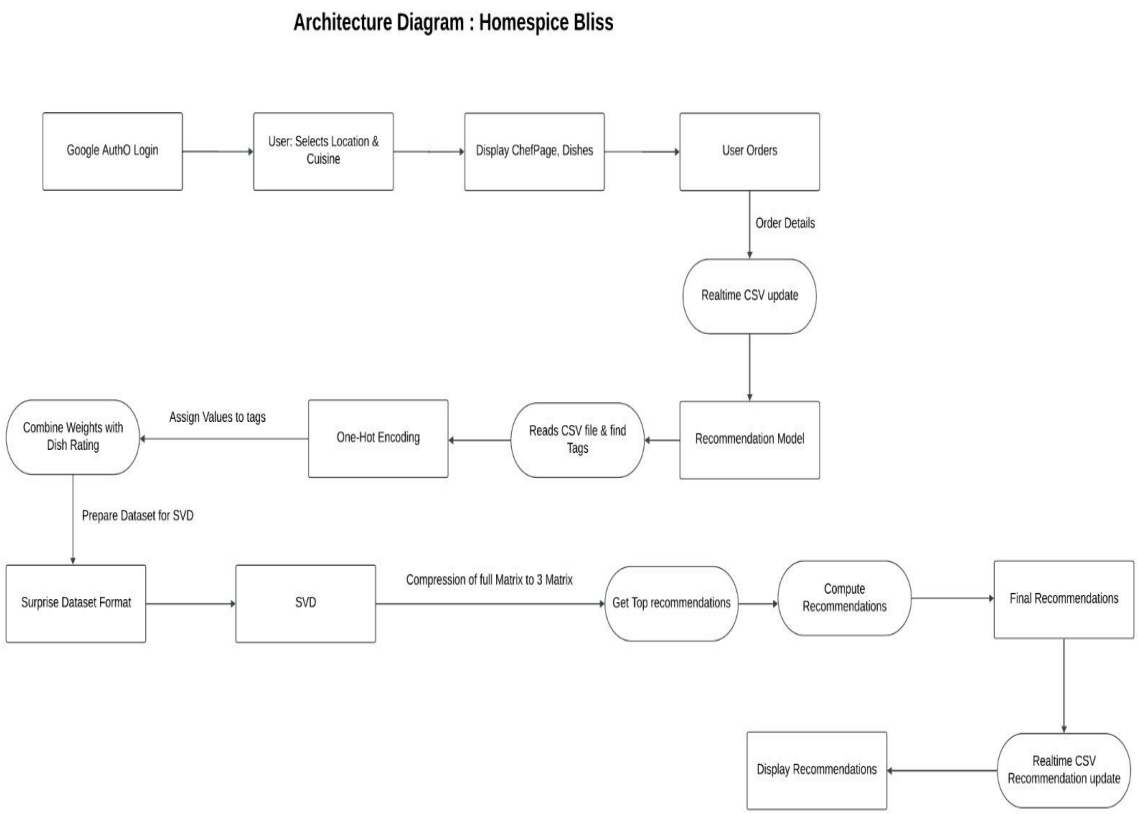
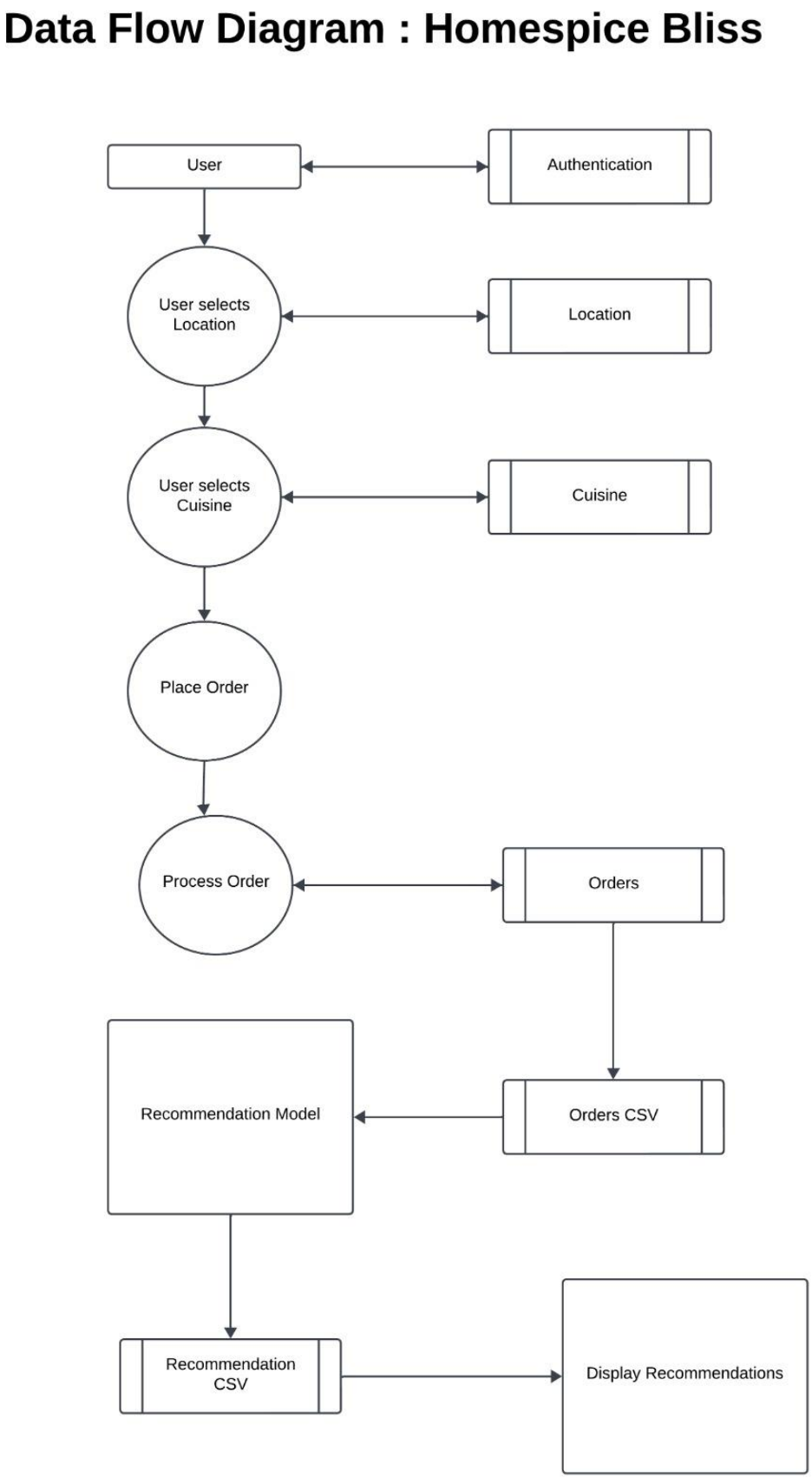
Chapter 3

