Assignment #03: CloudECG

DUE: Thursday, 2017-11-2 at 15:05.

Summary

In this assignment, you will create and deploy a well-tested web service that performs heart rate calculations on ECG data along with bradycardia and tachycardia detection. Your web service will be deployed on a public facing Virtual Machine (VM) as it would be in industry, allowing your service to process requests from any internet connected client (e.g. cloud connected ECG device, iOS/Android applications, Web applications, etc).

Instructions

Create a web service that implements the below RESTful API routes (a.k.a. endpoints). Please be sure to validate inputs and return the correct HTTP error codes with your responses and errors.

- **POST /api/heart_rate/summary** - responsible for returning instantaneous HR and brady/tachy summary. This endpoint takes the following as JSON input:

```
{
  "time": [1, 2, 3, 4, ...],
  "voltage": [20.1, 20, 14, ...],
}
```

  and returns JSON output in the following form:

```
{
  "time": [1, 2, 3, ...],
  "instantaneous_heart_rate": [100, 60, 62, ...],
  "tachycardia_annotations": [true, false, false, ...],
  "bradycardia_annotations": [false, false, false, ...],
}
```

- **POST /api/heart_rate/average** - return an array of average heart rate and brady/tachy annotations computed over a specified time interval. This endpoint takes the following as JSON input:

```
{
  "averaging_period": 20, // In seconds
  "time": [1, 2, 3, 4, ...],
  "voltage": [20.1, 20, 14, ...],
}
```

  and returns JSON output in the following form:

```
{
  "averaging_period": 20,
  "time_interval": [1, 2, 3, ...],
  "average_heart_rate": [100, 60, 62, ...],
  "tachycardia_annotations": [true, false, false, ...],
  "bradycardia_annotations": [false, false, false, ...],
}
```

- **GET /api/requests** - return the total number of requests the service has served since its most recent reboot. You may decide the best way to return this data as JSON.

Submission

- As with previous assignments, tag your git repository with an 'rc' semver tag (e.g. 'v1.0rc1').
- Ensure you have an up-to-date README that documents the endpoints your service provides. Be sure that it also includes a link to the virtual machine host where your final server is running and servicing requests.

Grading Criteria

- Proper version control usage [2]
- Adequate unit test coverage, modularity, continuous integration, and docstrings [2]
- Proper adherence to general python conventions (PEP8, snake_case, spacing around operators) [1]
- Achieves functional specifications [3]
- Is deployed and able to process API requests [1]