At the very beginning, user gen question ekk ahanda one “would you like to enjoy with flavorCraft or like to give your requirements?”kiyala..user ge response eka “like to enjoy with flavor craft nng” below function eka use wenda one.user reponse eke commas remove krla thama function ekata pass wenda one. parameters ganda one “location“,”restaurant type”,”allery_food_sentence”,”famous cusines”,”cost per person”.user respond kroth“I would like to give my requirements kiyala” ethokota use wenne meh function ekta below functions tika..ehema thama issalama question structure eka haadhanda one

```
####################################Location must be required
#location ,restaurant type,famouse cuisines,allergy food product and
cost_per_person

def longfamouscusinebased(Location, sentence,
cuisine_sentence,allergy_food_sentence,Cost_per_person_sentence):
    restaurants['Cleaned_Location'] =
restaurants['Cleaned_Location'].str.lower()
    restaurants['Cleaned_Restaurant_Type'] =
restaurants['Cleaned_Restaurant_Type'].str.lower()
    restaurants['Cleaned_Famous_Cuisines'] =
restaurants['Cleaned_Famous_Cuisines'].str.lower()
    restaurants['allergy_food_product'] =
restaurants['allergy_food_product'].str.lower()
    restaurants['Cost_per_person'] =
restaurants['Cost_per_person'].str.lower()


    sentence = sentence.lower()
    cuisine_sentence = cuisine_sentence.lower()
    allergy_food_sentence = allergy_food_sentence.lower()
    Cost_per_person_sentence = Cost_per_person_sentence.lower()

    sw = stopwords.words('english')
    lemm = WordNetLemmatizer()

    sentence_tokens = word_tokenize(sentence)
    cuisine_tokens = word_tokenize(cuisine_sentence)
    allergy_tokens = word_tokenize(allergy_food_sentence)
    Cost_tokens = word_tokenize(Cost_per_person_sentence)

    sentence_tokens = [lemm.lemmatize(word) for word in sentence_tokens if
word.isalnum() and word not in sw]
    cuisine_tokens = [lemm.lemmatize(word) for word in cuisine_tokens if
word.isalnum() and word not in sw]
    allergy_tokens = [lemm.lemmatize(word) for word in allergy_tokens if
word.isalnum() and word not in sw]
```

```
    Cost_tokens =[lemm.lemmatize(word) for word in Cost_tokens if
word.isalnum() and word not in sw]

    reqbased = restaurants[restaurants['Cleaned_Location'] ==
Location.lower()].copy()
    similarities = []
    for index, row in reqbased.iterrows():
        # Tokenize the restaurant type and famous cuisines
        temp_tokens = word_tokenize(row['Cleaned_Restaurant_Type'])
        temp_tokens = [lemm.lemmatize(word) for word in temp_tokens if
word.isalnum() and word not in sw]
        cuisine_tokens_copy = cuisine_tokens.copy()
        allergy_tokens_copy = allergy_tokens.copy()
        Cost_tokens_copy =Cost_tokens.copy()

        # Check for common tokens in restaurant type
        rvector = set(temp_tokens).intersection(set(sentence_tokens))

        # Check for common tokens in famous cuisines
        for cuisine_token in cuisine_tokens_copy:
            if cuisine_token in row['Cleaned_Famous_Cuisines']:
                rvector.add(cuisine_token)

         # Check for common tokens in allergy food products
        for allergy_token in allergy_tokens_copy:
            if allergy_token in row['allergy_food_product']:
                rvector.add(allergy_token)

        # Check for common tokens in allergy food products
        for Cost_token in Cost_tokens_copy:
            if Cost_token in row['Cost_per_person']:
                rvector.add(Cost_token)



        similarities.append(len(rvector))


    reqbased['similarity'] = similarities
    reqbased = reqbased.sort_values(by='similarity', ascending=False)

    return reqbased[['reviews', 'Restaurant_Name',
'Restaurant_description','Restaurant_type','recommended_dishes','menu_with
_allergy','menu_without_allergy','allergy_food_product',
                     'Cost_per_person','reviews','similarity']].head(10)
```

User reponse eka “I would like to give my requirements kiyala” nng methanin thama question starts wenne.