Methodology & Implementation

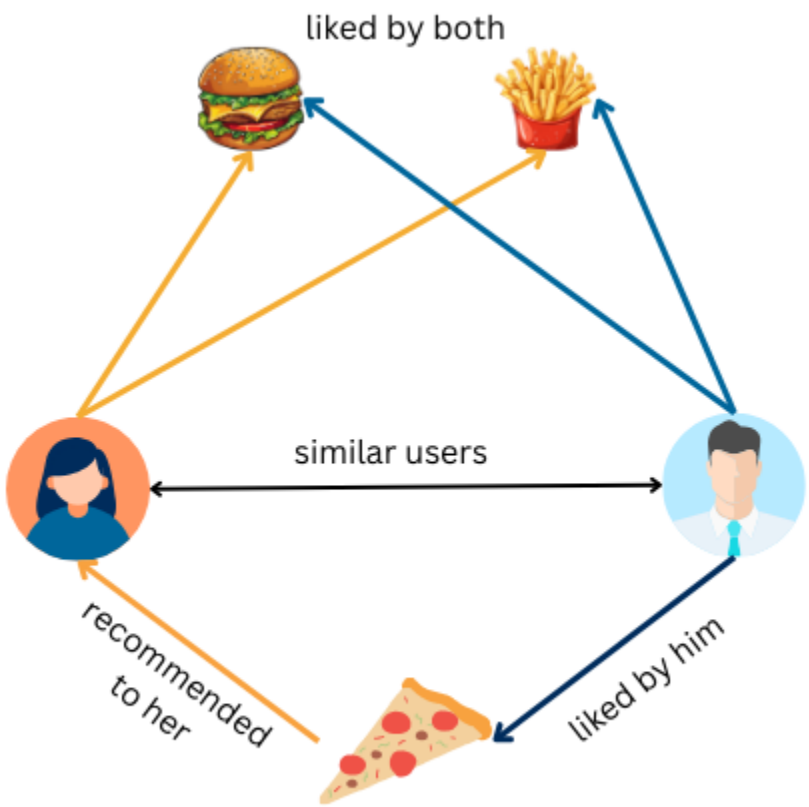
3.1 Block Diagram



Architecture Diagram



Data Flow Diagram



Collaborative Filtering

3.2 Hardware Description

3.3 Software Description

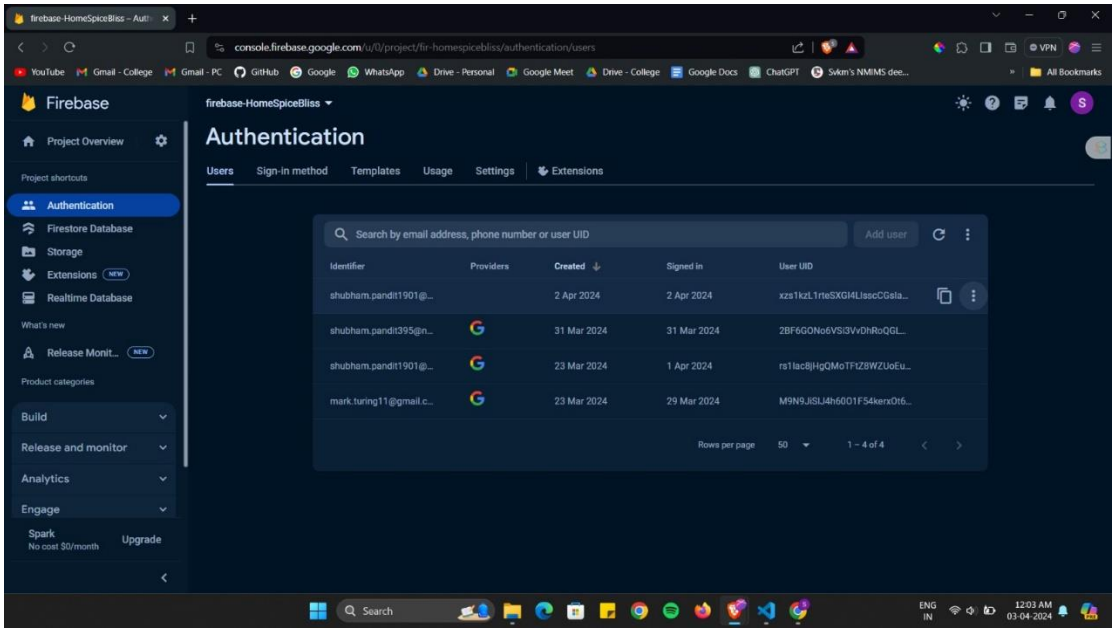
The algorithm used in the application is Tag based Collaborative Filtering System:

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.

