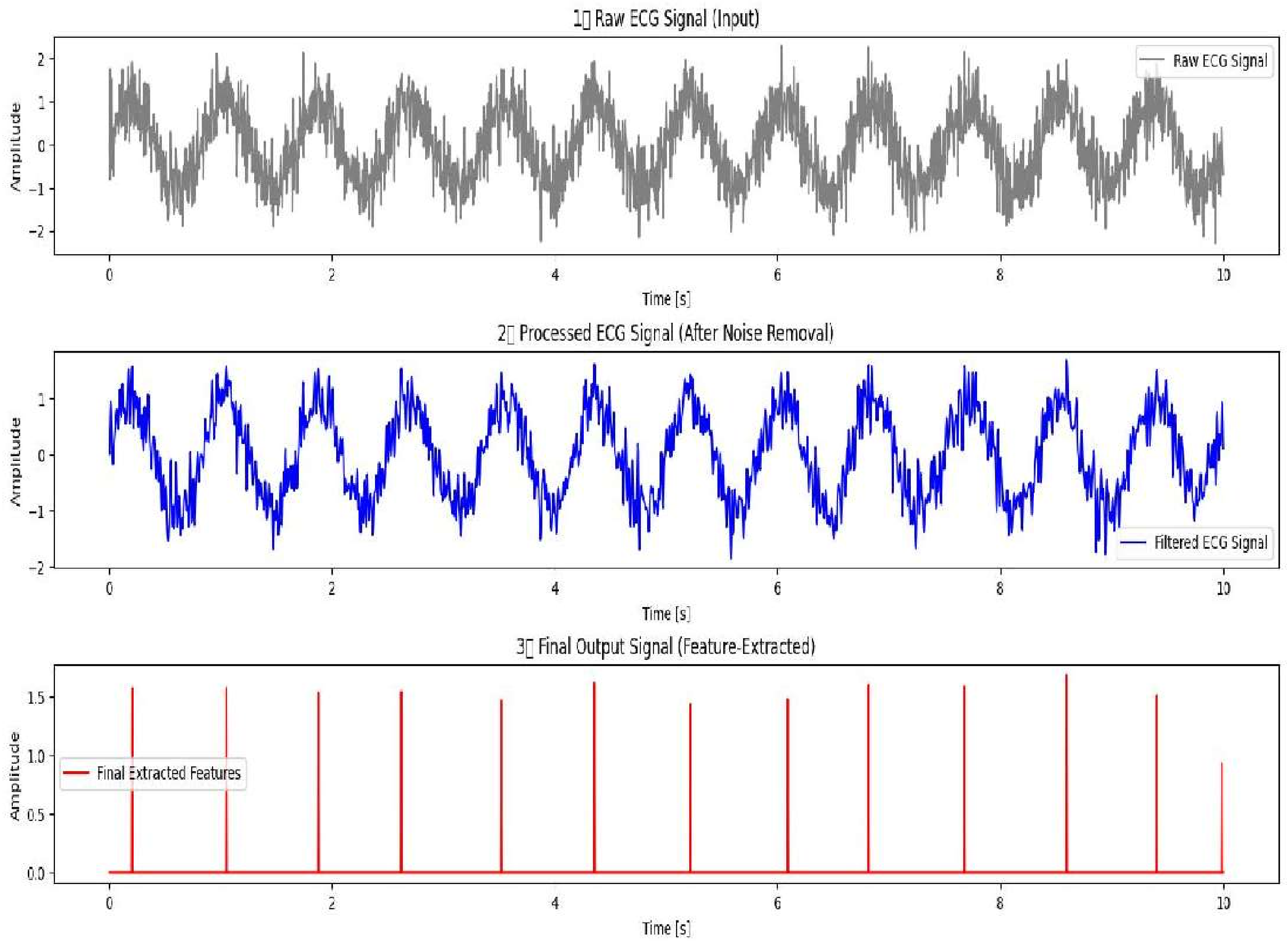
ECG SIGNAL FEATURE EXTRACTION

```
IMPORT NUMPY AS NP
IMPORT MATPLOTLIB.PYPLOT AS PLT
FROM SCIPY.FFT IMPORT FFT
FROM SCIPY.SIGNAL IMPORT FIND_PEAKS, BUTTER, FILTFILT

DEF BANDPASS_FILTER(SIGNAL, FS, LOWCUT=0.5, HIGHCUT=50,
ORDER=4):
  """
  APPLIES A BANDPASS FILTER TO REMOVE NOISE FROM THE ECG
SIGNAL.

