Junhan Liu

SPACE PROJECT

Technical Notes and User Manual

Table of Contents

[1 Project Requirements 2](#_Toc529731302)

[1.1 Backend Application 2](#_Toc529731303)

[1.1.1 Hardware Connection and Client Software 2](#_Toc529731304)

[1.1.2 Server Application 2](#_Toc529731305)

[1.1.3 Database and Data Content 2](#_Toc529731306)

[2. System Architecture Overview 2](#_Toc529731307)

[2.1 Project Code and Video Link 2](#_Toc529731308)

[2.2 Application Architecture 3](#_Toc529731309)

[2.3 Software Class Diagram 4](#_Toc529731310)

[3. Backend Application 5](#_Toc529731311)

[3.1 Hardware Connection 5](#_Toc529731312)

[3.1.1 Basic Working Principle 5](#_Toc529731313)

[3.1.2 Hardware Connection Product 5](#_Toc529731314)

[3.2 Client Software 5](#_Toc529731315)

[3.2.1 Environment and Invoke Method 5](#_Toc529731316)

[3.2.2 Software Functions 5](#_Toc529731317)

[3.3 Database - CouchDB 8](#_Toc529731318)

[3.3.1 CouchDB and Benefits 8](#_Toc529731319)

[3.3.2 Database Structure 8](#_Toc529731320)

[3.4 Server Application 10](#_Toc529731321)

[3.4.1 Environment – Amazon EC2 10](#_Toc529731322)

[3.4.2 Server Setup Method – Ansible 11](#_Toc529731323)

[3.4.3 Web Application Server and Corresponding Backend Logic 11](#_Toc529731324)

[3.5 RESTful API Implementation Details 13](#_Toc529731325)

[4. User Guide 15](#_Toc529731326)

[4.1 Disconnected Mode 15](#_Toc529731327)

[4.2 Remote Mode 15](#_Toc529731328)

# 1 Project Requirements

The SPACE training procedure involves the participants peddling on a stationary bicycle (physical exercise) while doing Sudoku (cognitive exercise).

In accordance, the project is divided into two components:

* Backend application, including the client software with hardware connection, the server application, and the supporting database.
* Frontend application, which is displayed at the client browser.

Below sections describes the requirements for each individual component.

## 1.1 Backend Application

### 1.1.1 Hardware Connection and Client Software

The client software needs to be able to:

* Maintain a stable connection to the provided heart rate monitor (Model: ANT+ LifeLine Heart Rate Transmitter).
* Display the real-time monitored heart rate on the interface, and keep updating as new data comes in.
* Send the heart rate data to central server on a regular time interval.

### 1.1.2 Server Application

A central server is required to receive and process the interfaced heart rate data, push the data into persistent storage, and retrieve on demands. It needs to support both the client software, where the data is gathered from the hardware, and the frontend application, where the following interactions happen:

* User related interactions, including registration, login, and procedure session start and end;
* Heart rate display and update;
* CSV file generate and download.

### 1.1.3 Database and Data Content

A persistent database is needed to store related information, including the experiment subject profiles, and detailed records on the corresponding experiment sessions.

The following information need to be saved:

* User ID
* Heart rate data during the session
* Flag when the monitored heart rate is within, above, below the predefined range

# 2. System Architecture Overview

## 2.1 Project Code and Video Link

Table 1 Project Code and Function Demonstration

|  |  |  |
| --- | --- | --- |
| Component | Type | Link |
| Backend Application | Code | https://github.com/Zayhan/SPACE-Project |
| Function Demonstration | https://www.youtube.com/watch?v=mY4oqZ\_TYgQ |
| Web Application | Code |  |
| Function Demonstration |  |

## 2.2 Application Architecture

The system of the project implements the typical three-tier architecture.

../FinalProject/ReportAssets/Chart2.pdf

Figure 1 System Architecture

In the centre sits the backend application, which contains the following sections:

* A web server which listens to the requests from both the hardware client application and the frontend application and provides response accordingly.
* A multi-threaded backend logic unit which handles the internal transactions.
* A database which persistently stores the user profile and records the corresponding experiment data. The database server responds to the requests from the backend application.

For each hardware device, a client software is launched. It validates that the user profile exists in database, gathers the real-time heart rate, and sends it to central server.

There is also a frontend application, using HTML, CSS and Javascript, where the participants can register, sign in, check the heart rate, and play Sudoku.

The communications between different components in the system are completed using RESTful API.

Below sections will explain each component in the system architecture graph above, including its functions and business logic, its invocation method, and its implementation methods.

