General Motors FinTech

Product Testing Plan

V 1.0

11/25/2019

Mohamad Saab
Aalem Singh
John Gettel
Abdul Ahmad

Document Version History

Version	Date	Author	Reviewer	Approved by	Description
0.1	11/10/2019	Abdul Ahmad	N/A	N/A	Created the new document, formatted, wrote starter text in all sections, wrote TC 1, 3
0.2	11/15/2019	Abdul Ahmad	N/A	N/A	Completed section 1 and 2 with all subsections, proofread
0.3	11/21/2019	Abdul Ahmad; John Gettel;	N/A	N/A	Formatting, TOC hyperlinks, wrote section 1.1, 2, reformatted test case boxes
0.4	11/22/2019	Abdul Ahmad; John Gettel; Mohamad Saab;	Abdul Ahmad	N/A	Several test cases with scripts, proofread/reviewed
0.5	11/23/2019	Abdul Ahmad; John Gettel; Aalem Singh;	Abdul Ahmad	N/A	Several test cases with scripts, added section 3, proofread/reviewed
0.6	11/24/2019	Abdul Ahmad; John Gettel;	Abdul Ahmad	N/A	Added remaining functional test cases, completed nonfunctional test cases, wrote section 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, appendix, proofread/reviewed
1.0	11/25/2019	Abdul Ahmad	N/A	John Gettel; Aalem Singh; Mohamad Saab;	TOC links, formatted all sections, page breaks, code snippets at the end, proofread/reviewed

TABLE OF CONTENTS

1 INTRODUCTION 4
1.1 Purpose of the Product Testing Plan Document 4
1.2 References 4

2 FUNCTION TESTING 4
2.1 Test Approach 4
2.2 Steps 4
2.3 Pass/Fail Criteria 4
2.4 Entry/Exit Criteria 4
2.5 Suspension/Resumption Criteria 5
2.6 Test Risks 5
2.7 Expected Results 5
2.8 Priority/Priority Reason 5
2.9 Pre-Condition and Post-Condition 5
2.10 Functional Test Cases 6

3 NON-FUNCTIONAL TESTING 62
3.1 Non-Functional Testing Approach 62
3.2 Non-Functional Testing Pass/Fail Criteria 62
3.3 Non-Functional Testing Risks 62
3.4 Non-Functional Testing Expected Results 62
3.5 Non-Functional Testing Priority/Priority Reason 62
3.6 Non-Functional Test Cases 63

4 PRODUCT TESTING CERTIFICATION 84

APPENDIX 85
APPENDIX A : SQL and Python Testing Scripts 85
APPENDIX B : Key Terms 113

1 INTRODUCTION

1.1 PURPOSE OF THE PRODUCT TESTING PLAN DOCUMENT

The purpose of the GM FinTech Product Testing Plan document is to test all the functionality, front-end, database and middle-layer calculations(Python) of the application. The test cases will go through each requirement laid out in the Software Requirements Specification document. The Test Plan document tracks the necessary information required to effectively define the approach to be used in the testing of the project's product. The Test Plan document is created during the Planning Phase of the project. The intended audience of this document includes but not limited to, the project manager, project team, and testing team. Some portions of this document may on occasion be shared with the client/user and other stakeholder whose input and approval into the testing process is needed.

1.2 REFERENCES

The test cases listed in this document have a direct impact on the data visualized in the Tableau user interface. The use cases listed in the design document are used to reference the test cases that are implemented.

2 FUNCTION TESTING

2.1 TEST APPROACH

Using the testing framework described in this document, the connection to the data source (YAHOO), middle-layer Python processes and database objects will be tested. A walkthrough of the testing process including SQL scripts for creating tables from the database, will be detailed throughout this document.

2.2 STEPS

Describe the overall approach to be used for the test in a step by step manner.

2.3 PASS/FAIL CRITERIA

If each individual test case matches the desired outcomes, the test is considered successful. If any of the test cases do not meet the predicted criteria, the test case is marked as a failure. A failed test must be reviewed and resolved.

2.4 ENTRY/EXIT CRITERIA

The Entry Criteria of functional testing is completion of Prototype 3. The Exit Criteria of functional testing is the completion of all stated test cases. This is the entry and exit criteria used to start testing and determine when to stop testing.

2.5 SUSPENSION/RESUMPTION CRITERIA

Testing will be suspended when a test case fails or when outside factors like change of product design occur. Testing will resume when these preventative factors are identified and resolved.

2.6 TEST RISKS

All machines utilizing this application must have the same software and connection setup. Any missing component will cause the application to not work as intended. The database must be setup properly, including the order of fields in the table. All hardware and software requirements as listed in the software requirement specifications document must be fulfilled. Availability of a decent internet connection is also required.

2.7 EXPECTED RESULTS

This section describes what is the expected result of the undergoing test. Most results are in the form of a data value return. Some tests are to check Python script functionality and validity.

2.8 PRIORITY/PRIORITY REASON

Priority number and the reason why the test is prioritized at that number.

2.9 PRE-CONDITON AND POST-CONDITION

These are the set of conditions that exist before a test begins. Username and password for the MySQL is created at the time of setup so it could be different for a tester/user than whats listed in this document. This document lists the generic password that a tester can use. Tester's machine's MySQL must be setup with this username and password first otherwise they will have to use their local username and password. This username and password should then be added to PyCharm code file named **'dbEngine.py'** in the **'conn-str'** variable. Post condition is the application's status and state after the test has completed(success or failure).

2.10 FUNCTIONAL TEST CASES

ID	**TC_1.0**		
Test Case Name	Database Engine has the correct MYSQL connection string		
Created By	Abdul Ahmad	**Date Created**	11/10/2019
Description	Application must use the proper MySQL database connection string.		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly 4 In PyCharm 'dbEngine.py' file has **Username** = root 5 In PyCharm 'dbEngine.py' file has **Password** = password		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database properly 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_1.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive one of the following message "incorrect engine" or "correct engine"		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order **Exit :** When the test completes or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** If the message is "correct engine" **Failed :** If the message is "incorrect engine"		
Fail Reasons	1 Wrong MySQL string in dbEngine.py 2 MySQL database is not function properly 3 MySQL database **username** or **password** is incorrect		
Priority	**High**	**Priority Reason**	**Important to have the database connection working properly**
Priority Sequence	**H-1**		

ID	**TC_2.0**		
Test Case Name	Correct financial data exchange for fetching base statistics		
Created By	Abdul Ahmad	**Date Created**	11-10-2019
Description	Application is designed to use a free financial data exchange. It is setup to use 'YAHOO'. The test needs to validate that 'YAHOO' is being used for fetching financial records.		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly 4 In PyCharm 'dbEngine.py' file has **Username** = root 5 In PyCharm 'dbEngine.py' file has **Password** = password		
Postconditions	1 Test will be completed successfully 2 Middle layer of Python will push data to the MySQL database properly using the expected configured data exchange 3 Database will be loaded with accurate records		
Test Steps	1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_2.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive one of the following message "incorrect exchange" or "correct exchange"		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the application. **Exit :** Test can be exited at any time during the run or after it finishes		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** If the message is "correct exchange" **Failed :** If the message is "incorrect engine"		
Fail Reasons	1 – Internet connection issues 2 – YAHOO exchange is down 3 – 'dbEngine.py' or 'DataFetch' python code is corrupted		
Priority	**High**	**Priority Reason**	**Availability of free financial data exchange is required**
Priority Sequence	**H-2**		

ID	**TC_3.0**		
Test Case Name	No duplicate instrument ids in 'instrumentmaster' table.		
Created By	Abdul Ahmad	**Date Created**	11-21-2019
Description	No duplicate instrument ids in 'instrumentmaster' table in the database.		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 Data table has values		
Postconditions	1 No duplicate records exist in the table 2 Database stays intact		
Test Steps	1 Open MySQL WB 2 Browse to Unit Test folder 3 Open TC_3.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	No records should show up in the Result Grid in the MySQL Workbench		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application **Exit :** Test can be exited at any time during the run or after it finishes		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** No records should show in the Results Grid in MySQL Workbench **Failed :** If any records show up in the Results Grid, the record will have a value of '2' in the 'cnt' column.		
Fail Reasons	1 MySQL database table 'dbo_instrumentmaster' has multiple records for each instrument 2 'dbo_instrumentmaster' table is not setup correctly 3 Manually inserted duplicate values into the 'dbo_instrumentmaster' table		
Priority	**High**	**Priority Reason**	**Duplicate records will throw off all related calculations**
Priority Sequence	**H-3**		

ID	**TC_4.0**		
Test Case Name	No duplicate values/records in future forecast for each symbol		
Created By	Abdul Ahmad	**Date Created**	11-21-2019
Description	No duplicate values/records in future forecast for each symbol in the database		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User		
Preconditions	1 MySQL **Username** = root **Password** = password 2 Database is created 3 Data table 'dbo_algorithmforecast' exists 4 Data table is not empty		
Postconditions	1 – No duplicate records exist in the table 2 – Database stays intact		
Test Steps	1 – Open MySQL WorkBench 2 – Browse to Unit Test folder 3 – Open TC_4.0.sql 4 – Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	5 records with each showing 'cnt' value of 10 on each record in MySQL WorkBench's Result Grid		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application **Exit :** Test can be exited at any time during the run or after it finishes		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows 5 records and 'cnt' value of 10 for each record **Failed :** The Result Grid in MySQL WorkBench shows more than 5 records or 'cnt' value not equal to 10 for each record		
Fail Reasons	1 MySQL database table 'dbo_algorithmforecast' has multiple records for each instrument for future dates 2 'dbo_algorithmforecast' table is not setup correctly		
Priority	**Medium**	**Priority Reason**	**Extra forecast values will not impact other calculations**
Priority Sequence	**M-1**		

ID	**TC_5.0**		
Test Case Name	No duplicate values/records in past forecast for each symbol		
Created By	Abdul Ahmad	**Date Created**	11-21-2019
Description	No duplicate values/records in past forecast for each symbol in the databse		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 Data table has values		
Postconditions	1 No duplicate records exist in the table 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_5.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	5 records with each showing 'cnt' value of 751 on each record in MySQL WorkBench's Result Grid		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application **Exit :** Test can be exited at any time during the run or after it finishes		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows 5 records and 'cnt' value of 751 for each record **Failed :** The Result Grid in MySQL WorkBench shows more than 5 records or 'cnt' value not equal to 751 for each record		
Fail Reasons	1 MySQL database table 'dbo_algorithmforecast' has multiple records for each instrument for future dates 2 'dbo_algorithmforecast' table is not setup correctly		
Priority	**Medium**	**Priority Reason**	**Extra forecast values will not impact other calculations**
Priority Sequence	**M-2**		

ID	**TC_6.0**		
Test Case Name	Future Forecast should exist for each symbol		
Created By	Mohamad Saab	**Date Created**	11-21-2019
Description	There should be forecasted close price data for future dates (1 or more days past the day the test is run) for each symbol in the tool		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application is opened		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Run project using DataMain in Python 3 Data table has values for symbols and past forecast data		
Postconditions	No duplications of forecast data past or future in database		
Test Steps	1. Open MySQL workbench 2. Browse to File > Open SQL Script > 3. Locate "TC_6.0" in SQL folder in project folders, Open 4. Run Script by pressing Execute (lightning symbol) 5. Note results in execution window		
Expected Result	Table showing: 1+ instrument ID; forecasted prices(avg); next day as a future date (day after testers present day); last day in the future; and 1+ count for future dates		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application **Exit :** Test can be exited at any time during the run or after it finishes		
Test Risks	No risk to project, SQL script is independent of project		
Pass/Fail Criteria	**Passed:** Table showing future dates and forecasted data **Failed:** Table returns empty or with no future dates		
Fail Reasons	1 Not connected to database 2 Database doesn't produce future data		
Priority	**High**	**Priority Reason**	**Duplication of master symbols could duplicate all calculations**
Priority Sequence	**H-2**		