  PARAMETERS:
    SIGNAL (ARRAY-LIKE): THE RAW ECG SIGNAL.
    FS (INT): SAMPLING FREQUENCY.
    LOWCUT (FLOAT): LOWER CUTOFF FREQUENCY IN HZ.
    HIGHCUT (FLOAT): HIGHER CUTOFF FREQUENCY IN HZ.
    ORDER (INT): ORDER OF THE BUTTERWORTH FILTER.

  RETURNS:
    ARRAY-LIKE: THE FILTERED SIGNAL.
  """
  NYQUIST = 0.5 * FS
  LOW = LOWCUT / NYQUIST
  HIGH = HIGHCUT / NYQUIST
  B, A = BUTTER(ORDER, [LOW, HIGH], BTYPE='BAND')
  RETURN FILTFILT(B, A, SIGNAL)

DEF EXTRACT_ECG_FEATURES(SIGNAL, FS):
  """
  EXTRACTS KEY FEATURES FROM AN ECG SIGNAL.
```

```
    Parameters:
      signal (array-like): The ECG signal.
      fs (int): Sampling frequency.

    Returns:
      dict: A dictionary containing extracted features.
    """
    features = {}

    # Time-domain features
    features['mean'] = np.mean(signal)
    features['std_dev'] = np.std(signal)
    features['rms'] = np.sqrt(np.mean(np.square(signal)))
    features['zero_crossings'] =
np.count_nonzero(np.diff(np.sign(signal)))
    features['energy'] = np.sum(np.square(signal))

    # Detect R-peaks
    peaks, _ = find_peaks(signal, height=np.max(signal) * 0.5,
distance=fs//2)
    features['num_r_peaks'] = len(peaks)

    if len(peaks) > 1:
      rr_intervals = np.diff(peaks) / fs
      features['mean_rr'] = np.mean(rr_intervals)
      features['std_rr'] = np.std(rr_intervals)
      features['heart_rate'] = 60 / np.mean(rr_intervals)
    else:
      features['mean_rr'] = 0
      features['std_rr'] = 0
      features['heart_rate'] = 0

    # Frequency-domain features using FFT
    n = len(signal)
    Freqs = np.fft.fftfreq(n, d=1/fs)
    Fft_values = fft(signal)
    Fft_magnitude = np.abs(fft_values)

    # Spectral features
    Spectral_centroid = np.sum(freqs * fft_magnitude) /
np.sum(fft_magnitude)
    Features['spectral_centroid'] = spectral_centroid
    Spectral_bandwidth = np.sqrt(np.sum(((freqs -
spectral_centroid)**2) * fft_magnitude) /
np.sum(fft_magnitude))
    Features['spectral_bandwidth'] = spectral_bandwidth
    Fft_magnitude_norm = fft_magnitude /
np.sum(fft_magnitude)
    Spectral_entropy = -np.sum(fft_magnitude_norm *
np.log2(fft_magnitude_norm + 1e-10))
    Features['spectral_entropy'] = spectral_entropy

    Return features

Def plot_ecg_signals(raw_signal, processed_signal,
final_signal, fs):
    """
    Plots three different ECG signals: Raw, Processed, and
Final Output.

    Parameters:
        raw_signal (array-like): Original ECG signal.
        processed_signal (array-like): Filtered ECG signal.
        final_signal (array-like): Feature-extracted signal.
        fs (int): Sampling frequency.
