Test File ------Test plans and implementation

Project name: MinersCoffeeTest

Main script: tst\_generaltest.cpp

1. The purpose of the test

Testing is the process of executing a program to discover errors in the program. The purpose of testing is to find as many errors in the software as possible before it is put into production operation. Successful test can find the errors in the system and make the system run correctly.

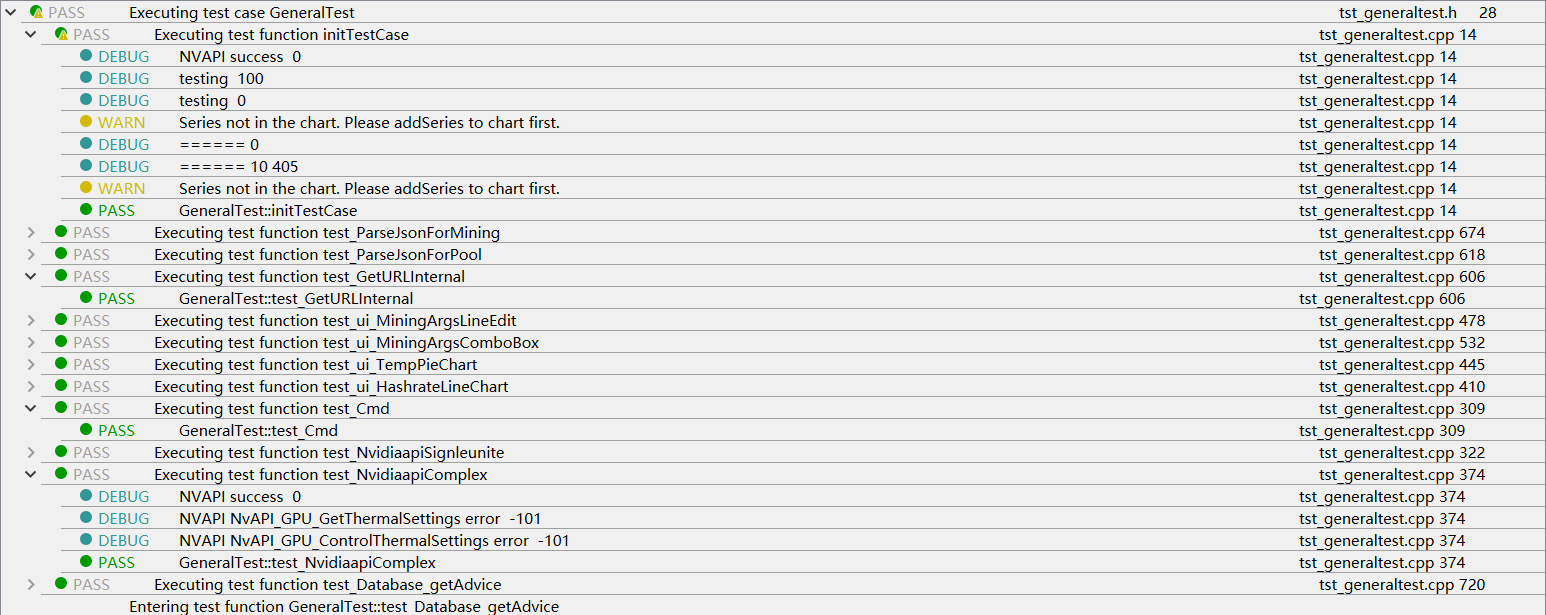
1. Test dependent conditions

The test project and the program project are two opposite Qt project. Rely on QTestLib, the software project needs to rely on this platform and the environment needs to be integrated.

3. Testing Framework:

QTestLib is a unit testing framework provided by Qt for programs or libraries written on Qt. QTestLib provides the basic functionality of the unit testing framework and provides extended functionality for GUI testing.

Use AutoTest plug-in to achieve visual effects.



1. Test initialization:

private:

MainWindow\* w;

Helper helper;

public:

**GeneralTest**();

~***GeneralTest***();

private:

void **initTestCase**();

Parameters: None.

Return: None.

Task: Initialize private variables MainWindow w.

void **cleanupTestCase**();

Parameters: None.

Return: None.

Task: Stop Mining-core and delete Main-window w.

