Of course. I've created a separate, final document for the web app creation process.

Here is the complete documentation for building and testing the model, now including the final step of creating the web app interface.

---

# ## Final Documentation: Building and Deploying the EMsync Severity Prediction Model

## Introduction

- **Goal**: This document provides a complete walkthrough for creating, testing, and deploying a machine learning model that predicts patient severity (Critical, Moderate, Low Urgency).
- **Final Model**: The final result is an XGBoost Classifier with a **71% Balanced Accuracy**.
- **Prerequisites**:
  - Anaconda installed on your system.
  - The following MIMIC-IV-ED data files (as .csv.gz) in your project folder: edstays.csv.gz, triage.csv.gz, vitalsign.csv.gz.

---

## Part 1: Building the Machine Learning Model

### Step 1: Environment Setup

1. **Open the Anaconda Prompt**.
2. **Create the environment**:
   Bash
   ```
   conda create --name emsync_env python=3.9
   ```
3. **Activate the environment**:
   Bash

```
conda activate emsync_env
```

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

---

### Step 2: Data Preprocessing & Cleaning

1. Create preprocess.py and check_nan_irregulars.py with the code from our previous discussions.
2. Run the scripts in order to generate cleaned_dataset.csv:
   Bash
   ```
   python preprocess.py
   python check_nan_irregulars.py
   ```

---

### Step 3: Training the Final Model

1. Create the train_model.py script with the final, corrected code we developed.
2. Run the training pipeline. This will generate the best model file (e.g., best_model_20250912_153322.pkl) and the feature importance chart.
   Bash
   ```
   python train_model.py
   ```

3. **Secure the model**: Rename the best model file for permanent use.
   Bash
   ```
   ren best_model_20250912_153322.pkl xgboost_baseline_71bal_acc.pkl
   ```

---

## Part 2: Deploying the Model with a Web App

### Step 4: Install Streamlit

1. In your active Anaconda Prompt (emsync_env), install the web app library:
   Bash
   ```
   pip install streamlit
   ```

---

### Step 5: Create the Web App Script

1. In your project folder, create a new file named app.py.
2. Copy and paste the entire code block below into this file.
   Python
   ```
   # app.py
   import streamlit as st
   import joblib
   import pandas as pd
   import numpy as np

   # --- Page Configuration ---
   st.set_page_config(
       page_title="EMsync Severity Predictor",
       page_icon="🚑",
       layout="wide"
   )

   # --- Load Model and Functions ---
   def add_engineered_features(df):
       df = df.copy()
       sbp_safe = df["sbp"].replace(0, np.nan)
       df["shock_index"] = df["heartrate"] / sbp_safe
       df["pulse_pressure"] = df["sbp"] - df["dbp"]
       df["hypoxia_flag"] = (df["o2sat"] < 90).astype(int)
       df["fever_flag"] = (df["temperature"] >= 38).astype(int)
       return df

   model_filename = 'xgboost_baseline_71bal_acc.pkl'
   try:
       model = joblib.load(model_filename)
   except FileNotFoundError:
       st.error(f"Model file '{model_filename}' not found. Please make sure it's in the same folder.")
       st.stop()
   ```

```
# --- Streamlit App Interface ---
st.title('EMsync: Real-Time Severity Prediction')
st.write("This tool uses a trained XGBoost model to predict patient severity based on their initial
vital signs. Adjust the sliders in the sidebar to match the patient's vitals and click 'Predict'.")

# --- Input Vitals via Sidebar ---
st.sidebar.header('Patient Vitals Input')
temperature = st.sidebar.slider('Temperature (°C)', 35.0, 42.0, 36.8, 0.1)
heartrate = st.sidebar.slider('Heart Rate (bpm)', 40, 200, 80)
resprate = st.sidebar.slider('Respiration Rate (breaths/min)', 10, 40, 18)
o2sat = st.sidebar.slider('Oxygen Saturation (%)', 80, 100, 98)
sbp = st.sidebar.slider('Systolic Blood Pressure (mmHg)', 70, 180, 120)
dbp = st.sidebar.slider('Diastolic Blood Pressure (mmHg)', 40, 120, 80)
pain_median = st.sidebar.slider('Pain Score (0-10)', 0, 10, 2)

if st.sidebar.button('**Predict Severity**', use_container_width=True):
    # Prepare data for the model
    new_patient_data = {
        'temperature': temperature, 'heartrate': heartrate, 'resprate': resprate,
        'o2sat': o2sat, 'sbp': sbp, 'dbp': dbp, 'pain_median': pain_median,
        'heartrate_median': heartrate, 'heartrate_min': heartrate - 5,
        'heartrate_max': heartrate + 5, 'resprate_median': resprate,
        'o2sat_median': o2sat, 'o2sat_min': o2sat - 1,
        'temperature_median': temperature, 'sbp_median': sbp, 'dbp_median': dbp,
        'shock_index': np.nan, 'pulse_pressure': np.nan,
        'hypoxia_flag': np.nan, 'fever_flag': np.nan
    }
    input_df = pd.DataFrame([new_patient_data])
    input_df_engineered = add_engineered_features(input_df)
    original_features = [
        'temperature', 'heartrate', 'resprate', 'o2sat', 'sbp', 'dbp', 'heartrate_median',
        'heartrate_min', 'heartrate_max', 'resprate_median', 'o2sat_median', 'o2sat_min',
        'temperature_median', 'sbp_median', 'dbp_median', 'pain_median', 'shock_index',
        'pulse_pressure', 'hypoxia_flag', 'fever_flag'
    ]
    input_df_final = input_df_engineered[original_features]

    # Make and display prediction
    prediction_index = model.predict(input_df_final)[0]
    prediction_probabilities = model.predict_proba(input_df_final)[0]
    severity_map = {0: 'Critical', 1: 'Moderate', 2: 'Low Urgency'}
    predicted_severity = severity_map[prediction_index]
    st.subheader('Prediction Result')
    if predicted_severity == 'Critical':
        st.error(f'### Predicted Severity: **{predicted_severity}**')
    elif predicted_severity == 'Moderate':
        st.warning(f'### Predicted Severity: **{predicted_severity}**')
    else:
        st.success(f'### Predicted Severity: **{predicted_severity}**')

    st.write('**Confidence Scores:**')
    prob_df = pd.DataFrame({
        'Severity': ['Critical', 'Moderate', 'Low Urgency'],
        'Confidence': prediction_probabilities
    })
    st.bar_chart(prob_df.set_index('Severity'))
```

---

### Step 6: Run Your Web App

1. **Save** the app.py file.
2. Go to your Anaconda Prompt and run this command:
   Bash
   ```
   streamlit run app.py
   ```
3. This will open a new tab in your web browser with the interactive application running.