Thread Profiler Instructions

How to use:

Include library:

```
#include "thread_profiler.h";
```

Take object from ThreadProfiler:

```
ThreadProfiler profiler;
```

Creating thread:

```
profiler.profilerThreadStart(workerFunction,&profiler,threadName(optional));
```

Creating mutex:

```
pthread_mutex_t mutex = profiler.profilerCreateMutex();
```

Acquire mutex:

```
Profiler.profileMutexAcquire(threadId, mutex);
```

Release mutex:

```
Profiler.profileMutexRelease(threadId, mutex);
```

End thread “should be called at worker function”:

```
profiler.profilerThreadEnd(threadId);
```

To get thread id:

```
pthread_t threadId = profiler.profilerGetThreadId();
```

To get thread name:

```
String threadName= profiler.profilerGetThreadName();
```

To get all threads id:

```
const auto &infoMap = profiler.profilerGetAllThread();
```

To join threads in Main thread:

```
profiler.profilerThreadJoin(thread_id);
```

Compile ,Run and Test Application: All steps in thread_profiler folder

Prerequisites:

Insall boost library

Install Reac.js

For C++ code:

Compile: g++ -o [output file name] [program file path] src/mutex_manager.cpp src/thread_manager.cpp src/thread_profiler.cpp -Iincludes -lboost_system -lboost_thread -lpthread

Run: ./ [output file name]

For GUI:

Run: npm start

Test:

1-write program by using “thread_profiler.h” functions

2- compile and run your code.

3- run GUI will get dashboard (can also refresh if data not appear)

Class Diagram and Architecture: