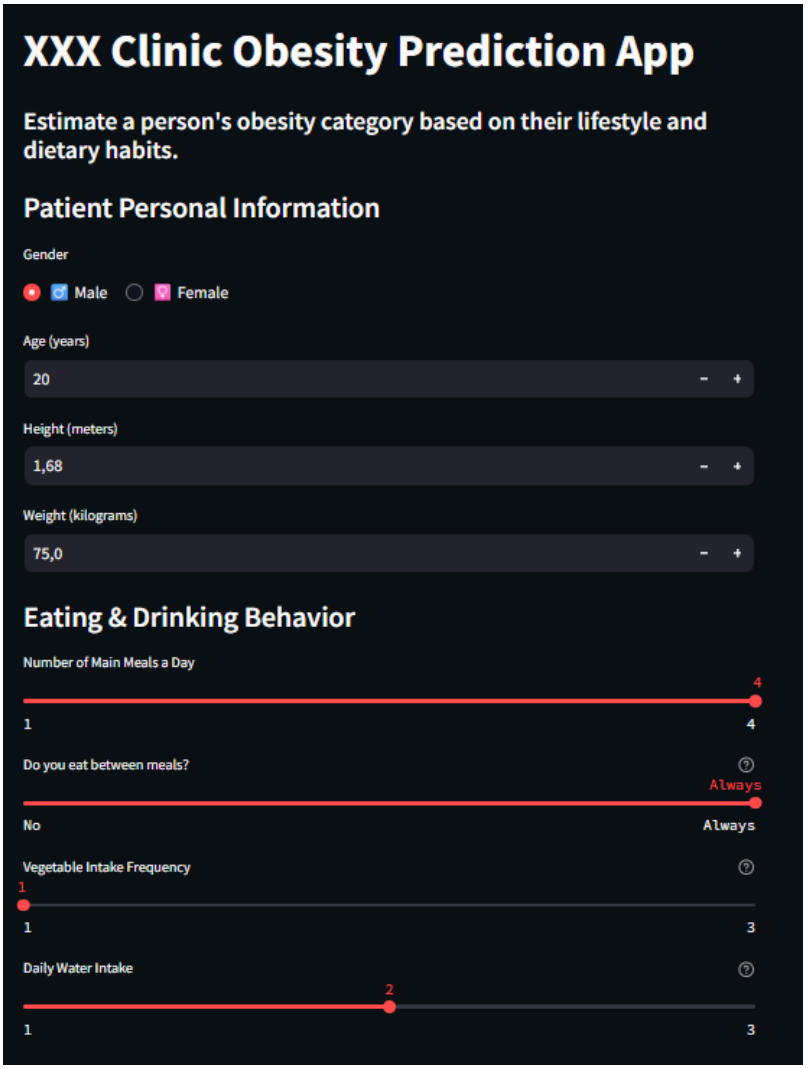
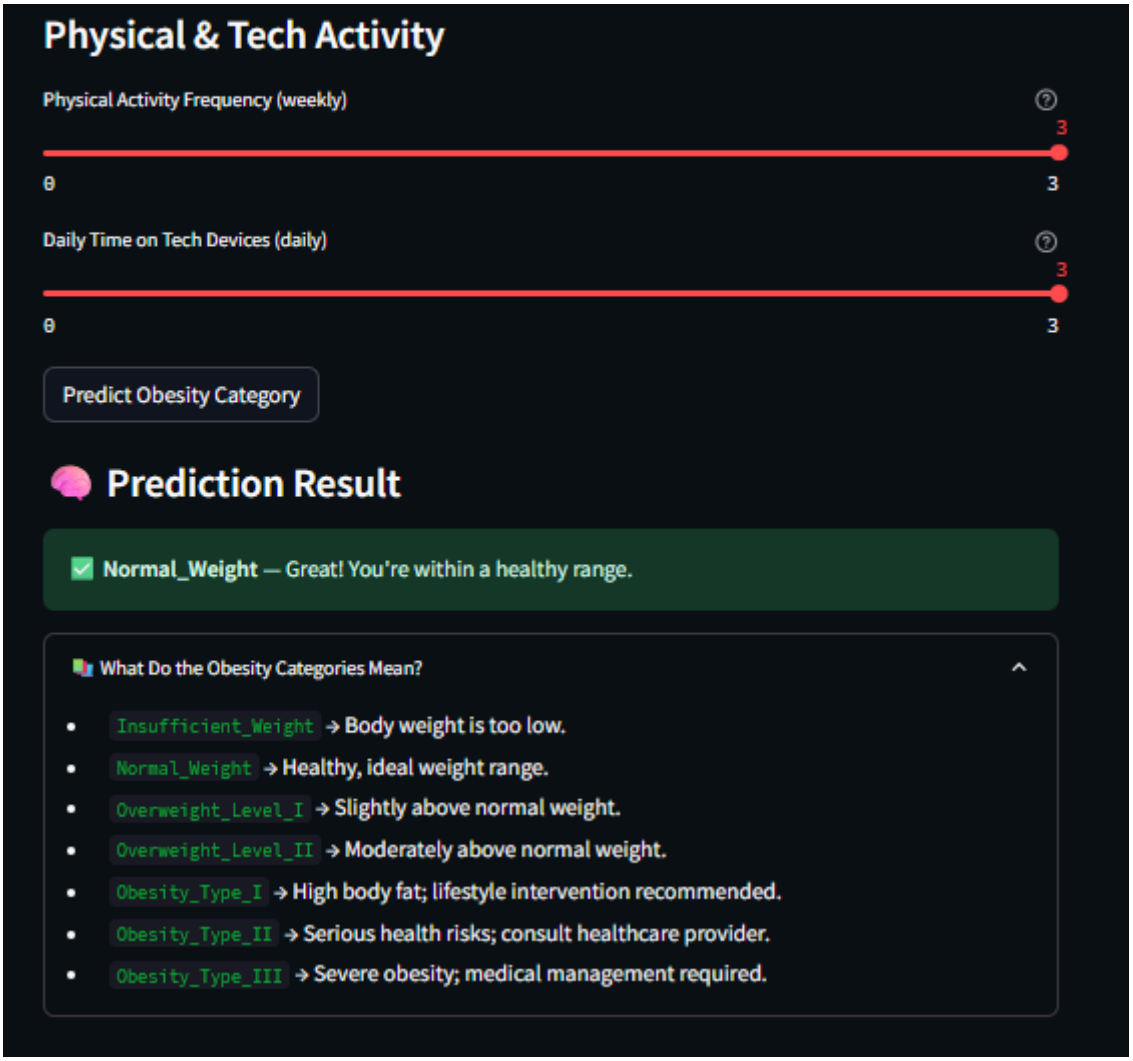
Test Case Result

