Software Requirement Specification

Quantitative Trading Platform

**軟體工程課程第四組**

指導教授：李允中教授

成員 : 鄒咏霖、汪宸宇、陳熙、李辰暘、侯思岑、李紹銘

# Outline

# 

[**Outline**](#_lhhdbdx9rq5u) **2**

[**1. System Architecture**](#_ljet50ah24u1) **3**

[1.1 Introduction](#_33z04wbpaiu5) 3

[1.2 Architecture Expression](#_a4cqx6bhca0x) 3

[1.3 System Architecture Diagram](#_27ye8n5v62q0) 5

[**2. Functional Requirement**](#_r4pyh2yli8hq) **6**

[2.1 Front-end Functional Requirements](#_yr885cmiixhs) 6

[2.2 Back-end Functional Requirements](#_hhxehyz5snxm) 8

[**3. Interface Requirement**](#_669bgg9uuspe) **11**

[3.1 External Interface Requirements](#_p937qwte7lkm) 11

[3.2 Internal Interface Requirements](#_6c689iegg6h7) 13

[**4. Nonfunctional Requirement**](#_b7p02wpd0s4a) **15**

[**5. Goal-Driven Use Case Diagram**](#_iukxvpygfk8s) **16**

[5.1 Objective Statement](#_jpu3g6pc3ezm) 16

[5.2 Goal-driven Use Case Diagram](#_udyuio5ky929) 17

[5.3 Actor Description and Actor Use Case Matrix](#_s7b3xcy6l09n) 18

[5.4 Use Case Specification](#_1yn1kt75vudi) 19

[**6. Traceability Matrix**](#_rrhb01lvi38g) **29**

[6.1 Requirements vs. Use Cases](#_u50r3lmx4499) 29

[6.2 Requirements vs. Modules](#_syxkev8fnmn8) 30

# 

# 1. System Architecture

## 1.1 Introduction

When it comes to investing, quantitative trading is perhaps the most reliable method widely used in major institutions on Wall Streets; hence, this project aims at building a quantitative platform to help users testing their trading ideas and accumulating their precious findings. Many quantitative research platforms have aroused and spread out on the Internet, like Quantopian, TradingView, RiceQuant and many more. However, some common drawbacks of such platforms are:

* many functionalities are costly
* visualization tools for performance analysis are inflexible or even lacking
* lack of a report-file management system; users have to organize their findings themselves, store them to their local hard drives and link them manually

We hope to build a system that can evaluate different trading methodologies and even manage the information gained from those testing results.

## 1.2 Architecture Expression

| **Module** | **Description** |
| --- | --- |
| **算法測試系統 Algorithm Testing Module** | Building and testing trading algorithms |
| **評估系統 Evaluation Module** | Evaluating performance of an algorithm |
| **條件測試系統 Conditional Testing Module** | Analyzing trading behavior of an algorithm |
| **文檔管理系統 File Management Module** | Provides the ability to manage algorithms and reports. |
| **標的資料處理與供給系統 Underlying Data Management Module** | Forward the underlying assets’ data to other modules |
| **參數設定系統 Parameter Setting Module** | Rendering and setting parameters of an algorithm |
| **績效計算系統 Performance Calculator Module** | Calculating Performance of an Algorithm |
| **文檔儲存系統 File Storage Module** | Store the file(algo., report, parameter) create by user action |

## 

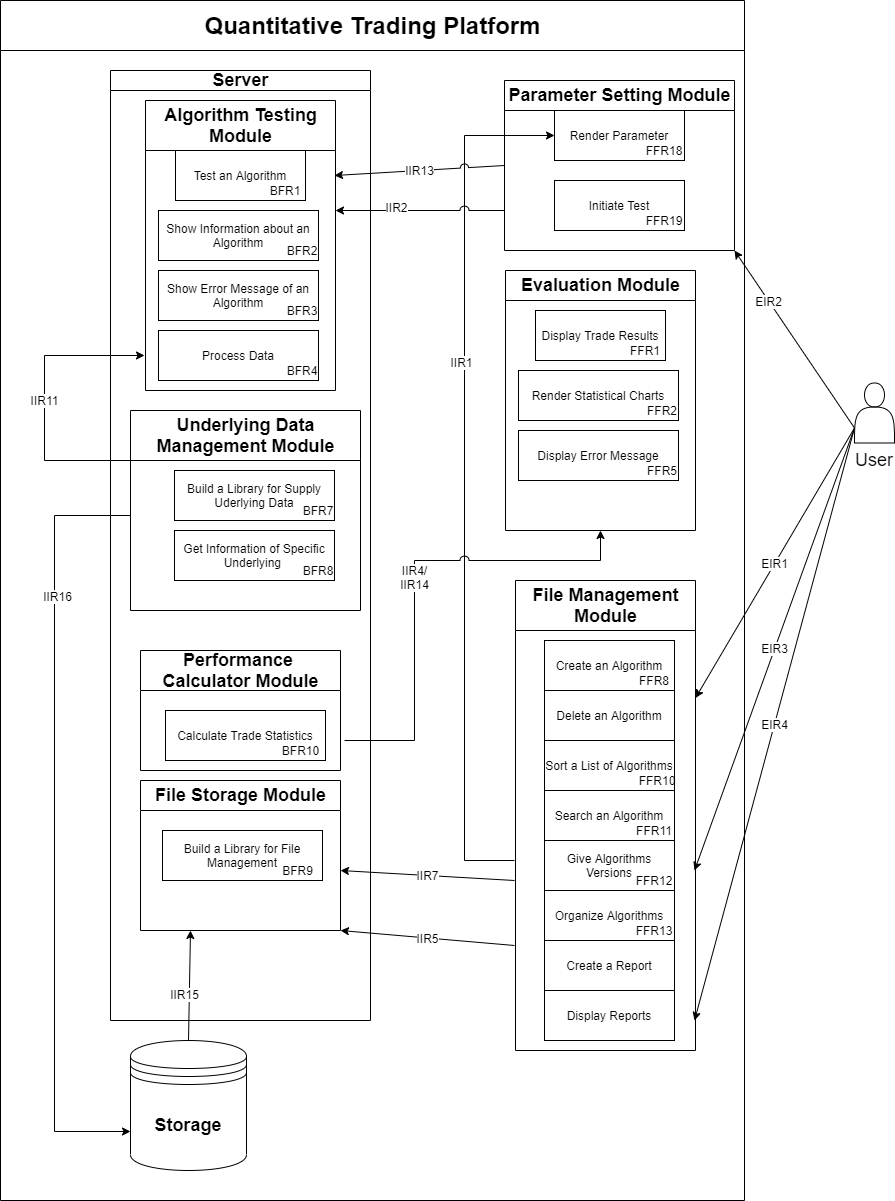
## 1.3 System Architecture Diagram

### 1.3.1 Expected

## 

## 

### 1.3.2 Realistic



# 2. Functional Requirement

## 2.1 Front-end Functional Requirements

| **評估系統 Evaluating System** | |
| --- | --- |
| **FFR1** | **Display Trade Results** |
|  | Once a test is done, or some xml files of previous tests are uploaded, results can be displayed on the webpage, including Trade Statistics and TradeActions; these results may be re-calculated at different time ranges. |
| **FFR2** | **Render Statistical Charts** |
|  | Once a test is done, or some xml files of previous tests are uploaded, common statistical charts can be displayed on the webpage. |
| **FFR3** | **Initiate Conditional Test** |
|  | Once a test is done, users can choose to further analyze the TradeActions by initiating the conditional test, which then transfers them to the Conditional Testing System. |
| **FFR4** | **Save Trade Result** |
|  | Once a test is done, users can save the TradeActions that this test generated as an xml file; they can also save the parameter settings that yield this result. |
| **FFR5** | **Display Error Message** |
|  | If any error occurs, an error message should be displayed on the web page accordingly. |

| **條件測試系統 Conditional Testing System** | |
| --- | --- |
| **FFR6** | **Perform Pivot Table Analysis** |
|  | Once initiated from Evaluating System, or uploaded from previously saved xml files, this system can generate a pivot table analyzing the profits, timestamp, tags, etc., of the TradeActions |
| **FFR7** | **Add New Columns to Pivot Table** |
|  | User can add new tags to existing pivot table via Parameter Setting System, so that new information can be added to each trade result, and user can use pivot table to study the effect of the information on these TradeActions |

| **文檔管理系統 File Management Module** | |
| --- | --- |
| **FFR8** | **Create an algorithm** |
|  | The user can create an algorithm. When the user creates an algorithm, he or she needs to provide necessary information, e.g., algorithm title and source code. |
| **FFR9** | **Delete an algorithm** |
|  | The user can delete an algorithm. When the user deletes an algorithm, this algorithm will be moved into the trash folder; if the user wants to permanently delete this algorithm, he or she can delete it in the trash folder. |
| **FFR10** | **Sort a list of algorithms** |
|  | The user can sort a list of created algorithms according to various fields, e.g., Last Modified, Title, or Profit. |
| **FFR11** | **Search algorithm** |
|  | The user can search algorithms based on titles. Results should be dynamically displayed. |
| **FFR12** | **Give algorithm versions** |
|  | In order to control the changes, the user can give an algorithm multiple versions. |
| **FFR13** | **Organize algorithms** |
|  | The user can organize his or her algorithms. For example, algorithms that have similar ideas can be placed together, so it won’t be a mess when the number of algorithms is getting larger and larger. |
| **FFR14** | **Download algorithm parameters** |
|  | The user can download the parameters of an algorithm so he or she can reproduce the same results later on. |
| **FFR15** | **Download trade actions** |
|  | Trade actions can be downloaded into XML files, for validating purposes, e.g., checking the impact of different data sources, ensuring that the logic of an algorithm is correct by comparing different implementations of it, debugging the performance system. |
| **FFR16** | **Create a report** |
|  | The user can create a report. |
| **FFR17** | **Display reports** |
|  | A created strategy is accompanied by related analysis reports. Those reports can be displayed or be downloaded. |

| **參數設定系統 Parameter Setting System** | |
| --- | --- |
| **FFR18** | **Render Parameter** |
|  | When (1) user selects an algorithm to perform single or batch tests and (2) user chooses to analyze the TradeActions of a tested algorithm, the parameter settings pane can be displayed on the webpage, for user to set numerical, textual , date-related, or categorical parameters, and check if a parameter is required or not. |
| **FFR19** | **Initiate Test** |
|  | Once parameters are set, users can start the test. |

## 2.2 Back-end Functional Requirements

| **算法測試系統 Algorithm Testing System** | |
| --- | --- |
| **BFR1** | **Test an Algorithm (Algo)** |
|  | Create an Algo instance, set it’s parameters, and execute it to get the TradeActions.  The Algo being tested can be stopped in execution.  Most Algo needs to preprocess on input data, and can be run with different parameter settings based on the processed data.  Input data of some Algos can be changed, so that the same Algo can be applied to different markets.  Some Algo can be applied to others, to compose into a bigger one. |
| **BFR2** | **Show Information about an Algo** |
|  | Display all the available Algos on the system, and parse the parameters declared in an Algo, so that other modules can use these information. |
| **BFR3** | **Show Error Message of an Algo** |
|  | If an Algo encounters a runtime error, exception messages can be passed to other modules. |
| **BFR4** | **Process Data** |
|  | Different kinds of time series statistics can be calculated, for Conditional Testing System, and for the preprocessing needs of some Algos. |

| **標的資料處理與供給系統 Underlying Data Management Module** | |
| --- | --- |
| **BFR5** | **Data Download** |
|  | Download the data needed daily, that means we need to write a function that will download data on the Internet automatically. |
| **BFR6** | **Data Format Transformation** |
|  | Transform the data downloaded into the format we want, and then store it in our database or a csv file. |
| **BFR7** | **Build a Library to Provide Underlying Assets’ Data** |
|  | Build a library that users can import in their script(e.g. import data\_for\_trade), and they can call it to get the underlying assets’ data they need. So what this library does is load the data stored in our database/csv file and then put it in some data structure(e.g. numpy array), and finally return it to other modules. |
| **BFR8** | **Get specific Underlying Asset’s information** |
|  | User can got the specific data they need by specify some parameter   * all information about TX on that day   + e.g. data\_for\_trade.TX.all() * specific information about TX on that day   + e.g. data\_for\_trade.TX.price() * information about TX from some date till now   + e.g. data\_for\_trade.TX.all(period=’{parameter}’)   + {parameter} can be ‘today’, ‘week’, ‘month’, etc. |

| **文檔儲存系統 File Storage Module** | |
| --- | --- |
| **BFR9** | **Build a Library for file management** |
|  | Other modules can read/write file(including report, parameter, algo.) via File Management Module |
| **BFR10** | **Store File** |
|  | Store file(including report, parameter, algo.) input from user/other modules in our file structure |
| **BFR11** | **Read file** |
|  | Read file(including report, parameter, algo.) out from our file source and provide it to other modules |

| **績效系統 Performance Evaluation System** | |
| --- | --- |
| **BFR10** | **Calculate Trade Statistics** |
|  | Trade Statistics can be calculated when TradeActions are given. Different TradeStatistics may be added in the future. |