Next, user gen question ekk aanwa ne “wht is you location?” kiyala, ekata use wenne one below function eka.pass krnda ona parameters “Location”.eke user input eka gadhdhi commas remove krla pass krnda one .For example; “Galle, Sri lanka” kiyala user kiwwoth the space between Galle and Sri lanka should be removed and passed into the function below

```
def locationbased(Location):
    restaurants['Cleaned_Location']=
restaurants['Cleaned_Location'].str.lower()
    locationbase=restaurants[restaurants['Cleaned_Location']==Location.low
er()]
    if(locationbase.empty==0):
        hname=locationbase[['reviews', 'Restaurant_Name',
'Restaurant_description','Restaurant_type','recommended_dishes','menu_with
_allergy','menu_without_allergy','allergy_food_product',
                     'Cost_per_person','reviews']]
        return hname.head()
    else:
        print('No restaurants Available')
```

second, user gen question ekk ahanda one “wht is the restaurant type you prefer kiyala?”.ekta include wena parameters ”location”,”restaurant_type”.meketh user input eken commas ain wela thama function ekta pass wenda one

```
#Location must be required
#location and restaurant type

def longrestauranttypebased(Location, sentence):
    restaurants['Cleaned_Location'] =
restaurants['Cleaned_Location'].str.lower()
    restaurants['Cleaned_Restaurant_Type'] =
restaurants['Cleaned_Restaurant_Type'].str.lower()

    sentence = sentence.lower()
    sw = stopwords.words('english')
    lemm = WordNetLemmatizer()
    sentence_tokens = word_tokenize(sentence)
    sentence_tokens = [lemm.lemmatize(word) for word in sentence_tokens if
word.isalnum() and word not in sw]

    reqbased = restaurants[restaurants['Cleaned_Location'] ==
Location.lower()].copy()
    similarities = []
    for index, row in reqbased.iterrows():
        temp_tokens = word_tokenize(row['Cleaned_Restaurant_Type'])
        temp_tokens = [lemm.lemmatize(word) for word in temp_tokens if
word.isalnum() and word not in sw]
        rvector = set(temp_tokens).intersection(set(sentence_tokens))
        similarities.append(len(rvector))
    reqbased['similarity'] = similarities
    reqbased = reqbased.sort_values(by='similarity', ascending=False)
    return reqbased[['reviews', 'Restaurant_Name',
'Restaurant_description','Restaurant_type','recommended_dishes','menu_with
_allergy','menu_without_allergy','allergy_food_product',
                     'Cost_per_person','reviews','similarity']].head(10)
```

second, user gen question ekk ahanwa ne “what is the restaurant type and famous cusines you are looking for?” kiyala, ekata pass wenda one below function eka.pass krnda ona parmeters “location”,”restaurant type”,”cuisine_sentence”.eketh user input gadhdhi commas ain wenda one below function ekta pass wenda kalin

```
#Location must be required
#location ,restaurant type and famouse cuisines

def longfamouscusinebased(Location, sentence, cuisine_sentence):
    restaurants['Cleaned_Location'] =
restaurants['Cleaned_Location'].str.lower()
    restaurants['Cleaned_Restaurant_Type'] =
restaurants['Cleaned_Restaurant_Type'].str.lower()
    restaurants['Cleaned_Famous_Cuisines'] =
restaurants['Cleaned_Famous_Cuisines'].str.lower()

    sentence = sentence.lower()
    cuisine_sentence = cuisine_sentence.lower()
    sw = stopwords.words('english')
    lemm = WordNetLemmatizer()
    sentence_tokens = word_tokenize(sentence)
    cuisine_tokens = word_tokenize(cuisine_sentence)
    sentence_tokens = [lemm.lemmatize(word) for word in sentence_tokens if 
word.isalnum() and word not in sw]
    cuisine_tokens = [lemm.lemmatize(word) for word in cuisine_tokens if 
word.isalnum() and word not in sw]

    reqbased = restaurants[restaurants['Cleaned_Location'] == 
Location.lower()].copy()
    similarities = []
    for index, row in reqbased.iterrows():
        # Tokenize the restaurant type and famous cuisines
        temp_tokens = word_tokenize(row['Cleaned_Restaurant_Type'])
        temp_tokens = [lemm.lemmatize(word) for word in temp_tokens if 
word.isalnum() and word not in sw]
        cuisine_tokens_copy = cuisine_tokens.copy()

        # Check for common tokens in restaurant type
        rvector = set(temp_tokens).intersection(set(sentence_tokens))

        # Check for common tokens in famous cuisines
        for cuisine_token in cuisine_tokens_copy:
            if cuisine_token in row['Cleaned_Famous_Cuisines']:
                rvector.add(cuisine_token)

        similarities.append(len(rvector))

    reqbased['similarity'] = similarities
    reqbased = reqbased.sort_values(by='similarity', ascending=False)

    return reqbased[['reviews', 'Restaurant_Name', 
'Restaurant_description','Restaurant_type','recommended_dishes','menu_with
out_allergy',
       'menu_with_allergy','Cost_per_person', 
'allergy_food_product','similarity']].head(10)
```