ID	**TC_7.0**		
Test Case Name	Future Forecast should exist for each algorithm		
Created By	Mohamad Saab	**Date Created**	11-21-2019
Description	There should be forecasted close price data for future dates (1 or more days past the day the test is run) for each algorithm in the tool		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application is opened		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Run project using DataMain in Python 3 Database is created 4 Data table has values for symbols and past forecast data		
Postconditions	No duplications of forecast data past or future in database		
Test Steps	6. Open mySQL workbench 7. Browse to File>Open SQL Script> 8. Locate "TC_7.0" in SQL folder in project folders, Open 9. Run Script by pressing Execute (lightning symbol) 10. Note results in execution window		
Expected Result	Table showing: 1+ algorithm name; forecasted prices(avg); next day as a future date (day after testers present day); last day in the future; and 1+ count for future dates		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application **Exit :** Test can be exited at any time during the run or after it finishes		
Test Risks	No risk to project, SQL script is independent of project		
Pass/Fail Criteria	**Passed: Table showing future dates and forecasted data** **Failed: Table returns empty or with no future dates**		
Fail Reasons	1 Not connected to database 2 Database doesn't produce future data		
Priority	**High**	**Priority Reason**	**Duplicate symbols will duplicate all calculations**
Priority Sequence	**H-2**		

ID	**TC_8.0**		
Test Case Name	Past forecast values must exist for last 3 years until today for each symbol		
Created By	Abdul Ahmad	**Date Created**	11-21-2019
Description	Past forecast values must exist for last 3 years until today for each symbol in the database		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 Data table has values		
Postconditions	1 Records exist in the table prior to today's date 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_8.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	5 records with each showing 'date' value of 751 in MySQL WorkBench's Result Grid		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application **Exit :** Test can be exited at any time during the run or after it finishes		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows value of 5 in the 'instid' column and 'date' value of 751 **Failed :** The Result Grid in MySQL WorkBench shows more than 1 record or 'date' value not equal to 751		
Fail Reasons	3 MySQL database table 'dbo_algorithmforecast' has missing records for the past 4 'dbo_algorithmforecast' table is not setup correctly		
Priority	**High**	**Priority Reason**	**Past forecasted values are very important**
Priority Sequence	**H-6**		

ID	**TC_9.0**		
Test Case Name	Past forecast values must exist for last 3 years until today for each algorithm		
Created By	Abdul Ahmad	**Date Created**	11-21-2019
Description	Past forecast values must exist for last 3 years until today for each algorithm in the database		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 Data table has values		
Postconditions	1 Records exist in the table prior to today's date 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_9.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	'algo' value must be 6 and 'date' value should be 751 in MySQL WorkBench's Result Grid		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application **Exit :** Test can be exited at any time during the run or after it finishes		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows value of 6 in the 'algo' column and 'date' value of 751 **Failed :** The Result Grid in MySQL WorkBench shows more than 1 record or 'date' value not equal to 751		
Fail Reasons	1 MySQL database table 'dbo_algorithmforecast' has missing records for the past 2 'dbo_algorithmforecast' table is not setup correctly		
Priority	**High**	**Priority Reason**	**Displaying past forecast values is required**
Priority Sequence	**H-7**		

ID	**TC_10.0**		
Test Case Name	Ten future days from today with no prediction error for each symbol		
Created By	Abdul Ahmad	**Date Created**	11-21-2019
Description	Ten future days from today with no prediction error for each symbol in the database. Future dates have no 'close' value so we can't calculate our Mean Base Error rate.		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 Data table has values		
Postconditions	1 Records exist in the table after today's date 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_10.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	'date' value must be 10 and 'prederror' should be 0 in MySQL WorkBench's Result Grid		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application **Exit :** Test can be exited at any time during the run or after it finishes		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows value of 10 in the 'date' column and 0 in the 'prederror' **Failed :** The Result Grid in MySQL WorkBench shows more than 1 record or 'date' value not equal to 10		
Fail Reasons	1 MySQL database table 'dbo_algorithmforecast' has missing records for the future 2 'dbo_algorithmforecast' table is not setup correctly		
Priority	**Medium**	**Priority Reason**	**Future forecast should not have Mean Base Error**
Priority Sequence	**M-3**		

ID	**TC_11.0**		
Test Case Name	Ten future days from today with no prediction error for each algorithm		
Created By	Abdul Ahmad	**Date Created**	11-22-2019
Description	Ten future days from today with no prediction error for each algorithm in the database		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 Data table has values		
Postconditions	1 Records exist in the table after today's date 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_11.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	'date' value must be 10 and 'prederror' should be 0 in MySQL WorkBench's Result Grid		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application **Exit :** Test can be exited at any time during the run or after it finishes		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows value of 10 in the 'date' column, 5 in the 'algo' column and 0 in the 'prederror' **Failed :** The Result Grid in MySQL WorkBench shows more than 1 record or 'date' value not equal to 10		
Fail Reasons	1 MySQL database table 'dbo_algorithmforecast' has missing records for the future 2 'dbo_algorithmforecast' table is not setup correctly		
Priority	**Medium**	**Priority Reason**	**Algorithm future forecast values should not have Mean Base Error**
Priority Sequence	**M-4**		

ID	**TC_12.0**		
Test Case Name	Nine tables exist in the database		
Created By	Abdul Ahmad	**Date Created**	11-22-2019
Description	Nine tables exist in the database. These are the main required tables.		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_12.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	'cnt' value must be 9 in MySQL WorkBench's Result Grid		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows value of 9 in the 'cnt' column. **Failed :** The Result Grid in MySQL WorkBench shows less or more than 9 in the 'cnt' column		
Fail Reasons	1 MySQL database has missing or extra tables 2 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Tables are required for the application middle-layer and front-end to work**
Priority Sequence	**H-4**		

ID	**TC_13.0**		
Test Case Name	ARIMA has first 9 days blank		
Created By	Abdul Ahmad	**Date Created**	11-22-2019
Description	ARIMA has first 9 days blank in the table		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data tables exists		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_13.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	'forecastcloseprice' value must be blank for first 10 days records in MySQL WorkBench's Result Grid		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows blank values in the first 10 days records in the 'forecastcloseprice' column. **Failed :** The Result Grid in MySQL WorkBench shows no blank values		
Fail Reasons	1 Python script did not pipe proper forecast values into the table 2 Database is not setup correctly 3 'instrumentstatistics' table is pulling less than 3 years of values		
Priority	**High**	**Priority Reason**	**First 9 days are used to feed for the forecast for next day**
Priority Sequence	**H-5**		

ID	**TC_14.0**		
Test Case Name	Random Forest has first 9 days blank		
Created By	Abdul Ahmad	**Date Created**	11-22-2019
Description	Random Forest has first 9 days blank		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data tables exists		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_14.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	'forecastcloseprice' value must be blank for first 10 days records in MySQL WorkBench's Result Grid		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows blank values in the first 10 days records in the 'forecastcloseprice' column. **Failed :** The Result Grid in MySQL WorkBench shows no blank values		
Fail Reasons	3 Python script did not pipe proper forecast values into the table 4 Database is not setup correctly 5 'instrumentstatistics' table is pulling less than 3 years of values		
Priority	**High**	**Priority Reason**	**First 9 days are used to feed for the forecast for next day**
Priority Sequence	**H-6**		

ID	**TC_15.0**		
Test Case Name	No missing values in the 'close' or 'date' column in the 'instrumentstatistics 'table		
Created By	Abdul Ahmad	**Date Created**	11-22-2019
Description	No missing values in the 'close' or 'date' column in the 'instrumentstatistics' table		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 1 Open TC_15.0.sql 2 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	No results should appear in MySQL WorkBench's Result Grid		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows no results, returns nothing. **Failed :** The Result Grid in MySQL WorkBench shows a record		
Fail Reasons	1 YAHOO exchange had issues 2 DataMain did not run properly 3 Python script did not pipe values properly into the table 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Each tuple should have atomic and accurate values**
Priority Sequence	**H-7**		

ID	**TC_16.0**		
Test Case Name	BuySell signals must exist for all records for past 3 years, values of -1,0,1 for each day		
Created By	Abdul Ahmad	**Date Created**	11-22-2019
Description	BuySell signals must exist for all records for past 3 years, values of -1,0,1 for each day.		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_16.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	Results should appear in MySQL WorkBench with Signal count		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the system. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows last 3 years dates with each date and count of signals between 1 and 3. **Failed :** The Result Grid in MySQL WorkBench shows any record where 'signals' column shows values <1 or >3		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 BuySell strategy did not work properly 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**BuySell signals are important to many other calculations**
Priority Sequence	**H-8**		

ID	**TC_17.0**		
Test Case Name	BuySell signals must exist for each strategy		
Created By	Abdul Ahmad	**Date Created**	11-22-2019
Description	BuySell signals must exist for each strategy		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_17.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	Results should appear in MySQL WorkBench with values in 'signals' column		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench shows last 3 years dates with each date and count of signals between 1-3 and 'strcd' value of 5 **Failed :** The Result Grid in MySQL WorkBench shows any record where 'signals' column shows values <1 or >3 or 'strcd' <> 5		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 BuySell strategy did not work properly 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Each strategy implemented must have Buy(1), Sell(-1) or Hold(0)**
Priority Sequence	**H-9**		

ID	**TC_18.0**		
Test Case Name	BuySell signals must exist for each symbol		
Created By	Abdul Ahmad	**Date Created**	11-22-2019
Description	BuySell signals must exist for each symbol		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_18.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	No result should appear in MySQL WorkBench with values		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench return no record **Failed :** The Result Grid in MySQL WorkBench shows a record		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 BuySell strategy did not work properly 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Each symbol must have Buy(1), Sell(-1) or Hold(0) signal**
Priority Sequence	**H-10**		

ID	**TC_19.0**		
Test Case Name	Six algorithm codes exist in 'algorithmmaster' table		
Created By	Abdul Ahmad	**Date Created**	11-22-2019
Description	Six algorithm codes exist in 'algorithmmaster' table		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_19.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	6 rows are returned in MySQL WorkBench with algorithm codes and names		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns 6 records **Failed :** The Result Grid in MySQL WorkBench shows <> 6 records		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 BuySell strategy did not work properly 4 Database is not setup correctly		
Priority	**Medium**	**Priority Reason**	**Algorithmcodes are picked up and piped to this table**
Priority Sequence	**M-5**		

ID	**TC_20.0**		
Test Case Name	'Instrumentmaster' must have only 5 stock symbols		
Created By	Abdul Ahmad	**Date Created**	11-22-2019
Description	'Instrumentmaster' must have only 5 stock symbols		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_20.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	5 rows are returned in MySQL WorkBench with instrument values		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns 5 records **Failed :** The Result Grid in MySQL WorkBench shows <> 6 records		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 BuySell strategy did not work properly 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Unique and required stock symbols are used in all calculations and measurements**
Priority Sequence	**H-11**		

ID	**TC_21.0**		
Test Case Name	'Instrumentstats' column named 'high' has values >= 'low' , 'open', 'close' columns on all records		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	'Instrumentstats' column named 'high' has values >= 'low' , 'open', 'close' columns on all records		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_21.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	No results are returned in MySQL WorkBench with instrument values		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns no records **Failed :** The Result Grid in MySQL WorkBench shows 1 or more records		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 YAHOO exchange encountered an issue 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Valid base statistics are vital to all the calculations**
Priority Sequence	**H-11**		