# 

# 3. Interface Requirement

## 3.1 External Interface Requirements

| **EIR1** | **Operations on algorithms presented in File Management Module done by the user** |
| --- | --- |
|  | The user can add, delete, search and select an algorithm from File Management Module.  Note: The operation "delete" is only available with his/her own algorithms, i.e., the user cannot delete algorithms uploaded from others. |
| **EIR2** | **Tune up the parameters of an algorithm** |
|  | After choosing an algorithm, the user can tune up its parameters. |
| **EIR3** | **Upload reports to File Management Module** |
|  | After evaluating results, the user can write a report and then upload it to  File Management Module. |
| **EIR4** | **Read reports from File Management Module** |
|  | The user can select a report and read it from File Management Module. |
| **EIR5** | **Reproduce the results of an algorithm** |
|  | The user can ask File Management Module to reproduce the results of an algorithm. |
| **EIR6** | **Pass TradeActions to Evaluation Module** |
|  | The user can pass TradeActions they want to evaluate to Evaluation Module. |
| **EIR7** | **Set the tags of TradeActions in Condition Testing Module** |
|  | The user can set the tags of TradeActions to get the analysis of it. |
| **EIR8** | **Store the parameters locally** |
|  | After getting ideal results from the visualization of Evaluation Module, the user can store the corresponding parameters and TradeActions of the algorithm. |
| **EIR9** | **Store the TradeActions locally** |
|  | The user can optionally save the results outputted from Evaluation Module as an .xml file. |
| **EIR10** | **Update data in File Storage Module** |
|  | Administrators are responsible for updating the data in File Storage Module. |
| **EIR11** | **Tune up the parameters of a batch of algorithms** |
|  | After choosing a batch of algorithms, the user can tune up the parameters of those parameters. |

## 

## 3.2 Internal Interface Requirements

| **IIR1** | **File management Module forward algo to Parameter Setting Module** |
| --- | --- |
|  | File management Module can notify Parameter Setting System to display the parameters of a selected algorithm |
| **IIR2** | **Parameter Setting Module forwards the (tuned) parameters to the Algo Testing System** |
|  | Parameter Setting Module passes the (tuned) parameters and the selected algo to the Algo Testing System. |
| **IIR3** | **Evaluation Module evaluates the TradeActions obtained from Algo Testing Module** |
|  | The obtained TradeActions from Algo Testing System will pass to the Evaluation Module for evaluation. |
| **IIR4** | **TradeStats calculation by Performance Calculator Module & forwarding to Evaluating System** |
|  | Algorithm Testing Module pass TradeActions to Performance Calculator Module for calculating TradStats, and Performance Calculator Module pass TradStats to Evaluating System for evaluation.. |
| **IIR5** | **File Management Module stores/reads reports to File Storage Module** |
|  | After File Management Module receives upload requests of reports from users, the system stores the file to File Storage Module. |
| **IIR6** | **File Management Module reads parameters from File Storage Module** |
|  | File Management Module retrieves parameters from File Storage Module for reproducing results. |
| **IIR7** | **File Management Module stores/reads algo. from File Storage Module** |
|  | Store : After File Management Module receives creating algo. requests from users, the system forwards the algo. to File Storage Module for further strage.  Read : When File Management Module needs algo. that had been stored, it will retrieve it from File Storage Module. |
| **IIR8** | **Performance Calculator passes the TradeResults to Condition Testing Module** |
|  | After calculating TradeResults, Performance Calculator Module will pass the TradeResults to Condition Testing Module for generate pivot table. |
| **IIR9** | **File Storage Module forward old parameter to Parameter Setting Module for reproducing result** |
|  | Parameter Setting Module send a request to File Storage Module for retrieve the old parameter for reproducing, and then File Storage Module forward it. |
| **IIR10** | **Evaluating System stores the results from Evaluating System** |
|  | After users get the desired results and ask to store them, Evaluating System will pass the parameter to File Storage Module. for storage. |
| **IIR11** | **Algo Testing System gets data from Underlying Data Management Module** |
|  | Algo Testing System gets the required data from Underlying Data Management Module. |
| **IIR12** | **Conditional Testing System gets data from Underlying Data Management Module** |
|  | Conditional Testing System gets the required data from Underlying Data Management Module. |
| **IIR13** | **Algo Testing System loads batch parameter from Parameter Setting Module** |
|  | Load batch parameter from Parameter Setting Module to Algo Testing System |
| **IIR14** | **Evaluate batch TradeStats** |
|  | batch version of IIR4 |
| **IIR15** | **File Storage Module Read/Store File from/to Storage** |
|  | * File Storage Module store file(including report, parameter, algo.) input from user/other modules in our file structure * File Storage Module read file(including report, parameter, algo.) out from our file source for providing it to other modules |
| **IIR16** | **Underlying Data Management Module Read File from Storage** |
|  | File Storage Module scratch underlying data out from our file source for providing it to other modules |