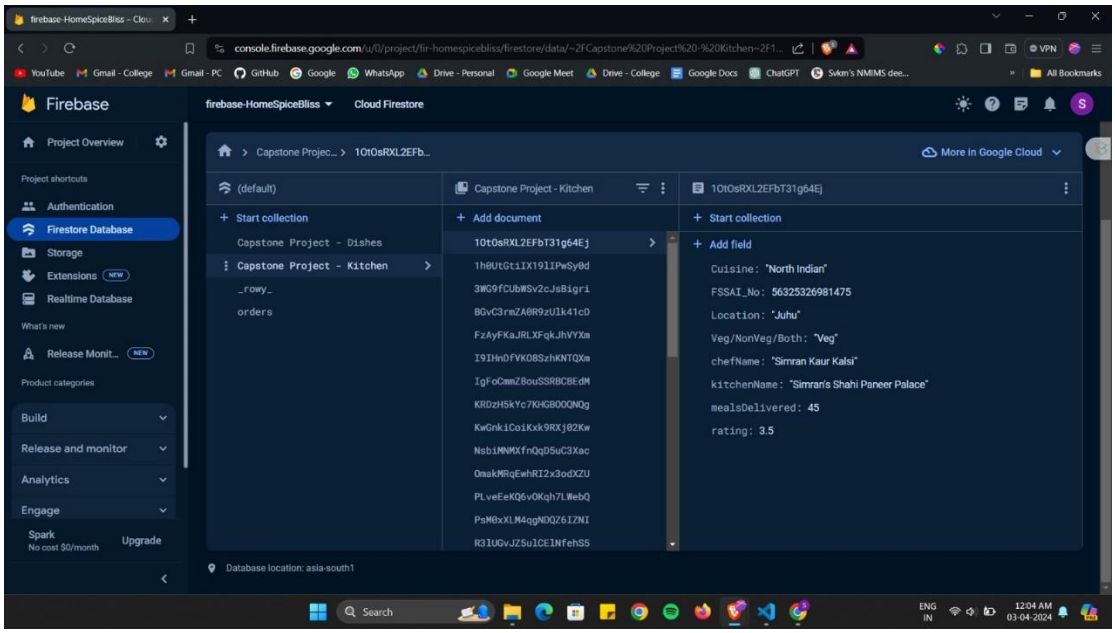
Frontend Technologies: Next.js

Database: Firebase (Firestore and Realtime Firebase), CSV

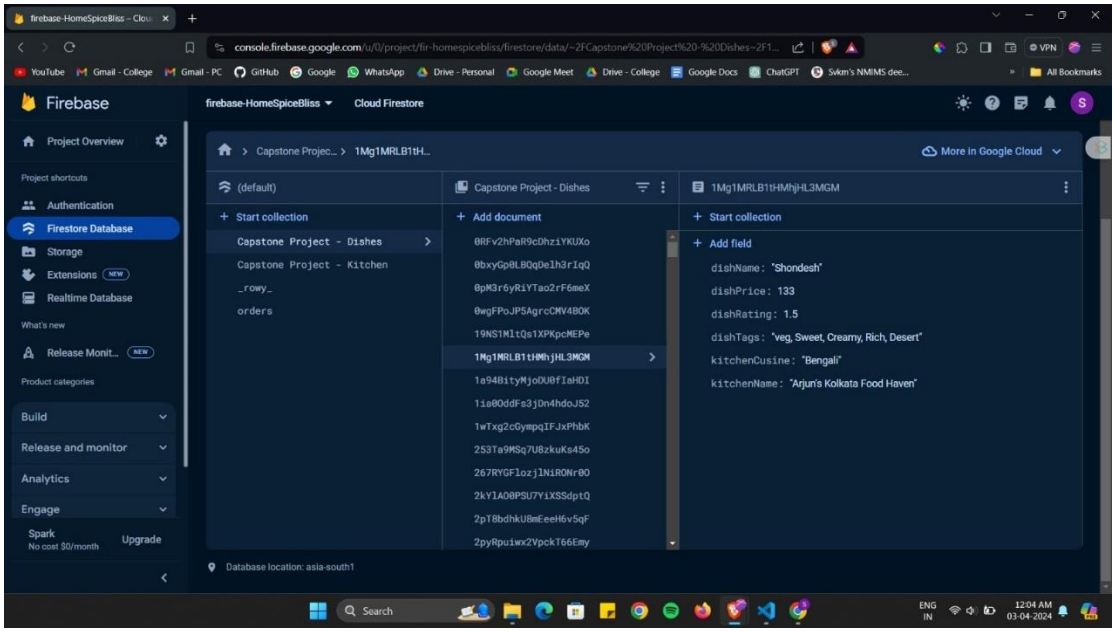
Recommendation System: One-hot encoding, Singular Value Decomposition (SVD), Pandas, Surprise, Itertools