ID	**TC_22.0**		
Test Case Name	Instrumentstats column names 'low' has values <= high, open, close columns on all records		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Instrumentstats column names 'low' has values <= high, open, close columns on all records		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_22.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	No results are returned in MySQL WorkBench with instrument values		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns no records **Failed :** The Result Grid in MySQL WorkBench shows 1 or more records. Sometimes due to decimal point a value may appear in the results of this test to be incorrect but the analyst can visually inspect it to check for validity.		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 YAHOO exchange encountered an issue 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Valid base statistics are vital to all the calculations**
Priority Sequence	**H-13**		

ID	**TC_23.0**		
Test Case Name	Instrumentstats column 'volume' values should always be >=0		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Instrumentstats column 'volume' values should always be >=0, volume is never negative in stock trading.		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_23.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	No results are returned in MySQL WorkBench with instrument values		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns no records **Failed :** The Result Grid in MySQL WorkBench shows 1 or more records.		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 YAHOO exchange encountered an issue 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Valid base statistics are vital to all the calculations**
Priority Sequence	**H-14**		

ID	**TC_24.0**		
Test Case Name	Instrumentstats must have 3 years of data till today, going back 3 years from today.		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Instrumentstats must have 3 years of data till today, going back 3 years from today		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_24.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a PC		
Expected Result	'startdate' must return a date that is 3 years ago from today and 'enddate' must return a date that is today/last business day if you are running it on the weekend in MySQL WorkBench with instrument values		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the system. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns minimum and maximum dates that comply with 3 year range **Failed :** The Result Grid in MySQL WorkBench shows values that are either over the 3 year limit or under		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 YAHOO exchange encountered an issue 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**3 years data is a requirement**
Priority Sequence	**H-15**		

ID	**TC_25.0**		
Test Case Name	Datedim must contain 50 future days from today		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Datedim must contain 50 future days from today		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_25.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a PC		
Expected Result	'daysdiff' must return a value of 50 in MySQL WorkBench		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the system. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns a single row with 'daysdiff' column containing value of 50 **Failed :** The Result Grid in MySQL WorkBench return a value other than 50 in 'daysdiff' column		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 DateFetch script function get_calendar() had issues 4 DateDim table is not setup properly 5 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Future dates are important to show forecasts and predictions**
Priority Sequence	**H-16**		

ID	**TC_26.0**		
Test Case Name	Datedim must contain last 3 years of dates, up till today		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Datedim must contain last 3 years of dates, up till today		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_26.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a PC		
Expected Result	'daysdiff' must return a value of 1096 in MySQL WorkBench		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns a single row with 'daysdiff' column containing value of 1096 **Failed :** The Result Grid in MySQL WorkBench return a value other than 1096 in 'daysdiff' column		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 DateFetch script function get_calendar() had issues 4 DateDim table is not setup properly 5 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Past 3 years date values are required**
Priority Sequence	**H-17**		

ID	**TC_27.0**		
Test Case Name	No duplicate dates or tuples in datedim table		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	No duplicate dates or tuples in datedim table		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_27.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a PC		
Expected Result	'cnt' column must return no value in MySQL WorkBench. There should be no record returned.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns no value or record **Failed :** The Result Grid in MySQL WorkBench return a value or record		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 DateFetch script function get_calendar() had issues 4 DateDim table is not setup properly 5 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Duplicate values can lead to inflated calculations and affect front-end**
Priority Sequence	**H-18**		

ID	**TC_28.0**		
Test Case Name	In Statisticalreturns table, cash on hand value must be >=0 for all records		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Statisticalreturns table, cash on hand value must be >=0 for all records		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_28.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	No value should be returned in the 'cashonhand' column in MySQL WorkBench. There should be no record returned.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns no value or record **Failed :** The Result Grid in MySQL WorkBench return a value or record		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 TradingSimulator.py had issues 4 Database is not setup correctly		
Priority	**Medium**	**Priority Reason**	**Cash On Hand is used behind the scenes in the trading portfolio**
Priority Sequence	**M-6**		

ID	**TC_29.0**		
Test Case Name	In Statisticalreturns table, 'portfoliovalue' data values must be >= 'cash on hand' values for all records		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	In Statisticalreturns table, 'portfoliovalue' data values must be >= 'cash on hand' values for all records		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists and all the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_29.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	No value should be returned in the MySQL WorkBench. There should be no record returned.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the system. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns no value or record **Failed :** The Result Grid in MySQL WorkBench return a value or record		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 TradingSimulator.py had issues 4 Database is not setup correctly		
Priority	**Medium**	**Priority Reason**	**'portfoliovalue' is always >= 'cash on hand' because it's the total of starter cash + (gain – loss)**
Priority Sequence	**M-7**		

ID	**TC_30.0**		
Test Case Name	In Statisticalreturns table, 'positionsize' column must be an integer data type		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	In Statisticalreturns table, 'positionsize' column must be an integer data type		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_30.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	'DATA_TYPE' column must show a value of 'int' in the MySQL WorkBench.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns single record and column 'DATA_TYPE' contains a value of 'int' **Failed :** The Result Grid in MySQL WorkBench shows 'DATA_TYPE' value other than 'int'		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 TradingSimulator.py had issues 4 Database is not setup correctly		
Priority	**Medium**	**Priority Reason**	**Shares are not bought or sold in fractions, position size represents number of shares**
Priority Sequence	**M-8**		

ID	**TC_31.0**		
Test Case Name	Statisticalreturns table must contain 7 strategies		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Statisticalreturns table must contain 7 strategies. These are strategy codes use for different buysell trading simulations.		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_31.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	'stratcode' column must show a value of 7 in the MySQL WorkBench.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns single record and column 'stratcode' contains a value of 7 **Failed :** The Result Grid in MySQL WorkBench shows 'stratcode' value other than 7		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 TradingSimulator.py had issues 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**All strategies should be computed**
Priority Sequence	**H-19**		

ID	**TC_32.0**		
Test Case Name	Fibonacci lines values are in order, 'highfrllinelong' values are >= 'lowfrllinelong' values for all records		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Fibonacci lines values are in order, 'highfrllinelong' values are >= 'lowfrllinelong' values for all records		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_32.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	No records should be returned in the MySQL WorkBench.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns no results **Failed :** The Result Grid in MySQL WorkBench returns 1 or more results		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 EngineeredFeatures.py had issues 4 Database is not setup correctly		
Priority	**Medium**	**Priority Reason**	**FRL strategy and technical indicators view will be affected**
Priority Sequence	**M-9**		

ID	**TC_33.0**		
Test Case Name	Bollinger Bands values are in order, boll_ub_v > boll_lb_v values for all records		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Bollinger Bands values are in order, boll_ub_v > boll_lb_v values for all records		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	3 Tables stay intact in the database Database stays intact		
Test Steps	5 Open MySQL WorkBench 6 Browse to Unit Test folder 7 Open TC_33.0.sql 8 Execute the script or press Ctrl+Shift+Enter key on a Windows machine		
Expected Result	No records should be returned in the MySQL WorkBench.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns no results **Failed :** The Result Grid in MySQL WorkBench returns 1 or more results		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 EngineeredFeatures.py had issues 4 Database is not setup correctly		
Priority	**Medium**	**Priority Reason**	**Technical Indicators view will be impacted**
Priority Sequence	**M-10**		

ID	**TC_34.0**		
Test Case Name	Each date must have a 0 , 1, -1 signal for each strategy and symbol in 'actionsignals' table		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Each date must have a 0 , 1, -1 signal for each strategy and symbol in 'actionsignals' table		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_34.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a PC		
Expected Result	'cnt' values for 'SubTotal of Records Grouped by Signal Above' row must be equal to ' Total Records in 'actionsignals' Table row in the MySQL WorkBench.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns 5 records with last two records showing same value for 'cnt' column **Failed :** The Result Grid in MySQL WorkBench returns different values for last two rows in 'cnt' column		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 BuySell.py had issues 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**BuySell signal to indicate Buy(1), Sell(-1) or Hold(0)**
Priority Sequence	**H-20**		

ID	**TC_35.0**		
Test Case Name	In Statisticalreturns table 'cash on hand' values must be between 0 and 'portfoliovalue', it should never exceed the 'portfolio' value		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	In Statisticalreturns table 'cash on hand' values must be between 0 and 'portfoliovalue', it should never exceed the 'portfolio' value		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_35.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a PC		
Expected Result	'mincoh' value must be >0 and 'maxcoh' must be < 'maxportfolio' in the MySQL WorkBench.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the system. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns single record with minimum 'cashonhand' > 0 and maximum 'cashonhand' must be less than 'maxportfolio' value. **Failed :** The Result Grid in MySQL WorkBench returns values that are greater than 'maxportfolio' compared to 'maxcoh'		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 TradingSimulator.py had issues 4 Database is not setup correctly		
Priority	**High**	**Priority Reason**	**Important for buy/selling trades**
Priority Sequence	**H-21**		

ID	**TC_36.0**		
Test Case Name	BuyHold strategy must have no signals, this strategy should only hold the stock and not sell it or buy more		
Created By	Aalem Singh	**Date Created**	11-23-2019
Description	BuyHold strategy must have no signals, this strategy should only hold the stock and not sell it or buy more		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_36.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a PC		
Expected Result	No value or record should be returned in the MySQL WorkBench.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns no values/records **Failed :** The Result Grid in MySQL WorkBench returns a value or a record		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 BuySell.py had issues 4 Database is not setup correctly		
Priority	**Medium**	**Priority Reason**	**BuyHold strategy is not supposed to do trades**
Priority Sequence	**M-11**		

ID	**TC_37.0**		
Test Case Name	In 'actionsignals' table frl, cma, ema, macd signals sum must be between -5 and 5		
Created By	Aalem Singh	**Date Created**	11-23-2019
Description	In 'actionsignals' table frl, cma, ema, macd signals sum must be between -5 and 5		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Data table exists 3 All the fields in the table contain values		
Postconditions	1 Tables stay intact in the database 2 Database stays intact		
Test Steps	1 Open MySQL WorkBench 2 Browse to Unit Test folder 3 Open TC_37.0.sql 4 Execute the script or press Ctrl+Shift+Enter key on a PC		
Expected Result	No value or record should be returned in the MySQL WorkBench.		
Test Entry/Exit Criteria	**Entry :** Test can be started at any time during testing phase or later when the application is deployed. Test itself has no direct impact on the working of the application. **Exit :** Test can be exited at any time during the run or after it finishes.		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database. Extra or less values could cause issues that will need to be corrected by a developer.		
Pass/Fail Criteria	**Passed :** The Result Grid in MySQL WorkBench returns no values/records **Failed :** The Result Grid in MySQL WorkBench returns a value or a record		
Fail Reasons	1 DataMain did not run properly 2 Python script did not pipe values properly into the table 3 BuySell.py had issues 4 Database is not setup correctly		
Priority	**Medium**	**Priority Reason**	**These strategies signals are combined into our custom algorithm for trading**
Priority Sequence	**M-12**		

ID	**TC_38.0**		
Test Case Name	getdatasources() function returns correct number of symbols		
Created By	John Gettel	**Date Created**	11/24/2019
Description	getdatasources() function returns correct number of symbols.		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database properly 3 Database will be loaded with accurate records		
Test Steps	1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_38.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “incorrect number of symbols:” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “incorrect number of symbols:”		
Fail Reasons	4 No symbols entered into database 5 Incorrect symbols entered into database		
Priority	**High**	**Priority Reason**	**Need correct symbols for application to run properly**
Priority Sequence	**H-22**		

ID	**TC_39.0**		
Test Case Name	getdatasources() function returns correct ticker symbols		
Created By	John Gettel	**Date Created**	11/24/2019
Description	getdatasources() function returns correct ticker symbols.		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database properly 3 Database will be loaded with accurate records		
Test Steps	1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_39.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “missing ticker:” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “missing ticker:”		
Fail Reasons	1 Incorrect symbols entered into database		
Priority	**High**	**Priority Reason**	**Need correct symbols for application to run properly**
Priority Sequence	**H-23**		