## 2.3 Software Class Diagram

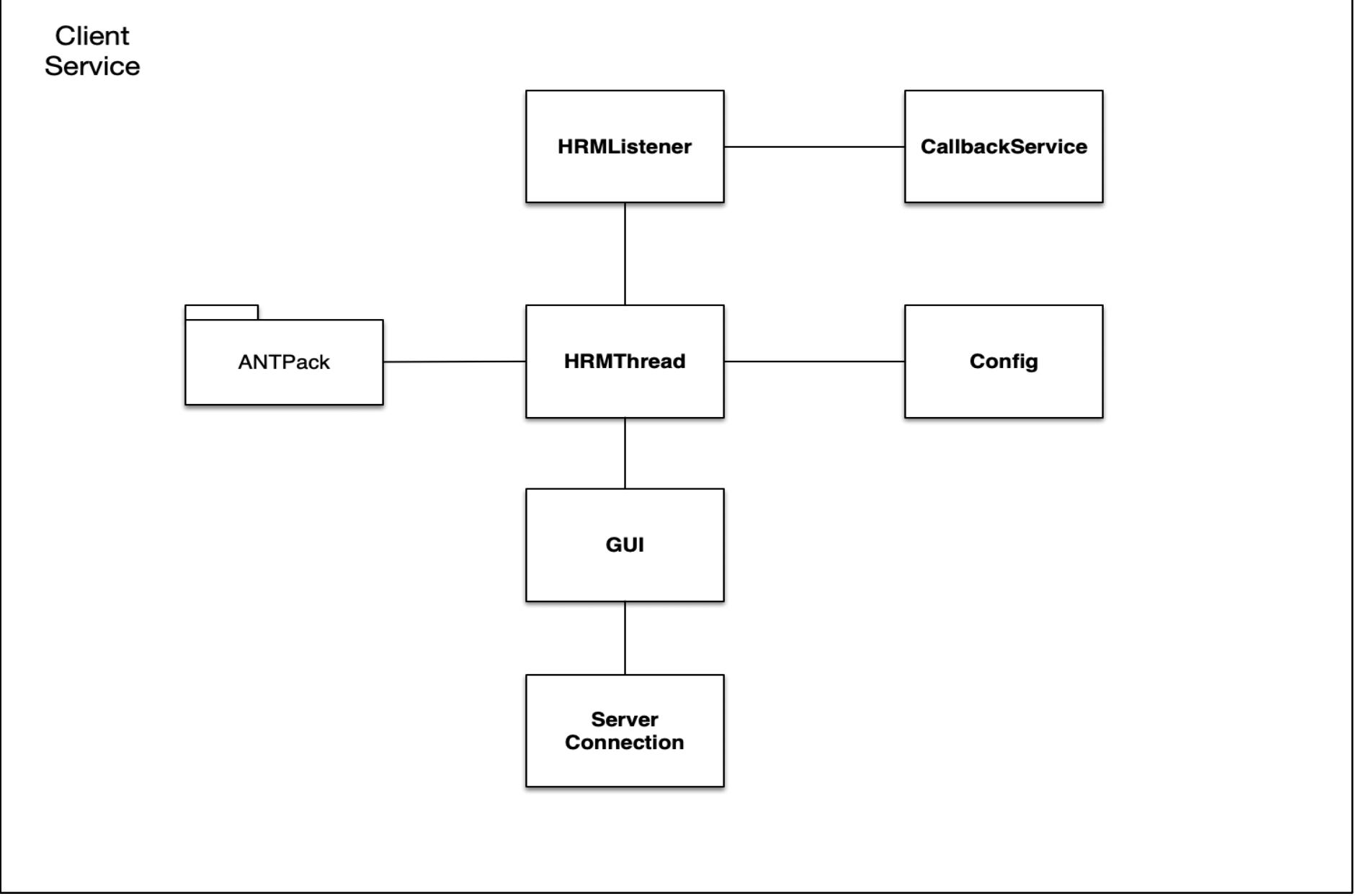


Figure 2 Client Software Class Diagram

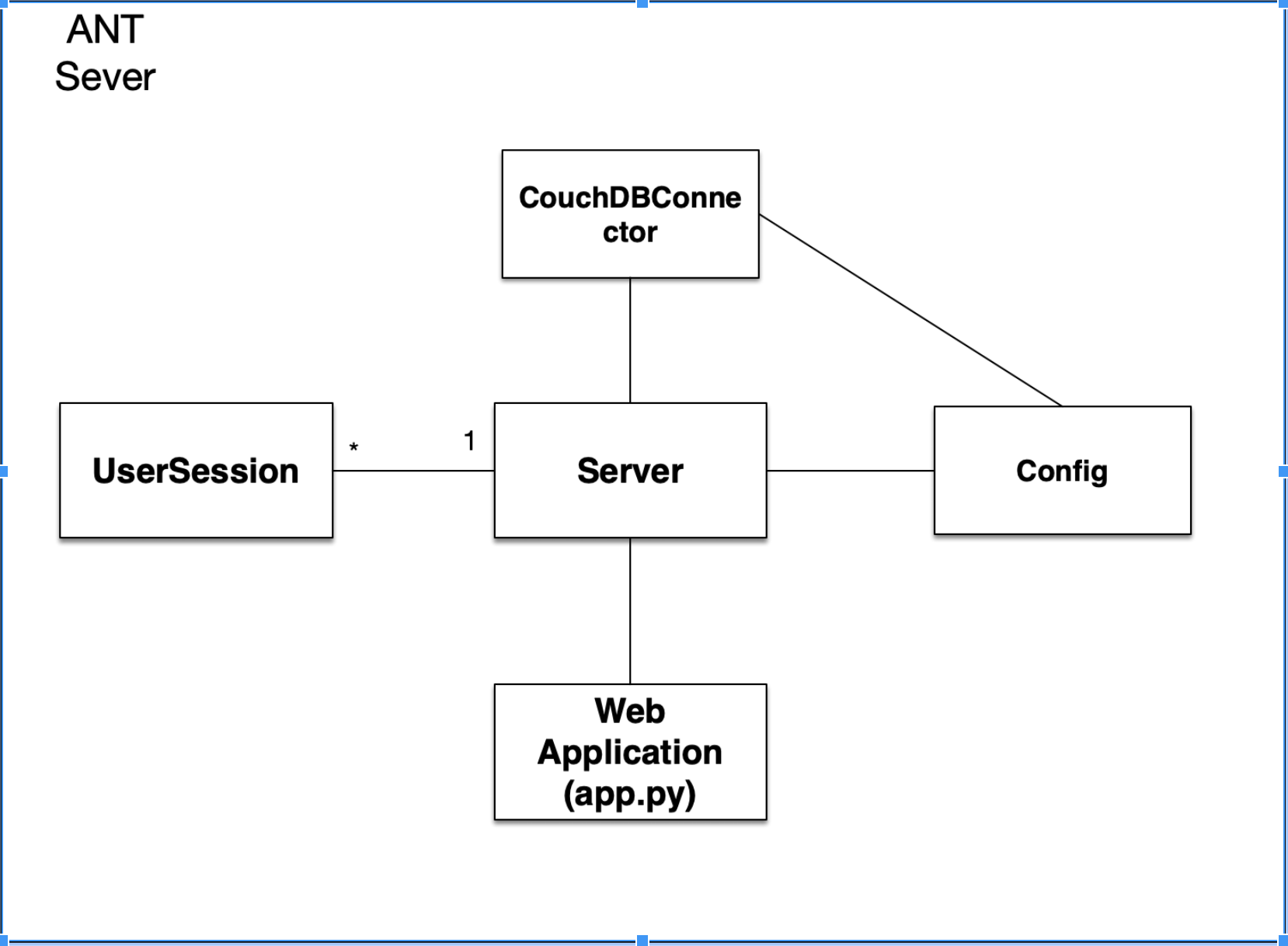


Figure 3 Server Class Diagram

# 3. Backend Application

Backend application, including its main components, its supported APIs and its deployment details will be covered in this chapter.