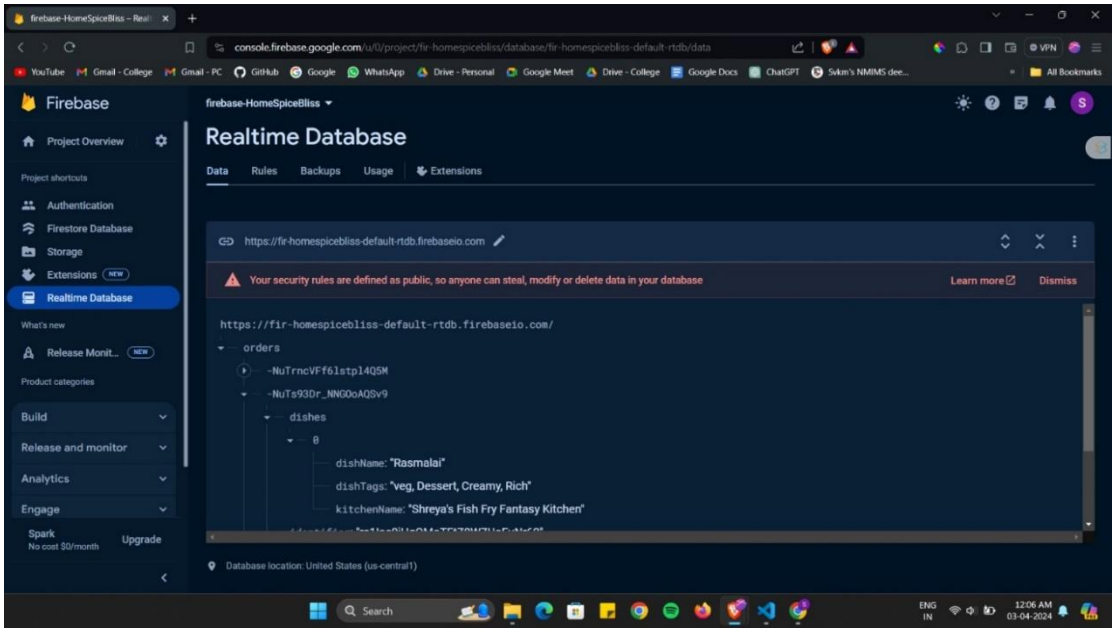
Authentication



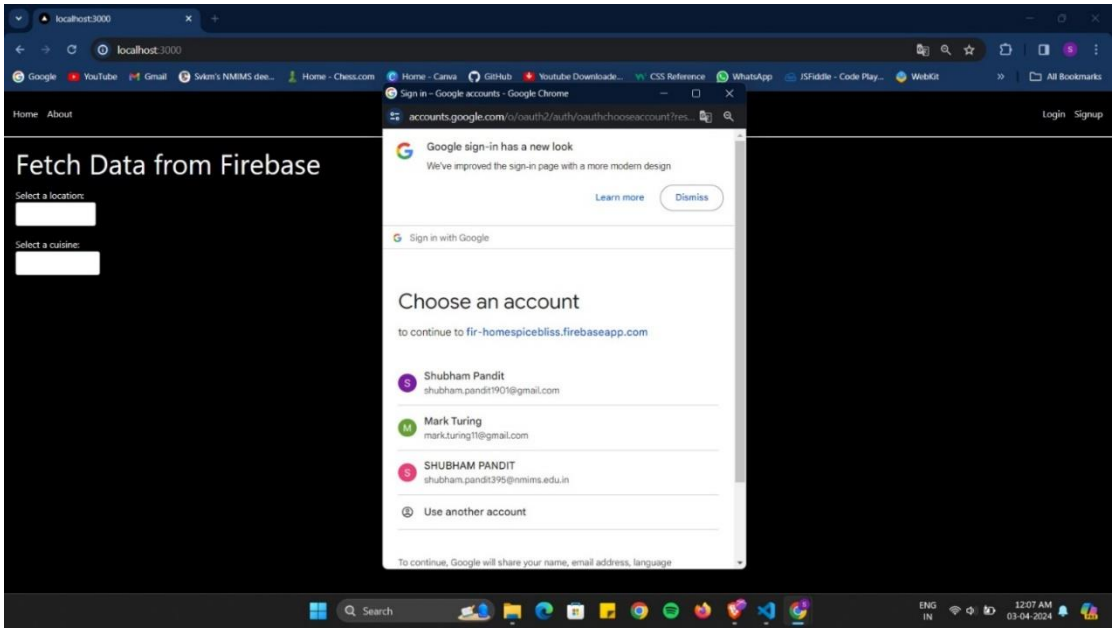
Kitchens Data



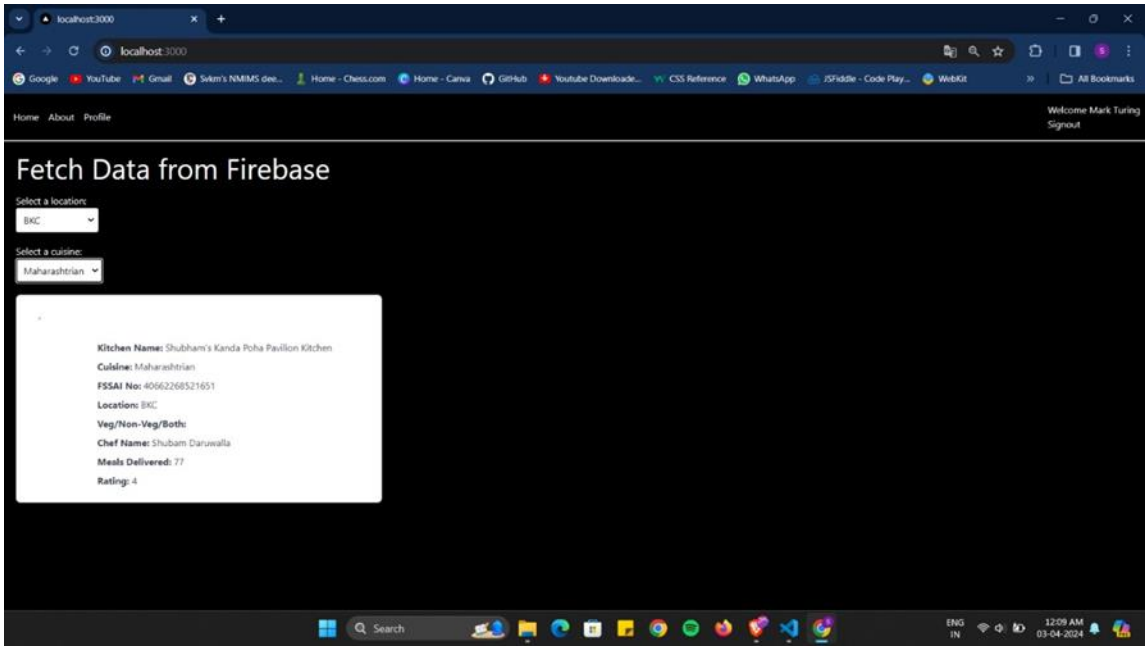
