INFS-740 Project

Anjali Sharma [asharm32@gmu.edu](mailto:asharm32@gmu.edu)

Table of Contents

[Technology Stake 1](#_Toc164775771)

[README 2](#_Toc164775772)

[Presentation 3](#_Toc164775773)

[MongoDB Database 3](#_Toc164775774)

[Setup 3](#_Toc164775775)

[Collections 3](#_Toc164775776)

[Complex Queries: 7](#_Toc164775777)

[Get Data for Home page: 7](#_Toc164775778)

[Get companies with Market Capitalization Over $1 Trillion: 7](#_Toc164775779)

[Get Highest Price from Historical Data Since 2023-01-01: 8](#_Toc164775780)

[Get All Price History for Comparison and Verification Against the Summary Report of Highest Prices: 8](#_Toc164775781)

[Python FLASK RESTful APIs 8](#_Toc164775782)

[CRUD Operations and Additional Query APIs 9](#_Toc164775783)

[Python Application for Machine Learning Model 14](#_Toc164775784)

[Angular Frontend Web Application Setup 16](#_Toc164775785)

[Testing of APIs - CRUD Operations- get, post, put, delete via postman. 32](#_Toc164775786)

[Integration of API with Frontend Angular Application 34](#_Toc164775787)

[Running the project 41](#_Toc164775788)

[Project Overview Video Recording 43](#_Toc164775789)

[References 43](#_Toc164775790)

# Technology Stake

* **Backend Database**: MongoDB
* **REST API**: Python Flask
* **Web User Interface**: Angular Material Web Development Framework
* **Visualization**: D3.js Visualization Libraries
* **Machine Learning**: Scikit-learn, Matplotlib, Seaborn, NumPy, Pandas

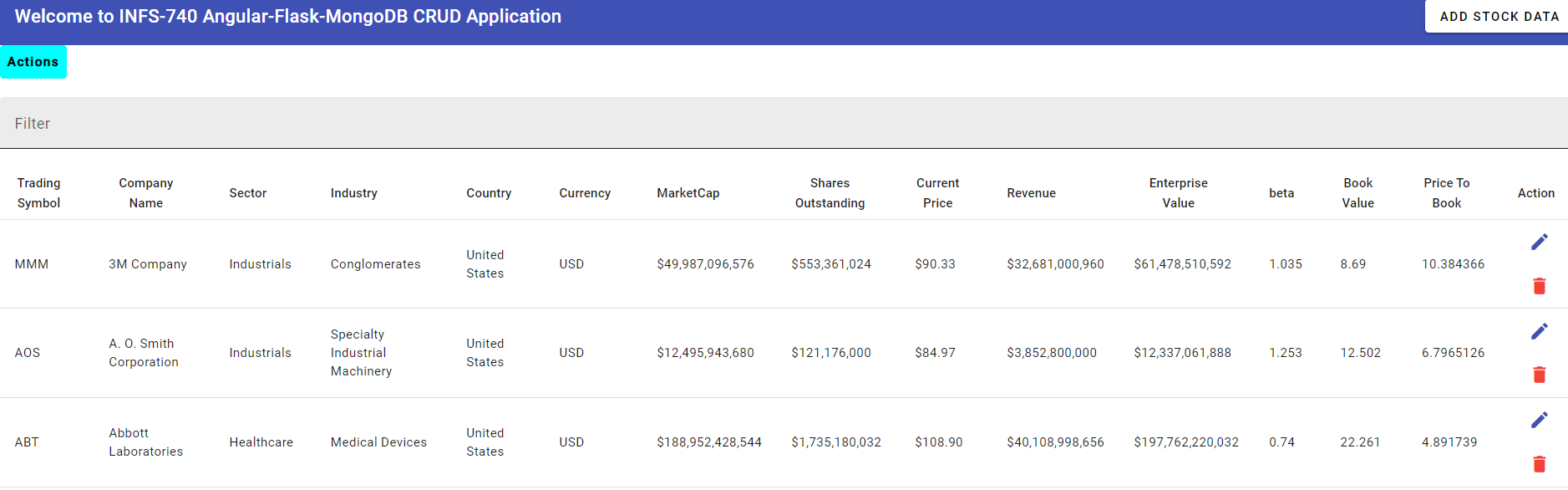
# README

1. Unzip the File:

* Unzip the downloaded project file to a desired location.
* Follow the respective sections of MongoDB, API and Angular Frontend in the document to perform setup.