# 

# 4. Nonfunctional Requirement

| **NFR1** | **Efficiency** |
| --- | --- |
|  | Test of an algorithm, either single or batch tested, should be as fast as possible. |
| **NFR2** | **Convenience** |
|  | Development of new algorithms should be convenient and straightforward. |
| **NFR3** | **Attractiveness** |
|  | Visual experience of the web page should be pleasant. |

# 

# 5. Goal-Driven Use Case Diagram

## 5.1 Objective Statement

| **Name** | **Property** | **Description** |
| --- | --- | --- |
| **Get underlying assets’ data** | (R,A,F) | The user can obtain the intended data about underlying assets they want. |
| **Parameters of Algorithms can be found** | (R,A,F) | The user can test algorithms and find the parameter sets that best suit the algorithm. |
| **Algorithms can be modified conveniently, or dynamically changed** | (S,A,N) | The user can modify behavior of an algorithm, which means algorithms should be flexible in use. |
| **Manage Algorithms and Reports** | (R,A,F) | Users can manage algorithms, browse reports on the system. |
| **Algorithms’ performance can be traced (since parameters can be saved and test can be reproduced)** | (R,A,F) | Users can reproduce the results of an algorithm. |
| **Algorithms can be improved.** | (S,A,N) | An algorithm can be evaluated and improved by analyzing its characteristics. |
| **New algorithms can be developed.** | (R,A,F) | User can upload an algo |
| **Algorithms can be evaluated.** | (R,A,F) | User can test & get the evaluation of their algo in our system |

## 

## 5.2 Goal-driven Use Case Diagram

### 5.2.1 Expected

## 

### 5.2.2 Realistic

### 

## 5.3 Actor Description and Actor Use Case Matrix

| **Name** | **Description** |
| --- | --- |
| **User** | People who use our system for testing their algorithms. |
| **System Maintainer** | People who maintain the system. |

| **Use Case** | **User** | **System Maintainer** |
| --- | --- | --- |
| User/Admin uploads an algorithm(紹明) | v | v |
| Single Test Algorithms(思岑) | v |  |
| Get underlying assets’ data(鄒) | v |  |
| Download & Store underlying assets’ data from Internet |  | v |
| Batch testing Algorithms(辰暘) | v |  |
| Testing Composite Algorithm | v |  |
| Reproduce parameters of an algorithm | v |  |
| Observe the characteristics of TradeActions | v |  |
| Study Reports on the Platform(宸宇) | v |  |

## 

## 5.4 Use Case Specification

| **Use Case ID** | **UC-001** |
| --- | --- |
| **Use Case Name** | User/Admin uploads an algorithm |
| **Goal** | New algorithms can be developed. |
| **Requirement** | [FFR8 : Create an algorithm]  [FFR12 : Give algorithm versions]  [BFR10 : Store File] |
| **Description** | **User/Admin can create an algorithm by providing algorithm information, e.g. algorithm title, source code and version.** |
| **Actor** | User/Admin |
| **Basic Flow** | 1. Click the “New button”, then show a “Input Modal” which requires the user/admin input algorithm information.   (Title, Version, Description)   1. Click the “Confirm button” , then show the “Input File Modal” which requires the user/admin to upload the source code. 2. User/Admin uploads the new algorithm. 3. Add the new algorithm to the “Algorithm Menu”. |
| **Alternative Flow** | 2.1 User input lacks in Title or Version, back to Step 1.  2.2 User Click “Exit Button” in Input Modal, Exit.  3.1 Duplicate “Title and Version”, Exit. |
| **Exceptional Flow** |  |
| **Includes Use Case** |  |
| **Extends Use Case** |  |