1. Test preparation:

private:

void **GetTest data**(QList<QString>& input, QList<QString>& result, const QString& in\_filename, const QString& res\_filename);

Parameters: Pointers to input and results’ data file data and filename.

Return: None.

Task: Input the parameter data and expected results in pairs to compare for tests

void **ShowDataError**(const QString& filename1, const QString& filename2);

Parameters: Pointers to input data filename and result data filename.

Return: None.

Task: Warn errors’ location when tests goes wrong.

1. Test hierarchy
2. Unit Tests:
   1. Command Line Module

void **test\_Cmd**():

Task: Check if we get right information of the disk memory size from Wincmd.

Test data: Directly passed

Test data size: 1

* 1. GPU Monitoring & Overclocking Module

void **test\_NvidiaapiSetTempLimit**();

Task: Check if program reset the gpu temprature limit correctly

Test data: Passed by function **test\_NvidiaapiSetTempLimit\_data**()

Test data size: 5

void **test\_NvidiaapiSetGPUoffset**();

Task: Check if program reset the gpu offset correctly

Test data: Passed by function **test\_NvidiaapiSetGPUoffset\_data**()

Test data size: 5

void **test\_NvidiaapiSetMemoffset**();

Task: Check if program reset the memory offset correctly

Test data: Passed by function **test\_NvidiaapiSetMemoffset\_data**()

Test data size: 5

void **test\_NvidiaapiGetTemp**();

Task: Check if program successfully access to the nvidia api and get gpuinfo.

Test data: Directly passed

Test data size: 1

void **test\_NvidiaapiComponent**();

Task: Check if program successfully access to the nvidia api and get the temperature.

Test data: Passed by function **test\_NvidiaapiComponent\_data**()

Test data size: 5

void **test\_NvidiaapiControlTest**():

Task: Check if program successfully control the gpuinfo

Test data: Directly passed

Test data size: 3

* 1. Network Module

void **test\_GetURLInternal**();

Task: Check if program successfully connect to sparkpool

Test data: Directly passed

Test data size: 1

* 1. Json Parsing Module

void **test\_ParseJsonForMining**();

Task: Check if program successfully access to the pool api and get mining information.

Test data: Passed by function **test\_ParseJsonForMining\_data**()

Test data size: 3

QString input\_filename = "test\_ParseJsonForMining\_input.txt";

QString result\_filename = "test\_ParseJsonForMining\_result.txt";

void **test\_ParseJsonForPool**();

Task: Check if program successfully access to the pool api and get pool information.

Test data: Passed by function **test\_ParseJsonForPool\_data**()

Test data size: 3

QString input\_filename = "test\_ParseJsonForPool\_input.txt";

QString result\_filename = "test\_ParseJsonForPool\_result.txt";

* 1. Database System

void **test\_Database\_getAdvice**();

Task: Check if program successfully connect to database and get the advice data

Test data: Directly passed

Test data size: 11

1. Component Tests:

void **test\_TempPieChart**();

Test Task: Check if program successfully get the temperature of the GPU and pass it to the right UI interface to display

Test data: Directly passed

Test data size: 1

void **test\_HashrateLineChart**();

Test Task:Check if program successfully get the Hashrate of the mining process and pass it to the right UI component to display

Test data: Directly passed

Test data size: 1

void **test\_MiningCore**();

Test Task:Check whether the user enters the correct wallet and username, and try to connect to the mining pool with the obtained data.

Test data: Directly passed

Test data size: 1

void **test\_MiningArgs**();

Test Task: Check whether the connection can be successfully connected and the mining data is obtained.

Test data: Directly passed

Test data size: 1

void **test\_ParsePoolInfo**();

Test Task: Check whether we successfully process the data and pass in the corresponding function module.