1. Navigate to the Project Directory:

* Open a terminal or command prompt.
* Enter the following command to navigate to the project-app folder:
  + cd project-app
* API Setup:
  + Install Required Dependencies:
    - pip install Flask pymongo pandas numpy scikit-learn matplotlib seaborn
* Start the API:
  + python api.py
* The application will start and produce message
  + (\* Running on [http://127.0.0.1:5000\*](http://127.0.0.1:5000*))
* At this point, the API server is successfully started.
* Once API started, navigate to the **Web Application Directory**:
  + Change to the web subdirectory within project-app:
    - cd project-app/web
    - ng serve
* After some time, you should see “Compiled successfully”.
* Verify <http://localhost:4200/>
* You should see stock data loaded on the UI, confirming that the MongoDB, API, and the front end are functioning correctly.



**The submission zip contains:**

1. A setup instructions document.
2. All source files, including the REST API, Angular Frontend, MongoDB Archive for setting up the database, and PowerPoint presentation.
3. Screenshots of all components of the application’s functionality
4. A video recording of the application’s functionality, titled “**video1133150885.mp4**”.

# Presentation

Open the PowerPoint presentation titled **INFS-740 Project Presentation.pptx** included with the submission. This will display all components of the project.

# MongoDB Database

## Setup

* Install MongoDB tools, including mongosh. Access the command line to proceed.
* Restore the MongoDB database from the provided archive (finance\_db\_infs740) located in the submission folder. This operation will recreate the database on localhost with all four collections used in the application.
* Run the following commands:
  + Open mongosh by typing mongosh in your command line.
  + Execute the database restore command
  + mongorestore --archive="finance\_db\_infs740" --host=localhost --port=27017

## Collections

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Complex Queries:

### Get Data for Home page:

Access data by querying the following three collections:

* stock\_metadata
* stock\_financial\_info
* stock\_default\_key\_stats

### Get companies with Market Capitalization Over $1 Trillion:

Start by using the stock\_financial\_info collection as the primary source to filter trading symbols representing companies with a market capitalization exceeding $1 trillion. Then, join this data with the stock\_metadata and stock\_default\_key\_stats collections to gather comprehensive data for the user interface.

**Collections involved**:

* stock\_financial\_info
* stock\_metadata
* stock\_default\_key\_stats

### Get Highest Price from Historical Data Since 2023-01-01:

Use the stock\_history\_data collection as the primary source to aggregate and summarize data. Group the data by symbol, and then join it with the stock\_metadata and stock\_financial\_info collections to compile the necessary information for the user interface.

**Collections involved**:

* stock\_history\_data
* stock\_metadata
* stock\_financial\_info

### Get All Price History for Comparison and Verification Against the Summary Report of Highest Prices:

Access data from four collections to compare and check results with the summary report of the highest price page:

* stock\_metadata
* stock\_history\_data
* stock\_financial\_info
* stock\_default\_key\_stats

# Python FLASK RESTful APIs

I have developed a Python Flask REST API application for fetching, adding, deleting, and updating data through an Angular Material web interface. The file **api.py** is included in the submission zip. Please ensure to install Flask, PyMongo, pandas, NumPy, scikit-learn, matplotlib, and seaborn by using the following pip install command:

**pip install Flask pymongo pandas numpy scikit-learn matplotlib seaborn**

A screenshot of a computer program

Description automatically generated

A close-up of a computer code

Description automatically generated

### CRUD Operations and Additional Query APIs

#### Get Data

A computer code with many text

Description automatically generated with medium confidence

#### Update Data by trading symbol

A screenshot of a computer code

Description automatically generated

#### Adding Data

A screen shot of a computer code

Description automatically generated

#### Deleting Data

A screen shot of a computer code

Description automatically generated

#### Get Companies with Market Capitalization Over $1 Trillion

A screenshot of a computer code

Description automatically generated

#### Get data for the Highest Price

A computer screen shot of a program

Description automatically generated

#### Get Complete Price History

A computer screen shot of a program

Description automatically generated

#### Get Full Price History by Trading Symbol

A screenshot of a computer code

Description automatically generated

#### Get Data for Visualization

A screen shot of a computer code

Description automatically generated

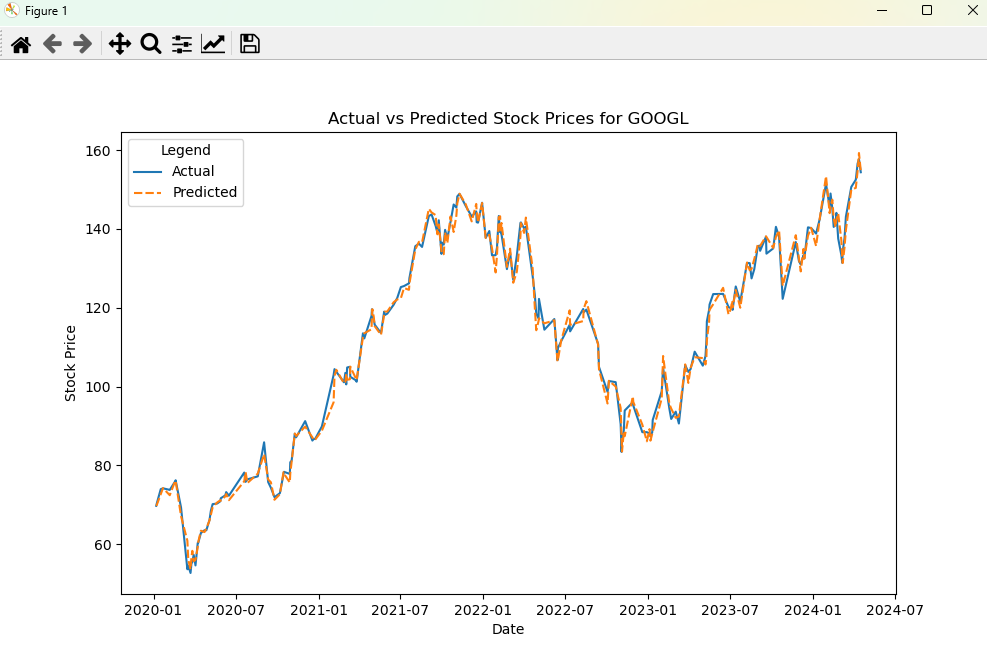
#### Get Predictions for the Next Day’s Closing Price

A screenshot of a computer program

Description automatically generated

# Python Application for Machine Learning Model

The file **infs\_740\_ml\_linear\_regression.py** is included in the submission. In this application, I have developed a Linear Regression model and calculated the Root Mean Square Error (RMSE). Based on the specified symbol, the application dynamically loads data, trains the model, makes predictions, calculates RMSE, and generates plots using the Matplotlib and Seaborn libraries



A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a computer screen

Description automatically generated

# Angular Frontend Web Application Setup

After successfully setting up the MongoDB database and starting and verifying the REST API application by following the steps provided in their respective sections, proceed with these steps to setup the Angular web application:

* Navigate to the web folder:
  + cd project-app/web
* Add Angular Material to your project:
  + ng add @angular/material
* Install the necessary dependencies from package.json:
  + npm install (it will install the dependencies from package. json)
* Start the Angular server:
  + ng serve
* After a short wait, you should see the application compile and start serving at

localhost:4200.

A black background with white text

Description automatically generated

* Verify the Application:
  + Access the application by visiting:
  + <http://localhost:4200/>

#### Home Page:

A screenshot of a computer

Description automatically generated

The web application's homepage offers sorting, filtering, and pagination features to enhance analysis. Additionally, Angular Material design ensures responsiveness for a seamless user experience across devices.

