Understood. Here is the updated documentation for our new Resource Prediction model.

As requested, this guide includes all the setup steps and the new prediction script, but summarizes the training step without including the full code for train_ventilator_model.py.

Documentation: EMsync Resource Prediction Model (Phase 2)

1. Goal & Scope

- **Objective**: To build a series of machine learning models that predict the need for critical resources and interventions for ICU patients.
- **Full Scope**: The project aims to predict the need for:
  - Mechanical Ventilator
  - ICU Bed (Admission)
  - IV Infusion Pumps (Vasopressors)
  - Blood Products (Transfusions)
  - Dialysis Machine
- **Current Focus**: This guide documents the creation of the first model in the series: **Ventilator Prediction**.

---

2. Project Setup

This model uses a separate, dedicated environment to keep dependencies organized.

1. **Create the environment**:
   Bash
   `conda create --name emsync_resource_env python=3.9`

2. **Activate the environment**:
   Bash
   `conda activate emsync_resource_env`

3. **Install required libraries**:
   Bash
   pip install pandas scikit-learn xgboost matplotlib seaborn jupyterlab pyarrow

---

3. Data Preparation

The model is trained on a single preprocessed file (e.g., icu_interventions.xlsx). This file must contain the following:

- A unique identifier for each ICU stay (stay_id).
- A set of columns containing patient vital signs and other data to be used as **features**.
- A set of binary (0/1 or TRUE/FALSE) columns for each intervention to be used as **labels**.

For the Ventilator Model, the specific target column (label) used is **ventilation_needed**.

---

4. Model Training (Summary)

A Python script named **train_ventilator_model.py** was created to orchestrate the model training process. The script performs the following actions:

1. Loads the preprocessed data from the specified input file.
2. Separates the data into features (X) and the target label (y = ventilation_needed).
3. Splits the data into training and testing sets.
4. Trains an XGBoost Classifier model, optimized to handle the imbalanced nature of the data.
5. Evaluates the model's performance on the test set and prints a full report.
6. Saves the trained model to a file named **ventilator_model.pkl**.
7. Generates and saves a chart of the model's feature importances named **ventilator_feature_importance.png**.

To run the training process, the following command is used:

Bash

python train_ventilator_model.py

(Note: The full code for train_ventilator_model.py is omitted from this document as requested.)

---

5. How to Use the Ventilator Model (Inference)

To use the trained ventilator_model.pkl file to make a prediction for a new patient, you can use the following script.

1. Create a new file named predict_ventilator.py.
2. Copy and paste the code below into the file.

```python
# predict_ventilator.py
import joblib
import pandas as pd
import numpy as np

# --- 1. LOAD THE SAVED MODEL ---
model_filename = 'ventilator_model.pkl'
try:
    model = joblib.load(model_filename)
except FileNotFoundError:
    print(f"ERROR: Model file not found.")
    print(f"Please make sure '{model_filename}' is in the same folder.")
    exit()

# --- 2. PREPARE NEW PATIENT DATA ---
# Example data for a new patient.
# The columns MUST match the features the model was trained on.
new_patient_data = {
    'mean_hr': 110, 'mean_mbp': 75, 'mean_spo2': 92, 'mean_temp_c': 38.5,
    'min_hr': 95, 'min_mbp': 65, 'min_spo2': 88, 'min_temp_c': 38.0,
    'max_hr': 125, 'max_mbp': 85, 'max_spo2': 95, 'max_temp_c': 39.0,
    'antibiotics': 1, 'sedatives': 1, 'opioids': 1, 'fluids_total': 2000,
    # Set the correct ICU unit (1 for yes, 0 for all others)
    'unit_Cardiac Vascular Intensive Care Unit (CVICU)': 0,
    'unit_Coronary Care Unit (CCU)': 0, 'unit_Intensive Care Unit (ICU)': 0,
    'unit_Med/Surg': 0, 'unit_Medical Intensive Care Unit (MICU)': 1,
    'unit_Medical/Surgical Intensive Care Unit (MICU/SICU)': 0,
```

```
    'unit_Medicine': 0, 'unit_Medicine/Cardiology Intermediate': 0,
    'unit_Neuro Intermediate': 0, 'unit_Neuro Stepdown': 0,
    'unit_Neuro Surgical Intensive Care Unit (Neuro SICU)': 0,
    'unit_Neurology': 0, 'unit_PACU': 0, 'unit_Surgery/Trauma': 0,
    'unit_Surgery/Vascular/Intermediate': 0,
    'unit_Surgical Intensive Care Unit (SICU)': 0,
    'unit_Trauma SICU (TSICU)': 0, 'unit_<NA>': 0
}

input_df = pd.DataFrame([new_patient_data])

# Ensure column order matches the training data
training_features = model.get_booster().feature_names
input_df = input_df[training_features]

# --- 3. MAKE THE PREDICTION ---
prediction = model.predict(input_df)[0]
probability = model.predict_proba(input_df)[0][1] # Probability of "Yes"

# --- 4. DISPLAY THE RESULT ---
print("\n--- VENTILATOR PREDICTION RESULT ---")
if prediction == 1:
    print(f"Prediction: YES, the patient will likely need a ventilator.")
else:
    print(f"Prediction: NO, the patient will likely NOT need a ventilator.")
print(f"Confidence (Probability of Needing Ventilator): {probability:.2%}")
print("------------------------------------\n")
```

3. **Run the prediction script**:
   Bash
   ```
   python predict_ventilator.py
   ```