| **Use Case ID** | **UC-002** |
| --- | --- |
| **Use Case Name** | Display results after the user executes evaluations. |
| **Goal** | Algorithms can be evaluated. |
| **Requirement** | [FFR1 : Display Trade Results]  [FFR2 : Render Statistical Charts]  [FFR5 : Display Error Messages] |
| **Description** | **The user can see trade results and statistical charts and error messages after he/she executes an evaluation.** |
| **Actor** | User |
| **Basic Flow** | 1. The user executes a new test. 2. The trade results include trade statistics and trade actions. 3. The user selects a period of time span to show trade results. 4. Common statistical charts can be displayed based on the trade results. |
| **Alternative Flow** |  |
| **Exceptional Flow** | If the user uploads a test with illegal data, error messages will display. |
| **Includes Use Case** |  |
| **Extends Use Case** |  |

| **Use Case ID** | **UC-003** |
| --- | --- |
| **Use Case Name** | Call API to obtain underlying assets’ data |
| **Goal** | Get underlying assets’ data |
| **Requirement** | [BFR7 : Build a Library for Provide Underlying Assets’ Data provide]  [BFR8 : Get specific Underlying Asset’s information] |
| **Description** | **User gets the underlying assets by calling the API provided by Underlying Assets’ Data Process and Provide Module.** |
| **Actor** | User |
| **Basic Flow** | 1. import the library built by Underlying Assets’ Data Process and Provide Module 2. Use the specific command to call Underlying Assets’ Data Process and Provide Module for the request data(e.g. data\_for\_trade.TX.all() ) 3. Underlying Assets’ Data Process and Provide Module gets the call and takes out the request data stored in file source and put them in some data structure(e.g. numpy array) 4. Underlying Assets’ Data Process and Provide Module returns that data back to where the call comes from |
| **Alternative Flow** |  |
| **Exceptional Flow** |  |
| **Includes Use Case** |  |
| **Extends Use Case** |  |

| **Use Case ID** | **UC-004** |
| --- | --- |
| **Use Case Name** | Download & Store underlying assets’ data from Internet |
| **Goal** | Get underlying assets’ data |
| **Requirement** | [BFR5 : Data Download]  [BFR6 : Data Format Transformation] |
| **Description** | **System Maintainer download the underlying assets’ data from the Internet daily and transformed the form of the data before storing it in local drive** |
| **Actor** | System Maintainer |
| **Basic Flow** | 1. Run a script that triggers the program for grabbing the data automatically at some specific time every day 2. The program download underlying assets’ data from Internet 3. The program converted the data downloaded to some specific form and then stored it in local |
| **Alternative Flow** |  |
| **Exceptional Flow** |  |
| **Includes Use Case** |  |
| **Extends Use Case** |  |

| **Use Case ID** | **UC-005** |
| --- | --- |
| **Use Case Name** | Batch testing Algorithms |
| **Goal** | Parameters of Algorithms can be found. |
| **Requirement** | [FFR18 : Render Parameter]  [FFR19 : Initiate Test]  [FFR1 : Display TradeResult]  [FFR2 : Render Statistical Charts]  [BFR1 : Testing an Algo]  [BFR2 : Show information about an Algo]  [BFR7 : Build a Library for Provide Underlying Assets’ Data] |
| **Description** | A user can test multiple parameters of an algorithm at once, and observe their performances. |
| **Actor** | User |
| **Basic Flow** | 1. Choose the algorithm to be tested 2. Set the batch-testing parameters 3. Execute the test 4. The web page displays several trade statistics for each parameter setting 5. Users can see the heatmap of some trade statistics of different parameters, and find the better ones |
| **Alternative Flow** | If for some parameter set, the algorithm being tested encountered error during execution:   1. trade statistics for that parameter set is skipped, until no other parameter sets are left |
| **Exceptional Flow** |  |
| **Includes Use Case** | 1. choosing the algorithm 2. evaluating algorithms |
| **Extends Use Case** | 1. trade statistics table can be saved after batch test is done |