Test data: Directly passed

Test data size: 1

1. System Tests:

void **test\_FullSystem**()

Task: View the order between threads for possible blocking situations, view the circulation of the regularization process and check the integrity of the system

1. GUI Test:

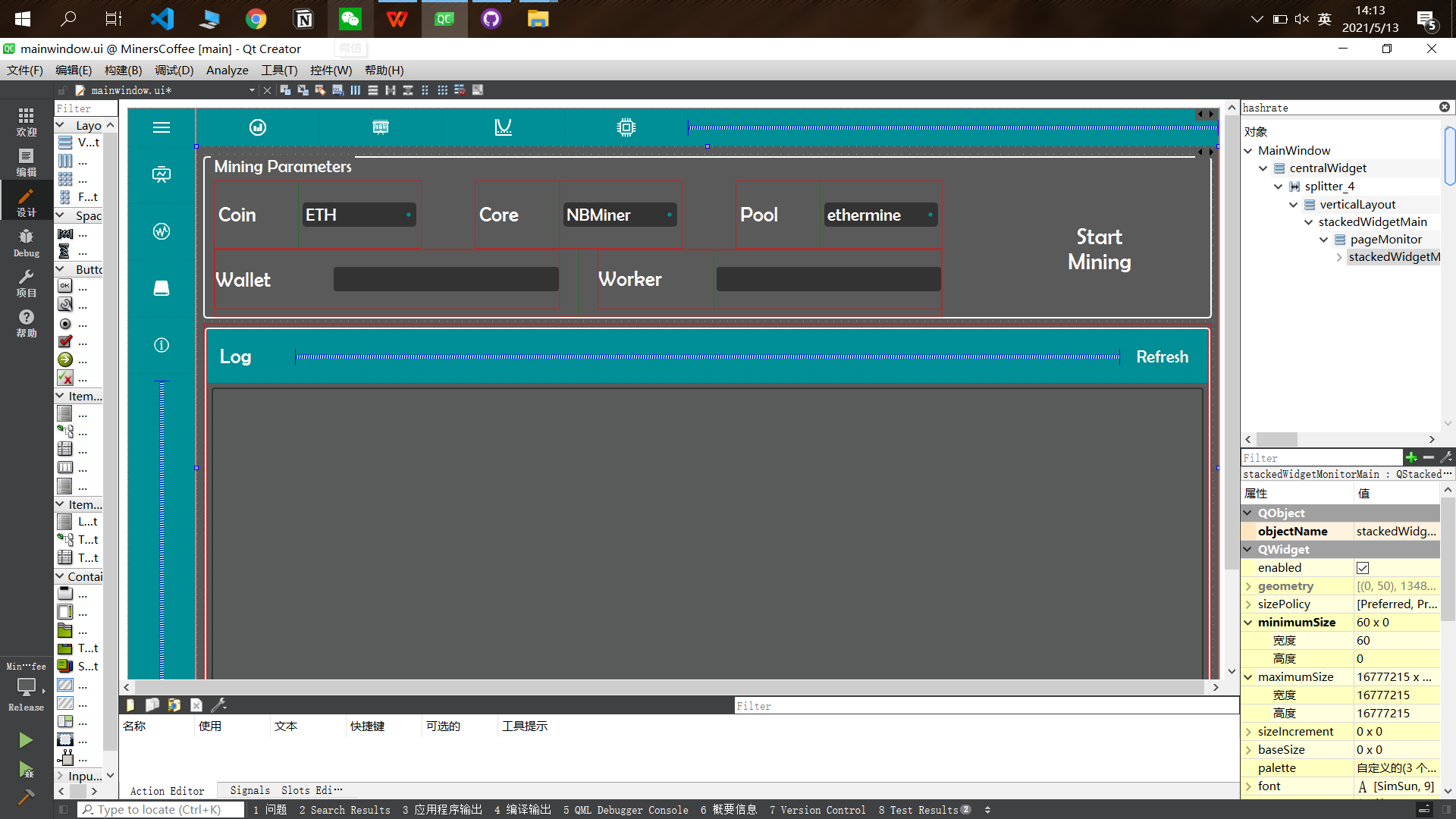
For the test of GUI graphic operation, set the data as the event list for simulation test. Use, for example, built-in functions, passed through internal events, to simulate the events of the local window system.