Case 1:


```
{
  "Gender": "Male",
  "Age": 20,
  "Height": 1.68,
  "Weight": 75.0,
  "NCP": 4,
  "CAEC": "Always",
  "FCVC": 1,
  "CH2O": 2,
  "FAF": 3,
  "TUE": 3
}
```


Result:





Case 2:


```
{
  "Gender": "Female",
  "Age": 30,
  "Height": 1.50,
  "Weight": 91.1,
  "NCP": 3,
  "CAEC": "Frequently",
  "FCVC": 1,
  "CH2O": 1,
  "FAF": 0,
  "TUE": 2
}
```


Result:

XXX Clinic Obesity Prediction App

Estimate a person's obesity category based on their lifestyle and dietary habits.

Patient Personal Information

Gender

○ ♂ Male ● ♀ Female

Age (years)

30 − +

Height (meters)

1,50 − +

Weight (kilograms)

91,1 − +

Eating & Drinking Behavior

Number of Main Meals a Day

3

1 4

Do you eat between meals?

Frequently

No Always

Vegetable Intake Frequency

1

1 3

Daily Water Intake

1

1 3

Physical & Tech Activity

Physical Activity Frequency (weekly)

0

0 3

Daily Time on Tech Devices (daily)

2

0 3

Predict Obesity Category

🧠 Prediction Result

🚨 **Obesity_Type_I** — There's a high risk. Medical attention is recommended.

📚 What Do the Obesity Categories Mean? ⌄

Essay

1. **Testing done to ensure the model will always work well and will have no error when entered the production system**
   To ensure that the obesity prediction model works properly and doesn't fail in a production system, it is crucial to perform **comprehensive testing,** which could be done in several aspect such as:
   a. **Unit testing**
      Are to ensure each component in the pipeline functions as expected, for example for those who are in the preprocessing, prediction, and output mapping. This are done in the pre- mass production era.
      example of this test are like testing if the .map() function to ensure the "male" and "female" are properly converted into 0 and 1
   b. **Input Validation and robustness**
      This is to ensure the model will not crash when receiveng unusual, invalid, or malicious inputs. Which in my case is enforced at the frontend level using Streamlit's input components, such as: **st.number_input()** ensures age, height, and weight are numeric and within allowed ranges, or st.select**_slider() and st.radio()** limit categorical inputs like MTRANS, SMOKE, or CAEC to only valid options.
   c. **Integration Testing**
      to ensure the integration between the frontend and backend and model functions smoothly, such as verifying the JSON input from the frontend is correctly received and parsed by the backend

   Through this testing, we can evaluate wether our mode is perfectly ready to be produced and deployed!

2. **Model drift and data drift**
   a. **Model Drift**
      Is when the relationship between input features and the target label changes over time. For example, in the obesity system, originally, low physical activity (FAF = 0) strongly correlated with obesity. But now, more sedentary patients have normal weight due to healthier diets
      The impact will leave the model can no longer capture the current behavior of features, reducing prediction quality
   b. **Data Drift**
      Is when the distribution of input data during prediction differs from the data the model was trained on. For example when the average patient age becomes significantly younger, whereas the model was trained on adult data. The impact from this is the model will apply old learned patterns to new types of data and become less accurate.

   While these two are very common in real world machine learning life, for this model remain effective, the system must be built adaptive, retrainable, and monitored regularly

Presentation Video Link:
https://drive.google.com/drive/folders/1mJt46SbbHsB0e2uPfvFWkqmNczS9UKNR?usp=sharing