Dishes Data



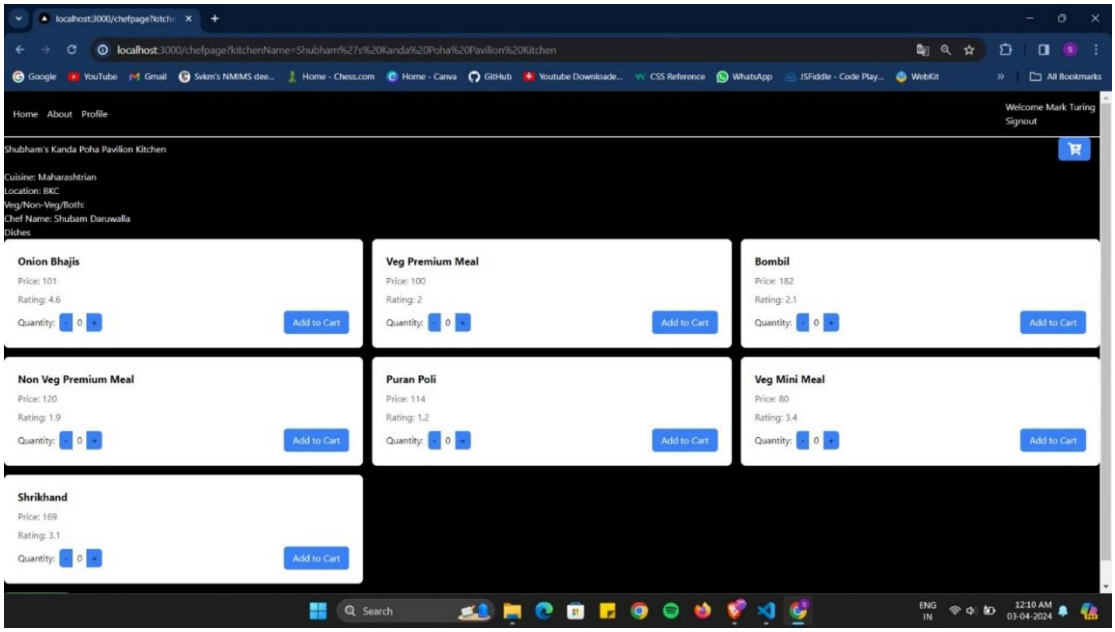
Order Placed-Details : Stored in Realtime Database in firebase