### 

| **Use Case ID** | **UC-006** |
| --- | --- |
| **Use Case Name** | Testing Composite Algorithm |
| **Goal** | Algorithms can be modified conveniently, or dynamically changed |
| **Requirement** | [FFR18 : Render Parameter]  [FFR19 : Initiate Test]  [FFR1 : Display TradeResult]  [FFR2 : Render Statistical Charts]  [BFR1 : Testing an Algo]  [BFR2 : Show information about an Algo]  [BFR7 : Build a Library for Provide Underlying Assets’ Data] |
| **Description** | Behaviors of algorithms can be dynamically modified by applying another algorithm onto existing ones, to enhance flexibility. |
| **Actor** | User |
| **Basic Flow** | 1. choose an “delegating” algorithm that can be applied to other algorithm 2. choose the algorithm that is to be modified 3. set the parameters for all the algorithms involved 4. execute the test 5. obtain trade statistics, trade results, and statistical charts from the web page |
| **Alternative Flow** | If error occurs during execution:   1. test is stopped automatically, and error message is displayed on the web page |
| **Exceptional Flow** |  |
| **Includes Use Case** | 1. choosing the algorithm 2. evaluating algorithms |
| **Extends Use Case** |  |

| **Use Case ID** | **UC-007** |
| --- | --- |
| **Use Case Name** | Reproduce parameters of an algorithm |
| **Goal** | Algorithms' performance can be traced (since parameters can be saved and test can be reproduced) |
| **Requirement** | [BFR11 : Read file]  [IIR6 : File Management Module reads parameters from File Storage Module] |
| **Description** | Users can reproduce parameters of an algorithm recorded in a report. |
| **Actor** | User |
| **Basic Flow** | 1. User read report and find parameter settings on that report 2. User reproduce the parameter settings |
| **Alternative Flow** | 1. **If parameter is not found on the File Storage Module, ignore the request** |
| **Exceptional Flow** |  |
| **Includes Use Case** |  |
| **Extends Use Case** |  |

| **Use Case ID** | **UC-008** |
| --- | --- |
| **Use Case Name** | Observe the characteristics of TradeActions |
| **Goal** | Algorithms can be improved |
| **Requirement** | [FFR6 : Perform Pivot Table Analysis]  [FFR7 : Add New Columns to Pivot Table] |
| **Description** | **Users are able to find the characteristics of TradeActions by operating the functions of Conditional Testing System, such as analyzing pivot table, or adding tags to the table.** |
| **Actor** | User |
| **Basic Flow** | 1. Upload TradeActions which are waiting to be evaluated to Conditional Testing System and get its TradeResult as a pivot table. 2. Set tags to the table as conditions to analyze the results under different conditions. 3. Add new tags to the table to get new trade results and more information. |
| **Alternative Flow** |  |
| **Exceptional Flow** |  |
| **Includes Use Case** |  |
| **Extends Use Case** | evaluating algorithms |

| **Use Case ID** | **UC-009** |
| --- | --- |
| **Use Case Name** | View reports and algorithms |
| **Goal** | Manage algorithms and Reports |
| **Requirement** | [FFR10 : Sort a list of algorithms]  [FFR11 : Search algorithms]  [FFR17: Display reports] |
| **Description** | The user views reports and algorithms via File Management Module. |
| **Actor** | User |
| **Basic Flow** | 1. The user finds the target algorithms on the file manager page, by following ways:    * sort the display list of algorithms by, for example, Last Modified;    * search the titles of algorithms. 2. After the algorithm items are found, the user selects the intended versions. 3. The user studies the algorithms by analysing the related data, for example, trade actions, or by reading the reports. |
| **Alternative Flow** |  |
| **Exceptional Flow** |  |
| **Includes Use Case** |  |
| **Extends Use Case** |  |