    """
    N = len(raw_signal)
    T = np.arange(0, n) / fs
    plt.figure(figsize=(12, 8))

    # Plot Raw Signal
    plt.subplot(3, 1, 1)
    plt.plot(t, raw_signal, color="gray", label="Raw ECG Signal")
    plt.title("1️⃣ Raw ECG Signal (Input)")
    plt.xlabel("Time [s]")
    plt.ylabel("Amplitude")
    plt.legend()

    # Plot Processed Signal
    plt.subplot(3, 1, 2)
    plt.plot(t, processed_signal, color="blue", label="Filtered ECG Signal")
    plt.title("2️⃣ Processed ECG Signal (After Noise Removal)")
    plt.xlabel("Time [s]")
    plt.ylabel("Amplitude")
    plt.legend()

    # Plot Final Output Signal
    plt.subplot(3, 1, 3)
    plt.plot(t, final_signal, color="red", label="Final Extracted Features")
    plt.title("3️⃣ Final Output Signal (Feature-Extracted)")
    plt.xlabel("Time [s]")
    plt.ylabel("Amplitude")
    plt.legend()

    plt.tight_layout()
    plt.show()

# Example usage
```

```
if __name__ == "__main__":
    # Generate a synthetic ECG-like signal
    fs = 250  # Sampling frequency (ECG signals typically
250-500 Hz)
    t = np.linspace(0, 10, fs * 10)  # 10-second ECG signal
    raw_signal = np.sin(2 * np.pi * 1.2 * t) + 0.5 *
np.random.randn(len(t))  # Simulated ECG with noise

    # Apply bandpass filter to clean the signal
    processed_signal = bandpass_filter(raw_signal, fs)

    # Extract features (final signal based on R-peaks)
