Testing Documentation

Checklist of common types of errors, relevant to the scheme of this project.

Data Faults	• Has the JSON output been verified? • Is the JSON encoding stable? • Can we assure that the JSON will be raw, for any third-party tool or GUI? • Is there any possibility of JSON corruption?
Control Faults	• For each conditional statement, is the condition correct and verified? Will inputting a company symbol lead to desired operation, or improperly branch towards elsewhere. • Is each loop certain to terminate, or will it be stuck in limbo? • Will each conditional statement outcome yield the proper result? Will the output for each stock company be unique to itself?
Input/Output Faults	• Are all input viable? • What to do in the event of improper input? • Can unexpected inputs cause corruption?
Interface Faults	• Do all function and method calls have the correct number of parameters? • If components access the same database, do they have the same model of the shared memory structure?

Storage Management Faults	• If a linked structure is modified, have all links been correctly reassigned? • If dynamic storage is used, has space been allocated correctly? • Is space explicitly deallocated after it is no longer required?
Exception Management Faults	• Has any possible exception been given proper handling by the developer, or by the environment?

********Summaries of Testings**********

Goals: The goals of these tests include checking the stability and desired functionality of our project manifestation. Primarily, seeing that our program outputs the proper JSON for selected company/date/etc. Not only must the JSON be proper, but the calculations for it must be accurate. While the APIs for this project can work stand alone, we need to ensure that we interact correctly with our web front end.

Verification Testing: Throughout our project, we shall sustain verification testing. Through consultation with our Professor Wai-Tak Wong, we ensure we are building the product right. Our professor will help us clear any ambiguity on the project guide sheet, to make sure we conform to specification.

We also held constant inspections, weekly, to ensure our code base was well formed. As a group, we peer reviewed to ensure we were all on the same page regarding direction of the project.

Validation Testing: We will ensure our tests validate the product. We will show that it outputs proper calculations for our consumer. Primarily if the calculations are correct.

We conducted black-box testing by just having multiple trials to ensure proper input/output without having the tester be keen to the exact implementation. This included our professor, as well as our peers.

I, Anthony, as well as others in our group conducted white box testing. We conducted trial tests of the project, knowing the details of this software and hardware. So if there were any issues, we would have an idea of where to look for, in the code

Defect Testing: Despite having proper calculations, there are many possible defects. This can include things like JSON output. Or JSON encoding/decoding. We also have to ensure our front-end displays in stable manner, and handles forms properly.

==

Unit Testing: We implemented unit testing by testing each of our RESTful APIs. We also tested the front end to make sure there was no abnormalities.

System Testing: We then tested how our front and back-end would interact, to make one whole composite web-site like project.

============================Scenario Testing============================

Test Scenario ID	Test Scenario	Test Case ID	Test Case Name	Test Steps	Test Data	Expected Outcome	Actual Outcome	Result
TS_01	Check Logos API	TC_01	Check logo with valid symbol	1.Go to Logos Page 2. Enter symbol in field 3.Hit enter	symbol=rfem	User should Receive JSON output with logo link	As Expected	Pass
	Check Company API	TC_02	Check to see if API shows company info	1.Go to Company Page 2. Enter symbol in field	symbol=googl	User should Receive JSON Output With company	As Expected	Pass

				3.Hit enter		info		
	Check Custom Dates API	TC_03	Check to see proper dates and calculations	1.Go to Custom Dates page 2.enter Symbol 3.enter Date Range 4.hit enter	symbol= aapl startDate = 2016-11-10 endDate = 2016-12-10	JSON output of high, low, median, etc displayed. Within the range input	As Expected	Pass
	Check 30 days API	TC_04	Check to see if the most recent 30 days are displayed, along with	1.Go to 30 days page 2.enter Symbol 3.Hit enter	symbol= aapl	The most recent 30 days are displayed, with added calculations from server side	As Expected	Pass
TS_02	Check to see if daily update scripts work	TC_01	Check to see if windows scheduler calls scripts to update database from IEX	1.Check Scheduler Logs 2.Check database to verify update	N/A	The database is updated, even when the computer is shut off(by turning itself on)	As Expected	Pass
TS_03	Check to see if APIs work in tool like Postman	TC_01	Validate that the APIs work standalone in a RESTful manner	1.Procure proper uri 2.open postman 3.set to GET 4.Enter in bar. 5.Hit enter	Multiple, different URIs	To have the output similar to the website, but in a PRETTY JSON format.	As Expected	Pass

				6.Set output to PRETTY JSON				