Third, user gen question ekk ahanda ona “what is your estimated budget to spend?” kiyala, ekata use wenda one below function eka .ekata pass wenda ona parameters “location”,”restaurant_type”,”famous cusines”,”cost per person”

```
def longfamouscusinebased(Location, sentence, cuisine_sentence, 
Cost_per_person_sentence):
    restaurants['Cleaned_Location'] =
restaurants['Cleaned_Location'].str.lower()
    restaurants['Cleaned_Restaurant_Type'] =
restaurants['Cleaned_Restaurant_Type'].str.lower()
    restaurants['Cleaned_Famous_Cuisines'] =
restaurants['Cleaned_Famous_Cuisines'].str.lower()
    restaurants['Cost_per_person'] =
restaurants['Cost_per_person'].str.lower()

    sentence = sentence.lower()
    cuisine_sentence = cuisine_sentence.lower()
    Cost_per_person_sentence = Cost_per_person_sentence.lower()

    sw = stopwords.words('english')
    lemm = WordNetLemmatizer()

    sentence_tokens = word_tokenize(sentence)
    cuisine_tokens = word_tokenize(cuisine_sentence)
    Cost_tokens = word_tokenize(Cost_per_person_sentence)

    sentence_tokens = [lemm.lemmatize(word) for word in sentence_tokens if
word.isalnum() and word not in sw]
    cuisine_tokens = [lemm.lemmatize(word) for word in cuisine_tokens if
word.isalnum() and word not in sw]
    Cost_tokens = [lemm.lemmatize(word) for word in Cost_tokens if
word.isalnum() and word not in sw]

    reqbased = restaurants[restaurants['Cleaned_Location'] ==
Location.lower()].copy()
    similarities = []

    for index, row in reqbased.iterrows():
        # Tokenize the restaurant type and famous cuisines
        temp_tokens = word_tokenize(row['Cleaned_Restaurant_Type'])
        temp_tokens = [lemm.lemmatize(word) for word in temp_tokens if
word.isalnum() and word not in sw]
        cuisine_tokens_copy = cuisine_tokens.copy()
        Cost_tokens_copy = Cost_tokens.copy()

        # Check for common tokens in restaurant type
        rvector = set(temp_tokens).intersection(set(sentence_tokens))

        # Check for common tokens in famous cuisines
        for cuisine_token in cuisine_tokens_copy:
            if cuisine_token in row['Cleaned_Famous_Cuisines']:
                rvector.add(cuisine_token)

        # Check for common tokens in Cost_per_person
        for Cost_token in Cost_tokens_copy:
            if Cost_token in row['Cost_per_person']:
                rvector.add(Cost_token)

        similarities.append(len(rvector))

    reqbased['similarity'] = similarities
    reqbased = reqbased.sort_values(by='similarity', ascending=False)

    return reqbased[['reviews', 'Restaurant_Name',
'Restaurant_description','Restaurant_type','recommended_dishes','menu_with
_allergy','menu_without_allergy','allergy_food_product',
                     'Cost_per_person','reviews','similarity']].head(10)
```

