Shrey Patel | B00960433 | sh644024@dal.ca

Milestone 2: Blackbox Tests and Feature Development Plan

Main task: To list out all the possible black box test cases for each function that could be implemented and also to create a rough timeline for the features that the system will serve.

Blackbox Tests for possible features:

In this we have to design the test cases such that we don't know the design of the code that is going to be implemented. The following are possible functions and their test cases. Below are the two general domains where test cases can be thought of.

Define Sectors and Stocks:

- By adding different values of stocks that exist and does not exist and how every stock is related with each other.
- Adding huge number of sectors and stock values to the system to ensure that it does not crash.

Investor Profiles

- To create different investment profiles with different desired investment sector distribuon and to check if they are easily readable or basically, they are properly stored and retrievable.

Below are the probable test cases for different functions that are mentioned in the document:

defineSector(String sectorName)

- Test with valid sector names.
- Test with null and empty strings.
- Test with duplicate sector names.

defineStock(String companyName, String stockSymbol, String sector)

- Test with valid company names, stock symbols, and sectors.
- Test with non-existent sectors.
- Test with null and empty values.

setStockPrice(String stockSymbol, double perSharePrice)

- Test with valid stock symbols and price values.
- Test with negative and zero price values.
- Test with stock symbols that do not exist.

defineProfile(String profileName, Map<String, Integer> sectorHoldings)

- Test with valid profile names and sector holdings.
- Test with sector holdings that do not add up to 100%.
- Test with an empty map.

addAdvisor(String advisorName)

- Test with valid advisor names.
- Test with null and empty strings.
- Test for the uniqueness of advisor IDs.

addClient(String clientName)

- Test with valid client names.
- Test with null and empty strings.
- Test for the uniqueness of client IDs.

createAccount(...)

- Test with valid parameters.
- Test with invalid client or advisor IDs.
- Test with invalid profile types.

tradeShares(...)

- Test buying and selling shares with valid account and stock symbols.
- Test buying without sufficient cash and selling shares not owned.
- Test with "cash" as the stock symbol for cash transfers.

changeAdvisor(int accountId, int newAdvisorId)

- Test with valid account and advisor IDs.
- Test with non-existent account or advisor IDs.
- Test changes are reflected correctly.

accountValue(int accountId)

- Test with a valid account ID with different stock holdings.
- Test with an account ID that does not exist.

advisorPortfolioValue(int advisorId)

- Test with valid advisor IDs.
- Test with advisor IDs that do not exist.

investorProfit(int clientId)

- Test with valid client IDs.
- Test with client IDs that do not exist.
- Test with no trading history.

profileSectorWeights(int accountId)

- Test with a valid account with various holdings.
- Test with an account that does not exist.

divergentAccounts(int tolerance)

- Test with various tolerance levels.
- Test with no divergent accounts.

disburseDividend(String stockSymbol, double dividendPerShare)

- Test with valid stock symbols and dividend amounts.
- Test with stock symbols that do not exist.

stockRecommendations(...)

- Test with various account IDs and comparator numbers.
- Test with non-existent account IDs.

advisorGroups(double tolerance, int maxGroups)

- Test with different tolerances and max group values.
- Test with scenarios expected to return multiple advisor groups.

Feature Development Timeline

April 1-2: Implement database design and set up tables. Designing of data base and creating tables accordingly for further implementation.

April 3: Development of functions that can create sectors and stocks of various types.

April 4: After the stocks and their sectors are added, development of function to manipulate the stock price and manage the profile of the client can be done.

April 5: Code addAdvisor, addClient, and createAccount methods.

April 6-7: Implement tradeShares and changeAdvisor methods.

April 8: Develop reporting functions like accountValue, advisorPortfolioValue and clusterFirms.

April 9-10: Code investorProfit

April 10-12: Code refactoring and code smells

April 12-15: Testing, and debugging the code. Fixing the bugs and preparing the document.