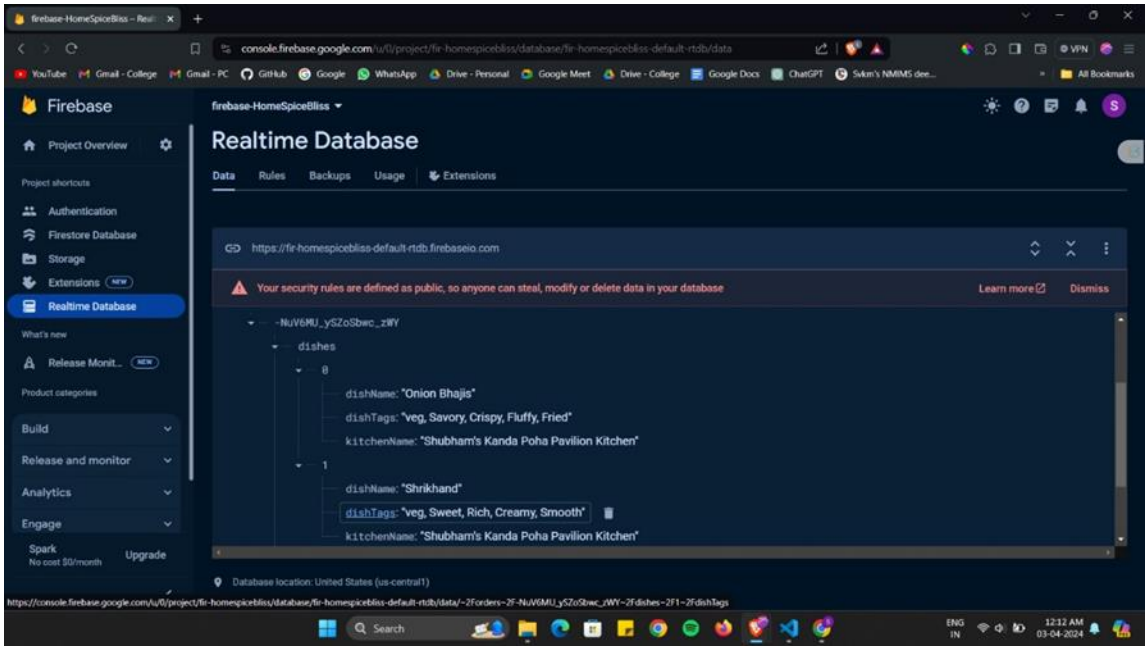
Filtered by Location and then Cusine



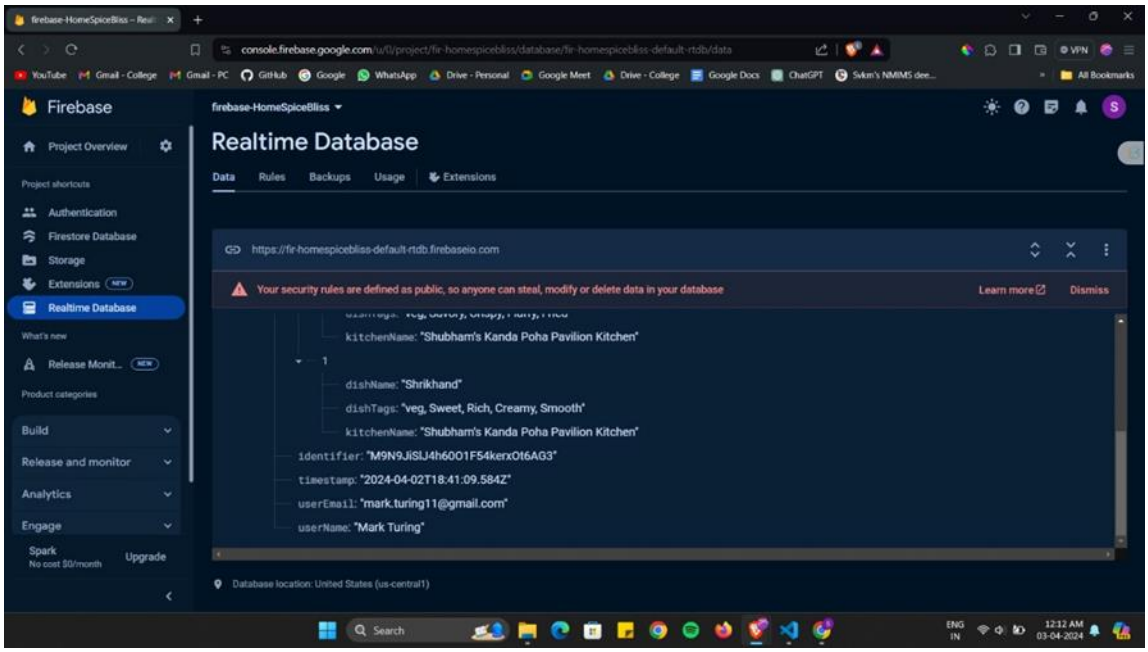
After selection of Kitchen., dishes are displayed