    features = extract_ecg_features(processed_signal, fs)
    final_signal = np.zeros_like(processed_signal)
    peaks, _ = find_peaks(processed_signal,
height=np.max(processed_signal) * 0.5, distance=fs//2)
    final_signal[peaks] = processed_signal[peaks]  # Show
only detected R-peaks

    # Print extracted features
    print("Extracted ECG Features:")
    for key, value in features.items():
        print(f"{key}: {value:.5f}")

    # Plot all three signals
    plot_ecg_signals(raw_signal, processed_signal,
final_signal, fs)
```

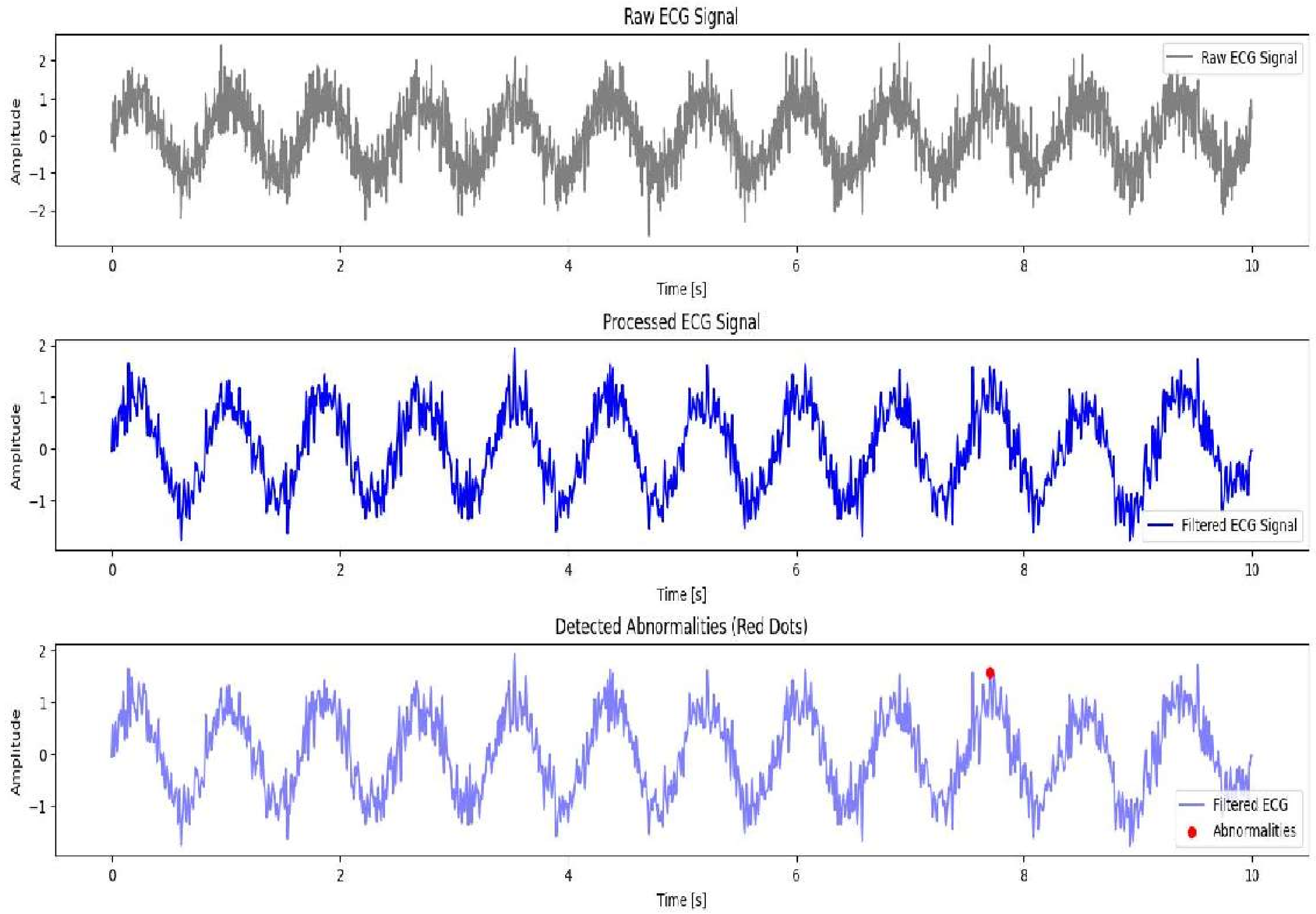

1 Raw ECG Signal (Input)
Raw ECG Signal
Amplitude
Time [s]
2 Processed ECG Signal (After Noise Removal)
Filtered ECG Signal
Amplitude
Time [s]
3 Final Output Signal (Feature-Extracted)
Final Extracted Features
Amplitude
Time [s]

ECG SIGNAL ABNORMALITY DETECTION

```
IMPORT NUMPY AS NP
IMPORT MATPLOTLIB.PYPLOT AS PLT
FROM SCIPY.SIGNAL IMPORT FIND_PEAKS, BUTTER, FILTFILT

DEF BANDPASS_FILTER(SIGNAL, FS, LOWCUT=0.5, HIGHCUT=50,
ORDER=4):
   """
   APPLIES A BANDPASS FILTER TO REMOVE NOISE FROM THE ECG
SIGNAL.
   """
   NYQUIST = 0.5 * FS
   LOW = LOWCUT / NYQUIST
   HIGH = HIGHCUT / NYQUIST
   B, A = BUTTER(ORDER, [LOW, HIGH], BTYPE='BAND')
   RETURN FILTFILT(B, A, SIGNAL)

DEF DETECT_ABNORMALITIES(SIGNAL, FS):
   """
   DETECTS ABNORMALITIES IN THE ECG SIGNAL BASED ON HEART
RATE VARIATIONS.