ID	**TC_40.0**		
Test Case Name	get_data () function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	get_data () function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly 4 Internet connection is working properly		
Postconditions	1 Test is completed successfully 2 Data is retrieved from data source 3 Middle layer of Python will push data to the MySQL database 4 Database will be loaded with accurate records		
Test Steps	1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_40.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: "Incorrect return value while running get_data():" or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is "Incorrect return value while running get_data():"		
Fail Reasons	1 MySQL database is not function properly 2 Internet connection not functioning properly		
Priority	**High**	**Priority Reason**	**Need to retrieve data from external data source**
Priority Sequence	**H-24**		

ID	**TC_41.0**		
Test Case Name	get_calendar() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	get_calendar() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly 4 Internet connection is working properly		
Postconditions	1 Test is completed successfully 2 Data is retrieved from data source 3 Middle layer of Python will push data to the MySQL database 4 Database will be loaded with accurate records		
Test Steps	1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_41.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: "Incorrect return value while running get_calendar():" or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is "Incorrect return value while running get_calendar():"		
Fail Reasons	1 MySQL database is not function properly 2 Internet connection not functioning properly		
Priority	**Medium**	**Priority Reason**	**Need to retrieve accurate dates to map trading calendar**
Priority Sequence	**M-13**		

ID	**TC_42.0**		
Test Case Name	calculate() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	calculate() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_42.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “Incorrect return value while running calculate():” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “Incorrect return value while running calculate():”		
Fail Reasons	1 MySQL database is not function properly		
Priority	**High**	**Priority Reason**	**Need to add technical indicator calculations to database for later calculations**
Priority Sequence	**H-25**		

ID	**TC_43.0**		
Test Case Name	calculate_forecast() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	calculate_forecast() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_43.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “Incorrect return value while running calculate_forecast():” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “Incorrect return value while running calculate_forecast():”		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Calculation for future price forecast**
Priority Sequence	**M-14**		

ID	**TC_44.0**		
Test Case Name	calculate_arima_forecast() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	calculate_arima_forecast() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_44.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: "Incorrect return value while running calculate_arima_forecast():" or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is "Incorrect return value while running calculate_arima_forecast():"		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Calculation for arima forecast**
Priority Sequence	**M-15**		

ID	**TC_45.0**		
Test Case Name	calculate_random_forest_forecast() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	calculate_random_forest_forecast() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_45.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: "Incorrect return value while running calculate_random_forest_forecast():" or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is "Incorrect return value while running calculate_random_forest_forecast():"		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Calculation for random forest forecast**
Priority Sequence	**M-16**		

ID	**TC_46.0**		
Test Case Name	calculate_svm_forecast() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	calculate_svm_forecast() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_46.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: "Incorrect return value while running calculate_svm_forecast():" or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is "Incorrect return value while running calculate_svm_forecast():"		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Calculation for support vector machine forecast**
Priority Sequence	**M-17**		

ID	**TC_47.0**		
Test Case Name	calculate_ forecast_old() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	calculate_forecast_old() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_47.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: "Incorrect return value while running calculate_forecast_old():" or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is "Incorrect return value while running calculate_ forecast_old():"		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Calculation for old forecast model**
Priority Sequence	**M-18**		

ID	**TC_48.0**		
Test Case Name	calculate_xgboost_forecast() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	calculate_xgboost_forecast() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_48.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: "Incorrect return value while running calculate_xgboost_forecast():" or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is "Incorrect return value while running calculate_xgboost_forecast():"		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Calculation for gradient boost forecast**
Priority Sequence	**M-19**		

ID	**TC_49.0**		
Test Case Name	cma_signal() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	cma_signal() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_49.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “Incorrect return value while running cma_signal():” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “Incorrect return value while running cma_signal():”		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**CMA signal generation**
Priority Sequence	**M-20**		

ID	**TC_50.0**		
Test Case Name	frl_signal() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	frl_signal() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_50.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “Incorrect return value while running frl_signal():” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “Incorrect return value while running frl_signal():”		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**FRL signal generation**
Priority Sequence	**M-21**		

ID	**TC_51.0**		
Test Case Name	ema_signal() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	ema_signal() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_51.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “Incorrect return value while running ema_signal():” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “Incorrect return value while running ema_signal():”		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**EMA signal generation**
Priority Sequence	**M-22**		

ID	**TC_52.0**		
Test Case Name	macd_signal() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	macd_signal() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_52.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “Incorrect return value while running macd_signal():” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “Incorrect return value while running macd_signal():”		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Moving average convergence/divergence signal generation**
Priority Sequence	**M-23**		

ID	**TC_53.0**		
Test Case Name	algo_signal() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	algo_signal() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_53.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “Incorrect return value while running algo_signal():” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “Incorrect return value while running algo_signal():”		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Algorithm forecast-base signal generation**
Priority Sequence	**M-24**		

ID	**TC_54.0**		
Test Case Name	trade_sim() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	trade_sim() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_54.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “Incorrect return value while running trade_sim():” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “Incorrect return value while running trade_sim():”		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Individual trade strategy simulation**
Priority Sequence	**M-25**		

ID	**TC_55.0**		
Test Case Name	comb_sim() function runs successfully		
Created By	Abdul Ahmad; John Gettel;	**Date Created**	11/24/2019
Description	comb_sim() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_55.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: “Incorrect return value while running comb_sim():” or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is “Incorrect return value while running comb_sim():”		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Combination trade strategy simulation**
Priority Sequence	**M-26**		

ID	**TC_56.0**		
Test Case Name	buy_hold_sim() function runs successfully		
Created By	John Gettel	**Date Created**	11/24/2019
Description	buy_hold_sim() function runs successfully		
Tester	Data Analyst or Developer		
Test Frequency	During the testing phase or anytime in the future, there are no restrictions		
Executed By	Application User, Developer or anyone performing QA Testing		
Preconditions	1 PyCharm is open 2 Project is open in PyCharm 3 MySQL is working properly		
Postconditions	1 Test is completed successfully 2 Middle layer of Python will push data to the MySQL database 3 Database will be loaded with accurate records		
Test Steps	Following are the steps 1 Open PyCharm 2 Navigate to Fistertab → F2019 → Unit Tests 3 Open TC_56.0.py 4 Click on Run menu → Run button 5 The test is executed		
Expected Result	Tester should receive the following message: "Incorrect return value while running buy_hold_sim ():" or no message at all		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes or it can be terminated during the run		
Test Risks	No risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** If no message **Failed:** If the message is "Incorrect return value while running buy_hold_sim():"		
Fail Reasons	1 MySQL database is not function properly		
Priority	**Medium**	**Priority Reason**	**Buy and hold trade strategy simulation**
Priority Sequence	**M-27**		

3 NON-FUNCTIONAL TESTING

3.1 NON-FUNCTIONAL TESTING APPROACH

Non-functional testing is done to verify the non-functional requirements of the application such as performance, usability, interface etc. All tests will be performed in Tableau. It is the front-end application platform to be used by the financial analyst/user. Tester will be performing all the test in Tableau while maintaining the connectivity to the locally installed MySQL database. Tester will have to setup a local instance of the MySQL database with **username** = root and **password** = password. This username and password could be different for each installation on each machine. In PyCharm **'dbEngine.py'** will have to be configured accordingly. Testing should not begin before the aforementioned configurations are setup in each software.

3.2 NON-FUNCTIONAL TESTING PASS/FAIL CRITERIA

Each test must produce the desired outcome or perform according to requirements to qualify for passed status. Testing is done in order to verify that in case of a failure the system is capable enough to keep functioning. If a test fails due to Tableau functionality issues, it is out of the scope of the project for the tester or developers to fix it as Tableau is a third-party proprietary software. Tester can notify the developers and client. They can report issue to Tableau vendor via email or online community forums.

3.3 NON-FUNCTIONAL TESTING RISKS

There is no direct risk to any portions of the application during any non-functional tests. In our application these tests are all performed on Tableau which is a third-party software. All its functionality is out of the box. The only serious consequence could be that the application might close due to memory or other local hardware issues. There are no tests that insert or delete data from the database so all calculations will stay intact after a test is complete.

3.4 NON-FUNCTIONAL TESTING EXPECTED RESULTS

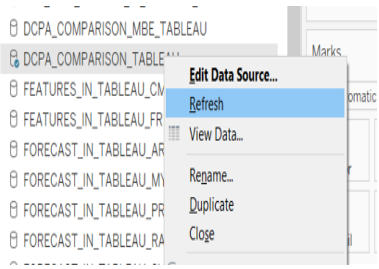
Each test must perform according to the expectation otherwise the feature or function will be classified as failed.

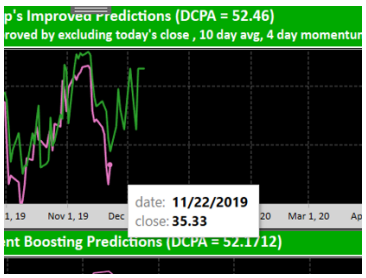
3.5 NON-FUNCTIONAL TESTING PRIORITY/PRIORITY REASON

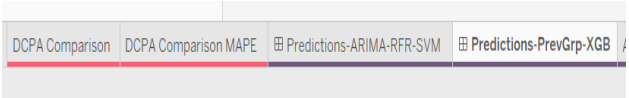
Each test is prioritized according to the feature and use cases.

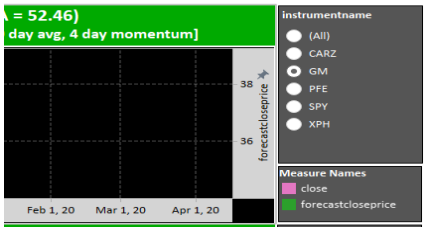
3.6 NON-FUNCTIONAL TEST CASES

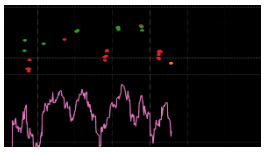
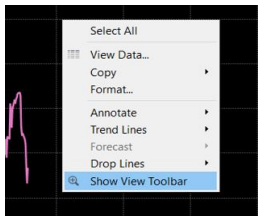
ID	**NFTC_1.0**		
Test Case Name	Tableau - Enter User Id and Password, configured locally for MySQL database named 'gmfsp_db'		
Created By	Abdul Ahmad	**Date Created**	11-23-2019
Description	Tableau - Enter User Id and Password, configured locally for MySQL database named 'gmfsp_db'		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Database and Tableau must be setup properly 3 Required drivers are installed properly		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 You should see a window like the one on the right 4 Enter **Username** = root **Password** = password 5 Or enter any password setup during MySQL database instance setup 		
Expected Result	Tableau has established connection with the locally setup database		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB 'gmfsp_db' **Failed :** Tableau is not able to connect to the database		
Fail Reasons	MySQL database is not setup properly		
Priority	**High**	**Priority Reason**	**Application will not work without a database connection**
Priority Sequence	**H-1**		

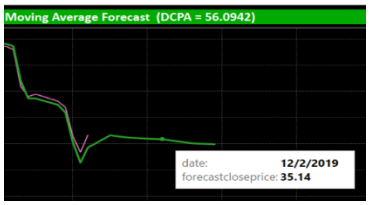
ID	**NFTC_2.0**		
Test Case Name	Tableau - data is refreshed, check any one data-model		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - data is refreshed, check any one data-model		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly 4 Required drivers are installed properly		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the **'Data Source'** tab to the very left of the screen 4 Right click on the data source with a little blue check at the bottom 5 Press **'Refresh'** 		
Expected Result	Data Source is refreshed without any error		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB **'gmfsp_db'** **Failed :** Tableau is not able to connect to the database or refresh		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source		
Priority	**High**	**Priority Reason**	**No related features will work without data source refresh.**
Priority Sequence	**H-2**		