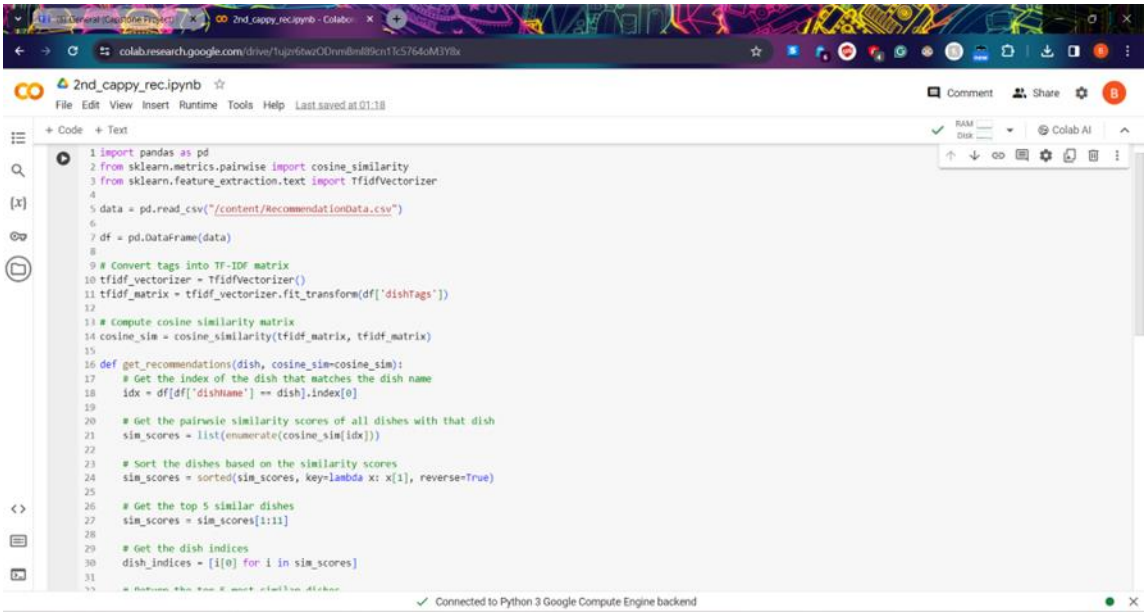
Dishes selected and added to cart



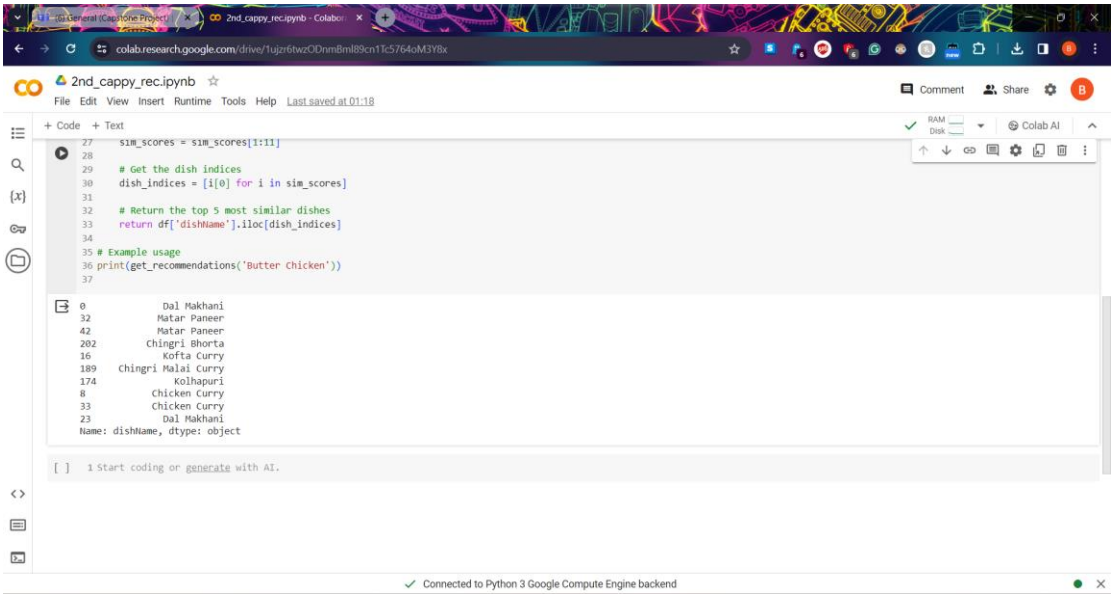
Order Updated into realtime database



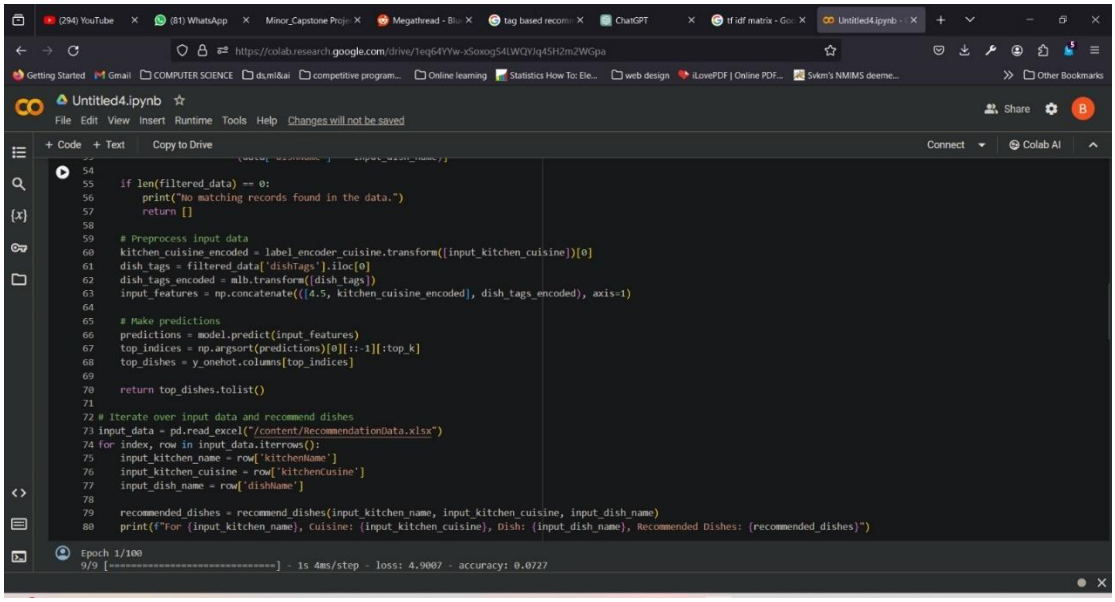
Order details updated with userEmail, userName and timestamps



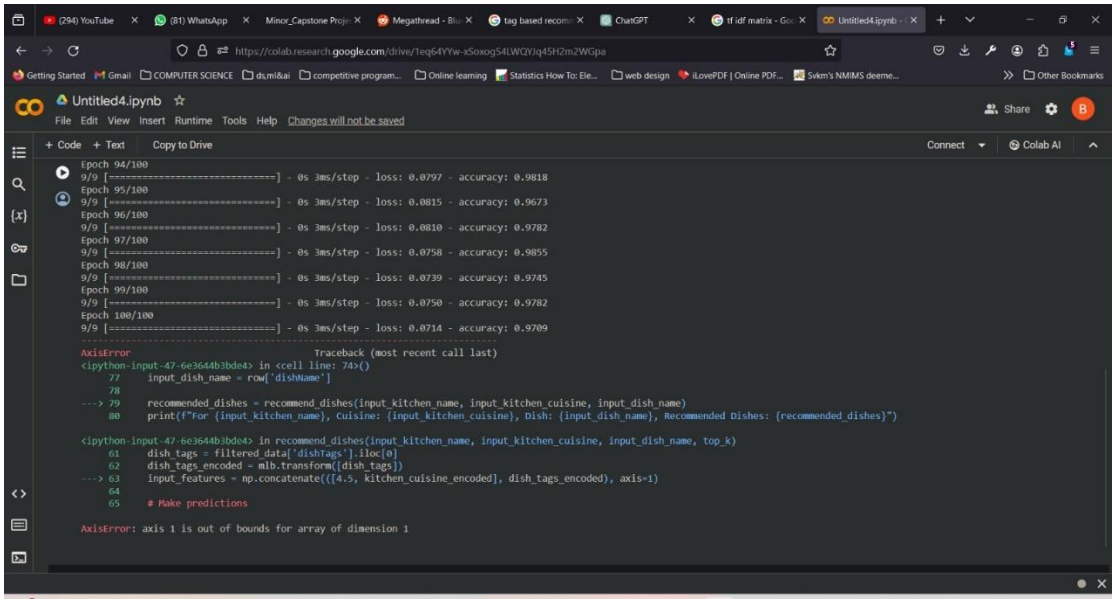
Tags based collaborative filtering source code



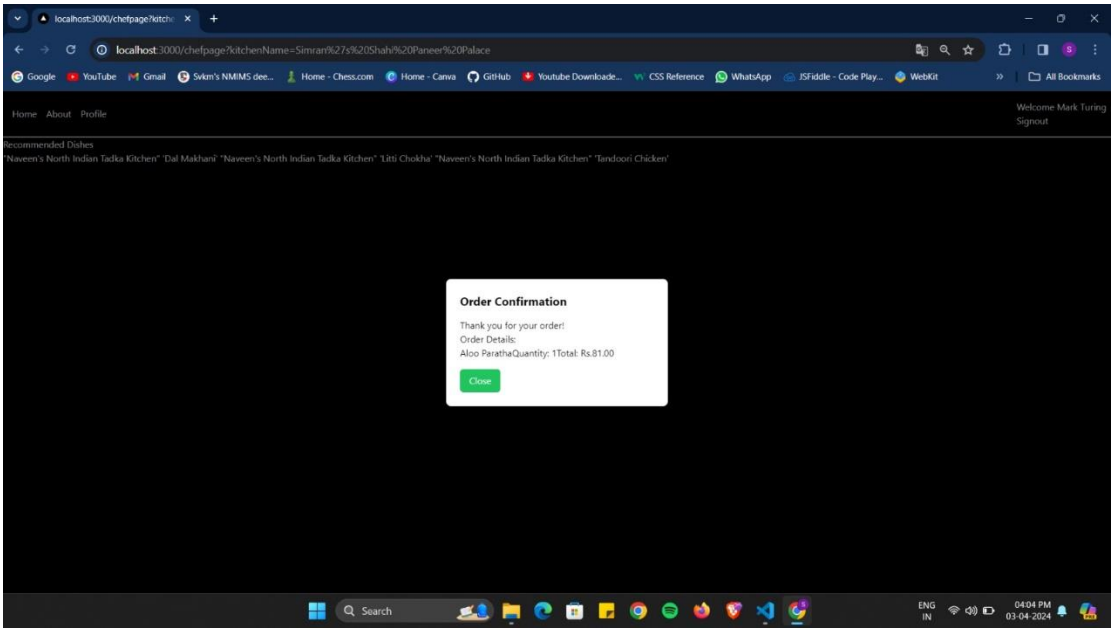
Dish recommendations based on tags for butter chicken



The recommendation model



Accuracy of the model



Recommended dishes in working app

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive

[ ] !cp -r "/content/drive/MyDrive/Capstone Project.xlsx" /content/

[ ] import pandas as pd

    # Load the Excel file into a DataFrame
    df = pd.read_excel('/content/Capstone_Project.xlsx')

    # Remove all rows where all columns are empty
    df = df.dropna(how='all')

    # Save the DataFrame back to an Excel file
    df.to_excel('/content/Capstone_Project_without_empty_rows.xlsx', index=False)

    import pandas as pd

    # Load the Excel file into a DataFrame
    df = pd.read_excel('/content/Capstone_Project_without_empty_rows.xlsx')

    # Display the DataFrame
    print(df)

                          kitchenName kitchenCusine  kitchenRating  \
    0    Naveen's North Indian Tadka Kitchen  North Indian           4.0
    1    Naveen's North Indian Tadka Kitchen  North Indian           4.0
    2    Naveen's North Indian Tadka Kitchen  North Indian           4.0
    3    Naveen's North Indian Tadka Kitchen  North Indian           4.0
    4    Naveen's North Indian Tadka Kitchen  North Indian           4.0
    ..                                   ...          ...           ...
    271      Pranav's Khandvi Corner Kitchen      Gujarati           3.0
    272      Pranav's Khandvi Corner Kitchen      Gujarati           3.0
    273      Pranav's Khandvi Corner Kitchen      Gujarati           3.0
    274      Pranav's Khandvi Corner Kitchen      Gujarati           3.0
    275      Pranav's Khandvi Corner Kitchen      Gujarati           3.0

                  dishName  dishRating  \
    0            Dal Makhani         3.6
    1           Litti Chokha         4.5
    2        Tandoori Chicken         4.7
    3           Aloo Paratha         4.9
```