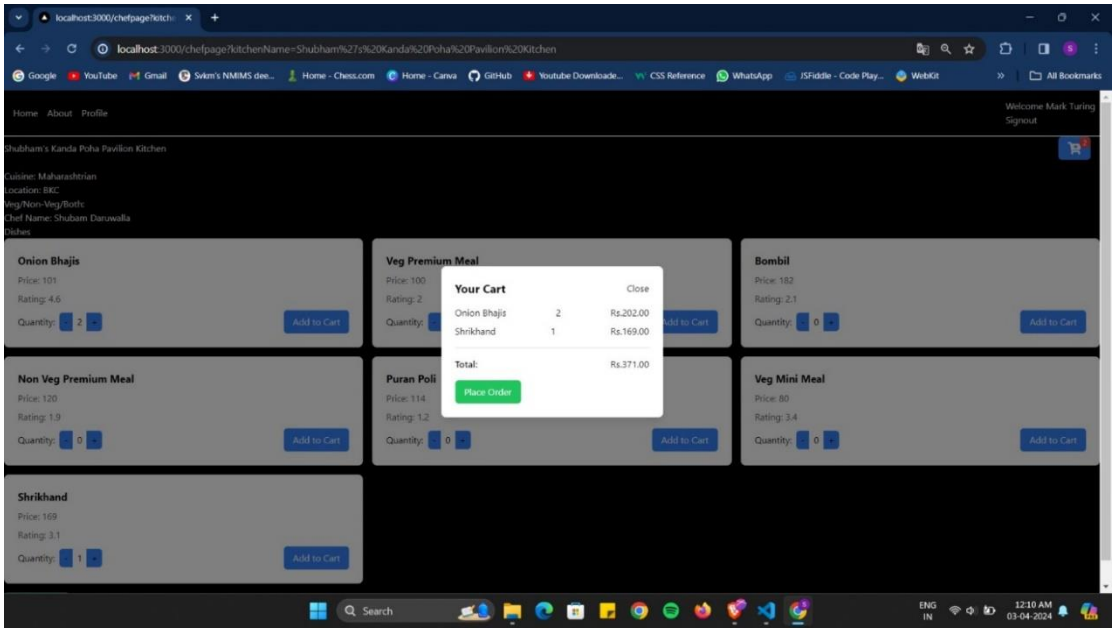
Login with Google AuthO



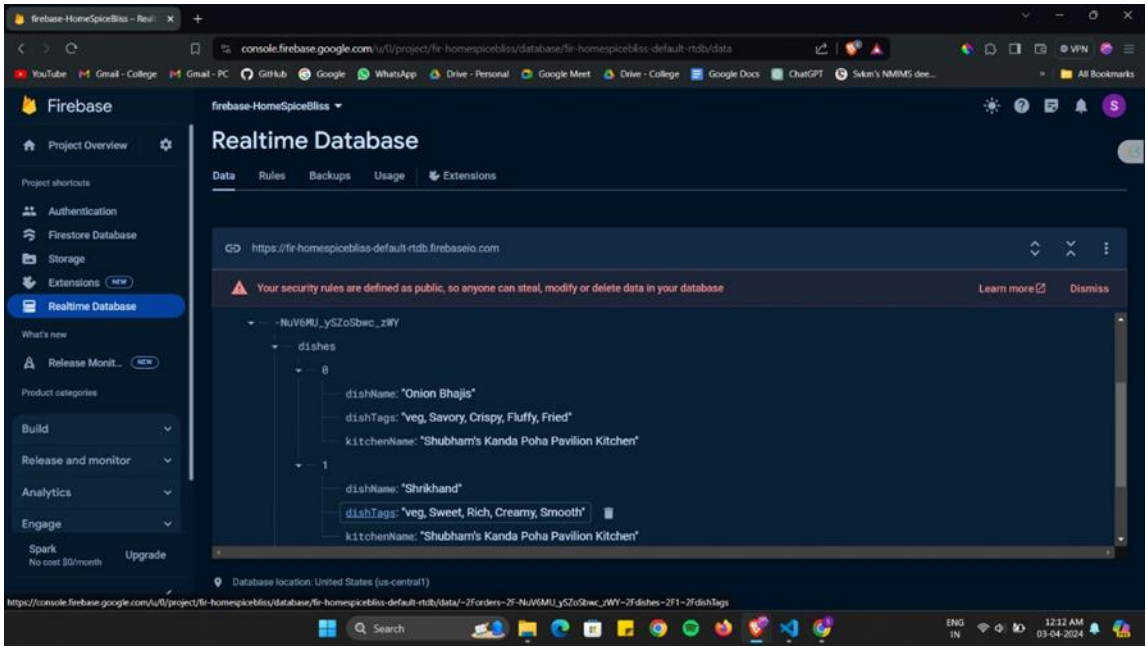
Filtered by Location and then Cusine



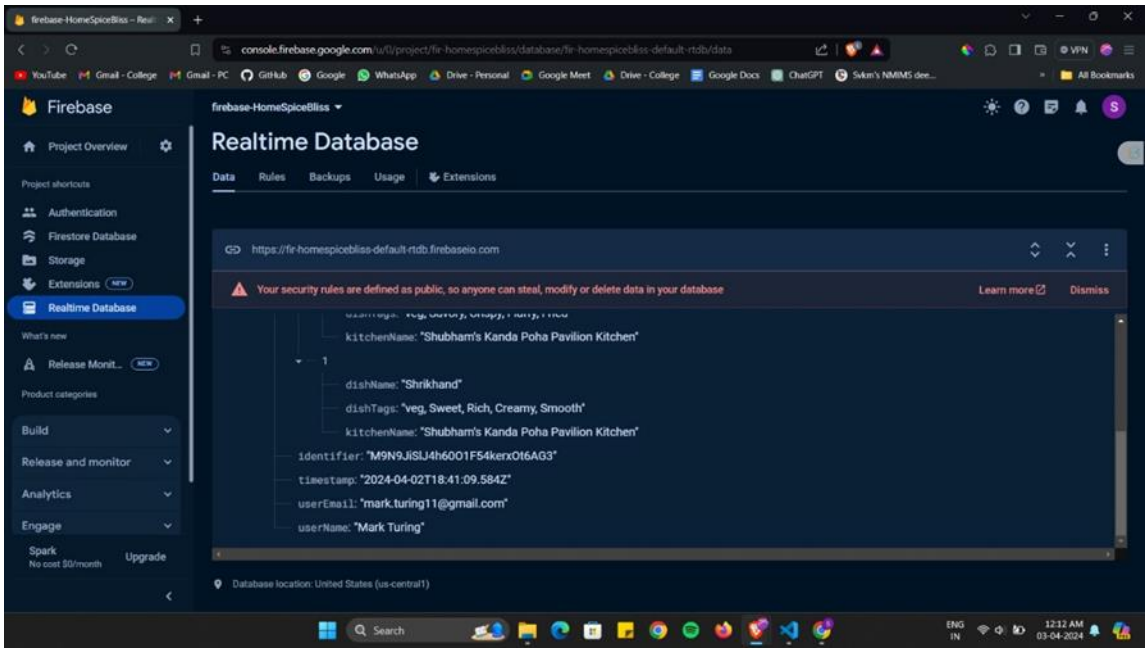
After selection of Kitchen., dishes are displayed