It is implemented using Python 2.7.

## 3.1 Hardware Connection

### 3.1.1 Basic Working Principle

The provided heart rate monitor is a LifeLine strap, and an ANT+ USB Stick. The transmitter on the strap passes the raw data gathered by the electrode pad to the ANT+ USB Stick via Bluetooth.

Upon receiving the byte stream from the monitor, the programme interprets the raw data, and extract the required heart rate.

### 3.1.2 Hardware Connection Product

After revising the original ANT+ library, the software can maintain a stable connection with the hardware and retrieve the accurate results. However, it has the following issues:

* Each USB stick can only connect to one heart rate monitor.
* Due to the limited platform support, the software can only run under Linux and Mac OSX.

## 3.2 Client Software

### 3.2.1 Environment and Invoke Method

Table 2 General Information on Implementation

|  |  |
| --- | --- |
| **Required Package** | ANTPack: the programme that handles the hardware connection.  ANTClient: the programme that provide the user interface, data interpretation, and the communication with the server. |
| **Required Environment** | Mac OSX operating system  Python version 2.7 |
| **Software Entrance** | GUI.py |
| **Invoke Method** | * Launch terminal * Enter to the ANTClient folder by typing “cd FILE\_PATH” * Type “python GUI.py” |

### 3.2.2 Software Functions

The main interface is divided into two sections.

On the left is the user control panel, and on the right, is the heart rate display panel.

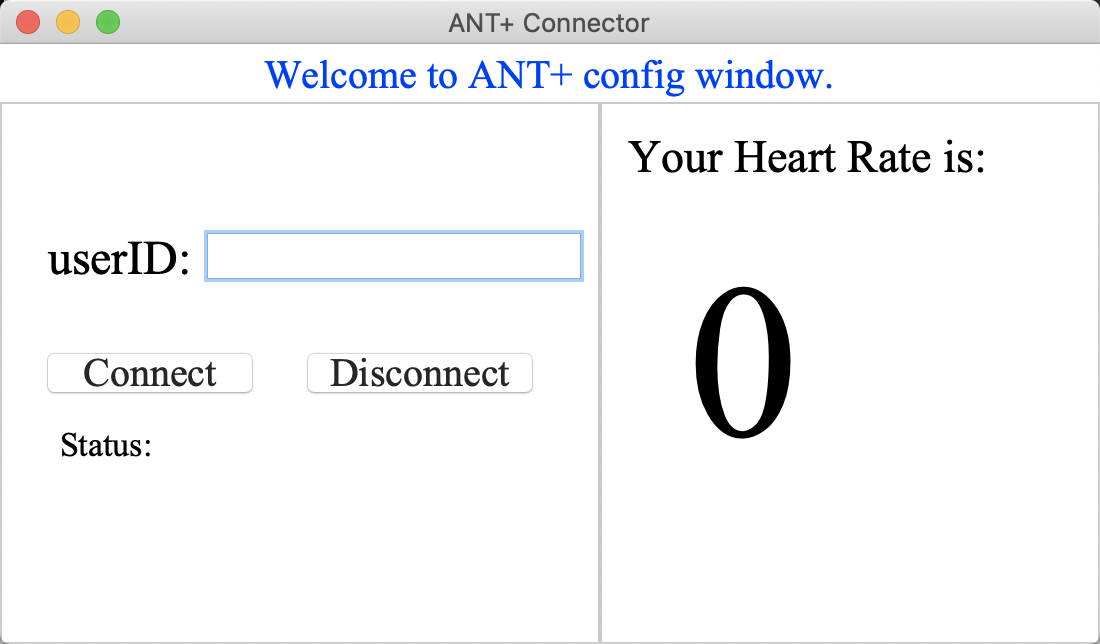


Figure 4 Client Software Application Interface

#### Heart Rate Display Panel

Once the software starts to receive heart rate data from the transmitter, the heart rate panel will keep updating with the latest heart rate. This panel does not support any user interactions.

#### Control Panel

The monitoring software provides two modes, which can be switched by interacting with the control panel:

* Local mode: only provides heart rate display, and not connected to server. Under local mode, the gathered data is only used for display purpose and will be discarded once the software is closed.
* Remote mode: besides the heart rate display, the software also pushes the heart rate data to the remote server, where the data gets processed and saved. To use the remote mode, the user must first create a profile via the frontend application, and verify the user ID with the remote server via the client software.

flowchart.pdf

Figure 5 Client Software User Flowchart

In the control panel, the experiment subjects can input the user IDs, and click the “Connect” button to verify with the server.

Once the user ID passes verification, the server will return the user name, and the software will start actively monitoring the heart rate of the subjects, and pass data to the server every 10 seconds.

The “Disconnect” button, once clicked, will trigger a “deactivate” request to the server, which will instruct the server to process the corresponding data, and delete the active session for that particular user.

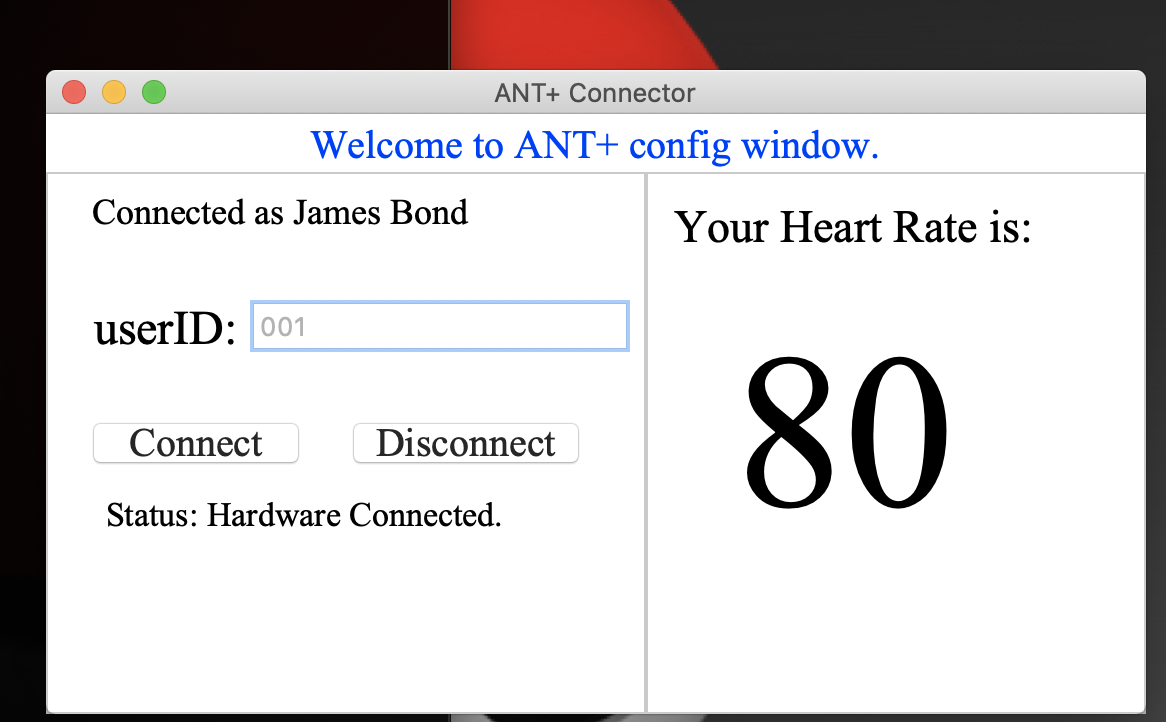


Figure 6 Client Software Application Interface - Connected Status

If the user fails the user ID validation, or the server is not responding, the software will invoke a prompt, asking if the user would like to switch to local mode. If “OK” is clicked, the software will initiate local mode, and display the heart rate data to the user.

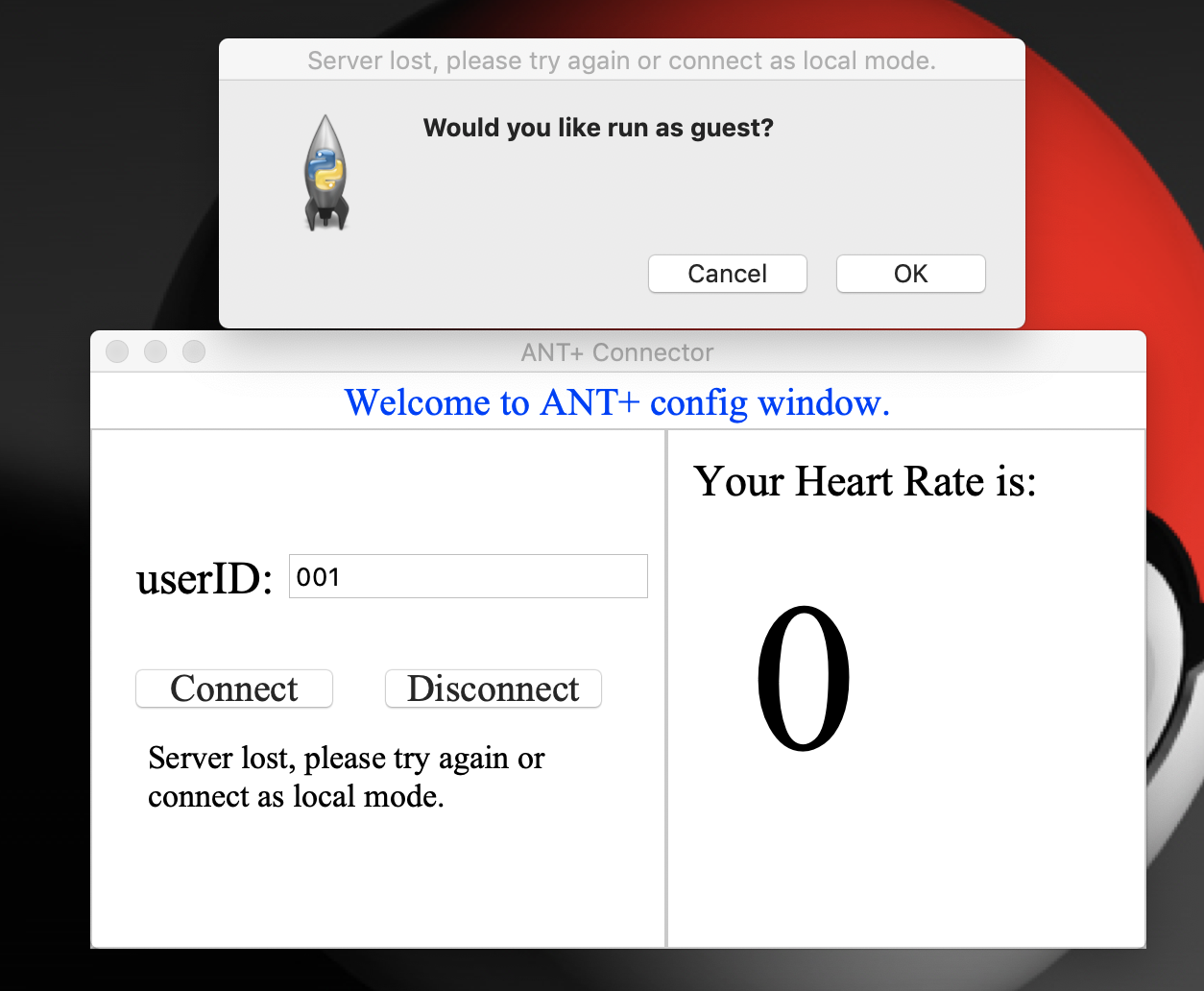


Figure 7 Client Software Application Interface - Mode Selection

## 3.3 Database - CouchDB

The application implements CouchDB as database for persistent storage.