```
3            Aloo Paratha        4.9
4    Chicken Keema Paratha       3.7
..                    ...        ...
271                 Patra        4.5
272        Gujarati Bonda        4.9
273                Aamras        4.7
274         Veg Mini Meal        3.7
275      Veg Premium Meal        4.1

                                              dishTags
0      veg, Creamy, Spicy, Flavorful, Rich, Satisfying
1     veg, Traditional, Spicy, Tangy, Smoky, Flavorful
2     non-veg, Grilled, Spicy, Smoky, Tangy, Flavorful
3    veg, Spicy, Buttery, Flavorful, Comforting, Sa...
4      non-veg, Spicy, Savory, Flavorful, Hearty, Rich
..                                                 ...
271                         veg, Savory, Soft, Rolled
272                       veg, Savory, Crispy, Fluffy
273                     veg, Sweet, Tangy, Refreshing
274  veg, Balanced, Light, Refreshing, Nutritious, ...
275  veg, Rich, Flavorful, Satisfying, Hearty, Indu...

[276 rows x 6 columns]
```

```
[ ] import pandas as pd

    # Load the Excel file into a DataFrame
    df = pd.read_excel('/content/Capstone_Project_without_empty_rows.xlsx')

    # Print all column names
    print(df.columns)
```

```
Index(['kitchenName', 'kitchenCusine', 'kitchenRating', 'dishName',
       'dishRating', 'dishTags'],
      dtype='object')
```

```
[ ] import pandas as pd

    # Load the Excel file into a DataFrame
    df = pd.read_excel('/content/Capstone_Project_without_empty_rows.xlsx')
```

```
import pandas as pd

# Load the Excel file into a DataFrame
df = pd.read_excel('/content/Capstone_Project_without_empty_rows.xlsx')

# Create an empty list to store unique dish tags
unique_dish_tags = []

# Iterate over each row in the 'dishTags' column
for tags in df['dishTags']:
    # Split the tags by comma and iterate over each tag
    for tag in tags.split(','):
        # Remove leading and trailing whitespace
        tag = tag.strip()
        # Append the tag to the unique_dish_tags list if it's not already present
        if tag not in unique_dish_tags:
            unique_dish_tags.append(tag)

# Print the unique dish tags
print(unique_dish_tags)
```

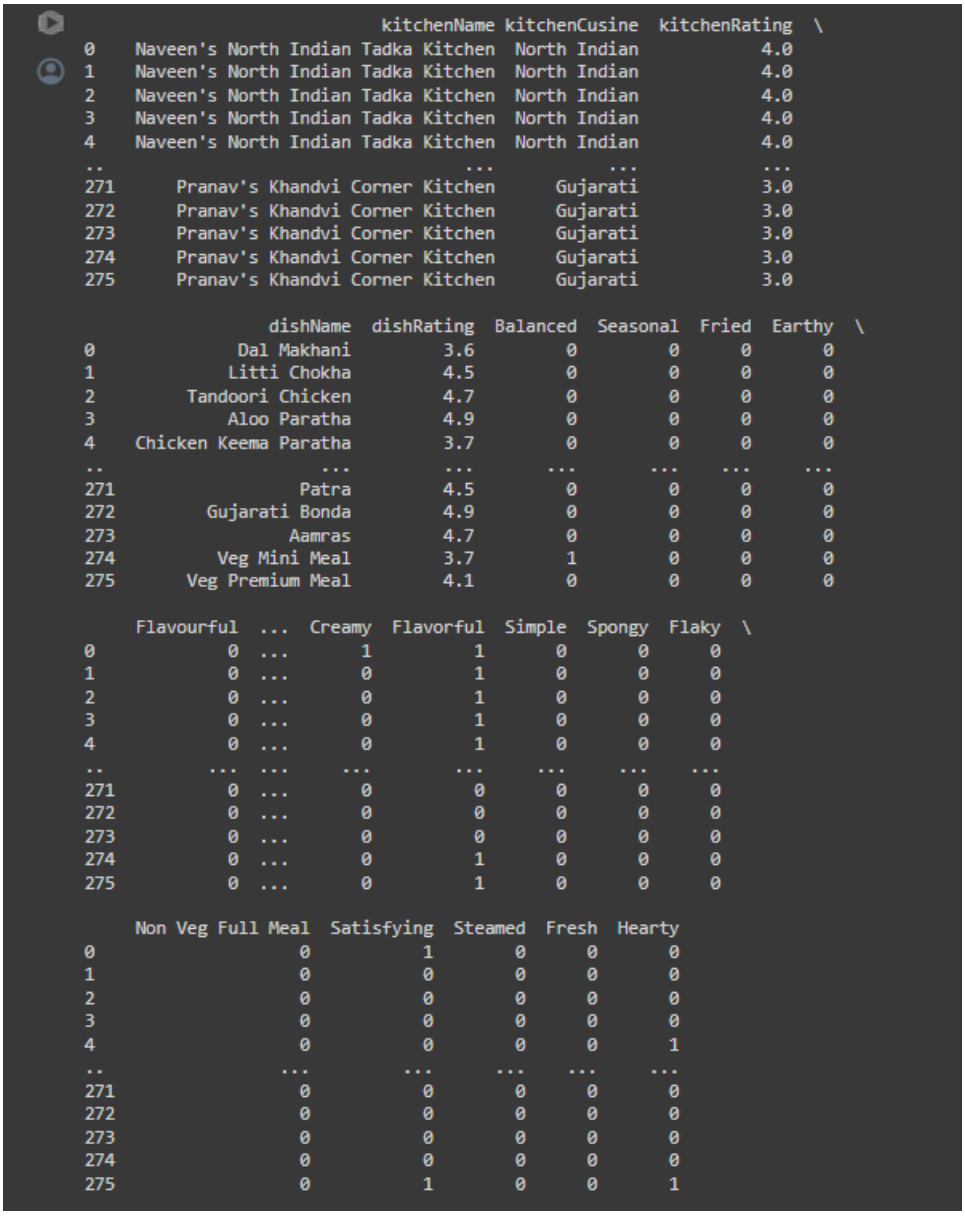
```
['veg', 'Creamy', 'Spicy', 'Flavorful', 'Rich', 'Satisfying', 'Traditional', 'Tangy', 'Smoky', 'non-veg', 'Grilled',
```

```python
from surprise import Reader, Dataset, SVD
import pandas as pd

# Load the Excel file
df = pd.read_excel('/content/Capstone_Project_without_empty_rows.xlsx')

# Find all unique tags
unique_tags = set()
for tags in df['dishTags']:
    unique_tags.update(tags.split(', '))

# Create columns for each tag and perform one-hot encoding
for tag in unique_tags:
    df[tag] = df['dishTags'].apply(lambda x: 1 if tag in x.split(', ') else 0)

# Assign weights to tags based on their presence in the dishTags column
tag_weights = {tag: 1 / len(tag.split(', ')) for tag in unique_tags}

# Combine weighted tags with dishRating
df['input'] = df['dishRating']
for tag, weight in tag_weights.items():
    df['input'] += df[tag] * weight

# Load the DataFrame into the surprise Dataset format
reader = Reader(rating_scale=(0, 1))
data = Dataset.load_from_df(df[['kitchenName', 'dishName', 'input']], reader)

# Use the SVD algorithm
algo = SVD()
trainset = data.build_full_trainset()
algo.fit(trainset)

# Function to predict rating for a single input
def predict_rating(kitchen_name, dish_name):
    prediction = algo.predict(kitchen_name, dish_name)
    return prediction.est

# Get user input
kitchen_name = input("Enter kitchen name: ")
dish_name = input("Enter dish name: ")

# Predict rating for the input
predicted_rating = predict_rating(kitchen_name, dish_name)
print(f"Predicted rating for {dish_name} in {kitchen_name}: {predicted_rating}")
```

```
[ ] # Find top recommendations based on predicted ratings
    def get_top_recommendations(user_kitchen_name,user_dish_name):
        predictions = []
        for kitchen_name, dish_name in product(df['kitchenName'].unique(), df['dishName'].unique()):
            if (kitchen_name != user_kitchen_name) or (dish_name != user_dish_name):
                rating = predict_rating(kitchen_name, dish_name)
                predictions.append((kitchen_name, dish_name, rating))

        top_recommendations = sorted(predictions, key=lambda x: x[2], reverse=True)[:3]
        return top_recommendations