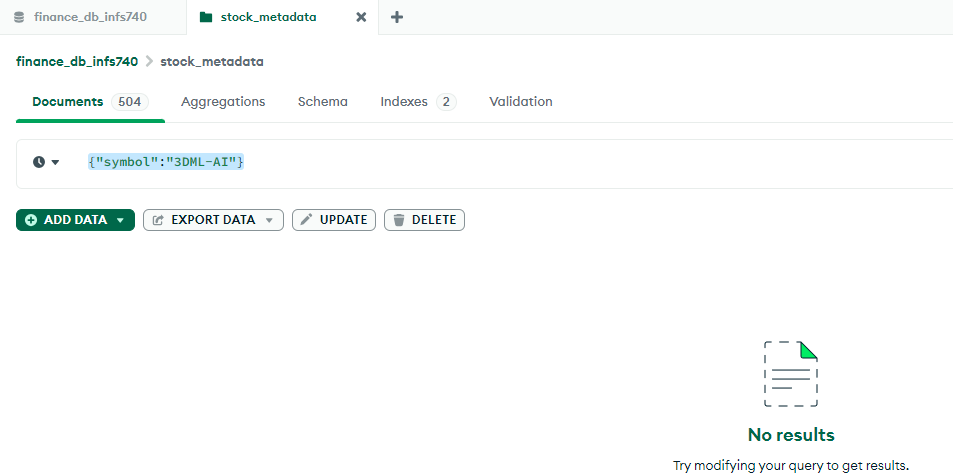
#### Add, Update, Delete from Web User Interface

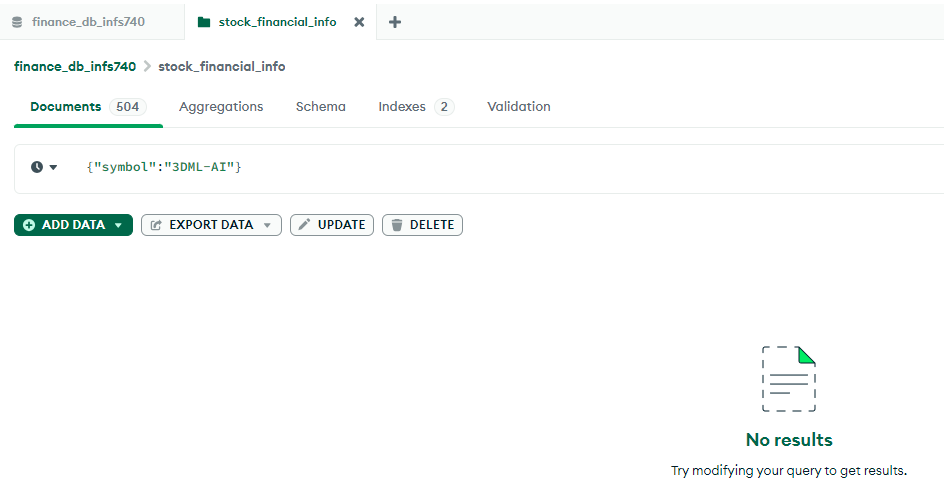
Let’s use the same symbol that does not currently exist to do full cycle testing of Add, Update, and Delete operations.

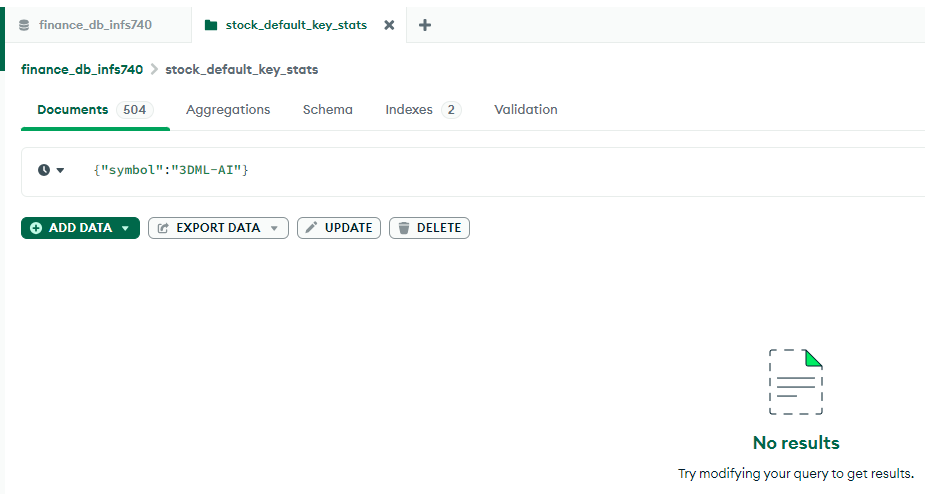
Verify first symbol **3DML-AI** is not present in the UI filter search and across all three MongoDB collections that serve as the data source for the frontend home page display:

A screenshot of a computer

Description automatically generated







Now let’s add the symbol **3DML-AI** from the Web User Interface.