### 3.3.1 CouchDB and Benefits

Apache CouchDB is a Non-SQL database. It is document oriented, and data is stored in the format of JSON files.

Besides the general benefits of CouchDB, such as high and stable performance, SPACE project chose CouchDB out of the following advantages over relational database and other Non-SQL databases.

* Natively Speaking HTTP/JSON

The communication between different components within the application is implemented using RESTful API, which is based on HTTP/JSON. The fact that data manipulation in CouchDB also utilises HTTP/JSON makes it a natural fit for the database solution.

* Easy to Scale

Although currently using single node database, CouchDB can be easily extend to a cluster database, and provide sharding and replica to achieve higher performance and reliability.

### 3.3.2 Database Structure

Each user forms a document in CouchDB. The document key is using a server generated number, which automatically increments as the number of user increases.

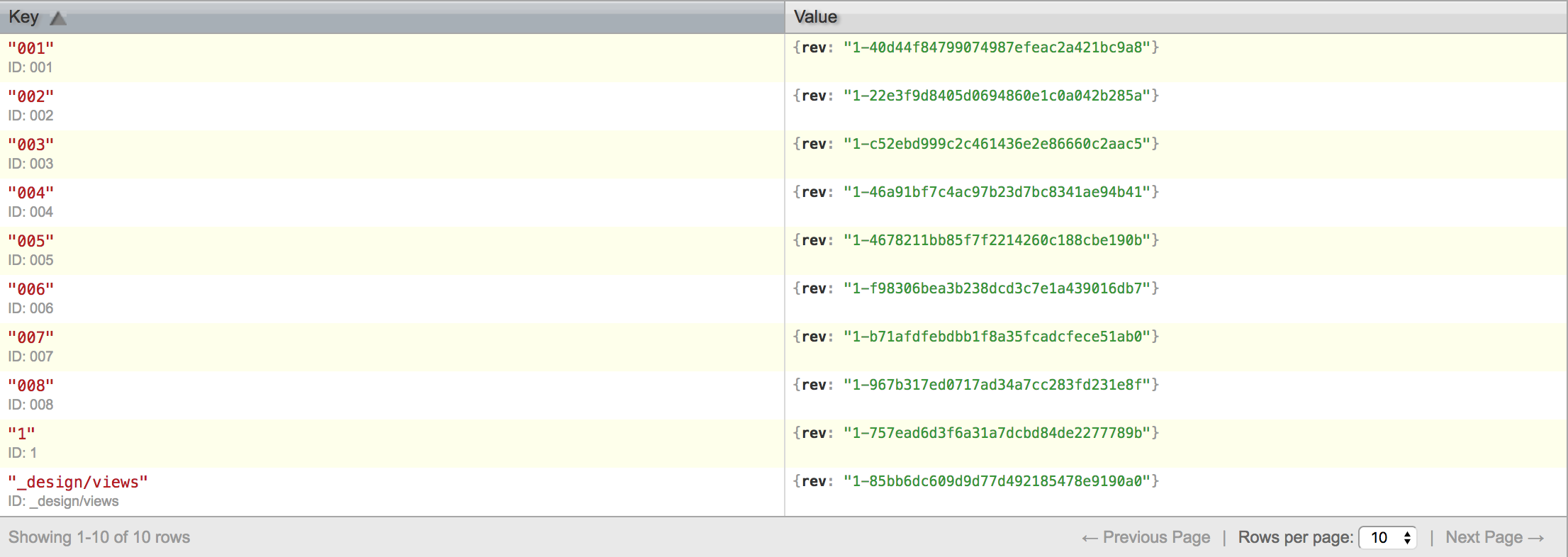


Figure 8 CouchDB Document

The table below lists the details of all the fields inside one user document.

Table 3 CouchDB Data Fields

|  |  |  |
| --- | --- | --- |
| **Name** | **Example** | **Comment** |
| \_id | “001” | Document ID, incremental number assigned by the server. |
| \_rev | “1-40d44f847990749  87efeac2a421bc9a8” | Generated and maintained by CouchDB. |
| Cool\_Zone\* | 0:54, 1:66 | The upper bound and lower bound of cool zone heart rate (45% to 55% of max heart rate) |
| Exercise\_zone\*\* | 0:67, 1:78 | The upper bound and lower bound of exercise zone heart rate (65% to 75% of the max heart rate) |
| HR\_Max | “120” | The maximum heart rate entered by user during registration. |
| PName | “James Bond” | The name of the experiment subject. |
| session | Dictionary, with one item for each session heart rates | Record details on different sessions. |
| \*Cool Zone: the expected heart rate range when the user is warming up or cooling down. | | |
| \*\*Exercise Zone: the heart rate range that the user is exercising. | | |

To improve the query speed, a CouchDB view is created to index all the document ID and the corresponding PName field.

## 3.4 Server Application

This section will explain in detail the server application, including the technical specification and its business logic. The server application is devoted to the following functions:

* Create and manipulate database;
* Act as web server, and respond to RESTful API calls, both from the frontend application and from the client software;
* Upon receiving API calls, process the requests and make responses accordingly.