    # Print top recommendations
    top_recommendations = get_top_recommendations(kitchen_name, dish_name)
    for rec in top_recommendations:
        print(f"Recommendation: {rec[1]} in {rec[0]}")
```

```
Enter kitchen name: Sanjay's South Indian Sappadu Kitchen
Enter dish name: Chana Masala
Predicted rating for Chana Masala in Sanjay's South Indian Sappadu Kitchen: 1
Recommendation: Dal Makhani in Naveen's North Indian Tadka Kitchen
Recommendation: Litti Chokha in Naveen's North Indian Tadka Kitchen
Recommendation: Tandoori Chicken in Naveen's North Indian Tadka Kitchen
```

```
import pandas as pd

# Load the Excel file
df = pd.read_excel('/content/Capstone_Project_without_empty_rows.xlsx')

# Find all unique tags
unique_tags = set()
for tags in df['dishTags']:
    unique_tags.update(tags.split(', '))

# Create columns for each tag and perform one-hot encoding
for tag in unique_tags:
    df[tag] = df['dishTags'].apply(lambda x: 1 if tag in x.split(', ') else 0)

# Drop the original 'dishTags' column
df.drop('dishTags', axis=1, inplace=True)

print(df)
```



```
                          kitchenName kitchenCusine  kitchenRating  \
0    Naveen's North Indian Tadka Kitchen  North Indian            4.0
1    Naveen's North Indian Tadka Kitchen  North Indian            4.0
2    Naveen's North Indian Tadka Kitchen  North Indian            4.0
3    Naveen's North Indian Tadka Kitchen  North Indian            4.0
4    Naveen's North Indian Tadka Kitchen  North Indian            4.0
..                                   ...          ...            ...
271      Pranav's Khandvi Corner Kitchen      Gujarati            3.0
272      Pranav's Khandvi Corner Kitchen      Gujarati            3.0
273      Pranav's Khandvi Corner Kitchen      Gujarati            3.0
274      Pranav's Khandvi Corner Kitchen      Gujarati            3.0
275      Pranav's Khandvi Corner Kitchen      Gujarati            3.0

                  dishName  dishRating  Balanced  Seasonal  Fried  Earthy  \
0              Dal Makhani         3.6         0         0      0       0
1             Litti Chokha         4.5         0         0      0       0
2         Tandoori Chicken         4.7         0         0      0       0
3             Aloo Paratha         4.9         0         0      0       0
4    Chicken Keema Paratha         3.7         0         0      0       0
..                     ...         ...       ...       ...    ...     ...
271                 Patra         4.5         0         0      0       0
272         Gujarati Bonda         4.9         0         0      0       0
273                Aamras         4.7         0         0      0       0
274          Veg Mini Meal         3.7         1         0      0       0
275       Veg Premium Meal         4.1         0         0      0       0

     Flavourful  ...  Creamy  Flavorful  Simple  Spongy  Flaky  \
0             0  ...       1          1       0       0      0
1             0  ...       0          1       0       0      0
2             0  ...       0          1       0       0      0
3             0  ...       0          1       0       0      0
4             0  ...       0          1       0       0      0
..          ...  ...     ...        ...     ...     ...    ...
271           0  ...       0          0       0       0      0
272           0  ...       0          0       0       0      0
273           0  ...       0          0       0       0      0
274           0  ...       0          1       0       0      0
275           0  ...       0          1       0       0      0

     Non Veg Full Meal  Satisfying  Steamed  Fresh  Hearty
0                    0           1        0      0       0
1                    0           0        0      0       0
2                    0           0        0      0       0
3                    0           0        0      0       0
4                    0           0        0      0       1
..                 ...         ...      ...    ...     ...
271                  0           0        0      0       0
272                  0           0        0      0       0
273                  0           0        0      0       0
274                  0           0        0      0       0
275                  0           1        0      0       1
```

```
!pip install surprise
```

```
Collecting surprise
  Downloading surprise-0.1-py2.py3-none-any.whl (1.8 kB)
Collecting scikit-surprise (from surprise)
  Downloading scikit-surprise-1.1.3.tar.gz (771 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 772.0/772.0 kB 5.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-su
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-su
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-sur
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.3-cp310-cp310-linux_x86_64.whl size=
  Stored in directory: /root/.cache/pip/wheels/a5/ca/a8/4e28def53797fdc4363ca4af740db15a9c2f1595ebc51f
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.3 surprise-0.1
```

```
[ ] from flask import Flask, jsonify, request
    import pandas as pd

    app = Flask(__name__)

    # Load the Excel file
    df = pd.read_excel('/content/Capstone_Project_without_empty_rows.xlsx')

    @app.route('/recommendations', methods=['POST'])
    def get_recommendations():
        # Code to get top recommendations
        # You can reuse the function get_top_recommendations() from the previous code
        top_recommendations = get_top_recommendations()
        return jsonify({'recommendations': top_recommendations})

    if __name__ == '__main__':
        app.run(debug=True)
```

```
[ ]  import pandas as pd
     from surprise import Reader, Dataset, SVD
     from itertools import product

     # Load the Excel file
     df = pd.read_excel('/content/Capstone_Project_without_empty_rows.xlsx')

     # Find all unique tags
     unique_tags = set()
     for tags in df['dishTags']:
         unique_tags.update(tags.split(', '))

     # Create columns for each tag and perform one-hot encoding
     for tag in unique_tags:
         df[tag] = df['dishTags'].apply(lambda x: 1 if tag in x.split(', ') else 0)

     # Assign weights to tags based on their presence in the dishTags column
     tag_weights = {tag: 1 / len(tag.split(', ')) for tag in unique_tags}

     # Combine weighted tags with dishRating
     df['input'] = df['dishRating']
     for tag, weight in tag_weights.items():
         df['input'] += df[tag] * weight

     # Load the DataFrame into the surprise Dataset format
     reader = Reader(rating_scale=(0, 1))
     data = Dataset.load_from_df(df[['kitchenName', 'dishName', 'input']], reader)

     # Use the SVD algorithm
     algo = SVD()
     trainset = data.build_full_trainset()
     algo.fit(trainset)

     # Function to predict rating for a single input
     def predict_rating(kitchen_name, dish_name):
         prediction = algo.predict(kitchen_name, dish_name)
         return prediction.est

     # Function to get top recommendations from the same cuisine type
     def get_top_recommendations(user_kitchen_name, user_dish_name):
         user_cuisine = df[df['kitchenName'] == user_kitchen_name]['kitchenCusine'].iloc[0]

         # Filter out dishes from different cuisines
         filtered_df = df[df['kitchenCusine'] == user_cuisine]

         predictions = []
```

```
         for _, row in filtered_df.iterrows():
             kitchen_name = row['kitchenName']
             dish_name = row['dishName']
             if (kitchen_name != user_kitchen_name) or (dish_name != user_dish_name):
                 rating = predict_rating(kitchen_name, dish_name)
                 predictions.append((kitchen_name, dish_name, rating))

         top_recommendations = sorted(predictions, key=lambda x: x[2], reverse=True)[:3]
         return top_recommendations

     # Compute recommendations for all combinations
     all_recommendations = {}
     for kitchen_name, dish_name in product(df['kitchenName'].unique(), df['dishName'].unique()):
         recommendations = get_top_recommendations(kitchen_name, dish_name)
         all_recommendations[(kitchen_name, dish_name)] = recommendations

     # Store recommendations in a CSV column
     df['recommendations'] = df.apply(lambda row: all_recommendations[(row['kitchenName'], row['dishName'])], axis=1)

