Richard Maddern

CS 499

11/29/2020

Milestone 4

1. Briefly describe the artifact. What is it? When was it created?

This artifact was created in my CS-340 class to use MongoDB and python to create a database. I used the MongoDB tool for creating a database from the json file. I then provided a database name “Market” and a collection name “Stocks” then specify the path of the json file, which a database will be created. First up was then to import database.

1. Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?

Honestly, I selected this artifact because it was the only database I have ever worked on. I know it had some issues with the search and delete function, but I already repaired that before this submission.

1. Did you meet the course objectives you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

Yes, it is all functioning now and can be taken out of codio and would work flawlessly. At this point since everything is working and it is doing what it was designed to do, I will be leaving it be for the portfolio.

1. Reflect on the process of enhancing and/or modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

I learned that I was poorly handling the QA on this project due to most of the mistakes I made were syntax errors in the python documents and in the actual database code as well making it so it always displayed the same answer. Now it shows the correct answers shown below in the explanation of the artifact.

**Collection Management**

I used the MongoDB tool for creating a database from the json file. I then provided a database name “Market” and a collection name “Stocks” then specify the path of the json file, which a database will be created. First up was then to import database.

Text

Description automatically generated

**Document Manipulation**

For this section I had to insert update and delete documents in the collection database. So, I created a file insertDocument.py then added it to the database. Once added I go this message.

Text

Description automatically generated

After inserting it I needed to update the “volume” key-value pair that was identified by the string “ticker”. The update command changes or edits the existing documents. I created the python file named update document. This allows the user to submit the value, and the application will display the top ten results in the documents. I was then met with this message after updating.

Text

Description automatically generated

**Document Retrieval**

Now getting into the more complicated stuff. First is finding the statement arguments.

Text

Description automatically generated

I needed to find documents with 50-day moving average for supplied values. To find I used find({}) statement. And whatever parameters I need it to search by. This was all done on a python findDocuments.py document. Next was the python document to find documents for the document key “Sector” to find the grouped “industry” documents if need be. This asks the user to provide a string to find the document. Like query = {“Industry” : industryname}. This would obtain the ticker symbols with industry matches the string.

Text

Description automatically generated

I then need to write a MongoDB aggression pipeline statement that transformed documents into aggregated results using multiple pipeline stages. The method would find documents for which the input strings match the document key “sector” nand return the outstanding shares grouped by “industry”. { ‘$match’: {“Sector”:sectorname }} first part that allows only the documents with sectorname to be sent to the next part, {‘$group’:{‘\_id’: “$Industry”,’Total Outstanding Shares:’:{‘$sum’: “$Shares Outstanding”}}} this step the pipeline groups the documents that passed part one using the industry key. This gives us groups of industries with a total outstanding shares.

Graphical user interface, text

Description automatically generated

**Advanced Programming Project**

I used Python to develop the RESTful.API. I then created a new stock. The API allows the user to add new stock documents to the database by just providing the url.

Text

Description automatically generated

After the document is added to the database as the above image shows, the function will take the data from the document and add it to the collection stock. Next is to update the database, the update functionality edits the document, with the given information from the user. Using the URL Curl -H “Content-type:application/json” -X post -d ‘{“Volume”:40,”Sector”:”MySector”,”Industry”:industryName”}’ <http://localhost:8080/update/BRLI>. This URL and code provides an additional parameter BRLI as a ticker value. This will help to filter documents so it only edits what you want it to.

A picture containing text

Description automatically generated

After the update I then worked on getting delete to work as well. The delete function allows the user to give a criteria for the application to delete certain documents from the database using an URL. <http://localhost:8080/remove/TA> giving me the below image after it got deleted.

Text

Description automatically generated

Finally the application is capable of summarizing the new document groups using the URL curl-H”Content-Type:application/json”-X Post -d’{“list”:”[AA,TA,BRLI,GA]”}’ <http://localhost:8080/stockSummery/> the URL has the Ticker Value list we are looking for, so that the application groups the documents up and produces a summary.