### 3.4.1 Environment – Amazon EC2

The server application is deployed to Amazon Elastic Compute Cloud (EC2). Amazon EC2 is a web service that provides secure, resizable computation capacity in the cloud [5].

A t2.micro instance is created to host the web application, with Ubuntu Xenial 16.04 as the operating system. It has 1 CPU, 1 gig byte main memory, and 8 gig bytes persistent storage. Due to the limitations of AWS free trial, the server is placed in US east subnet, which means that there exists considerable amount of network delay. Still, hosting server on Amazon EC2 has its competitive advantages:

* Fast-to-scale. The main advantage of using the cloud is the scalability. If the number of the experiments expands drastically, adding new instances to scale up is incredibly easy.
* Ready-to-use infrastructure, including the operating system, etc. so that more focus is put on the application deployment.
* Easy-to-manage security setting. The security configuration is maintained in the AWS console, which includes a detailed list of rules for both inbound and outbound traffic. It is easy to modify the security configuration, limiting traffic according to port, source IP, etc.

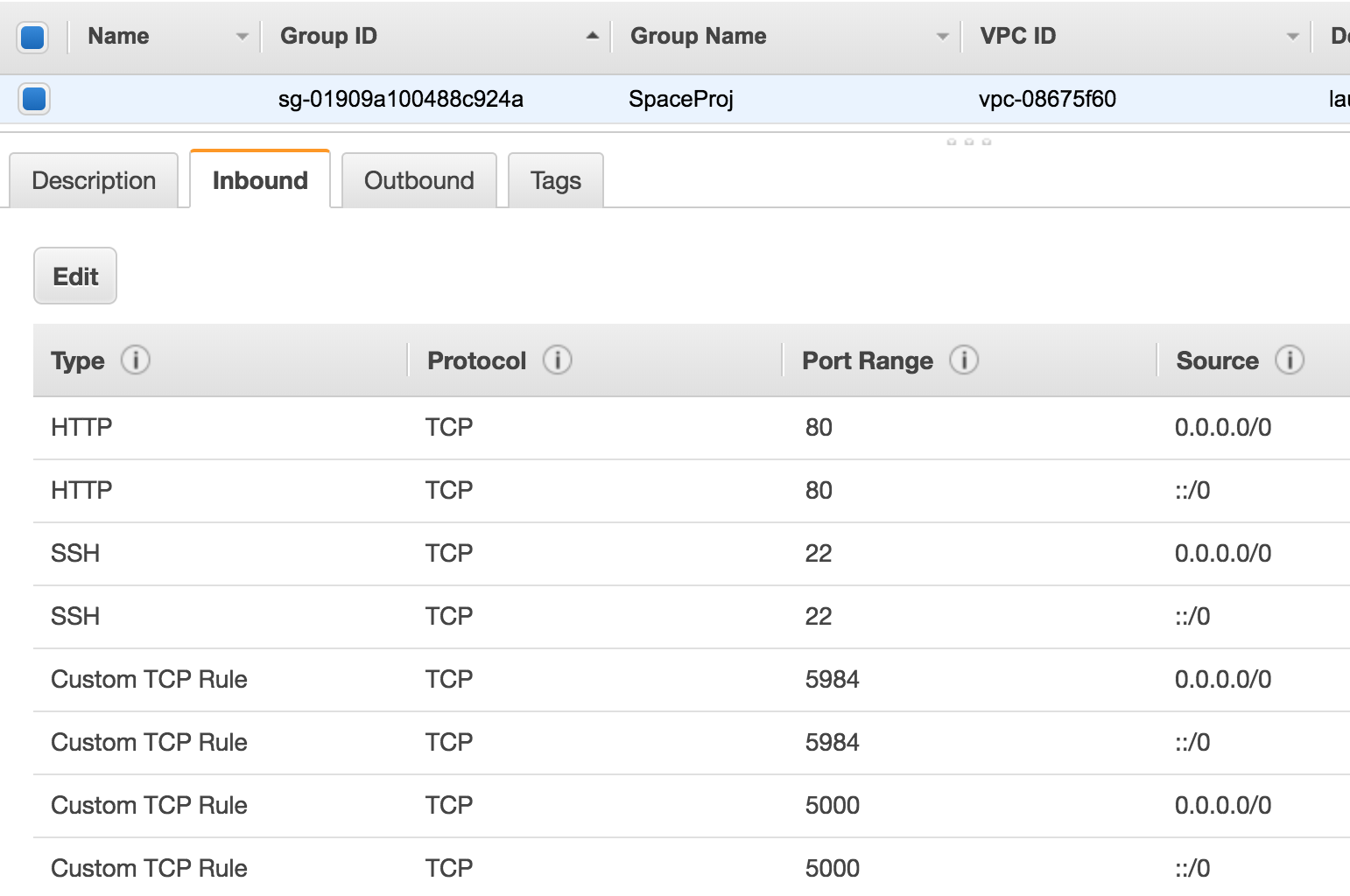


Figure 9 Amazon Security Setup

* Detailed monitoring. Amazon console provides a detailed record and visualisation for the instances, including network traffic, CPU performance and so on.

### 3.4.2 Server Setup Method – Ansible

To increase the scalability of the project, the application uses Ansible script to setup the server environment. Ansible is the automation tool that allows configuring multiple servers with one command line.

### 3.4.3 Web Application Server and Corresponding Backend Logic

The web application server is written under Flask framework, and supported by Gunicorn as WSGI server. It caters requests from two sources – namely the client software, and the frontend pages on browser.

#### Client Software Server

For client software, the application server needs to handle three types of requests.

* User Verification

In order to use the remote mode in client software, the user ID must first be verified with the server. Therefore, upon receiving the verification request with the user ID, the server application will connect to CouchDB view, and search for the corresponding user ID. If the user ID does exist in database, the user name saved against the profile will be returned to the client application, and the server application will register the current user ID to the server-maintained user session, preparing to receive the heart rate data. If the passed user ID cannot be found, an error code will be returned to the software application, and the client software will prompt the local mode message.