# 

# 6. Traceability Matrix

## 6.1 Requirements vs. Use Cases

|  | **UC-001** | **UC-002** | | | **UC-003** | **UC-004** | **UC-005** | **UC-006** | **UC-007** | **UC-008** | **UC-009** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FFR1** |  | v | | |  |  | v | v |  |  |  |
| **FFR2** |  | v | | |  |  | v | **v** |  |  |  |
| **FFR3** |  |  | | |  |  |  |  |  |  |  |
| **FFR4** |  |  | | |  |  |  |  |  |  |  |
| **FFR5** |  | v | | |  |  |  |  |  |  |  |
| **FFR6** |  |  | | |  |  |  |  |  | v |  |
| **FFR7** |  |  | | |  |  |  |  |  | v |  |
| **FFR8** | v |  | | |  |  |  |  |  |  |  |
| **FFR9** |  |  | | |  |  |  |  |  |  |  |
| **FFR10** |  |  | | |  |  |  |  |  |  | v |
| **FFR11** |  |  | | |  |  |  |  |  |  | v |
| **FFR12** | v |  | | |  |  |  |  |  |  |  |
| **FFR13** |  |  | | |  |  |  |  |  |  |  |
| **FFR14** |  |  | | |  |  |  |  |  |  |  |
| **FFR15** |  |  | | |  |  |  |  |  |  |  |
| **FFR16** |  |  | | |  |  |  |  |  |  |  |
| **FFR17** |  |  | | |  |  |  |  |  |  | v |
| **FFR18** |  |  | | |  |  | v | v |  |  |  |
| **FFR19** |  |  | | |  |  | v | v |  |  |  |
|  | **UC-001** | **UC-002** | | | **UC-003** | **UC-004** | **UC-005** | **UC-006** | **UC-007** | **UC-008** | **UC-009** |
| **BFR1** |  |  | | |  |  | v | v |  |  |  |
| **BFR2** |  |  | | |  |  | v | v |  |  |  |
| **BFR3** |  |  | | |  |  |  |  |  |  |  |
| **BFR4** |  |  | | |  |  |  |  |  |  |  |
| **BFR5** |  |  | | |  | v |  |  |  |  |  |
| **BFR6** |  |  | | |  | v |  |  |  |  |  |
| **BFR7** |  |  | | | v |  | v | v |  |  |  |
| **BFR8** |  |  | | | v |  |  |  |  |  |  |
| **BFR9** |  |  | | |  |  |  |  |  |  |  |
| **BFR10** | v |  | | |  |  |  |  |  |  |  |
| **BFR11** |  |  | | |  |  |  |  | v |  |  |
| **BFR12** |  |  | | |  |  |  |  |  |  |  |

## 

## 

## 6.2 Requirements vs. Modules

|  | **Algorithm Testing Module** | **Evaluation Module** | | | **Conditional Testing Module** | **File Management Module** | **Underlying Data Management Module** | **Parameter Setting Module** | **Performance Calculator Module** | **File Storage Module** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FFR1** |  | v | | |  |  |  |  |  |  |
| **FFR2** |  | v | | |  |  |  |  |  |  |
| **FFR3** |  | v | | |  |  |  |  |  |  |
| **FFR4** |  | v | | |  |  |  |  |  |  |
| **FFR5** |  | v | | |  |  |  |  |  |  |
| **FFR6** |  |  | | | v |  |  |  |  |  |
| **FFR7** |  |  | | | v |  |  |  |  |  |
| **FFR8** |  |  | | |  | v |  |  |  |  |
| **FFR9** |  |  | | |  | v |  |  |  |  |
| **FFR10** |  |  | | |  | v |  |  |  |  |
| **FFR11** |  |  | | |  | v |  |  |  |  |
| **FFR12** |  |  | | |  | v |  |  |  |  |
| **FFR13** |  |  | | |  | v |  |  |  |  |
| **FFR14** |  |  | | |  | v |  |  |  |  |
| **FFR15** |  |  | | |  | v |  |  |  |  |
| **FFR16** |  |  | | |  | v |  |  |  |  |
| **FFR17** |  |  | | |  | v |  |  |  |  |
| **FFR18** |  |  | | |  |  |  | v |  |  |
| **FFR19** |  |  | | |  |  |  | v |  |  |
| **BFR1** | v |  | | |  |  |  |  |  |  |
| **BFR2** | v |  | | |  |  |  |  |  |  |
| **BFR3** | v |  | | |  |  |  |  |  |  |
| **BFR4** | v |  | | |  |  |  |  |  |  |
| **BFR5** |  |  | | |  |  | v |  |  |  |
| **BFR6** |  |  | | |  |  | v |  |  |  |
| **BFR7** |  |  | | |  |  | v |  |  |  |
| **BFR8** |  |  | | |  |  | v |  |  |  |
| **BFR9** |  |  | | |  |  |  |  |  | v |
| **BFR10** |  |  | | |  |  |  |  |  | v |
| **BFR11** |  |  | | |  |  |  |  |  | v |
| **BFR12** |  |  | | |  |  |  |  | v |  |

## 