   """
   PEAKS, _ = FIND_PEAKS(SIGNAL,
HEIGHT=NP.PERCENTILE(SIGNAL, 95), DISTANCE=FS//2)

   IF LEN(PEAKS) < 2:
      RETURN NP.ZEROS_LIKE(SIGNAL), 0, 0, "NO R-PEAKS
DETECTED"

   RR_INTERVALS = NP.DIFF(PEAKS) / FS
```

```
    AVG_RR = NP.MEAN(RR_INTERVALS)
    STD_RR = NP.STD(RR_INTERVALS)
    HEART_RATE = 60 / AVG_RR IF AVG_RR > 0 ELSE 0

    ABNORMAL_SIGNAL = NP.ZEROS_LIKE(SIGNAL)
    FOR I, RR IN ENUMERATE(RR_INTERVALS):
        IF ABS(RR - AVG_RR) > 1.5 * STD_RR:
            ABNORMAL_SIGNAL[PEAKS[I]] = SIGNAL[PEAKS[I]]

    # CLASSIFY ABNORMALITY
    IF HEART_RATE > 100:
        CONDITION = "TACHYCARDIA (FAST HEART RATE)"
    ELIF HEART_RATE < 60:
        CONDITION = "BRADYCARDIA (SLOW HEART RATE)"
    ELSE:
        CONDITION = "NORMAL ECG"

    RETURN ABNORMAL_SIGNAL, HEART_RATE, STD_RR, CONDITION

DEF PLOT_ECG_SIGNALS(RAW_SIGNAL, PROCESSED_SIGNAL,
ABNORMAL_SIGNAL, FS):
    """
    PLOTS THREE ECG SIGNALS: RAW, PROCESSED, AND
ABNORMALITY DETECTION.
    """
    N = LEN(RAW_SIGNAL)
    T = NP.ARANGE(N) / FS

    PLT.FIGURE(FIGSIZE=(12, 8))

    PLT.SUBPLOT(3, 1, 1)
    PLT.PLOT(T, RAW_SIGNAL, COLOR="GRAY", LABEL="RAW ECG
SIGNAL")
    PLT.TITLE("RAW ECG SIGNAL")
    PLT.XLABEL("TIME [S]")
    PLT.YLABEL("AMPLITUDE")
    PLT.LEGEND()

    PLT.SUBPLOT(3, 1, 2)
    PLT.PLOT(T, PROCESSED_SIGNAL, COLOR="BLUE",
LABEL="FILTERED ECG SIGNAL")
    PLT.TITLE("PROCESSED ECG SIGNAL")
    PLT.XLABEL("TIME [S]")
    PLT.YLABEL("AMPLITUDE")
    PLT.LEGEND()

    PLT.SUBPLOT(3, 1, 3)
    PLT.PLOT(T, PROCESSED_SIGNAL, COLOR="BLUE", ALPHA=0.5,
LABEL="FILTERED ECG")
    PLT.SCATTER(T[ABNORMAL_SIGNAL > 0],
ABNORMAL_SIGNAL[ABNORMAL_SIGNAL > 0], COLOR="RED",
LABEL="ABNORMALITIES", ZORDER=3)
    PLT.TITLE("DETECTED ABNORMALITIES (RED DOTS)")
    PLT.XLABEL("TIME [S]")
    PLT.YLABEL("AMPLITUDE")
    PLT.LEGEND()

    PLT.TIGHT_LAYOUT()
    PLT.SHOW()

# EXAMPLE USAGE
IF __NAME__ == "__MAIN__":
    FS = 250  # SAMPLING FREQUENCY
    T = NP.LINSPACE(0, 10, FS * 10)  # 10-SECOND ECG SIGNAL
    RAW_SIGNAL = NP.SIN(2 * NP.PI * 1.2 * T) + 0.5 *
NP.RANDOM.RANDN(LEN(T))  # SIMULATED ECG WITH NOISE

    PROCESSED_SIGNAL = BANDPASS_FILTER(RAW_SIGNAL, FS)
    ABNORMAL_SIGNAL, HEART_RATE, STD_RR, CONDITION =
DETECT_ABNORMALITIES(PROCESSED_SIGNAL, FS)
```

```
    print(f"Heart Rate: {heart_rate:.2f} bpm")
    print(f"RR Interval Variability: {std_rr:.2f}")
    print(f"Condition: {condition}")

    plot_ecg_signals(raw_signal, processed_signal,
abnormal_signal, fs)
```


Raw ECG Signal
Raw ECG Signal
Amplitude
Time [s]
Processed ECG Signal
Filtered ECG Signal
Amplitude
Time [s]
Detected Abnormalities (Red Dots)
Filtered ECG
Abnormalities
Amplitude
Time [s]