# Abstract

This report details the analysis of Photoplethysmogram (PPG) signals for heart rate monitoring and abnormality detection. It includes preprocessing, peak detection, heart rate calculation, and identification of abnormalities such as bradycardia, tachycardia, and irregular heartbeats. The report includes placeholders for visualizations.

# Scope

1. Data Acquisition: Load PPG signal data from a CSV file.
2. Preprocessing: Apply bandpass filtering to remove noise.
3. Peak Detection: Identify R-peaks using the find\_peaks function.
4. Heart Rate Calculation: Compute RR intervals and heart rate.
5. Abnormality Detection: Detect bradycardia, tachycardia, and irregular heartbeats.
6. Visualization: Plot raw and filtered signals, R-peaks, heart rate trends, and abnormalities.
7. Report Generation: Provide heart rate statistics and detected abnormalities.

# Code Implementation Import Required Libraries

import numpy as np from scipy.signal import butter, filtfilt, find\_peaks import matplotlib.pyplot as plt import pandas as pd

# Define PPGAnalyzer Class

class PPGAnalyzer: def \_\_init\_\_(self, sampling\_rate=100):

self.fs = sampling\_rate self.ppg\_data = None self.filtered\_data = None self.r\_peaks = None self.rr\_intervals = None self.heart\_rates = None

# Load Data

|  |
| --- |
| def load\_data(self, file\_path): try:  self.ppg\_data = pd.read\_csv(file\_path).iloc[:, 0].values return True except Exception as e:  print(f'Error loading data: {e}') return False |

# Bandpass Filter

|  |
| --- |
| def bandpass\_filter(self, lowcut=0.7, highcut=7.5):  nyquist = 0.5 \* self.fs low = lowcut / nyquist |

high = highcut / nyquist b, a = butter(2, [low, high], btype='band') self.filtered\_data = filtfilt(b, a, self.ppg\_data)

# Detect R-Peaks

|  |
| --- |
| def detect\_r\_peaks(self, height=None, distance=None): if height is None:  height = 0.5 \* np.max(self.filtered\_data) if distance is None:  distance = int(0.4 \* self.fs) self.r\_peaks, \_ = find\_peaks(self.filtered\_data, height=height, distance=distance) |

# Analyze Heart Rate

def analyze\_heart\_rate(self):

self.rr\_intervals = np.diff(self.r\_peaks) / self.fs self.heart\_rates = 60 / self.rr\_intervals

# Detect Abnormalities

def detect\_abnormalities(self):

abnormalities = {'bradycardia': [], 'tachycardia': [], 'irregular': []} bradycardia\_idx = np.where(self.heart\_rates < 60)[0] abnormalities['bradycardia'] = self.r\_peaks[bradycardia\_idx] tachycardia\_idx = np.where(self.heart\_rates > 100)[0] abnormalities['tachycardia'] = self.r\_peaks[tachycardia\_idx]

# Main Function

def analyze\_ppg\_file(file\_path, sampling\_rate=100):

analyzer = PPGAnalyzer(sampling\_rate) if analyzer.load\_data(file\_path): analyzer.bandpass\_filter() analyzer.detect\_r\_peaks() analyzer.analyze\_heart\_rate() abnormalities = analyzer.detect\_abnormalities() print('Analysis Complete', abnormalities) return abnormalities

# Results

A screenshot of a graph

AI-generated content may be incorrect.

# Conclusion

The PPG signal analysis system successfully extracts heart rate, detects abnormalities, and provides insights into heart health. The placeholders provided in this document should be replaced with actual plots for a complete analysis. Future improvements could include real-time processing and enhanced filtering techniques.