(1)

void **test\_ui\_MiningArgsLineEdit**();

void **test\_ui\_MiningArgsLineEdit\_data**();

Test component:



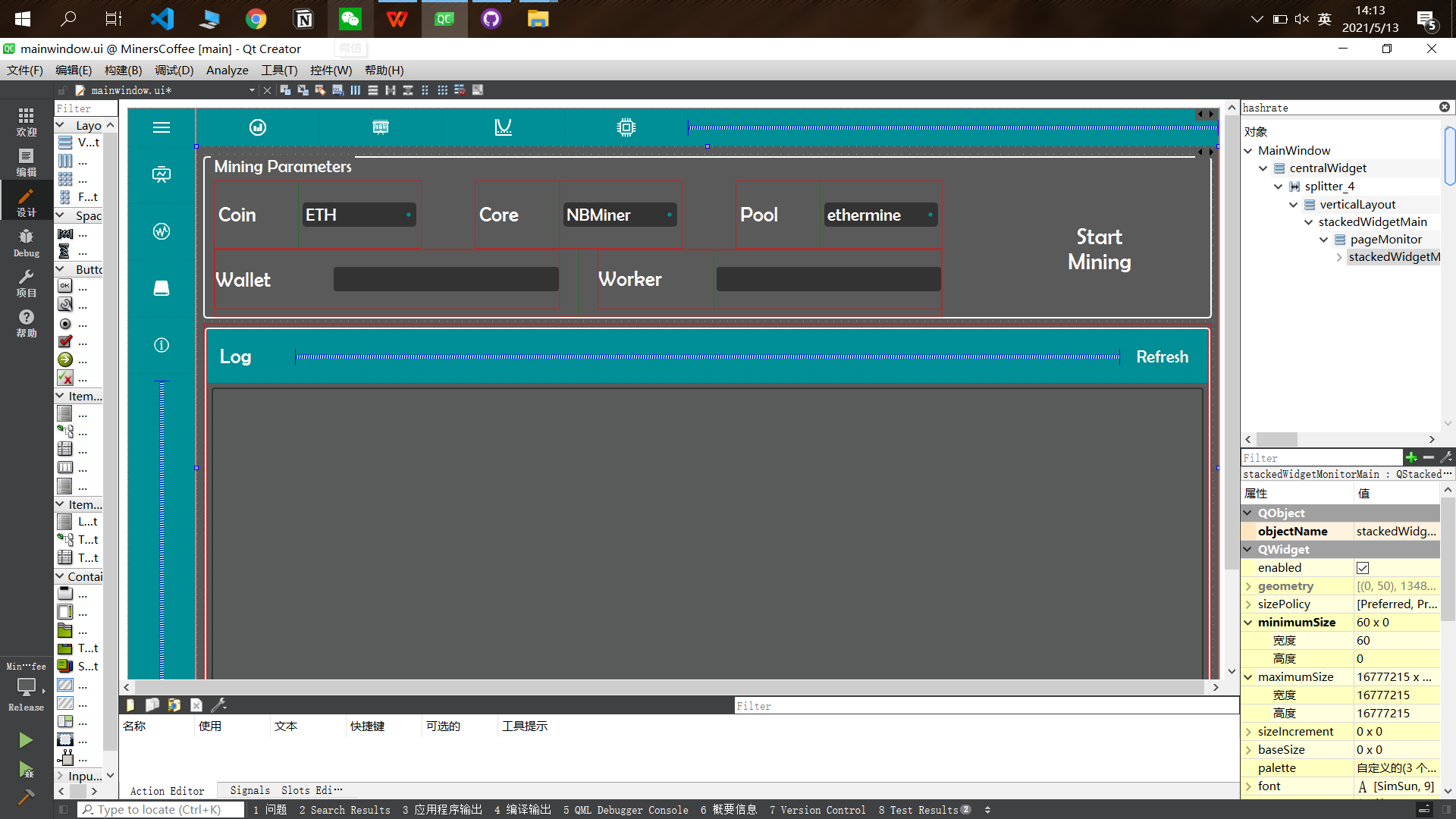
Test task:Check whether the wallet and worker filling and data acquisition are successful.