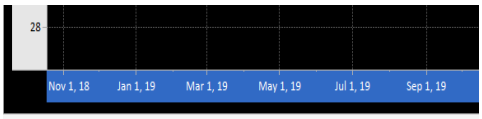
ID	**NFTC_3.0**		
Test Case Name	Tableau - latest data for the base statistics is updated		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - latest data for the base statistics is updated		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly 4 Required drivers are installed properly		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the tab named **'Predictions-PrevGrp-XGB'** 4 Hover over any Pink line graph to the point on the most right as shown on the right 5 A tooltip should pop up 		
Expected Result	Tooltip shows the most recent expected date, weekends and holidays have no data in this application.		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB **'gmfsp_db'** and shows updated data **Failed :** Tableau is not able to connect to the database or pull latest data		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 SQL Scripts for this tab has issues, multiple scripts 4 Python scripts related to this data did not run properly		
Priority	**High**	**Priority Reason**	**Latest data and forecasts requried**
Priority Sequence	**H-3**		

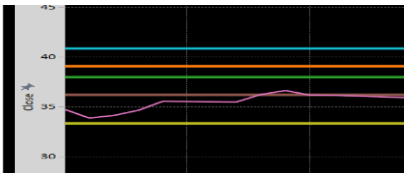
ID	**NFTC_4.0**		
Test Case Name	Tableau - User can view all tabs		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - User can view all tabs		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly 4 Required drivers are installed properly		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Look at the bottom of the application  4 There should be tabs present with different names		
Expected Result	Tabs available in the application, tester should be able to click on each tab.		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB 'gmfsp_db' and shows tabs without error **Failed :** Tableau is not able to connect to the database and show tabs		
Fail Reasons	1 MySQL database is not setup properly 3 Tableau data source is not setup properly, recreated the data source 2 Tableau has encountered an issue(we are limited in correcting Tableau related technical issues as it's a third-party software)		
Priority	**High**	**Priority Reason**	**Tabs house the graphs and measures**
Priority Sequence	**H-4**		

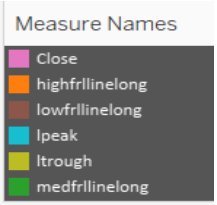
ID	**NFTC_5.0**		
Test Case Name	Tableau - Toggle radio buttons are working, mostly available on the right		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - Toggle radio buttons are working, mostly available on the right		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly and required drivers are installed		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the tab named **'Predictions-PrevGrp-XGB'** 4 On the right, radio buttons are visible 5 Make a different selection other than the default selected option 		
Expected Result	Toggle buttons are filtering data as per design. The graph should change when the tester toggles the button.		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB 'gmfsp_db' and shows tabs without error **Failed :** Tableau is not able to connect to the DB and does not show tabs		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue(we are limited in correcting it) 4 SQL scripts to build this tab has issues, multiple scripts on this tab 5 Filters setting in Tableaus is not setup properly		
Priority	**High**	**Priority Reason**	**Ability to view different datasets and graphs**
Priority Sequence	**H-5**		

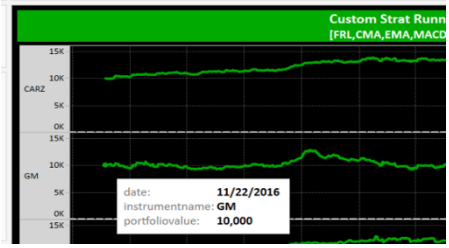
ID	**NFTC_6.0**		
Test Case Name	Tableau - Zoom feature is working on all applicable line graphs		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - Zoom feature is working on all applicable line graphs		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly and required drivers are installed		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on '**BuySellCMA'** tab 4 Bring the cursor to the dark area 5 Using a touchpad if available use two fingers to make a zoom in or zoom-out motion, similar to like a smartphone 6 Or right click in the dark area, a menu will pop-up, click Show View Toolbar 7 A vertical bar like the one shown here should appear 8 Press the '+' or '-' buttons to zoom-in or zoom-out   		
Expected Result	Zoomed in or out of the graph or object		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application.		
Pass/Fail Criteria	**Passed :** Tableau is successfully able to zoom-in or out of the graphs **Failed :** Tableau issue or data not loaded properly to create the object		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau has encountered an issue(we are limited in correcting Tableau related technical issues as it's a third-party software) 3 **'BUY_SELL_SIGNALS_IN_TABLEAU_CMA.sql'** script has issues 4 **'BuySell.py'** Python script has issues		
Priority	**Medium**	**Priority Reason**	**zoom-in or out**
Priority Sequence	**M-1**		

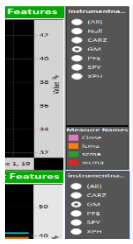
ID	**NFTC_7.0**		
Test Case Name	Tableau - Forecast line graph shows future dates		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - Forecast line graph shows future dates		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly and required drivers are installed		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the **'ARIMA'** tab 4 Hover the mouse pointer over the 'Green' line where there is no 'Pink' line 		
Expected Result	When the mouse hovers over the Green line future dates should be visible		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order on any tab that shows predictions based on an algorithm **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB 'gmfsp_db' and future forecast graph and data is configured properly **Failed :** Tableau is not able to show future forecast		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue(we are limited in correcting Tableau related technical issues as it's a third-party software) 4 **'FORECAST_IN_TABLEAU_ARIMA_ONLY.sql'** script has issues 5 Python script **'DataForecast.py'** has issues 6 **'DateDim'** table has issues in MySQL **'gmfsp_db'** database		
Priority	**Medium**	**Priority Reason**	**Forecast values must be visible**
Priority Sequence	**M-2**		

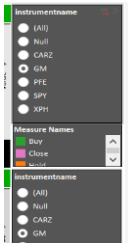
ID	**NFTC_8.0**		
Test Case Name	Tableau - Date is always on x-axis on all graphs		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - Date is always on x-axis on all graphs		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly and required drivers are installed		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on **'BuySellCustom'** tab 4 Look at the bottom of the graph 5 The date values should be aligned horizontally with the object 		
Expected Result	Date values show on the horizontal x-axis		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order on any tab that shows date values **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB 'gmfsp_db' and date values are displayed on the horizontal x-axis **Failed :** Tableau is not able to connect to the database and show date values on the x-axis		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue(we are limited in correcting Tableau related technical issues as it's a third-party software) 4 **'BUY_SELL_SIGNALS_IN_TABLEAU_ALGO.sql'** script to has issues 5 **'BuySell.py'** Python script related to this tab has issues 6 **'DateDim'** table has issues in MySQL **'gmfsp_db'** database		
Priority	**High**	**Priority Reason**	**Dates must be shown on x-axis**
Priority Sequence	**H-6**		

ID	**NFTC_9.0**		
Test Case Name	Tableau - Close price is always on the y-axis on the left		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - Close price is always on the y-axis on the left		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly and required drivers are installed		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the **'FRLFeatures'** tab 4 Look for the 'Close' values on the left side y-axis of the graph 		
Expected Result	'Close' values show on the left vertical y-axis		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order on any tab showing 'Close' value **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB 'gmfsp_db' and shows 'Close' values on the left vertical y-axis. **Failed :** Tableau is not able to connect to the database and show 'Close' values on the left y-axis.		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue(we are limited in correcting Tableau related technical issues as it's a third-party software) 4 'FEATURES_IN_TABLEAU_FRL.sql' script to build this tab has issues 5 'DataFetch.py' Python script has issues 6 YAHOO data exchange had issues 7 'instrumentstatistics' table has issues in MySQL 'gmfsp_db' database		
Priority	**High**	**Priority Reason**	**Close price is vital to all visuals**
Priority Sequence	**H-7**		

ID	**NFTC_10.0**		
Test Case Name	Tableau - Measure names legends must show on each graph		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - Measure names legends must show on each graph		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly and required drivers are installed		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the **'FRLFeatures'** tab 4 On the right side of the screen, look for a box labeled **'Measure Names'** 		
Expected Result	Measure Names, which are legend codes show next to each graph on the right-hand side		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order on any tab **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB **'gmfsp_db'** and line graphs for different values are built accurately **Failed :** Tableau is not able to connect to the database and show graph measure legend names		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue(we are limited in correcting Tableau related technical issues as it's a third-party software) 4 **'FEATURES_IN_TABLEAU_FRL.sql'** script to build this tab has issues 5 Python script **'EngineeredFeatures.py'** has issues		
Priority	**High**	**Priority Reason**	**Legends are important for all the graphs**
Priority Sequence	**H-8**		

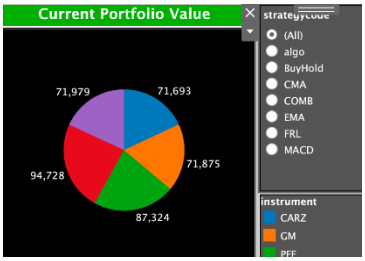
ID	**NFTC_11.0**		
Test Case Name	Tableau - All portofolio values graphs must start with $10000		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - All portofolio values graphs must start with $10000		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly and required drivers are installed		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the tab **'PortfolioCOMB'** 4 Hover the mouse pointer to the far left of any of the **'Green'** lines 5 A tooltip with 3 values will appear 		
Expected Result	The tooltip must show a value of **10,000** in 'portfoliovalue' data item		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order on any of the Portfolios **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB **'gmfsp_db'** and line graphs for **'portfoliovalue'** **Failed :** Tableau is not able to connect to the database and pull correct **'portfoliovalue'** from 3 years ago		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue(we are limited in correcting Tableau related technical issues as it's a third-party software) 4 **'PORTFOLIO_VALUE_IN_TABLEAU_COMB.sql'** script has issues 5 **'TradingSimulator.py'** Python script has issues 6 **'DateDim'** table in the **'gmfsp_db'** MySQL instance has issues		
Priority	**High**	**Priority Reason**	**Portfolio's initial value is $10,000**
Priority Sequence	**H-9**		

ID	**NFTC_12.0**		
Test Case Name	Tableau - CMA-FRL-Signals tab has multiple instrument name radio toggle buttons functional		
Created By	Abdul Ahmad	**Date Created**	11-24-2019
Description	Tableau - CMA-FRL-Signals tab has multiple instrument name radio toggle buttons functional. This is a dashboard tab not a single data tab.		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly and database is configured correctly		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on **'CMA-FRL-Signals'** tab 4 Look at the right or left toggle button panels 5 Separate toggle radio button stacks should appear(4 in total) 6 Click on different options to change graphs 		
Expected Result	Graph values must change if the tester changes radio button selections		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB **'gmfsp_db'** and Tableau dashboard page is built correctly **Failed :** Tableau not able to connect to the DB and show radio buttons		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue(we are limited in correcting Tableau) 4 Issues with multiple SQL scripts that are used to built this page 5 Issues with multiple Python scripts that are used for different calculations combined on this page		
Priority	**Medium**	**Priority Reason**	**Radio buttons for each graph**
Priority Sequence	**M-3**		

ID	**NFTC_13.0**		
Test Case Name	Tableau - EMA-MACD-CUSTOM-Signals tab has multiple instrument name radio toggle buttons functional		
Created By	Mohamad Saab	**Date Created**	11-24-2019
Description	Tableau - EMA-MACD-CUSTOM-Signals tab has multiple instrument name radio toggle buttons functional. This tab is a dashboard, not a single data tab.		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly and required drivers are installed		
Postconditions	1 Tableau will be connected to the database 2 Tableau will be able to pull all data and show the updated tabs 3 Application will be functional		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on **'EMA-MACD-CUSTOM-Signals'** tab 4 Look at the right side for toggle button panels 5 Separate toggle radio button stacks should appear(3 in total) 6 Click on different options to change graphs 		
Expected Result	Graph values must change if the tester changes radio button selections		
Test Entry/Exit Criteria	**Entry :** Test can be executed at any time in any order on this tab **Exit :** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed :** Tableau is successfully connected to MySQL local DB **'gmfsp_db'** and the Tableau tab is configured accurately. **Failed :** Tableau is not able to connect to the database and show graph		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue(we are limited in correcting Tableau) 4 Multiple SQL scripts that built this view have issues 5 **'BuySell.py'** Python script has issues 6 **'DataForecast.py'** function **'calculate_arima_forecast()'** has issues		
Priority	**Medium**	**Priority Reason**	**Dynamic graphs**
Priority Sequence	**M-4**		