     # Save the DataFrame to a CSV file
     df.to_csv('/content/recommendations.csv', index=False)
```

Complete working of the recommendation model

	A	B	C	D	E	F	G	H	I	J
1	kitchenNa	kitchenCus	kitchenRat	dishName	dishRating	dishTags	Balanced	Seasonal	Fried	Earthy
2	Naveen's I	North Indi	4	Dal Makha	3.6	veg, Crean	0	0	0	0
3	Naveen's I	North Indi	4	Litti Chokh	4.5	veg, Tradit	0	0	0	0
4	Naveen's I	North Indi	4	Tandoori C	4.7	non-veg, G	0	0	0	0
5	Naveen's I	North Indi	4	Aloo Parat	4.9	veg, Spicy,	0	0	0	0
6	Naveen's I	North Indi	4	Chicken Ke	3.7	non-veg, S	0	0	0	0
7	Naveen's I	North Indi	4	Veg Mini M	3.9	veg, Balan	1	0	0	0
8	Naveen's I	North Indi	4	Veg Premiu	4.4	veg, Rich, I	0	0	0	0
9	Naveen's I	North Indi	4	Non Veg P	4.3	non-veg, R	0	0	0	0
10	Meera's Di	North Indi	4	Chicken Cu	4.2	non-veg, S	0	0	0	0
11	Meera's Di	North Indi	4	Shahi Pane	4.8	veg, Crean	0	0	0	0
12	Meera's Di	North Indi	4	Khichdi	3.6	veg, Comfo	0	0	0	0
13	Meera's Di	North Indi	4	Kadai Pane	4.7	veg, Spicy,	0	0	0	0
14	Meera's Di	North Indi	4	Rajma Cha	3.7	veg, Comfo	0	0	0	0
15	Meera's Di	North Indi	4	Veg Mini M	4.2	veg, Balan	1	0	0	0
16	Meera's Di	North Indi	4	Veg Premiu	4.4	veg, Rich, I	0	0	0	0
17	Meera's Di	North Indi	4	Non Veg P	4.6	non-veg, R	0	0	0	0
18	Rohan's Ra	North Indi	5	Kofta Curr	4.1	veg, Flavor	0	0	0	0
19	Rohan's Ra	North Indi	5	Kadai Pane	3.9	veg, Spicy,	0	0	0	0
20	Rohan's Ra	North Indi	5	Rajma Cha	4.7	veg, Savor	0	0	0	0
21	Rohan's Ra	North Indi	5	Hara Bhara	4.5	veg, Flavor	0	0	1	0
22	Rohan's Ra	North Indi	5	Shahi Pane	4.4	veg, Crean	0	0	0	0
23	Rohan's Ra	North Indi	5	Veg Mini M	4.3	veg, Balan	1	0	0	0
24	Rohan's Ra	North Indi	5	Veg Premiu	3.7	veg, Rich, I	0	0	0	0
25	Pooja's Pa	North Indi	4.5	Dal Makha	4.7	veg, Crean	0	0	0	0
26	Pooja's Pa	North Indi	4.5	Tandoori C	4.2	non-veg, G	0	0	0	0
27	Pooja's Pa	North Indi	4.5	Sarson Da	4.8	veg,Spicy,	0	0	0	0

K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Flavourful	Cooling	Tasty	Smoky	Filing	Refreshing	non-veg	Gravy	Crispy	Crsipy	Spiced	Nutritious	Rolled,	Rolled
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0

Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO
Wholesom	Tempered	Comfortin	Tangy	Veg Full M	Spice	Baked	Cool	Savory	Soupy	Lentil-base	Layered	Spicy,Tang	Drink	veg	Indulgent,F	Sweet
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF
Fluffy	Spicy	Veg Half M	Grilled	Fermentec	Crunchy	Buttery	Rich	Luxurious	Pickled	Flatbread	Varied	Indulgent	Traditiona	Soft	Nutty	Snack
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ	BR	BS
veg,Spicy	Dessert	Mild	Variety	Flavorful	Sticky	Desert	Aromatic	Smooth	Stuffed	Light	Creamy	Flavorful
0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	1

BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC
Simple	Spongy	Flaky	Non Veg F	Satisfying	Steamed	Fresh	Hearty	input	recommenda
0	0	0	0	1	0	0	0	9.6	[("Naveen's N
0	0	0	0	0	0	0	0	10.5	[("Naveen's N
0	0	0	0	0	0	0	0	10.7	[("Naveen's N
0	0	0	0	0	0	0	0	10.9	[("Naveen's N
0	0	0	0	0	0	0	1	9.7	[("Naveen's N
0	0	0	0	0	0	0	0	10.9	[("Naveen's N
0	0	0	0	1	0	0	1	11.4	[("Naveen's N
0	0	0	1	1	0	0	0	11.3	[("Naveen's N
0	0	0	0	1	0	0	0	10.2	[("Naveen's N
0	0	0	0	1	0	0	0	10.8	[("Naveen's N
1	0	0	0	0	0	0	0	9.6	[("Naveen's N
0	0	0	0	1	0	0	0	10.7	[("Naveen's N
0	0	0	0	1	0	0	0	9.7	[("Naveen's N
0	0	0	0	0	0	0	0	11.2	[("Naveen's N
0	0	0	0	1	0	0	1	11.4	[("Naveen's N
0	0	0	1	1	0	0	0	11.6	[("Naveen's N
0	0	0	0	1	0	0	1	11.1	[("Naveen's N
0	0	0	0	1	0	0	0	10.9	[("Naveen's N
0	0	0	0	1	0	0	0	10.7	[("Naveen's N
0	0	0	0	1	0	0	0	10.5	[("Naveen's N
0	0	0	0	1	0	0	0	10.4	[("Naveen's N
0	0	0	0	0	0	0	0	11.3	[("Naveen's N
0	0	0	0	1	0	0	1	10.7	[("Naveen's N
0	0	0	0	1	0	0	0	11.7	[("Naveen's N
0	0	0	0	0	0	0	0	10.2	[("Naveen's N
0	0	0	0	0	0	0	0	8.8	[("Naveen's N

Tags and Recommendation Database

Chapter 4

Result & Analysis

We have implemented Tags based Collaborative Filtering technique for the recommendation system.

From the above application we have utilised the algorithm which gives us an accuracy rate of 97% which leads us to believe it’s a highly accurate model.

We have also been able to see the recommended dishes after a customer makes an order. On the basis of the order the application gives a recommendation based on the recommendation system which is being used in the application.

We have created the application using Next.js for frontend, Firebase (Firestore and Realtime Firebase), CSV.

We have also used Recommendation systems which included One-hot enconding, Singular Valuue Decomposition (SVD), Pandas, Surprise, Itertools.

This has allowed us to create an application which allows users to select their location, select the available cuisines, select the desired dishes and order. It would also recommend dishes based on the above order.

This Tags based Collaborative Filtering technique for the recommendation system is implemented via our application.

Chapter 5

Advantages:

Healthier alternatives as compared to traditional food delivery choices of fast-food chains and restaurants

Supports the local chefs, allowing them to showcase their culinary skills and generate additional income

Variety in the choice of cuisines which can be ordered by the customer due to the various types of kitchens serving in a particular region

Create a sustainable and healthy habit.

Limitations:

Limited availability due to it being home cooked food since they are home chefs.

Consistent quality across all the chefs is difficult to achieve.

Cost consideration could be slightly higher as compared to other mass production restaurants and fast-food chains.

Data and addition of new home chefs could be difficult.

Chapter 6

Conclusion & Future Scope

We have implemented the above application using the Tag based Collaborative Filtering.

We took down all the data regarding the chefs and various dishes.

We then used Firebse, Firestore for the backend of the application.

The future scope of the application is that it can be expanded for various regions and incorporate more chefs and expand the user base as well.

We also plan to implement a Calendar functionality which would allow users to buy meal plans and allocate what meals and when exactly they need them to be delivered. This would allow us to increase the usability of the application and provide more customization and flexibility of the application.

We can also increase the functionalities in the application as well as incorporate various other algorithms to better improve the application.

References

1. Food Recommender Systems Important Contributions, Challenges and Future Research Directions. Author: Christoph Trattner, David Elsweiler
2. Food Recommendation: Framework, Existing Solutions and Challenges Weiqing Min, Member, IEEE, Shuqiang Jiang, Senior Member, IEEE, Ramesh Jain, Fellow, IEEE
3. Interaction Design in a Mobile Food Recommender System, Mehdi Elahi, Mouzhi Ge, Francesco Ricci
4. Market2Dish: Health-aware Food Recommendation, WENJIE WANG, National University of Singapore, LING-YU DUAN, Peking University, HAO JIANG, Shandong University, PEIGUANG JING, Tianjin University, XUEMENG SONG and LIQIANG NIE, Shandong University
5. Using Tags and Latent Factors in a Food Recommender System, Mouzhi Ge, Mehdi Elahi, Francesco Ricci
6. Food Recommendation System Based on Content Based and Collaborative Filtering Techniques - Singh, Reetu & Dwivedi, Pragya. (2023). Food Recommendation Systems Based On Content-based and Collaborative Filtering Techniques. 10.1109/ICCCNT56998.2023.10307080.
7. Hybrid Recommendation System with Graph based and Collaborative Filtering Recommendation Systems - S. Khanduri and P. Saravanan, “Hybrid Recommendation System with Graph based and Collaborative Filtering Recommendation Systems,” 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), Oct. 2022, doi: 10.1109/mysurucon55714.2022.9972677. Available: https://doi.org/10.1109/mysurucon55714.2022.9972677
8. User Profile Feature-Based Approach to Address the Cold Start Problem in Collaborative Filtering for Personalized Movie Recommendation - Uyangodage, Lasitha & Ahangama, Supunmali & Ranasinghe, Tharindu. (2019). User Profile Feature-Based Approach to Address the Cold Start Problem in Collaborative Filtering for Personalized Movie Recommendation.
9. Movie Recommendation System using RNN and Cognitive thinking - Labde, Shubhada & Karan, Vishesh & Shah, Shubham & Krishnan, Dhruv. (2023).

Movie Recommendation System using RNN and Cognitive thinking. 1-7. 10.1109/INCET57972.2023.10170572.

10. Coffee Shop Recommendation System Using an Item-Based Collaborative Filtering Approach - Astri, Renita & Kamal, Ahmad & Sura, Suaini. (2022). Coffee Shop Recommendation System Using an Item-Based Collaborative Filtering Approach. 65-67. 10.1109/ISITDI55734.2022.9944403.
11. Food Recommendation System Based on Collaborative Filtering and Taste Profiling -
12. Intelligent Movie Recommendation System Based on Hybrid Recommendation Algorithms - Pu, Qingna & Hu, Bin. (2023). Intelligent Movie Recommendation System Based on Hybrid Recommendation Algorithms. 1-5. 10.1109/AIKIIE60097.2023.10389982.
13. A systematic review on food recommender systems - Bondevik, Jon & Bennin, Kwabena & Babur, Önder & Ersch, Carsten. (2023). A systematic review on food recommender systems. Expert Systems with Applications. 238. 122166. 10.1016/j.eswa.2023.122166.
14. A Food Recommender System Considering Nutritional Information and User Preferences - Yera Toledo, Raciel & Alzahrani, Ahmad & Martinez, Luis. (2019). A Food Recommender System Considering Nutritional Information and User Preferences. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2929413.
15. A Review of Academic Recommendation Systems Based on Intelligent Recommendation Algorithms - Yang, Huaiyuan & Zhou, Hua & Li, Yucheng. (2022). A Review of Academic Recommendation Systems Based on Intelligent Recommendation Algorithms. 958-962. 10.1109/ICIVC55077.2022.9886104.
16. A Review of Healthcare Recommendation Systems Using Several Categories of Filtering and Machine Learning-Based Methods - Kumar, Pardeep & Kumar, Ankit. (2022). A Review of Healthcare Recommendation Systems Using Several Categories of Filtering and Machine Learning-Based Methods. 762-768. 10.1109/ICCCIS56430.2022.10037604.
17. A Study on Product Recommendation System based on Deep Learning and Collaborative Filtering -
18. An Intelligent Recommendation System for Performance Equipment Operation and Maintenance via Deep Neural Network and Attention Mechanism - Li, Huimin & Chen, Yongyi & Zhang, Dan & Wu, Lifeng.

(2021). An Intelligent Recommendation System for Performance Equipment Operation and Maintenance via Deep Neural Network and Attention Mechanism. 1464-1468. 10.1109/DDCLS52934.2021.9455655.

19. Collaborative Filtering Recommendation with Fluctuations of User' Preference -
20. Content-Based Recommendation Using Machine Learning -
21. E-Commerce Intelligent Recommendation System Based on Deep Learning - Huang, Gang. (2022). E-Commerce Intelligent Recommendation System Based on Deep Learning. 1154-1157. 10.1109/IPEC54454.2022.9777500.
22. Food Recommendation using Ontology and Heuristics - El-dosuky, Mohamed & Rashad, Magdi & Hamza, Taimoor & El-Bassiouny, Ahmed. (2013). Food Recommendation using Ontology and Heuristics.
23. User Profile-Based Recommendation Engine Mitigating the Cold-Start Problem - Mayrhuber, Elisabeth & Krauss, Oliver. (2022). User Profile-Based Recommendation Engine Mitigating the Cold-Start Problem. 1-6. 10.1109/ICECCME55909.2022.9988037.

Appendix

Recommendation Code:

```
# -*- coding: utf-8 -*-
"""2nd_cappy_rec.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1ujzr6twzODnmBml89cn1Tc5764oM3Y8x
"""

import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer

# Sample data (You can replace this with your own dataset)
data = pd.read_csv("/content/RecommendationData.csv")

df = pd.DataFrame(data)

# Convert tags into TF-IDF matrix
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(df['dishTags'])

# Compute cosine similarity matrix
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

def get_recommendations(dish, cosine_sim=cosine_sim):
    # Get the index of the dish that matches the dish name
    idx = df[df['dishName'] == dish].index[0]
```

```
    # Get the pairwsie similarity scores of all dishes with that dish
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the dishes based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the top 5 similar dishes
    sim_scores = sim_scores[1:6]

    # Get the dish indices
    dish_indices = [i[0] for i in sim_scores]

    # Return the top 5 most similar dishes
    return df['dishName'].iloc[dish_indices]

# Example usage
print(get_recommendations('Dal Makhani'))
```

ChefPage

```
'use client'
import React, { useState } from 'react';
import { addDoc, collection } from 'firebase/firestore';
import { db, auth } from '../firebase/config';

const CartModal = ({ cartItems, handleClose }) => {
  const [isOrderPlaced, setIsOrderPlaced] = useState(false);

  const placeOrder = async () => {
```

```
  try {
    const user = auth.currentUser;
    if (user) {
      const { uid, displayName, email } = user;
      await Promise.all(
        cartItems.map(async (item) => {
          await addDoc(collection(db, 'orders'), {
            identifier: uid,
            userName: displayName,
            userEmail: email,
            dishName: item.dishName,
            dishTags: item.dishTags,
            kitchenName: item.kitchenName,
          });
        })
      );
      setIsOrderPlaced(true);
      // Redirect back to the homepage after placing the order
      window.location.href = '/';
    } else {
      console.error('User not authenticated.');
    }
  } catch (error) {
    console.error('Error placing order:', error);
  }
};

// Calculate total amount
const totalAmount = cartItems.reduce((total, item) => {
  return total + item.quantity * item.dishPrice;
```

```
  }, 0);


  return (
    <div className="fixed inset-0 z-50 overflow-hidden bg-black bg-opacity-50 flex justify-center items-center">
      <div className="bg-white rounded-lg shadow-lg p-6 w-96">
        <div className="flex justify-between mb-4">
          <h2 className="text-xl font-bold text-black">Your Cart</h2>
          <button onClick={handleClose} className="text-black">
            Close
          </button>
        </div>
        <div>
          {cartItems.map((item) => (
            <div key={item.id} className="flex justify-between mb-2 text-black">
              <span>{item.dishName}</span>
              <span>{item.quantity}</span>
              <span>Rs.{(item.quantity * item.dishPrice).toFixed(2)}</span>
            </div>
          ))}
        </div>
        <div className="mt-4 border-t border-gray-300 pt-4 text-black">
          <div className="flex justify-between">
            <span className="font-semibold">Total:</span>
            <span>Rs.{totalAmount.toFixed(2)}</span>
          </div>
        </div>
        <button
          onClick={() => {
            placeOrder();
            handleClose();
```

```
        }}
        className="bg-green-500 text-white px-4 py-2 rounded-md hover:bg-green-600 mt-4"
        disabled={isOrderPlaced}
      >
        {isOrderPlaced ? 'Order Placed' : 'Place Order'}
      </button>
    </div>
  </div>
 );
};

export default CartModal;
```

29

Firebase Config File

```
// Import the functions you need from the SDKs you need
import { initializeApp, getApps } from "firebase/app";
import { getAuth } from "firebase/auth";
import { Firestore, getFirestore } from "firebase/firestore";

const firebaseConfig = {
  apiKey: process.env.NEXT_PUBLIC_FIREBASE_API_KEY,
  authDomain: process.env.NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN,
  projectId: process.env.NEXT_PUBLIC_FIREBASE_PROJECT_ID,
  storageBucket: process.env.NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: process.env.NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID,
  appId: process.env.NEXT_PUBLIC_FIREBASE_APP_ID,
  // measurementId: process.env.NEXT_PUBLIC_FIREBASE_MEASUREMENT_ID,
};
```

```
// Initialize Firebase
let firebase_app = getApps().length === 0 ? initializeApp(firebaseConfig) :
getApps()[0];

const app = initializeApp(firebaseConfig);



export default firebase_app;
export const auth = getAuth(app);

const db = getFirestore(app)
export {db};
```

NavBar

```
import React from "react";
import Link from "next/link";
import { UserAuth } from "../context/AuthContext";

const navbar = () => {
  const { user, googleSignIn, logOut } = UserAuth();

  const handleSignIn = async () => {
    try {
      await googleSignIn();
    } catch (error) {
      console.log(error);
    }
  };

  const handleSignOut = async () => {
```

```
  try {
    await logOut();
  } catch (error) {
    console.log(error);
  }
};

return (
  <div className="h-20 w-full border-b-2 flex items-center justify-between p-2">
    <ul className="flex">
      <li className="p-2 cursor-pointer">
        <Link href="/">Home</Link>
      </li>
      <li className="p-2 cursor-pointer">
        <Link href="/aboutpage">About</Link>
      </li>
      {!user ? null : (
        <li className="p-2 cursor-pointer">
          <Link href="/profilepage">Profile</Link>
        </li>
      )}
    </ul>
    {!user ? (
      <ul className="flex">
        <li onClick={handleSignIn} className="p-2 cursor-pointer">
          Login
        </li>
        <li onClick={handleSignIn} className="p-2 cursor-pointer">
          Signup
        </li>
```