(2)

void **test\_ui\_MiningArgsComboBox**();

void **test\_ui\_MiningArgsComboBox\_data**();

Test component:



Test task:Check whether the drop-down box can be displayed when clicked, whether the

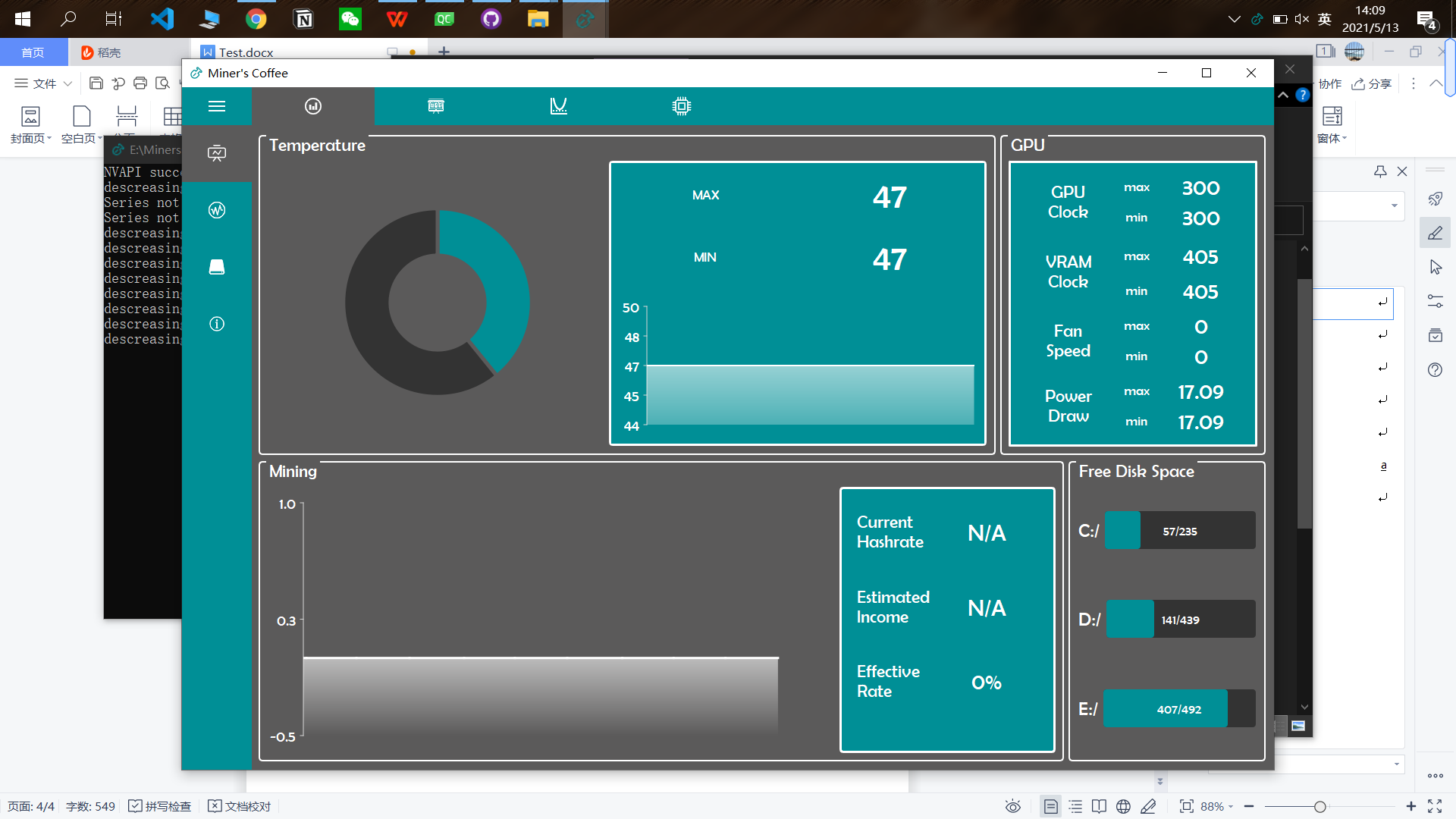
normal selection can be performed and the data after selection can be returned.