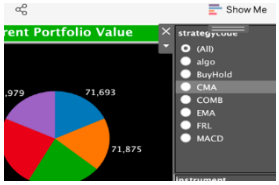
ID	**NFTC_14.0**		
Test Case Name	Tableau - Portfolio-CMA_FRL-EMA tab show data grid with values in each array, intersection		
Created By	John Gettel	**Date Created**	11-24-2019
Description	Tableau - Portfolio-CMA_FRL-EMA tab show data grid with values in each array, intersection		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly with required drivers installed		
Postconditions	4 The user will be able to view portfolio returns for each strategy and each symbol		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the tab named **'Portfolio-CMA_FLR-EMA'** 		
Expected Result	A grid is displayed that gives the total returns for each strategy and each ticker symbol		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** Tableau is successfully connected to MySQL local database 'gmfsp_db' and data on tab changes appropriately **Failed:** Tableau is not able to connect to the database, incomplete data, or no data at all is displayed		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue(we are limited in correcting Tableau related technical issues as it's a third-party software) 4 SQL Script to build this tab has issue		
Priority	**High**	**Priority Reason**	**Viewing portfolio returns is required**
Priority Sequence	**H-10**		

	Strategy						
	algo	BuyHold	CMA	COMB	EMA	FRL	MACD
CARZ	312	-97	590	105	486	428	-768
GM	1,385	-842	-205	-961	-471	-28	-1,686
PFE	-821	-922	-1,328	-804	-621	-62	572
SPY	1,299	777	620	1,043	912	184	645
XPH	152	-43	249	15	260	909	189

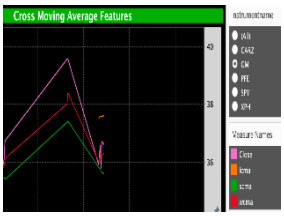
ID	**NFTC_15.0**		
Test Case Name	Tableau – Portfolio-CMA_FRL-EMA tab has instrument name radio toggle buttons functional		
Created By	John Gettel	**Date Created**	11-24-2019
Description	Tableau – Portfolio-CMA_FRL-EMA tab has instrument name radio toggle buttons functional		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly with required drivers installed		
Postconditions	1 User will be able to switch view between each instrument tracked by the application		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the tab named **'Portfolio-CMA_FRL-EMA'** 4 On the right, there should be list of radio buttons 5 Make a different selection other than the default selected option 		
Expected Result	Toggle buttons are filtering data as per design. The graph should change when the tester toggles the button.		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** Tableau is successfully connected to MySQL local database 'gmfsp_db' and data on tab changes appropriately **Failed:** Tableau is not able to connect to the database or no changes occur on tab		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue (we are limited in correcting Tableau) 4 SQL Script to build this tab has issue 5 Filters setting in Tableaus is not setup properly		
Priority	**High**	**Priority Reason**	**Viewing data for symbols is required**
Priority Sequence	**H-11**		

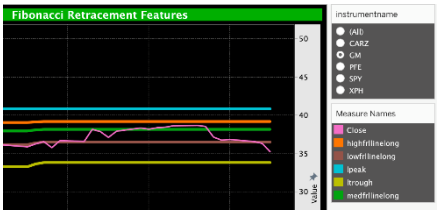
ID	**NFTC_16.0**		
Test Case Name	Tableau - Portfolio-MACD-COMB-BuyHold tab show data grid with values in each array, intersection		
Created By	John Gettel	**Date Created**	11-24-2019
Description	Tableau - Portfolio-MACD-COMB-BuyHold tab show data grid with values in each array, intersection		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly with required drivers installed		
Postconditions	The user will be able to view portfolio returns for each strategy and each symbol		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the tab named **'Portfolio-MACD-COMB-BuyHold'** 		
Expected Result	A grid is displayed that gives the total returns for each strategy and each ticker symbol		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** Tableau is successfully connected to MySQL local database 'gmfsp_db' and data on tab changes appropriately **Failed:** Tableau is not able to connect to the database, incomplete data, or no data at all is displayed		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue (we are limited in correcting it) 4 SQL Script to build this tab has issue		
Priority	**High**	**Priority Reason**	**Viewing portfolio returns is a main feature of the application**
Priority Sequence	**H-12**		

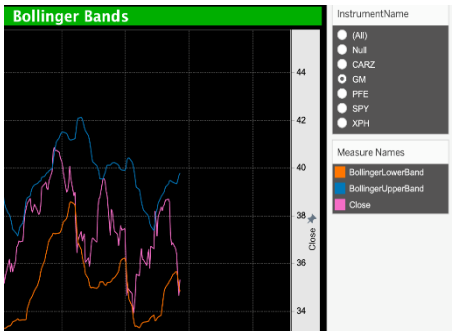
	Strategy						
	algo	BuyHold	CMA	COMB	EMA	FRL	MACD
CARZ	312	-97	590	105	486	428	-768
GM	1,385	-842	-205	-961	-471	-28	-1,686
PFE	-821	-922	-1,328	-804	-621	-62	572
SPY	1,299	777	620	1,043	912	184	645
XPH	152	-43	249	15	260	909	189

ID	**NFTC_17.0**		
Test Case Name	Tableau – Portfolio-MACD-COMB-BuyHold tab has instrument name radio buttons functional		
Created By	John Gettel	**Date Created**	11-24-2019
Description	Tableau – Portfolio-MACD-COMB-BuyHold tab has instrument name radio buttons functional		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly with required drivers installed		
Postconditions	5 User will be able to switch view between each instrument tracked by the application		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 Click on the tab named **'Portfolio-MACD-COMB-BuyHold'** 4 On the right, there should be list of radio buttons  5 Make a different selection other than the default selected option		
Expected Result	Toggle buttons are filtering data as per design. The graph should change when the tester toggles the button.		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** Tableau is successfully connected to MySQL local database 'gmfsp_db' and data on tab changes appropriately **Failed:** Tableau is not able to connect to the database or no changes occur on tab		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue (we are limited in correcting it) 4 SQL Script to build this tab has issue, multiple SQL scripts 5 Filters setting in Tableaus is not setup properly		
Priority	**High**	**Priority Reason**	**Viewing data for different symbols is required**
Priority Sequence	**H-13**		

ID	**NFTC_18.0**		
Test Case Name	Tableau – Small play and reverse buttons on the bottom right of the screen are functional and move the tabs to the right or left		
Created By	John Gettel	**Date Created**	11-24-2019
Description	Tableau – Small play and reverse buttons on the bottom right of the screen are functional and move the tabs to the right or left		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly with required drivers installed		
Postconditions	1 User will be able to scroll through the menu of sheets at the bottom of the tableau application		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 On the lower right hand corner of the screen click the 'play' button to view the next sheet in the list  4 Click the 'fast forward' button to scroll to the last sheet 5 Click the 'reverse' button to view the previous sheet 6 Click the 'rewind' button to scroll back to the first sheet 7 Each of the above steps should allow the user to view a new sheet and click on it to view		
Expected Result	Each of the four scroll buttons should displace the sheets bar by a predefined distance configured by Tableau		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** A new sheet is available to be clicked on by the user **Failed:** No new sheets appear unless at the beginning or end of the list of sheets		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly, recreated the data source 3 Tableau has encountered an issue (we are limited in correcting it) 4 SQL Script to build each individual tab has failed to execute properly		
Priority	**Medium**	**Priority Reason**	**Viewing all data tabs is optimal**
Priority Sequence	**M-5**		

ID	**NFTC_19.0**		
Test Case Name	Tableau - CMAFeatures tab shows close, lcma, scma and wcma values and radio buttons for the instrument name		
Created By	John Gettel	**Date Created**	11-24-2019
Description	Tableau - CMAFeatures tab shows close, lcma, scma and wcma values and radio buttons for the instrument name		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly with required drivers installed		
Postconditions	User will be able to view the close, lcma, scma, and wcma values on each date for each individual symbol		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 On the right side of the application click through each radio button for the ticker symbols 4 The data showing on the the graph should change for each symbol 5 Each symbol should have a lcma, scma, wcma, and close data point for each day. 		
Expected Result	The data displayed in each graph should change for each ticker symbol selected with a radio button		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** The user can select and see the appropriate data on each day for each symbol **Failed:** No data or incomplete data appears, or no toggling occurs		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly 3 Tableau has encountered an issue (we are limited in correcting Tableau) 4 SQL Script to build each individual tab has failed to execute properly		
Priority	**Medium**	**Priority Reason**	**Technical data view is a good feature**
Priority Sequence	**M-6**		

ID	**NFTC_20.0**		
Test Case Name	Tableau - FRLFeatures tab shows Close, five different FRL values and radio buttons for the instrument name		
Created By	John Gettel	**Date Created**	11-24-2019
Description	Tableau - FRLFeatures tab shows Close, five different FRL values and radio buttons for the instrument name		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly with required drivers installed		
Postconditions	Users will be able to view the close price and each of the five different Fibonacci Retracement levels for each date and each ticker symbol		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 On the right side of the application click through each radio button for the ticker symbols 4 The data showing on the the graph should change for each symbol 5 Each symbol should have five different Fibonnacci retracement levels for each date 		
Expected Result	The data displayed in each graph should change for each ticker symbol selected with a radio button		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** The user can click through and see the appropriate data on each day for each symbol **Failed:** No data or incomplete data appears or no toggling occurs		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly 3 Tableau has encountered an issue (we are limited in correcting it) 4 SQL Script to build each individual tab has failed to execute properly		
Priority	**Medium**	**Priority Reason**	**Viewing technical data is a good feature**
Priority Sequence	**M-7**		

ID	**NFTC_21.0**		
Test Case Name	Tableau - Tableau - BollingerBands tab shows Close, lower and, upper bound values appropriately and radio buttons for the instrument name		
Created By	Aalem Singh	**Date Created**	11-24-2019
Description	Tableau - BollingerBands tab shows Close, lower and, upper bound values appropriately and radio buttons for the instrument name		
Tester	Data Analyst or Developer		
Test Frequency	Anytime the application testing is required		
Executed By	Application User		
Preconditions	1 Database is created with MySQL **Username** = root **Password** = password 2 Tableau must be setup properly 3 Database must be setup properly with required drivers installed		
Postconditions	User will be able to view the close price and each of the five different Fibonacci Retracement levels for each date and each ticker symbol		
Test Steps	1 Browse to Tableau application file name **'GM_FinTech_Application'** 2 Double click to open it 3 On the right side of the application click through each radio button for the ticker symbols 4 The data showing on the the graph should change for each symbol 5 Each symbol should have an upper and a lower Bolligner Band displayed for each date 		
Expected Result	The data displayed in each graph should change for each ticker symbol selected with a radio button		
Test Entry/Exit Criteria	**Entry:** Test can be executed at any time in any order **Exit:** When the test completes, or it can be terminated during the run		
Test Risks	This test itself causes no risk to the operation of the application or the existing data in the database.		
Pass/Fail Criteria	**Passed:** The user can click through and see data on each day for each symbol **Failed:** No data or incomplete data appears or no toggling occurs		
Fail Reasons	1 MySQL database is not setup properly 2 Tableau data source is not setup properly 3 Tableau has encountered an issue (we are limited in correcting it) 4 SQL Script to build each individual tab has failed to execute properly		
Priority	**Medium**	**Priority Reason**	**Viewing technical data is important**
Priority Sequence	**M-8**		

4 PRODUCT TESTING CERTIFICATION

The undersigned acknowledge that they have overseen or themselves performed each test listed in this document for the GM FinTech Tableau/PyCharm/MySQL based application and all tests were successful and passed the expected results criteria.