```
      </ul>
     ) : (
      <div>
       <p>Welcome {user.displayName}</p>
       <p onClick={handleSignOut} className="cursor-pointer">
        Signout
       </p>
      </div>
     )}
    </div>
  );
};

export default navbar;
```

Authentication

```
import { useContext, createContext, useState, useEffect } from "react";
import { signInWithPopup, signOut, onAuthStateChanged, GoogleAuthProvider }
from "firebase/auth";
import {auth} from '../firebase/config';

const AuthContext = createContext();

export const AuthContextProvider = ({children}) => {
   const [user, setUser] = useState(null);

   const googleSignIn = () => {
      const provider = new GoogleAuthProvider()
      signInWithPopup(auth, provider)
   }
```

```
    const logOut = () => {
      signOut(auth)
    }

    useEffect(() =>{
        const unsubscribe = onAuthStateChanged(auth,(currentUser) => {
            setUser(currentUser);
        })
        return () => unsubscribe()
    },[user] )

    return(
        <AuthContext.Provider value={{user, googleSignIn,
logOut}}>{children}</AuthContext.Provider>
    )
}


export const UserAuth = () => {
    return useContext(AuthContext)
}
```

Add To Cart

```
'use client'
import React, { useState } from 'react';
import { addDoc, collection } from 'firebase/firestore';
import { db, auth } from '../firebase/config';

const CartModal = ({ cartItems, handleClose }) => {
  const [isOrderPlaced, setIsOrderPlaced] = useState(false);
```

```
const placeOrder = async () => {
  try {
    const user = auth.currentUser;
    if (user) {
      const { uid, displayName, email } = user;
      await Promise.all(
        cartItems.map(async (item) => {
          await addDoc(collection(db, 'orders'), {
            identifier: uid,
            userName: displayName,
            userEmail: email,
            dishName: item.dishName,
            dishTags: item.dishTags,
            kitchenName: item.kitchenName,
          });
        })
      );
      setIsOrderPlaced(true);
      // Redirect back to the homepage after placing the order
      window.location.href = '/';
    } else {
      console.error('User not authenticated.');
    }
  } catch (error) {
    console.error('Error placing order:', error);
  }
};

// Calculate total amount
const totalAmount = cartItems.reduce((total, item) => {
```

```
    return total + item.quantity * item.dishPrice;
  }, 0);


  return (
    <div className="fixed inset-0 z-50 overflow-hidden bg-black bg-opacity-50 flex justify-center items-center">
      <div className="bg-white rounded-lg shadow-lg p-6 w-96">
        <div className="flex justify-between mb-4">
          <h2 className="text-xl font-bold text-black">Your Cart</h2>
          <button onClick={handleClose} className="text-black">
            Close
          </button>
        </div>
        <div>
          {cartItems.map((item) => (
            <div key={item.id} className="flex justify-between mb-2 text-black">
              <span>{item.dishName}</span>
              <span>{item.quantity}</span>
              <span>Rs.{(item.quantity * item.dishPrice).toFixed(2)}</span>
            </div>
          ))}
        </div>
        <div className="mt-4 border-t border-gray-300 pt-4 text-black">
          <div className="flex justify-between">
            <span className="font-semibold">Total:</span>
            <span>Rs.{totalAmount.toFixed(2)}</span>
          </div>
        </div>
        <button
          onClick={() => {
            placeOrder();
```

```
          handleClose();
        }}
        className="bg-green-500 text-white px-4 py-2 rounded-md hover:bg-green-600 mt-4"
        disabled={isOrderPlaced}
      >
        {isOrderPlaced ? 'Order Placed' : 'Place Order'}
      </button>
    </div>
  </div>
 );
};

export default CartModal;
```

Order Confirmation

```
import React from 'react';

const OrderConfirmation = ({ orderDetails, onClose }) => {
 const handleClose = () => {
  onClose();
  // Redirect to homepage after closing the modal
  window.location.href = '/';
 };
 return (
  <div className="fixed inset-0 z-50 overflow-hidden bg-black bg-opacity-50 flex justify-center items-center">
   <div className="bg-white rounded-lg shadow-lg p-6 w-96 text-black">
    <h2 className="text-xl font-bold text-black mb-4">Order Confirmation</h2>
    <div className="mb-4">
     <p>Thank you for your order!</p>
```

```
        <p>Order Details:</p>
        <ul>
          {orderDetails && orderDetails.length > 0 ? (
            orderDetails.map((order, index) => (
              <li key={index}>
                <span>{order.dishName}</span>
                <span>Quantity: {order.quantity}</span>
                <span>Total: Rs.{(order.quantity * order.dishPrice).toFixed(2)}</span>
              </li>
            ))
          ) : (
            <li>No order details available</li>
          )}
        </ul>
      </div>
      <button
        onClick={handleClose}
        className="bg-green-500 text-white px-4 py-2 rounded-md hover:bg-green-
600"
      >
        Close
      </button>
    </div>
  </div>
 );
};

export default OrderConfirmation;
```

Structure of Webpage

'use client'

```
import './globals.css'
import { AuthContextProvider } from './context/AuthContext'
import Navbar from './components/Navbar'


export default function RootLayout({ children }) {
 return (
   <html lang="en">
     {/*
       <head /> will contain the components returned by the nearest parent
       head.js. Find out more at https://beta.nextjs.org/docs/api-reference/file-conventions/head
     */}
     <head />
     <body>
       <AuthContextProvider>
         <Navbar/>
         {children}
       </AuthContextProvider>
     </body>
   </html>
 )
}
```

Home Page

'use client'

```
import React, { useState, useEffect } from "react";
import { usePathname } from "next/navigation";
import { db } from "./firebase/config";
import { getDocs, collection, where, query } from "firebase/firestore";
```

```
import chefImage from './MaleChef.png';
import CartModal from "./cartModal/page";

async function fetchKitchensByLocation(selectedLocation) {
  const q = query(
    collection(db, "Capstone Project - Kitchen"),
    where("Location", "==", selectedLocation)
  );
  const querySnapshot = await getDocs(q);
  const data = [];
  querySnapshot.forEach((doc) => {
    data.push({ id: doc.id, ...doc.data() });
  });
  return data;
}

async function fetchKitchensByCuisine(selectedCuisine, selectedLocation) {
  const q = query(
    collection(db, "Capstone Project - Kitchen"),
    where("Location", "==", selectedLocation),
    where("Cuisine", "==", selectedCuisine)
  );
  const querySnapshot = await getDocs(q);
  const data = [];
  querySnapshot.forEach((doc) => {
    data.push({ id: doc.id, ...doc.data() });
  });
  return data;
}
```

```
async function fetchRecommendations(kitchenName) {
  const q = query(
    collection(db, "recommendations"),
    where("kitchenName", "==", kitchenName)
  );
  const querySnapshot = await getDocs(q);
  const data = [];
  querySnapshot.forEach((doc) => {
    data.push({ id: doc.id, ...doc.data() });
  });
  return data;
}

export default function Home() {
  const pathname = usePathname();
  const [selectedLocation, setSelectedLocation] = useState("");
  const [selectedCuisine, setSelectedCuisine] = useState("");
  const [userData, setUserData] = useState([]);
  const [recommendations, setRecommendations] = useState([]); // State for
recommendations
  const [cartItems, setCartItems] = useState([]); // State for cart items
  const [cartOpen, setCartOpen] = useState(false); // State for cart modal visibility

  useEffect(() => {
    const fetchData = async () => {
      try {
        // Fetch kitchens based on selected cuisine and location
        if (selectedCuisine && selectedLocation) {
          const data = await fetchKitchensByCuisine(
            selectedCuisine,
            selectedLocation
```

```
        );
        setUserData(data);

        // Fetch recommendations for the selected cuisine and location
        const recommendationsData = await fetchRecommendations(
          selectedCuisine
        );
        setRecommendations(recommendationsData);
      } else if (selectedLocation) {
        const data = await fetchKitchensByLocation(selectedLocation);
        setUserData(data);
      }
    } catch (error) {
      console.error("Error fetching data:", error);
    }
  };

  fetchData();
}, [selectedCuisine, selectedLocation]);


const handleLocationSelection = async (event) => {
  const selectedLocationValue = event.target.value;
  setSelectedLocation(selectedLocationValue);
  console.log("Selected location:", selectedLocationValue);
  const data = await fetchKitchensByLocation(selectedLocationValue);
  setUserData(data);
};

const handleCuisineSelection = async (event) => {
```

```
    const selectedCuisineValue = event.target.value;
    setSelectedCuisine(selectedCuisineValue);
    console.log("Selected cuisine:", selectedCuisineValue);
    const data = await fetchKitchensByCuisine(
      selectedCuisineValue,
      selectedLocation
    );
    setUserData(data);

