Howard Li

**Heart Rate Monitor Functional Block Diagram**

Read ECG Data from CSV File

* Identify CSV File with data
  + ~~Ensure that file selected is CSV~~
  + Ensure that file contains data that is readable and appropriate
* Read data from the CSV file into a python array
  + Ensure that read data is a numpy array
  + Ensure that the read data is an array filled with float data points (time, voltage)
* Pass data from the READ block to the PROCESS block

Process ECG Data

* ~~Split the ECG Data~~ 
  + ~~Split the time data into an individual array~~
  + ~~Split the voltage data into an individual array~~
* Ensure that the data from the READ block represents a reasonable ECG data set
  + Voltage values are within range
  + Peaks are detected (heart beats are detected)
  + Time range is not too short
* Detect heart beats
  + Detect peaks in the ECG signal
    - Account for shifting baseline
    - Account for shifted voltage values
    - Account for spastic ECG data
    - Account for missing data
    - Account for breaks in ECG signal
  + **~~Num\_beats: the number of beats is the number of recorded heart beats~~**
    - ~~Output the number of beats as an integer~~
    - ~~Save the number of beats into the dictionary~~
* ~~Record time value of the heart beat peaks as an array of time values~~
  + **~~Duration: the time duration of the ECG strip using start and end time~~**
    - ~~Output the time duration as a float in ms~~
    - ~~Save the float time into the dictionary~~
  + **~~Beats: numpy array of times when a beat occurred~~** 
    - ~~Output the value as an array~~
    - ~~Save the array into the dictionary~~
* **~~Voltage extremes: tuple containing minimum and maximum lead voltages~~**
  + ~~Output the value as a tuple~~
  + ~~Save the tuple into the dictionary~~
* **~~Calculate mean\_hr\_bpm: mean\_hr\_bpm: estimated average heart rate over a user-specified number of minutes (can choose a default interval)~~**
  + ~~Output the value as an integer BPM~~
  + ~~Save the BPM into the dictionary~~

Output metrics as JSON File

* ~~Your metrics dictionary should be output as a~~[~~JSON~~](https://json.org/)~~file should be saved with the same name as the input CSV file that contains the values of all of your object attributes. Note that there is a [json module](https://docs.python.org/3.6/library/json.html) that will make your life easier.~~

Other Considerations & ToDo:

* Git Practices:
  + Good commit messages
  + Don’t delete branches
  + Make a lot of branches for different features
  + Descriptive Readme file
  + Software license in the Readme file
  + Use project management milestones and issues
  + Travis CI
  + Unit tests for each functional element
* Python coding practices:
  + Virtual environment
  + Sphinx Docstrings
  + Achieve the functional specifications with passing unit tests. **~~Make sure that you include a test for writing the output JSON file.~~**
  + Pep8
  + Handle Exceptions
  + Create meaningful logs