Team Lead, Front End Lead, Back End Lead, QA Lead(if available)

Signature		**Date**	
Print Name			
Title			
Role			

Signature		**Date**	
Print Name			
Title			
Role			

Signature		**Date**	
Print Name			
Title			
Role			

Signature		**Date**	
Print Name			
Title			
Role			

APPENDIX

APPENDIX A : SQL AND PYTHON TESTING SCRIPTS

ID	TC_1.0		
Test Case Name	Database Engine has the correct MYSQL connection string		
Created By	Abdul Ahmad	Date Created	11-10-2019

```
# TC_1.0_Database Engine

import sqlalchemy as sal
from FinsterTab.F2019.dbEngine import DBEngine

conn_str = 'mysql+pymysql://root:password@localhost:3306/gmfsp_db'
actual_engine = DBEngine().mysql_engine()
expected_engine = sal.create_engine(conn_str)

if actual_engine == expected_engine:
    print("correct engine")
else:
    print("incorrect engine")
```

ID	TC_2.0		
Test Case Name	Correct financial data exchange for fetching base statistics		
Created By	Abdul Ahmad	Date Created	11-10-2019

```
# Script ID : TC_2.0
from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataFetch import DataFetch
conn_str = 'mysql+pymysql://root:password@localhost:3306/gmfsp_db'
engine = DBEngine().mysql_engine()
db_engine = DBEngine().mysql_engine()
instrument_master = 'dbo_instrumentmaster'
master_data = DataFetch(engine, instrument_master)

if master_data.datasource != 'yahoo':
    print("incorrect data source: %s" % master_data.datasource)
```

ID	TC_3.0		
Test Case Name	No duplicate instrument ids in 'instrumentmaster' table		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID     : TC_3.0                                    --

SELECT
    instrumentid,
          count(instrumentname) cnt

FROM dbo_instrumentmaster as m

group by
                    instrumentid
having count(instrumentname) > 1
;
```

ID	TC_4.0		
Test Case Name	No duplicate values/records in future forecast for each symbol		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID     : TC_4.0 --

SELECT
    a.instrumentid,
    count(distinct a.forecastdate) cnt

FROM dbo_algorithmforecast as a

where a.forecastdate > current_date()

group by
     a.instrumentid
;
```

ID	TC_5.0		
Test Case Name	No duplicate values/records in past forecast for each symbol		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID     : TC_5.0 --

SELECT
    a.instrumentid,
    count(distinct a.forecastdate) cnt

FROM dbo_algorithmforecast as a

where a.forecastdate < current_date()

group by
     a.instrumentid
;
```

ID	TC_6.0		
Test Case Name	No duplicate values/records in past forecast for each symbol		
Created By	Mohamad Saab	Date Created	11-22-2019

```
-- Script ID : TC_6.0 --

use GMFSP_db;

SELECT  instrumentid InstrumentID,
                    avg(forecastcloseprice) ForecastPriceAverage,
      min(forecastdate) NextDate,
      max(forecastdate) LastDate,
      count(distinct forecastdate) NumOfDaysAhead
FROM dbo_algorithmforecast
WHERE forecastdate > current_date()
GROUP BY instrumentid
```

ID	TC_7.0		
Test Case Name	No duplicate values/records in past forecast for each symbol		
Created By	Mohamad Saab	Date Created	11-21-2019

```
-- Script ID  :  TC_7.0 --

use GMFSP_db;

SELECT  algorithmcode AlgorithmName,
                    avg(forecastcloseprice) ForecastPriceAverage,
      min(forecastdate) NextDate,
      max(forecastdate) LastDate,
      count(distinct forecastdate) NumOfDaysAhead
FROM dbo_algorithmforecast
WHERE forecastdate > current_date()
GROUP BY algorithmcode
```

ID	TC_8.0		
Test Case Name	No duplicate values/records in past forecast for each symbol		
Created By	Abdul Ahmad	Date Created	11-21-2019

```
-- Script ID     : TC_8.0 --

SELECT
    count(distinct instrumentid) instid,
    count(distinct forecastdate) date,
    avg( forecastcloseprice) avgprice

FROM dbo_algorithmforecast as a
      left outer join dbo_datedim as d on a.forecastdate = d.date

where forecastdate < current_date()
  and d.isholiday = 0
  and d.weekend = 0
;
```

ID	TC_9.0		
Test Case Name	Past forecast values must exist for last 3 years until today for each algorithm		
Created By	Abdul Ahmad	Date Created	11-21-2019

```
-- Script ID     : TC_9.0 --

SELECT
    count(distinct algorithmcode) algo,
    count(distinct forecastdate)  date,
    avg(forecastcloseprice)      avgprice

FROM dbo_algorithmforecast as a
      left outer join dbo_datedim as d on a.forecastdate = d.date

where forecastdate < current_date()
  and d.isholiday = 0
  and d.weekend = 0
;
```

ID	TC_10.0		
Test Case Name	Ten future days from today with no prediction error for each symbol		
Created By	Abdul Ahmad	Date Created	11-21-2019

```
-- Script ID     : TC_10.0 --

SELECT
    count(distinct forecastdate)  date,
    avg(prederror)            prederror

FROM dbo_algorithmforecast as a
      left outer join dbo_datedim as d on a.forecastdate = d.date

where forecastdate > current_date()
  and d.isholiday = 0
  and d.weekend = 0
;
```

ID	TC_11.0		
Test Case Name	Ten future days from today with no prediction error for each algorithm		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID     : TC_11.0 --

SELECT
    count(distinct forecastdate)  date,
    count(distinct algorithmcode) algo,
    avg(prederror)              prederror

FROM dbo_algorithmforecast as a
      left outer join dbo_datedim as d on a.forecastdate = d.date

where forecastdate > current_date()
  and d.isholiday = 0
  and d.weekend = 0
;
```

ID	TC_12.0		
Test Case Name	Nine tables exist in the database		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID     : TC_12.0 --

use gmfsp_db;

SELECT COUNT(*) as cnt
FROM information_schema.tables
WHERE table_schema = 'gmfsp_db' and table_name like 'dbo_%'
;
```

ID	TC_13.0		
Test Case Name	ARIMA has first 9 days blank		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID : TC_13.0 --
use gmfsp_db;
SELECT i.date,
	i.instrumentid,
	i.close,
		a.algorithmcode,
	a.forecastcloseprice
FROM dbo_instrumentstatistics as i
	 left outer join dbo_algorithmforecast as a  on i.instrumentid = a.instrumentid and
                                        i.`date` = a.forecastdate and
                                        a.algorithmcode ='arima'

order by i.date
;
```

ID	TC_14.0		
Test Case Name	Random Forest has first 9 days blank		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID : TC_14.0 --
use gmfsp_db;
SELECT i.date,
	i.instrumentid,
	i.close,
		a.algorithmcode,
	a.forecastcloseprice
FROM dbo_instrumentstatistics as i
	 left outer join dbo_algorithmforecast as a  on i.instrumentid = a.instrumentid and
                                        i.`date` = a.forecastdate and
                                        a.algorithmcode ='randomforest'

order by i.date
;
```

ID	TC_15.0		
Test Case Name	No missing values in the 'close' or 'date' column in the instrumentstatistics table		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID : TC_15.0 --

use gmfsp_db;

SELECT i.`date`,
      i.instrumentid,
      i.`close`

FROM dbo_instrumentstatistics as i
where i.`close` is null or i.date is null
;
```

ID	TC_16.0		
Test Case Name	BuySell signals must exist for all reccords for past 3 years, values of -1,0,1 for each day		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID : TC_16.0 --

use gmfsp_db;

SELECT i.`date`,
      count(distinct i.signal)  signals

FROM dbo_actionsignals as i
group by i.date
order by i.date
;
```

ID	TC_17.0		
Test Case Name	BuySell signals must exist for each strategy		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID : TC_17.0 --

use gmfsp_db;

SELECT i.`date`,
      count(distinct i.strategycode) strcd,
      count(distinct i.signal)       signals

FROM dbo_actionsignals as i
group by i.date
order by i.date
;
```

ID	TC_18.0		
Test Case Name	BuySell signals must exist for each symbol		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID : TC_18.0 --

use gmfsp_db;

SELECT i.`date`,
      count(distinct i.instrumentid) instid,
      count(distinct i.signal)       signals

FROM dbo_actionsignals as i
group by i.`date`
having count(distinct i.instrumentid) <> 5
;
```

ID	TC_19.0		
Test Case Name	Six algorithm codes exist in algorithmmaster table		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID : TC_19.0 --

use gmfsp_db;

SELECT *

FROM dbo_algorithmmaster
;
```

ID	TC_20.0		
Test Case Name	Instrumentmaster must have only 5 stock symbols		
Created By	Abdul Ahmad	Date Created	11-22-2019

```
-- Script ID : TC_20.0 --

use gmfsp_db;

SELECT *

FROM dbo_instrumentmaster
;
```

ID	TC_21.0		
Test Case Name	Instrumentstats column named 'high' has values >= 'low' , 'open', 'close' columns on all records		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_21.0 --

use gmfsp_db;

SELECT *

FROM dbo_instrumentstatistics

where
        high < low or
   high < `open` or
   high < `close`
;
```

ID	TC_22.0		
Test Case Name	Instrumentstats column names 'low' has values <= high, open, close columns on all records		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_22.0 --

use gmfsp_db;

SELECT *

FROM dbo_instrumentstatistics

where
        low > high or
   low > `open` or
   low > `close`
;
```

ID	TC_23.0		
Test Case Name	Instrumentstats column 'volume' values should always be >=0		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_23.0 --

use gmfsp_db;

SELECT *

FROM dbo_instrumentstatistics

where
    volume < 0
;
```

ID	TC_24.0		
Test Case Name	Instrumentstats must have 3 years of data till today, going back 3 years from today		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_24.0 --

use gmfsp_db;

SELECT  min(date) startdate,
       max(date) enddate

FROM dbo_instrumentstatistics

;
```

ID	TC_25.0		
Test Case Name	Datedim must contain 50 future days from today		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_25.0 --

use gmfsp_db;

SELECT  min(date) todaysdate,
      max(date) tendaysintofuture,
      datediff(cast(date_add(now(), interval 50 day) as date) , current_date()) daysdiff

FROM dbo_datedim

WHERE
          `date` >= current_date() and
    `date` < cast(date_add(now(), interval 50 day) as date)
;
```

ID	TC_26.0		
Test Case Name	Datedim must contain last 3 years of dates, up till today		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_26.0 --

use gmfsp_db;

SELECT  min(date) todaysdate,
      max(date) tendaysintofuture,
      datediff(cast(date_add(now(), interval 3 year) as date) , current_date()) daysdiff

FROM dbo_datedim

WHERE
          `date` <= current_date() and
    `date` >= cast(date_add(now(), interval -3 year) as date)
;
```

ID	TC_27.0		
Test Case Name	No duplicate dates or tuples in datedim table		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_27.0 --

use gmfsp_db;

SELECT  count(*) cnt,
      date
FROM dbo_datedim
GROUP by date
HAVING count(*) > 1
;
```

ID	TC_28.0		
Test Case Name	In Statisticalreturns table, cash on hand value must be >=0 for all records		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_28.0 --

use gmfsp_db;

SELECT  date,
      instrumentid,
      strategycode,
      cashonhand

FROM dbo_statisticalreturns

WHERE cashonhand < 0
;
```