* Click on **ADD STOCK DATA** on the right side of the tool bar. The application will present an empty stock entry data form.
* As this is a new entry form, the submit button will display **Save** as text.

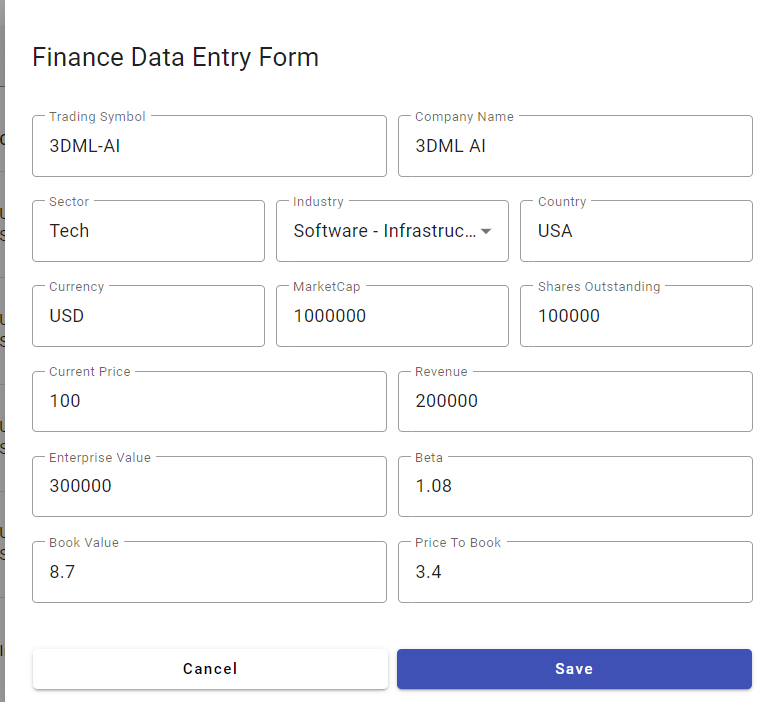
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Go ahead and enter data for symbol **3DML-AI** and click on **Save**.
* **Cancel** Operation - If you decide to cancel the operation, click on the **Cancel** button. This action will close the entry form and return the user to the home page.