Next user gen question ekk ahnda one "do you any allergies or dietary restrictions ?" kiyala,user "yes " kiwwoth only , ekata use wenda one below function eka.eke parameters "location","restaurant_type","famous_cuisines","allergy_food_product".eketh user input eka gadhdhi commas ain krla thama function ekta ewanda one

```
def longfamouscusinebased(Location, sentence, cuisine_sentence,
allergy_food_sentence):
    restaurants['Cleaned_Location'] =
restaurants['Cleaned_Location'].str.lower()
    restaurants['Cleaned_Restaurant_Type'] =
restaurants['Cleaned_Restaurant_Type'].str.lower()
    restaurants['Cleaned_Famous_Cuisines'] =
restaurants['Cleaned_Famous_Cuisines'].str.lower()
    restaurants['allergy_food_product'] =
restaurants['allergy_food_product'].str.lower()

    sentence = sentence.lower()
    cuisine_sentence = cuisine_sentence.lower()
    allergy_food_sentence = allergy_food_sentence.lower()

    sw = stopwords.words('english')
    lemm = WordNetLemmatizer()

    sentence_tokens = word_tokenize(sentence)
    cuisine_tokens = word_tokenize(cuisine_sentence)
    allergy_tokens = word_tokenize(allergy_food_sentence)

    sentence_tokens = [lemm.lemmatize(word) for word in sentence_tokens if
word.isalnum() and word not in sw]
    cuisine_tokens = [lemm.lemmatize(word) for word in cuisine_tokens if
word.isalnum() and word not in sw]
    allergy_tokens = [lemm.lemmatize(word) for word in allergy_tokens if
word.isalnum() and word not in sw]

    reqbased = restaurants[restaurants['Cleaned_Location'] ==
Location.lower()].copy()
    similarities = []
    for index, row in reqbased.iterrows():
        # Tokenize the restaurant type and famous cuisines
        temp_tokens = word_tokenize(row['Cleaned_Restaurant_Type'])
        temp_tokens = [lemm.lemmatize(word) for word in temp_tokens if
word.isalnum() and word not in sw]
        cuisine_tokens_copy = cuisine_tokens.copy()
        allergy_tokens_copy = allergy_tokens.copy()

        # Check for common tokens in restaurant type
        rvector = set(temp_tokens).intersection(set(sentence_tokens))

        # Check for common tokens in famous cuisines
        for cuisine_token in cuisine_tokens_copy:
            if cuisine_token in row['Cleaned_Famous_Cuisines']:
                rvector.add(cuisine_token)

         # Check for common tokens in allergy food products
        for allergy_token in allergy_tokens_copy:
            if allergy_token in row['allergy_food_product']:
                rvector.add(allergy_token)

        similarities.append(len(rvector))


    reqbased['similarity'] = similarities
    reqbased = reqbased.sort_values(by='similarity', ascending=False)

    return reqbased[['reviews', 'Restaurant_Name',
'Restaurant_description','Restaurant_type','recommended_dishes','menu_with
_allergy','menu_without_allergy','allergy_food_product',
                     'Cost_per_person','reviews','similarity']].head(10)
```

Next, user gen user gen question ekk ahanda one,”do you have any other custom preferences when chosing a restaurant ?”kiyala.ethokta okkoma paramaters include krla function ekk hadhala thynwa as below.eka thama use wenda one.eka ganda ona “sentence” withrai.eketh user input eken commas remove krla thama function ekta pass wenda one

```
#part 3
#if user adked long based questions

import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

def longfamouscusinebased(sentence):
    restaurants['merged_info'] = restaurants[['Location', 
'Restaurant_type', 'famous_Cuisines', 'allergy_food_product', 
'Cost_per_person', 'recommended_dishes']].fillna('').agg(', '.join, 
axis=1)
    restaurants['merged_info'] = restaurants['merged_info'].str.lower()

    sentence = sentence.lower()

    sw = stopwords.words('english')
    lemm = WordNetLemmatizer()

    sentence_tokens = word_tokenize(sentence)
    sentence_tokens = [lemm.lemmatize(word) for word in sentence_tokens if 
word.isalnum() and word not in sw]

    reqbased = restaurants.copy()
    similarities = []
    for index, row in reqbased.iterrows():
        # Tokenize the merged information
        temp_tokens = word_tokenize(row['merged_info'])
        temp_tokens = [lemm.lemmatize(word) for word in temp_tokens if
word.isalnum() and word not in sw]

        # Check for common tokens
        rvector = set(temp_tokens).intersection(set(sentence_tokens))
        similarities.append(len(rvector))

    reqbased['similarity'] = similarities
    reqbased = reqbased.sort_values(by='similarity', ascending=False)

    return reqbased[['reviews', 'Restaurant_Name',
'Restaurant_description','Restaurant_type','recommended_dishes','menu_with
_allergy','menu_without_allergy','allergy_food_product',
                     'Cost_per_person','similarity']].head(10)
```

Also,first of all, above thibba haama function ekema restaurant recommendation welata non related words thibboth ewata “no responses” kiyala output ekk dhenda function ekk hadhala thynwa.ekta below function eka use wenne.ekta ona wena parameters “sentence” ..issalla user input eke bad words naathan wihtrai uda thyna functions walin output dhenne naathan print wenda ona me function eke thiyana whidhiyata “I’m sorry.I wont be able to assist you with that”…….

```
import pandas as pd
from nltk.tokenize import word_tokenize

# Read the CSV file containing bad words
badwords_df = pd.read_csv('nowords.csv')

# Assuming the column name containing bad words is 'badwords'
badwords = set(badwords_df['splitword'].str.lower())

def check_for_bad_words(sentence):
    # Tokenize the user's sentence and lowercase the words
    sentence_tokens = word_tokenize(sentence.lower())

    # Check if any of the words from the user's sentence match the bad
words
    for word in sentence_tokens:
        if word in badwords:
            return True  # Bad word found
    return False  # No bad words found

# Example usage:
user_input = "I'm in New York City book."
if check_for_bad_words(user_input):
    print("I'm sorry, I'm not able to assist with that. Could you please
ask me something related to restaurants?")
```