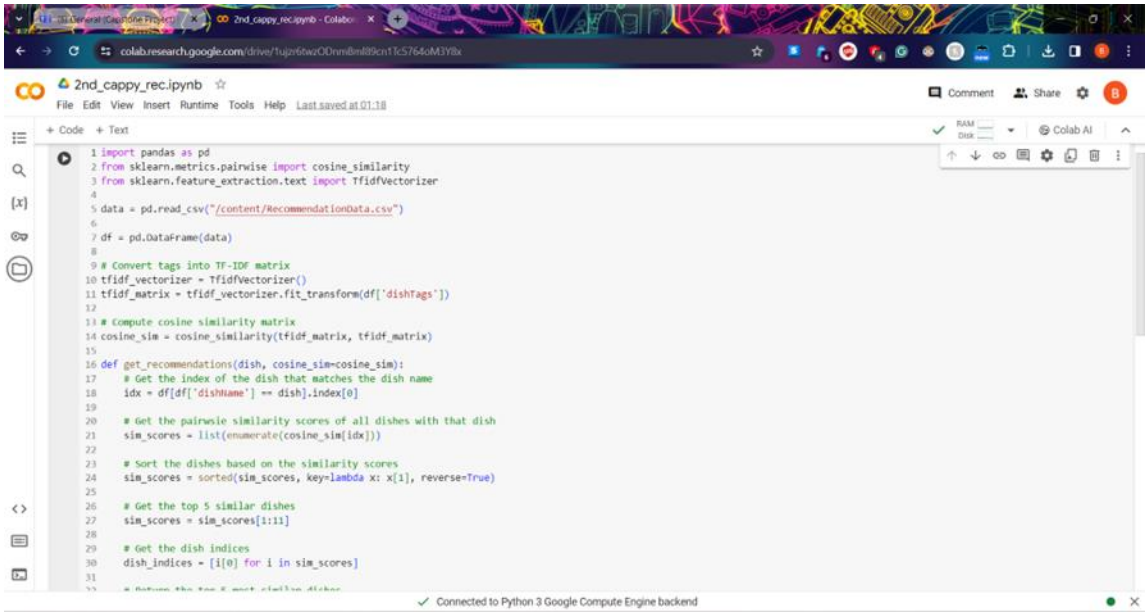
Dishes selected and added to cart



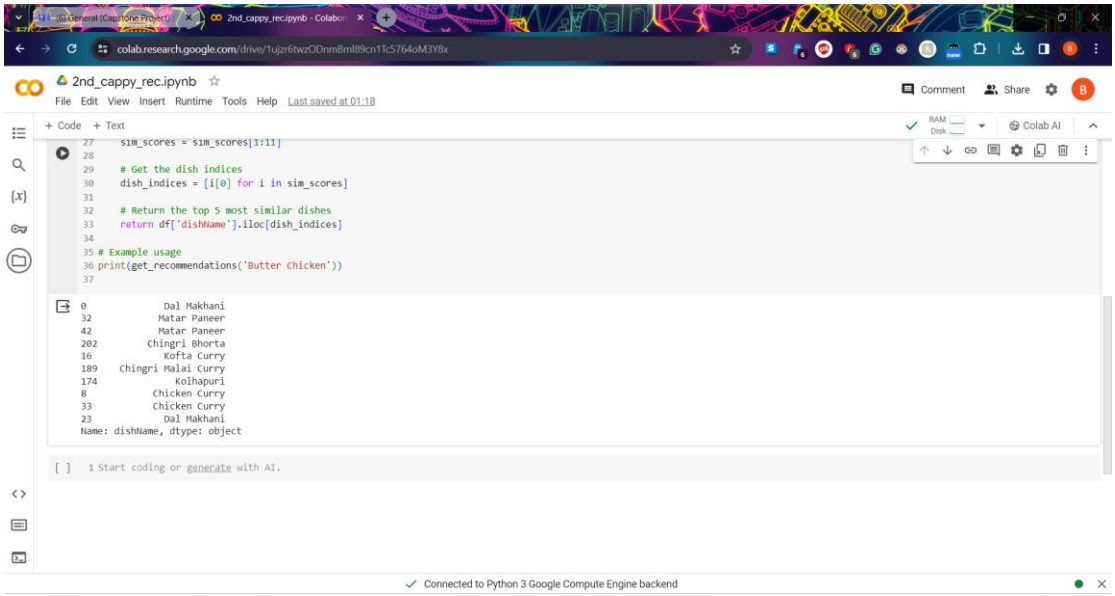
Order Updated into realtime database



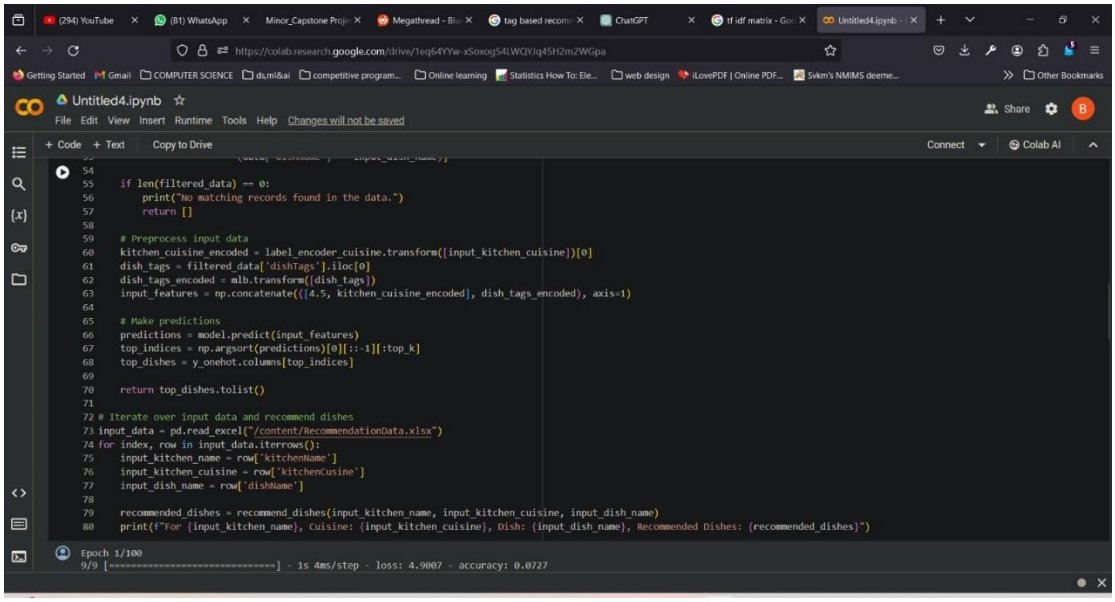
Order details updated with userEmail, userName and timestamps