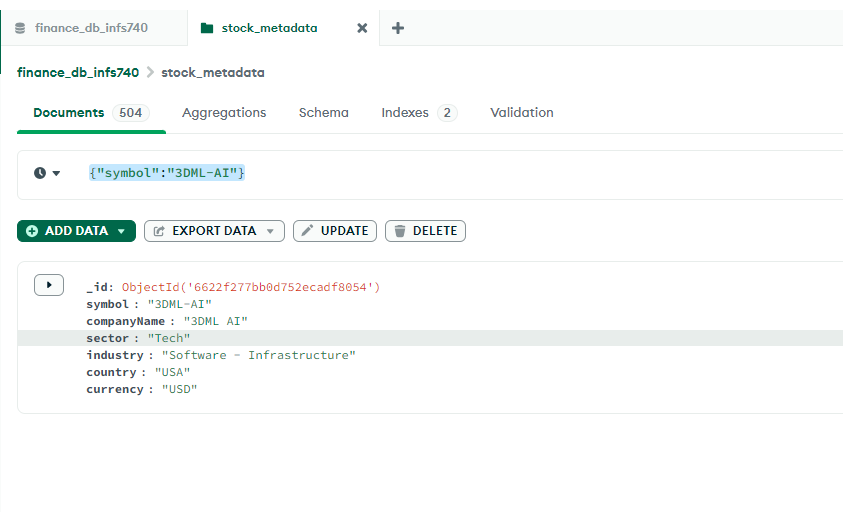


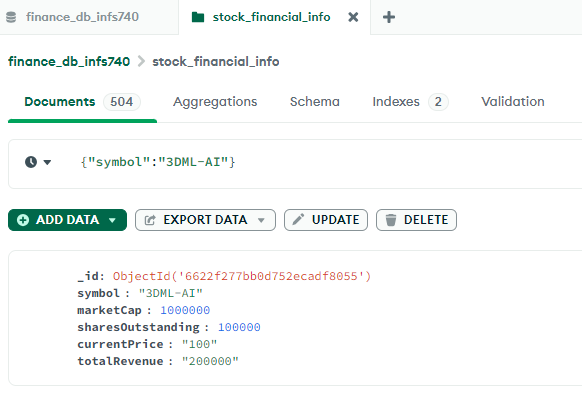
* Now, let’s verify that the symbol **3DML-AI** is available in the UI and in all three collections in the MongoDB, which are the sources of the home page information.
* Enter 3DML-AI in the Filter on the home page and verify:

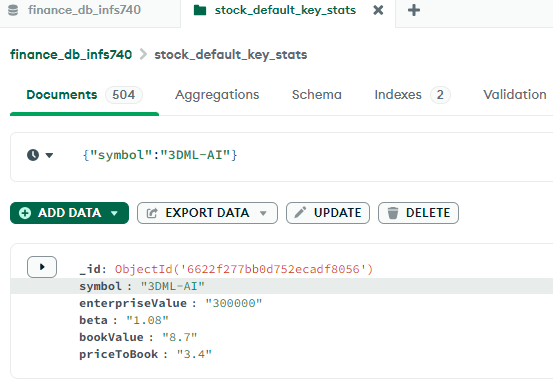
**A screenshot of a computer

Description automatically generated**

* Verify from the backend:







Data is entered to all respective collections.

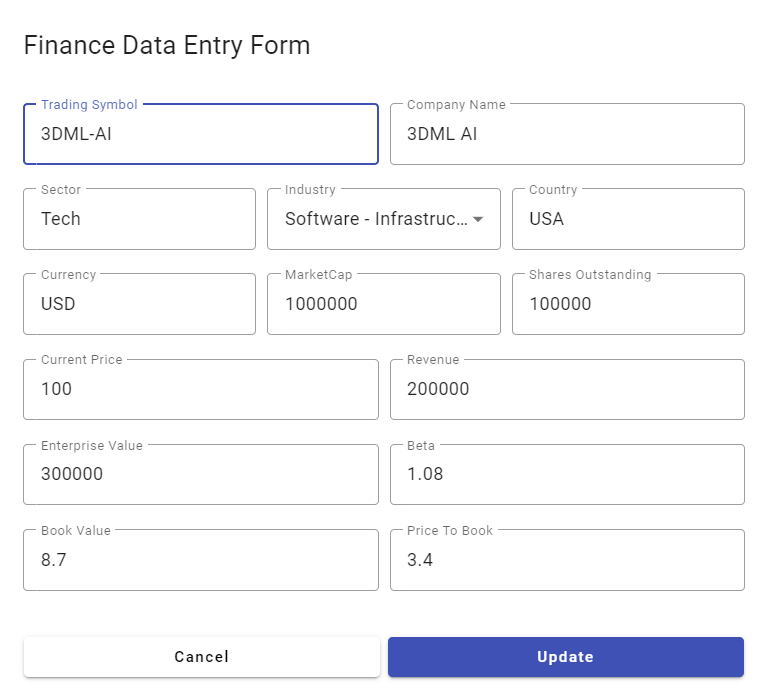
Now, we will use the same symbol **3DML-AI** for our Update and Delete operations.

Filter the result for **3DML-AI** by entering the text in Filter field. Once the record is displayed, click on the pencil icon to edit the information for the trading symbol. The application will present the form to edit the information.

This time, the submit button text will show **Update** as we are editing the existing record.