(3)

void **test\_ui\_TempPieChart**();

void **test\_ui\_TempPieChart\_data**();

Test component:



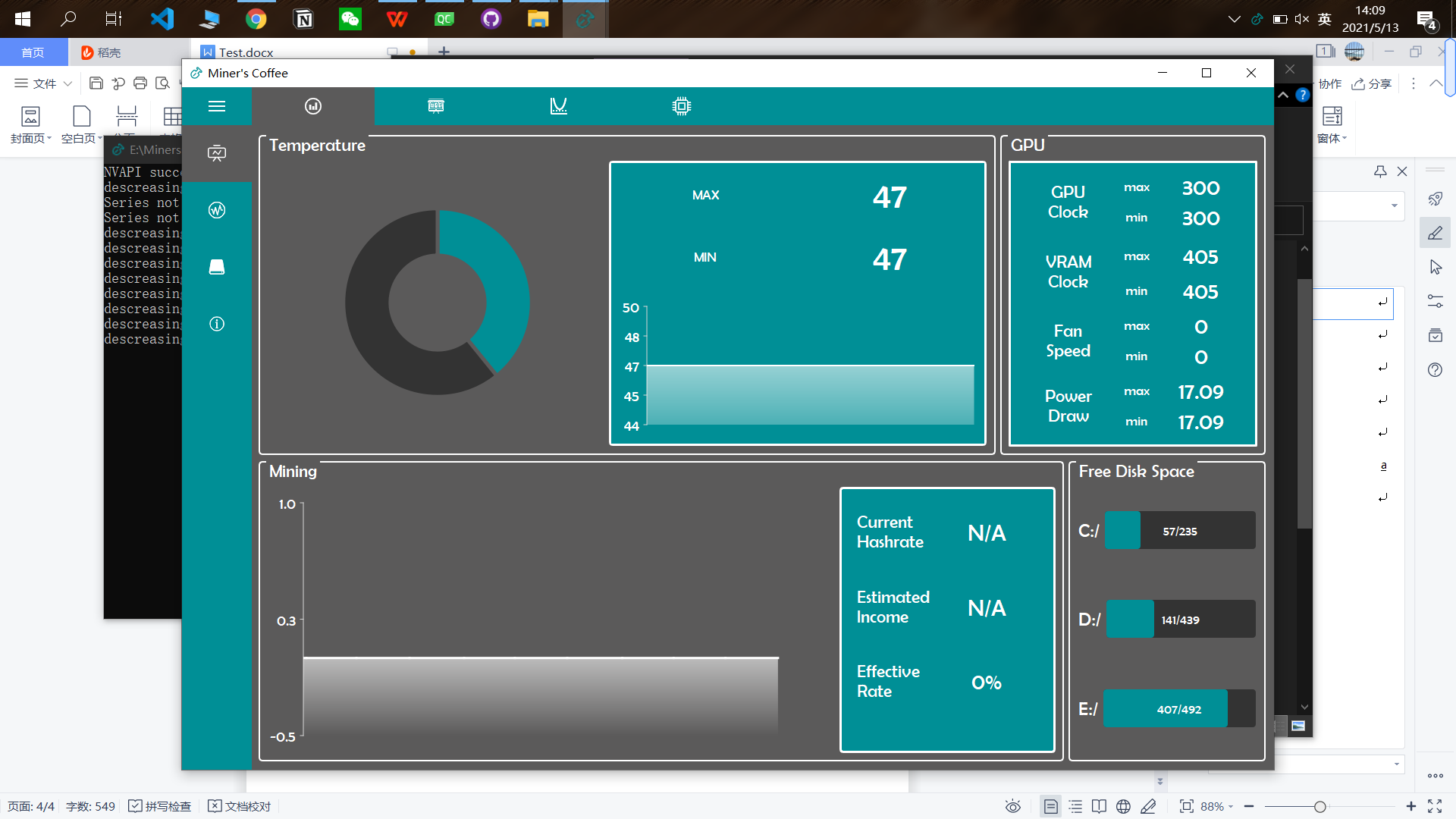
Test task: Check whether the pie-shaped color distribution can change with the change of the temperature value.

(4)

void **test\_ui\_HashrateLineChart**();

void **test\_ui\_HashrateLineChart\_data**();

Test component:



Test task: Check whether the line-shaped color distribution can change with the change of the Hashrate value.

1. Nonfunctional Parameters Analysis.

Security: Thanks to SQLite, we don’t need users’ database password to access to database.

Flexibility: The term refers to the ease with which the application can work with different models of NVIDIA graphics cards.