# Design of Tests

## 1 Test Cases

### 1.1 Function Test

The use cases are divided into independent parts, which mean that the use cases can be easily tested as a unit. So in this part, we decide to list and describe the test cases by the use case list.

Use Case Function list:

1. Leaderboard
2. Histogram
3. Calculator
4. Calendar
5. Twitter collection

### Test ID: TC1\_Leaderboard

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Requirement** | **Expected Output** | **Pass/Fail** | **Comments** |
| User enters the website and clicks on the leaderboard. | The database sends back the data of leaderboard.  The website shows the leaderboard as a graph. | Pass if the website shows the graph of the current leaderboard.  Fail if the website shows nothing or out-of-date leaderboard. | The test is to make sure that the data collected from twitters can be updated on time and displayed as leaderboard on the website. |

### Test ID: TC2\_WorkoutHistogram

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Requirement** | **Expected Output** | **Pass/Fail** | **Comments** |
| User enters the website and clicks on the button of different histogram (diet, exercise and smoke). | The database sends back the specific data of histogram.  The website shows the histogram. | Pass if the website shows the graph of the current histogram.  Fail if the website shows nothing or out-of-date histogram. | The test is to make sure that the website can receive the histogram choose and display the required histogram. |

### Test ID: TC3\_Caluator

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Requirement** | **Expected Output** | **Pass/Fail** | **Comments** |
| User uploads the individual situation in the calculator blank. | The calculator calculates the individual situation by mathematics formula and shows the result. | Pass if the website shows calculator result correctly.  Fail if the website does not receive the uploaded data or the website cannot show the result correctly. | The test is to make sure that the website can calculate the data that uploaded by the users. |

### Test ID: TC4\_Calendar

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Requirement** | **Expected Output** | **Pass/Fail** | **Comments** |
| User enters the website. | The website displays the calendar correctly. | Pass if the website shows calendar correctly.  Fail if the website cannot show the calendar correctly. | The test is to make sure that the website can display the calendar. |

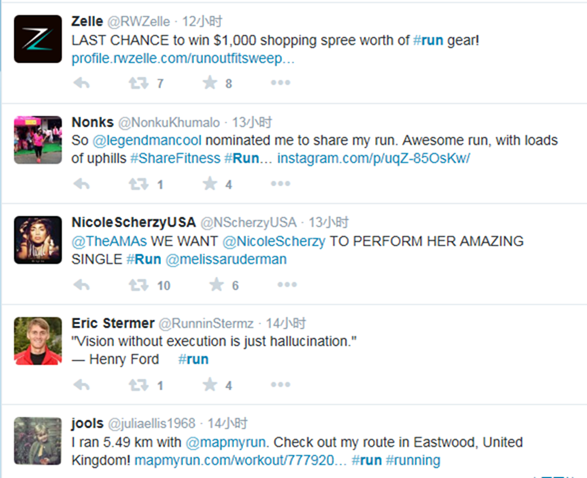
### Test ID: TC5\_CollecTwitterInformation

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Requirement** | **Expected Output** | **Pass/Fail** | **Comments** |
| The data needed by the Health Monitor Analytics occurs on the twitter. | The data needed by the Health Monitor Analytics is collected successfully in JSON and stored completely in the Mongodb. | Pass if the Mongodb stores all the data that the system needed.  Fail if data collects the wrong data or the system crashes. | The test is to make sure that the data can be collected correctly and successfully. The database can store all the data collected. |

### 1.2 Data Relevant Test

As a Health Monitor Analytics System, we should keep our system data source more accurate to ensure our website to be credible. Since we collect the information from the twitter, the information noise is unavoidable. So we make the effort to filter the twitter data by some Algorithms (like ML-KNN method).It is necessary to do the Data Relevant Test except the Function test (6.1.1).

The test is started by the raw data collected by the twitter. We search the #word in twitter. We will get the raw relevant data like the figure below. In this example, we search #run.



**Figure 16 Example Tweet**

The data in the figure will be collected completely in the database as the raw data. We use our analytics algorithms to filter the data. The filtered data is stored as a list.

As a test, it is impossible to analyze all the data. We will take a hundred filtered twitters from the list randomly. Then we judge the data relevance manually and count the number of the relevant data. At last, we can get the accuracy percentage of the analytics algorithms.

Certainly, we will make several parallel groups of the same analytics algorithm and achieve the mean accuracy percentage. If the accuracy percentage passes the threshold value, we will regard the analytics algorithm as a qualified algorithm.

We will test all the filter algorithms in our system in this way.

## 2 Test Coverage

We use test coverage to get the degree to which the specification or code of a software program has been exercised by tests.

As mentioned in 6.1.2, we divide the test by use case list. So it belongs to the State-based testing. State-based testing defines a set of abstract states that a software unit can take and tests the unit’s behavior by comparing its actual states to the expected states.

We use the State-based unit for test coverage because the website system is like the objected-oriented system. The function is defined as the object. It works as an individual state independently. So it will be easy to test in this way.

## 3 Integration Testing Strategy

In this system, we decide to use ***Vertical Integration Testing Strategies*** to test our codes for several reasons.

The codes in our system have several characteristics below:

1. The use case functions are divided independently.
2. Most of the use cases in our system give the feedback of the data as the function of the website.
3. The system is designed for the customers who want the health data.
4. The system is designed for daily uses.

The ***Vertical Integration Testing Strategies*** have several characteristics below:

1. The vertical integration approaches to develop the user stories in parallel for testing the code.

2. Each story is developed in a feedback loop, where the developers use unit tests in the inner loop and the customer runs the acceptance test in the outer loop.

3. Each cycle starts with the customer/user writing the acceptance test that will test a particular user story. Based on the acceptance test, the developer writes the unit tests and develops only the code that is relevant, i.e., needed to pass the unit tests.

4. The unit tests are run on daily basis, soon after the code is written, and the code is committed to the code base only after it passes the unit tests. The acceptance test is run at the end of each cycle (order of weeks or months).

So the ***Vertical Integration Testing Strategies*** fit our aim to test the code in our system. We believe that Vertical Integration Testing Strategies can fit our code completely and properly.