    // Fetch recommendations for the selected cuisine and location
    const recommendationsData = await
fetchRecommendations(selectedCuisineValue);
    setRecommendations(recommendationsData);
  };

  const handleKitchenSelection = async (selectedKitchen) => {
    console.log("Selected kitchen:", selectedKitchen);
    window.location.href = `./chefpage?kitchenName=${selectedKitchen}`;
  };

  // Function to toggle cart modal visibility
  const toggleCartModal = () => {
    setCartOpen(!cartOpen);
  };

  return (
    <main className="p-5">
      <h1 className="text-5xl mb-5">Fetch Data from Firebase</h1>

      <div className="mb-5">
        <h1>Select a location:</h1>
```

```
<select
  value={selectedLocation}
  onChange={handleLocationSelection}
  className="border border-gray-300 rounded p-2"
>
  <option value="">Select...</option>
  <option value="Borivali (W)">Borivali West</option>
  <option value="Borivali (E)">Borivali East</option>
  <option value="Andheri (W)">Andheri West</option>
  <option value="Juhu">Juhu</option>
  <option value="Mahim">Mahim</option>
  <option value="BKC">BKC</option>
 </select>
</div>

<div className="mb-5">
 <h1>Select a cuisine:</h1>
 <select
  value={selectedCuisine}
  onChange={handleCuisineSelection}
  className="border border-gray-300 rounded p-2"
 >
  <option value="">Select...</option>
  <option value="Maharashtrian">Maharashtrian</option>
  <option value="South Indian">South Indian</option>
  <option value="Goan">Goan</option>
  <option value="Gujarati ">Gujarati </option>
  <option value="Bengali">Bengali</option>
  <option value="North Indian">North Indian</option>
 </select>
```

```
    </div>


    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
      {userData.map((kitchen) => (
        <div key={kitchen.kitchenName} className="cursor-pointer" onClick={() =>
handleKitchenSelection(kitchen.kitchenName)}>
          <div className="border border-gray-300 rounded-lg shadow-md p-6 bg-
white flex">
            <img
              src={chefImage}
              alt="Chef"
              className="w-24 h-24 rounded-full mr-4"
            />
            <div className="flex flex-col gap-2">
              <p className="font-bold text-xl mb-4">{kitchen.kitchenName}</p>
              <div className="flex flex-col gap-2">
                <p className="text-gray-700">
                  <span className="font-bold">Kitchen Name:</span>
{kitchen.kitchenName}
                </p>
                <p className="text-gray-700">
                  <span className="font-bold">Cuisine:</span> {kitchen.Cuisine}
                </p>
                <p className="text-gray-700">
                  <span className="font-bold">FSSAI No:</span> {kitchen.FSSAI_No}
                </p>
                <p className="text-gray-700">
                  <span className="font-bold">Location:</span> {kitchen.Location}
                </p>
                <p className="text-gray-700">
                  <span className="font-bold">Veg/Non-Veg/Both:</span>
{kitchen.Veg_NonVeg_Both}
```

```
              </p>
              <p className="text-gray-700">
                <span className="font-bold">Chef Name:</span> {kitchen.chefName}
              </p>
              <p className="text-gray-700">
                <span className="font-bold">Meals Delivered:</span>
{kitchen.mealsDelivered}
              </p>
              <p className="text-gray-700">
                <span className="font-bold">Rating:</span> {kitchen.rating}
              </p>
            </div>
          </div>
        </div>
      </div>
    ))}
  </div>


  {cartOpen && <CartModal cartItems={cartItems} handleClose={() =>
setCartOpen(false)} />} {/* Pass cartItems and handleClose function */}
 </main>
 );
}
```

20% Overall Similarity

Top sources found in the following databases:

- 12% Internet database
- Crossref database
- 12% Submitted Works database
- 11% Publications database
- Crossref Posted Content database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	arxiv.org Internet	1%
2	Shubhada Labde, Vishesh Karan, Shubham Shah, Dhruv Krishnan. "Mov... Crossref	<1%
3	Reetu Singh, Pragya Dwivedi. "Food Recommendation Systems Based ... Crossref	<1%
4	medium.com Internet	<1%
5	freecodecamp.org Internet	<1%
6	Gang Huang. "E-Commerce Intelligent Recommendation System Based... Crossref	<1%
7	National University of Ireland, Galway on 2023-04-03 Submitted works	<1%
8	Qingna Pu, Bin Hu. "Intelligent Movie Recommendation System Based ... Crossref	<1%

9	University of London External System on 2023-05-10 Submitted works	<1%
10	University of Greenwich on 2023-05-18 Submitted works	<1%
11	researchgate.net Internet	<1%
12	research-test.aston.ac.uk Internet	<1%
13	University of Northampton on 2024-01-28 Submitted works	<1%
14	dev.to Internet	<1%
15	Emirates International School on 2023-01-18 Submitted works	<1%
16	Jing Yu, Jinaiing Shi, Yunwen Chen, Daqi Ji, Wenhai Liu, Zhijun Xie, Kai ... Crossref	<1%
17	Yifan Tai, Zhenyu Sun, Zixuan Yao. "Content-Based Recommendation ... Crossref	<1%
18	University of Technology, Sydney on 2023-05-05 Submitted works	<1%
19	pure.fh-ooe.at Internet	<1%
20	Huimin Li, Yongyi Chen, Dan Zhang, Lifeng Wu. "An Intelligent Recomm... Crossref	<1%

21	Shubham Khanduri, S. Prabakeran. "Hybrid Recommendation System ... Crossref	<1%
22	ABES Engineering College on 2023-05-29 Submitted works	<1%
23	Liverpool John Moores University on 2023-03-08 Submitted works	<1%
24	ieeexplore.ieee.org Internet	<1%
25	ea on 2024-04-02 Submitted works	<1%
26	gitlab2.eeecs.qub.ac.uk Internet	<1%
27	Raciel Yera Toledo, Ahmad A. Alzahrani, Luis Martinez. "A Food Recom... Crossref	<1%
28	scilit.net Internet	<1%
29	University of Northampton on 2023-04-23 Submitted works	<1%
30	ijeecs.iaescore.com Internet	<1%
31	arno.uvt.nl Internet	<1%
32	Roehampton University on 2024-03-05 Submitted works	<1%

33	University of Bedfordshire on 2023-11-03 Submitted works	<1%
34	Elisabeth Mayrhuber, Oliver Krauss. "User Profile-Based Recommendat... Crossref	<1%
35	Jinjuan Hu, Chao Xie. "Research and implementation of e-commerce in... Crossref	<1%
36	ijai.iaescore.com Internet	<1%
37	Intercollege on 2023-05-10 Submitted works	<1%
38	University of Derby on 2022-08-11 Submitted works	<1%
39	garuda.kemdikbud.go.id Internet	<1%
40	American University of Beirut on 2015-10-26 Submitted works	<1%
41	University of Hertfordshire on 2022-12-14 Submitted works	<1%
42	Kingston University on 2012-09-20 Submitted works	<1%
43	Nottingham Trent University on 2023-11-23 Submitted works	<1%
44	Universiti Kebangsaan Malaysia on 2013-04-18 Submitted works	<1%

45	mdpi.com Internet	<1%
46	slideshare.net Internet	<1%
47	Aston University on 2023-05-03 Submitted works	<1%
48	Communications in Computer and Information Science, 2012. Crossref	<1%
49	Coventry University on 2023-04-03 Submitted works	<1%
50	ksascholar.dri.sa Internet	<1%
51	s3.eu-central-1.amazonaws.com Internet	<1%
52	ijitee.org Internet	<1%
53	infoworld.com Internet	<1%
54	Asia Pacific University College of Technology and Innovation (UCTI) on... Submitted works	<1%
55	Asia Pacific University College of Technology and Innovation (UCTI) on... Submitted works	<1%
56	CSU, San Jose State University on 2024-03-26 Submitted works	<1%

57	Huaiyuan Yang, Hua Zhou, Yucheng Li. "A Review of Academic Recom... Crossref	<1%
58	Universiti Teknologi MARA on 2016-01-13 Submitted works	<1%
59	University of Hertfordshire on 2023-08-18 Submitted works	<1%
60	University of Westminster on 2019-05-05 Submitted works	<1%
61	Yan Xiao, Congdong Li, Vincenzo Liu. "DFM-GCN: A Multi-Task Learnin... Crossref	<1%