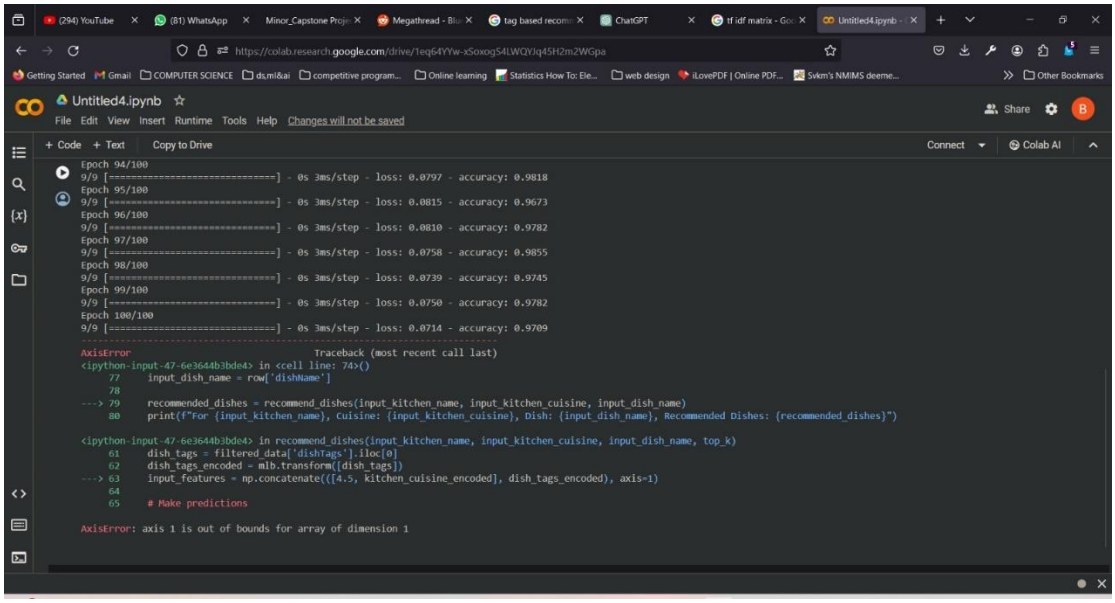
Tags based collaborative filtering source code



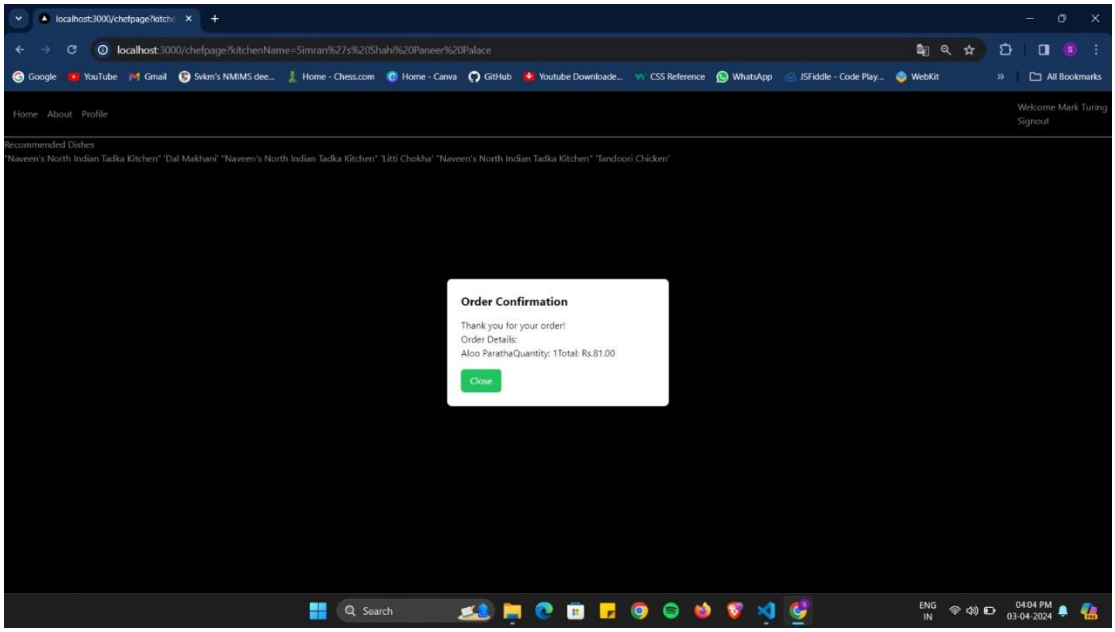
Dish recommendations based on tags for butter chicken



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)
```

```python
[ ] 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 = []
```

```python
        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 Premi	4.4	veg, Rich, F	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 D	North Indi	4	Chicken Cu	4.2	non-veg, S	0	0	0	0
11	Meera's D	North Indi	4	Shahi Pane	4.8	veg, Crean	0	0	0	0
12	Meera's D	North Indi	4	Khichdi	3.6	veg, Comf	0	0	0	0
13	Meera's D	North Indi	4	Kadai Pane	4.7	veg, Spicy,	0	0	0	0
14	Meera's D	North Indi	4	Rajma Cha	3.7	veg, Comf	0	0	0	0
15	Meera's D	North Indi	4	Veg Mini M	4.2	veg, Balan	1	0	0	0
16	Meera's D	North Indi	4	Veg Premi	4.4	veg, Rich, F	0	0	0	0
17	Meera's D	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 Premi	3.7	veg, Rich, F	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
0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	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	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
0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	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.