ID	TC_29.0		
Test Case Name	In Statisticalreturns table, 'portfoliovalue' data values must be >= 'cash on hand' values for all records		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_29.0 --

use gmfsp_db;
SELECT  `date`,
      instrumentid,
      strategycode,
      cashonhand,
      portfoliovalue

FROM dbo_statisticalreturns
WHERE cashonhand > portfoliovalue
;
```

ID	TC_30.0		
Test Case Name	In Statisticalreturns table, 'positionsize' column must be an integer data type		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_30.0 --

use gmfsp_db;

SELECT  TABLE_NAME,
      COLUMN_NAME,
      DATA_TYPE

FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_name = 'dbo_statisticalreturns' AND
            COLUMN_NAME = 'positionsize'
;
```

ID	TC_31.0		
Test Case Name	Statisticalreturns table must contain 7 strategies		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_31.0 --

use gmfsp_db;

SELECT count(distinct strategycode) stratcode

FROM dbo_statisticalreturns
;
```

ID	TC_32.0		
Test Case Name	Fibonacci lines values are in order, 'highfrllinelong' values are >= 'lowfrllinelong' values for all records		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_32.0 --

use gmfsp_db;

SELECT *

FROM dbo_engineeredfeatures

WHERE lowfrllinelong > highfrllinelong
;
```

ID	TC_33.0		
Test Case Name	Bollinger Bands values are in order, boll_ub_v > boll_lb_v values for all records		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_33.0 --

use gmfsp_db;

SELECT *

FROM dbo_engineeredfeatures

WHERE boll_lb_v > boll_ub_v
;
```

ID	TC_34.0		
Test Case Name	Each date must have a 0 , 1, -1 signal for each strategy and symbol in 'actionsignals' table		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_34.0 --
use gmfsp_db;
drop table if exists `a`;
drop table if exists `b`;
drop table if exists `c`;
create temporary table a
SELECT   `signal`,
 count(`date`) cnt
FROM dbo_actionsignals
GROUP BY `signal`;
create temporary table b
select 'SubTotal of Records Grouped by Signal Above' as `signal`,
      sum(cnt) as cnt
from a;
select *
from a UNION ALL
select *
from b UNION ALL
select      'Total Records in ''actionsignals'' Table' as `signal`,
        count(*)
from dbo_actionsignals;
```

ID	TC_35.0		
Test Case Name	In Statisticalreturns table 'cash on hand' values must be between 0 and 'portfoliovalue', it should never exceed the 'portfolio' value		
Created By	Abdul Ahmad	Date Created	11-23-2019

```
-- Script ID : TC_35.0 --

use gmfsp_db;

SELECT min(cashonhand) mincoh,
            max(cashonhand) maxcoh,
     max(portfoliovalue) maxportfolio

FROM dbo_statisticalreturns
;
```

ID	TC_36.0		
Test Case Name	BuyHold strategy must have no signals, this strategy should only hold the stock and not sell it or buy more		
Created By	Aalem Singh	Date Created	11-23-2019

```
-- Script ID : TC_36.0 --

use gmfsp_db;

SELECT *

FROM dbo_actionsignals

WHERE strategycode = 'buyhold'
;
```

ID	TC_37.0		
Test Case Name	In 'actionsignals' table frl, cma, ema, macd signals sum must be between -5 and 5		
Created By	Aalem Singh	Date Created	11-23-2019

```
-- Script ID : TC_37.0 --

use gmfsp_db;

select `date`,
      instrumentid,
      sum(`signal`) as sgnl

FROM dbo_actionsignals

WHERE strategycode in ('frl' , 'cma', 'ema', 'macd')
group by `date`,
       instrumentid
having sum(`signal`) < -5 or
      sum(`signal`) > 5
;
```

ID	TC_38.0		
Test Case Name	getdatasources() function returns correct number of symbols		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_38.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataFetch import DataFetch

engine = DBEngine().mysql_engine()
fetch = DataFetch(engine, 'dbo_instrumentmaster')
tickers = fetch.get_datasources()

if len(tickers) == 5:
    print('incorrect number of symbols: %s' % str(len(tickers)))
```

ID	TC_39.0		
Test Case Name	getdatasources() function returns correct ticker symbols		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_39.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataFetch import DataFetch

engine = DBEngine().mysql_engine()
fetch = DataFetch(engine, 'dbo_instrumentmaster')
tickers = fetch.get_datasources()

symbols = ['GM', 'PFE', 'SPY', 'XPH', 'CARZ']

for i in range(len(symbols)):
    if symbols[i] != tickers['instrumentname'][i]:
        print('missing ticker: %s' % symbols[i])
```

ID	TC_40.0		
Test Case Name	getdata () function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_40.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataFetch import DataFetch

engine = DBEngine().mysql_engine()
fetch = DataFetch(engine, 'dbo_instrumentmaster')
tickers = fetch.get_datasources()
data = fetch.get_data(tickers)

if data is not None:
    print('Incorrect return value while running get_data function: %s' % data)

    if symbols[i] != tickers['instrumentname'][i]:
        print('missing ticker: %s' % symbols[i])
```

ID	TC_41.0		
Test Case Name	get_calendar() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_41.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataFetch import DataFetch

engine = DBEngine().mysql_engine()
fetch = DataFetch(engine, 'dbo_instrumentmaster')
calendar = fetch.get_calendar()

if calendar is not None:
    print('Incorrect return value while running get_calendar function: %s' % calendar)
```

ID	TC_42.0		
Test Case Name	calculate() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_42.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.EngineeredFeatures import EngineeredFeatures

engine = DBEngine().mysql_engine()
eng_feat = EngineeredFeatures(engine, 'dbo_instrumentmaster')
eng_feat = eng_feat.calculate()

if eng_feat is not None:
    print('Incorrect return value while running calculate function: %s' % eng_feat)
```

ID	TC_43.0		
Test Case Name	calculate_forecast() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_43.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataForecast import DataForecast

engine = DBEngine().mysql_engine()
forecast = DataForecast(engine, 'dbo_instrumentmaster')
forecast = forecast.calculate_forecast()

if forecast is not None:
    print('Incorrect return value while running calculate_forecast function: %s' % forecast)
```

ID	TC_44.0		
Test Case Name	calculate_arima_forecast() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_44.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataForecast import DataForecast

engine = DBEngine().mysql_engine()
forecast = DataForecast(engine, 'dbo_instrumentmaster')
forecast = forecast.calculate_arima_forecast()

if forecast is not None:
    print('Incorrect return value while running calculate_arima_forecast function: %s' % forecast)
```

ID	TC_45.0		
Test Case Name	calculate_random_forest_forecast() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_45.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataForecast import DataForecast

engine = DBEngine().mysql_engine()
forecast = DataForecast(engine, 'dbo_instrumentmaster')
forecast = forecast.calculate_random_forest_forecast()

if forecast is not None:
    print('Incorrect return value while running calculate_random_forest_forecast function: %s' % forecast)
```

ID	TC_46.0		
Test Case Name	calculate_svm_forecast() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_46.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataForecast import DataForecast

engine = DBEngine().mysql_engine()
forecast = DataForecast(engine, 'dbo_instrumentmaster')
forecast = forecast.calculate_svm_forecast()

if forecast is not None:
    print('Incorrect return value while running calculate_svm_forecast function: %s' % forecast)
```

ID	TC_47.0		
Test Case Name	calculate_forecast_old() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_47.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataForecast import DataForecast

engine = DBEngine().mysql_engine()
forecast = DataForecast(engine, 'dbo_instrumentmaster')
forecast = forecast.calculate_forecast_old()

if forecast is not None:
    print('Incorrect return value while running calculate_old_forecast function: %s' % forecast)
```

ID	TC_48.0		
Test Case Name	calculate_rxgboost_forecast() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_48.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.DataForecast import DataForecast

engine = DBEngine().mysql_engine()
forecast = DataForecast(engine, 'dbo_instrumentmaster')
forecast = forecast.calculate_xgboost_forecast()

if forecast is not None:
    print('Incorrect return value while running calculate_xgboost_forecast function: %s' % forecast)
```

ID	TC_49.0		
Test Case Name	cma_signal() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_49.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.BuySell import BuySell

engine = DBEngine().mysql_engine()
signal = BuySell(engine, 'dbo_instrumentmaster')
signal = signal.cma_signal()

if signal is not None:
    print('Incorrect return value while running cma_signal function: %s' % signal)
```

ID	TC_50.0		
Test Case Name	frl_signal() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_50.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.BuySell import BuySell

engine = DBEngine().mysql_engine()
signal = BuySell(engine, 'dbo_instrumentmaster')
signal = signal.frl_signal()

if signal is not None:
    print('Incorrect return value while running frl_signal function: %s' % signal)
```

ID	TC_51.0		
Test Case Name	ema_signal() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_51.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.BuySell import BuySell

engine = DBEngine().mysql_engine()
signal = BuySell(engine, 'dbo_instrumentmaster')
signal = signal.ema_signal()

if signal is not None:
    print('Incorrect return value while running ema_signal function: %s' % signal)
```

ID	TC_52.0		
Test Case Name	macd_signal() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_52.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.BuySell import BuySell

engine = DBEngine().mysql_engine()
signal = BuySell(engine, 'dbo_instrumentmaster')
signal = signal.macd_signal()

if signal is not None:
    print('Incorrect return value while running macd_signal function: %s' % signal)
```

ID	TC_53.0		
Test Case Name	algo_signal() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_53.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.BuySell import BuySell

engine = DBEngine().mysql_engine()
signal = BuySell(engine, 'dbo_instrumentmaster')
signal = signal.algo_signal()

if signal is not None:
    print('Incorrect return value while running algo_signal function: %s' % signal)
```

ID	TC_54.0		
Test Case Name	trade_sim() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_54.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.TradingSimulator import TradingSimulator

engine = DBEngine().mysql_engine()
simulator = TradingSimulator(engine, 'dbo_instrumentmaster')
simulator = simulator.trade_sim()

if simulator is not None:
    print('Incorrect return value while running trade_sim function: %s' % simulator)
```

ID	TC_55.0		
Test Case Name	comb_sim() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_55.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.TradingSimulator import TradingSimulator

engine = DBEngine().mysql_engine()
simulator = TradingSimulator(engine, 'dbo_instrumentmaster')
simulator = simulator.combination_trade_sim()

if simulator is not None:
    print('Incorrect return value while running combination_trade_sim function: %s' % simulator)
```

ID	TC_56.0		
Test Case Name	buy_hold_sim() function runs successfully		
Created By	John Gettel	Date Created	11-24-2019

```
# TC_56.0

from FinsterTab.F2019.dbEngine import DBEngine
from FinsterTab.F2019.TradingSimulator import TradingSimulator

engine = DBEngine().mysql_engine()
simulator = TradingSimulator(engine, 'dbo_instrumentmaster')
simulator = simulator.buy_hold_sim()

if simulator is not None:
    print('Incorrect return value while running buy_hold_sim function: %s' % simulator)
```

APPENDIX B : KEY TERMS

Term	Description
ARIMA	Auto-Regressive Integrated Moving Average
Back-End	Database that supports the middle layer and front-end
CMA	Cross Moving Average
EMA	Exponential Moving Average
FRL	Fibonacci Retracement Line
GMFSP_db	Name of the database used to store data
Instrument	Financial instruments are monetary contracts between parties. They can be created, traded, modified and settled. They can be cash, evidence of an ownership interest in an entity, or a contractual right to receive or deliver cash.
Middle-Layer	Python code that runs the algorithms, data fetch and, data loads
Non-Functional Requirement	A requirement that is deemed part of important and critical functions of a Software
Random Forest	An algorithm for making predictions on the data using the learning method of classification.
Tableau	Software used to visualize data
Test Case	Steps containing actions to test a functionality of the Software
Ticker	Stock symbol or instrument

THIS PAGE LEFT INTENTIONALLY BLANK