* Heart Rate Data Push

When the software successfully connects to the server, and is running under remote mode, upon receiving the heart rate data from the transmitter, the software will fire a heart rate update request to the server, together with the corresponding user ID. When the server receives the heart rate, it will invoke the heart rate record function to store the data in the user session. However, with the consideration to business requirements, the data will be ignored by the backend logic until the web page notify the server to start recording.

As the experiment session is strictly limited to one hour, once the clock is reached, the server will no longer receive new heart rate data, and will notify the client software with the corresponding error message.

* Hardware Disconnect

When the “Disconnect” button is clicked on the client software, the deactivation request will be sent to the server. On this request, the server will delete the corresponding user from the active user session and remove all related heart rate data.

#### Frontend Web Server

Based on the requirements, the frontend pages serve two purposes. First, the training interface for the experiment participants, which displays the heart rate, and controls the session start time. Second, the monitoring interface for researchers, where the researchers can see a list of experiment participants and their details and download the recorded data in CSV format.

The functions of the frontend web server are designed in accordance with the requirements above.

* User Registration

When a new experiment subject joins the trial, he/she first needs to register a user profile with the server in order to obtain a user ID. When the server receives the registration request from the frontend page, it will generate a user ID, create a new document for that user in CouchDB, and return the assigned user ID in the response.

* User Login

When a returning experiment subject starts a new session, he/she must input the user ID in the web page. The web page will send a verification request to the sever. Upon checking if the user ID exists in database, the server will return the corresponding response to the frontend page.

* Activate User Session

As each training session is strictly limited to 1 hour, the experiment participant must be able to control the session starting time. Therefore, when the user clicks the “Start Session” button on the frontend page, an activation request will be triggered to the server. And the server will start recording the heart rate passed by the client software and starts a clock for that user so that the training session can be terminated in one hour.

* Get Heart Rate

During the SPACE training session, the frontend page is required to display the latest heart rate obtained from the hardware. Therefore, when the server receives the request, it will search for the latest heart rate gathered from the client software and return to the web page.

* Get All Users and Details

Reserved for the monitoring page, the server provides responses with either a full list of user IDs in the database, or all the details of a particular user, depending on the requests.

* Generate and Download CSV

When the one-hour limit is reached, the server will automatically stop the clock, and generate a CSV file, which contains the following fields:

Table 4 CSV Fields

|  |  |
| --- | --- |
| **Field Name** | **Comment** |
| PName | Name of the experiment subject |
| HR\_MAX | Maximum heart rate of the experiment subject |
| Cool\_Zone | Heart rate range when the subject is doing warm-up or cool-down activities |
| Exercise\_zone | Heart rate range when the subject is doing exercises |
| Time\_in\_warmup\_zone | Total time when the heart rate is within the Cool\_Zone range before starting exercise |
| Time\_in\_exercise\_zone | Total time when the heart rate is within Exercise\_zone |
| Time\_in\_cool\_zone | Total time when the heart rate is within the Cool\_Zone range after finishing exercise |
| TimeLine | Date of the session |
| Session Data | Heart\_Rate  Indication Flag: The heart rate is above, below or within the corresponding heart rate range.  Timestamp  Second |

Upon the click of the download button, the server will return the corresponding file for a specific session of one user.

## 3.5 RESTful API Implementation Details

The application utilises RESTful API for communications between different components and functions.

Table 5 API Details

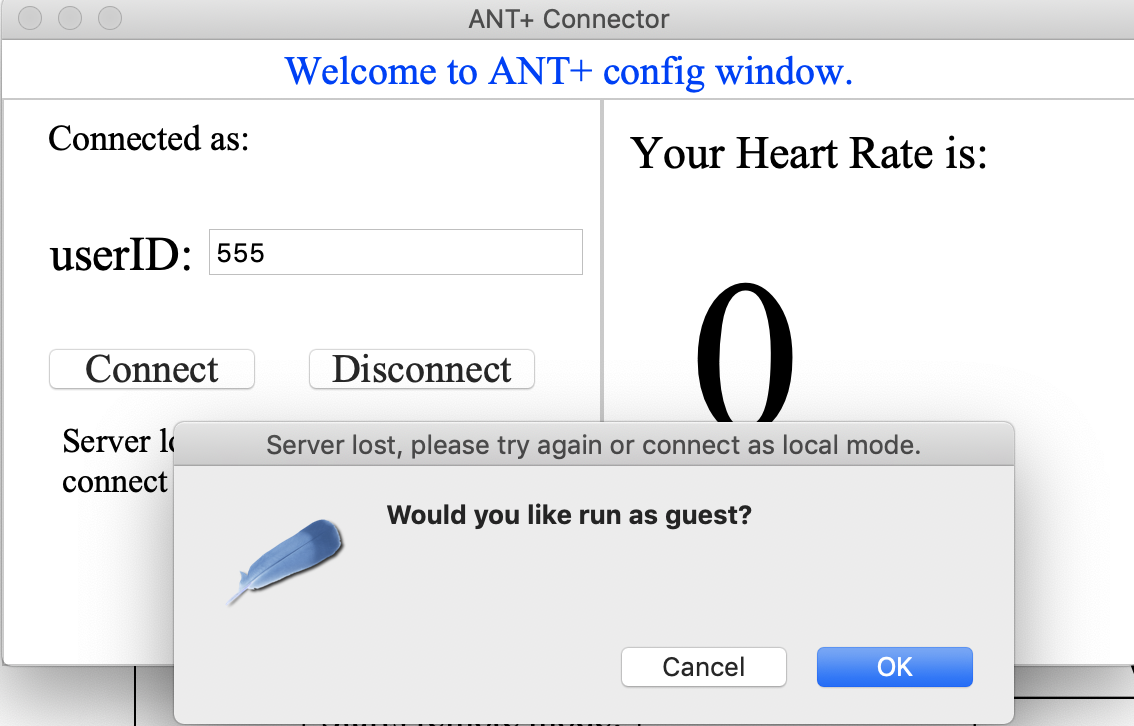
|  |  |  |  |
| --- | --- | --- | --- |
| **End Point URL: http://18.219.29.53:5000/** | | | |
| **Path** | **Methods** | **Usage** | **Request Format (JSON)** |
| **/** | GET | Welcome message from the server | N/A |
| /users | GET | Get a list of all user IDs | N/A |
|  | POST | Register user in the server | {"PID": "user\_id","PName": "user\_name","HR\_Max": "max\_heart\_rate","Cool\_Zone": [45, 55],"Exercise\_Zone": [65, 75],"Session": {}} |
| /users/<string:user\_id>/login | POST | Check whether user in database and login | N/A |
| /users/<string:user\_id>/  detail | GET | Get details of one user | N/A |
| /active\_user | GET | Get a list of active user IDs | N/A |
| /active\_session/<String: user\_id> | POST | Active user session to start recording | N/A |
| /active\_user/<string:user\_id>/ deactive | POST | Delete the user from the active user session | N/A |
| /register\_session | POST | Push the user into the active user session | {"id": "user\_id"} |
| /active\_user/<string:user\_id>/ heart\_rate | GET | Get the latest heart rate for one user | N/A |
|  | POST | Send the retrieved heart rate for one user | {"id": "user\_id","heart\_rate": "heat\_rate"} |
| /<string:userid>/sessions/  <string:fileName> | GET | Download the CSV file | N/A |

