Here's a **detailed step-by-step documentation** for building your BTech final year project:

🥗 Project Title: Nutrition Tracker with Disease Risk Alert

📝 Project Overview

This application allows users to input their meals and personal health details to:

- Analyze nutritional intake
- Predict health risks (like diabetes, heart issues, obesity)
- Get healthy recommendations

⚙️ Step-Wise Implementation

✅ STEP 1: Project Setup

1. **Tools to Install**:
2. pip install streamlit pandas scikit-learn matplotlib joblib
3. **Folder Structure**:
4. nutrition-tracker/
5. ├── app.py
6. ├── model/
7. │ ├── disease_model.pkl
8. │ └── scaler.pkl
9. ├── data/
10. │ └── food_nutrients.csv
11. ├── utils/
12. │ └── nutrition_lookup.py
13. └── requirements.txt

✅ STEP 2: Build User Input Form (Streamlit)

Use Streamlit to create an interactive UI.

Fields:

- Name, Age, Gender

- Height, Weight, Activity Level
- Existing Diseases
- Daily Meals (text input)

```
import streamlit as st

st.title("Nutrition Tracker with Disease Risk Alert")

name = st.text_input("Name")
age = st.number_input("Age", min_value=1)
gender = st.selectbox("Gender", ["Male", "Female"])
weight = st.number_input("Weight (kg)")
height = st.number_input("Height (cm)")
activity = st.selectbox("Activity Level", ["Sedentary", "Light", "Moderate", "Active"])
diseases = st.multiselect("Existing Diseases", ["Diabetes", "Hypertension", "Heart Issues", "None"])

meal_input = st.text_area("Enter your meals (e.g., 2 chapatis, dal, rice)")
```

✅ **STEP 3: Nutrition Extraction**

Two options:

- Use **API** (Edamam, Spoonacular)
- Use **mock CSV file** (food_nutrients.csv) with nutrients for common Indian food

Example CSV:

```
food,calories,protein,fat,carbs,sugar,sodium
chapati,80,2.5,1.5,15,0,120
dal,150,9,2,20,2,200
rice,200,4,0.4,45,0,2
```

Write nutrition_lookup.py:

```
import pandas as pd

def get_nutrition(meal_string, csv_path="data/food_nutrients.csv"):
    food_db = pd.read_csv(csv_path)
```

```
    total = {"calories": 0, "protein": 0, "fat": 0, "carbs": 0, "sugar": 0, "sodium": 0}
    for food in food_db["food"]:
        if food in meal_string.lower():
            row = food_db[food_db["food"] == food].iloc[0]
            for key in total:
                total[key] += row[key]
    return total
```

✅ **STEP 4: Rule-Based or ML-Based Risk Prediction**

Option A: Rule-Based Logic

```
def get_health_risk(nutrients):
    alerts = []
    if nutrients["sugar"] > 50:
        alerts.append("⚠️ High sugar intake - Diabetes risk")
    if nutrients["sodium"] > 2300:
        alerts.append("⚠️ High sodium - Heart disease risk")
    if nutrients["calories"] > 2500:
        alerts.append("⚠️ High calorie intake - Obesity risk")
    return alerts
```

Option B: ML-Based Model

1. **Prepare Dataset** (CSV)

```
calories,protein,fat,carbs,sugar,sodium,label
2200,40,85,300,65,2400,1
1800,50,60,250,30,1800,0
```

2. **Train Model**

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import joblib

df = pd.read_csv("disease_risk.csv")
```

```
X = df.drop("label", axis=1)
y = df["label"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2)
model = RandomForestClassifier()
model.fit(X_train, y_train)

joblib.dump(model, "model/disease_model.pkl")
joblib.dump(scaler, "model/scaler.pkl")
```

3. **Predict in App**

```
import joblib
model = joblib.load("model/disease_model.pkl")
scaler = joblib.load("model/scaler.pkl")

input_data = scaler.transform([[nutrients['calories'], nutrients['protein'],
                                nutrients['fat'], nutrients['carbs'],
                                nutrients['sugar'], nutrients['sodium']]])
risk = model.predict(input_data)
```

✅ **STEP 5: Output Display and Visualization**

- Show nutrient chart
- Show risk messages
- Show recommendations

```
import matplotlib.pyplot as plt

st.subheader("Nutrient Breakdown")
st.write(nutrients)
st.write("Health Alerts:")
```

st.warning(alert) if risk else st.success("No major risk detected")

✅ **STEP 6: Optional Add-ons**

- Save history to CSV for each user
- Show weekly trend chart
- Recommend low-fat/low-sugar alternatives

✅ **STEP 7: Documentation and Presentation**

Prepare the following:

- ✅ Abstract / Synopsis
- ✅ PPT (10–12 slides)
- ✅ Final Report
- ✅ Screenshots of working UI
- ✅ Demo video (optional but powerful)

📂 **Deliverables**

File	Description
app.py	Main Streamlit app
nutrition_lookup.py	Food → nutrient logic
disease_model.pkl	Trained ML model
scaler.pkl	Scaler for input
food_nutrients.csv	Food nutrition data
disease_risk.csv	Dataset to train model
report.pdf	Project documentation
presentation.pptx	Final viva PPT

Would you like me to:

- Generate a mock **food nutrient CSV**?
- Create a **sample dataset to train your model**?

- Build the **Streamlit app code** for you?

Just say the word and I'll start with the next piece 🔧💻