# 4. User Guide

The basic functionalities follow the instructions based on Disconnected mode and Remote mode. The Client-side software is easily runnable by just double-clicking the software.

## 4.1 Disconnected Mode

When the desktop has no internet connection or cannot connect to the server in one minute, the programme will run as the disconnected mode. Under this mode, the patient`s heart rate will not be saved to the remote server. And this software will run as the heart-rate reader.



If user want to close the programme, the user can either click the disconnect button, or click the close button on the left top corner.

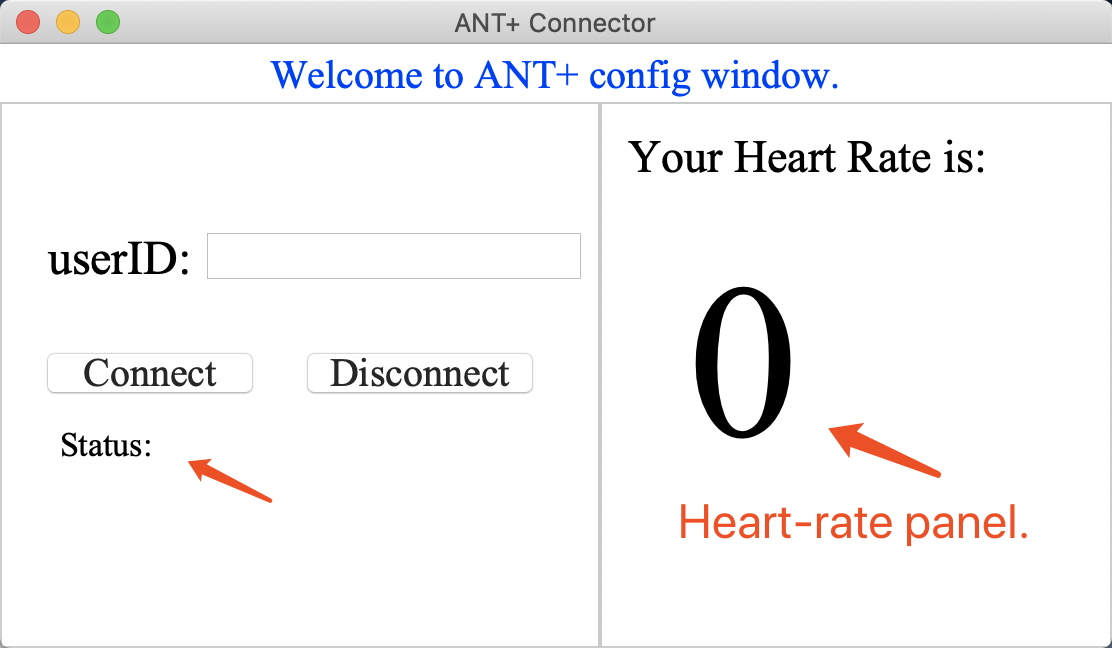
## 4.2 Remote Mode

To use the remote mode, the user needs to be registered into the database, so that the programme can retrieve the user name and information for the purpose of scientific use and analysis.

The user should follow the steps as shown below:

* The user should register to the database using webpage.
* When the user is registered into the database, user can type their user ID in the text Field and click connect. If user ID is correct and the internet connection is stable, then the programme will start hardware connection and try to connect with heart-rate transmitter. The connection status will show on status. Once the hardware is established, user heart rate will be shown on the panel.

In this step, heart rate will not be captured by the server. In this step, the client software will continuously post user’s heart rate but will be denied by the server, due to the inactive of the user session.



* In order to allow the server record patient`s heart-rate and analysis, the user should manually activate their session through the webpage. Once the session is activated, the server will record user heart rate posted by the client-end software.
* Once the user finished their one-hour sessions, the user session will be automatically stored into the Cloud file system （Linux based） and the database. The user will receive the message saying the session time reached.
* If the user wants to download the CSV file for analysis the user can type the HTTP request on the browser like typing a website address (detailed format see the table below).

|  |  |
| --- | --- |
| **Field Name** | **Example** |
| Address | http://<IPaddress>:5000/<UserID>/sessions/<FileName> |
| File Name | S6\_2018-10-17\_19:14:01 |
| Example: https:// 192.168.0.1:5000/001/sessions/ S6\_2018-10-17\_19:14:01 | |