A screen shot of a phone

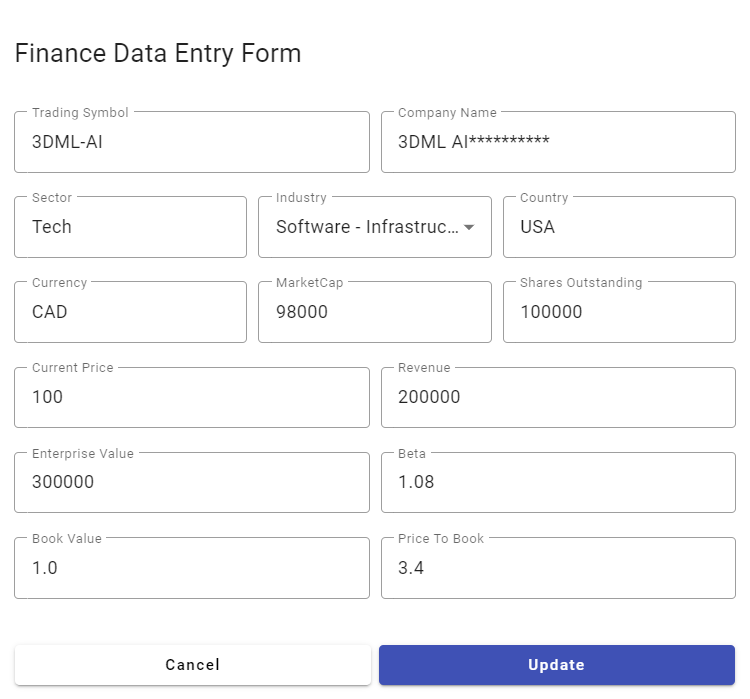
Description automatically generated



Let’s go ahead and click on the “**Update**” to edit the information belonging to all source collections so that we can verify data is simultaneously updated to all collections.

We changed the Company Name, Currency, MarketCap and Book Value fields. Let’s go ahead and click on **Update** and verify updated information.

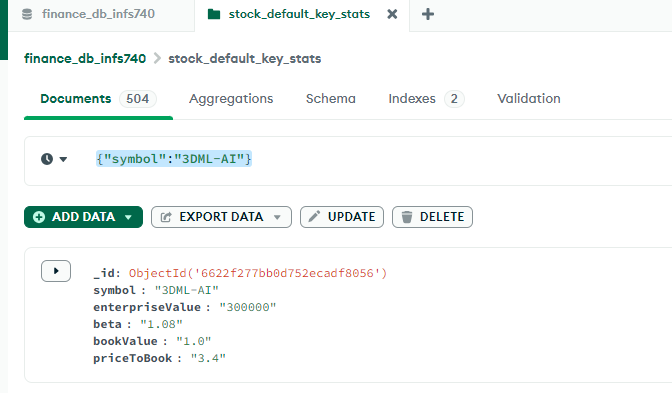
**Cancel** Operation - If you decide to cancel the operation, you can click on the **Cancel** button. This action will close the entry form and return the user to the home page, leaving the original data unchanged.



To ensure our edits have been applied correctly, let's perform verifications by filtering for the symbol. Check the edit screen, UI record, and MongoDB database to confirm that the updates have been successfully propagated across all relevant collections.

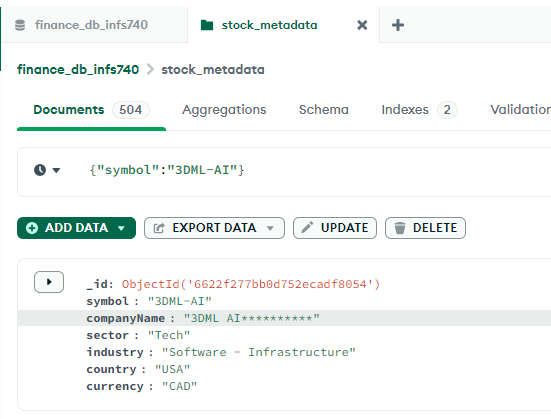
A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



The data is updated to UI and all three collections.

Now, let’s perform Delete operation on **3DML-AI** and verify the results. The information for symbol **3DML-AI** should be deleted from all places.

**Locate the Record**: Filter and find the record for the symbol "3DML-AI" in the system.

**Perform the Delete Operation**: Click on the trashcan icon corresponding to the row of "3DML-AI". This should trigger the deletion process.

**Observe the Success Message**: Immediately after deletion, watch for the success message to appear, confirming the operation was successful.

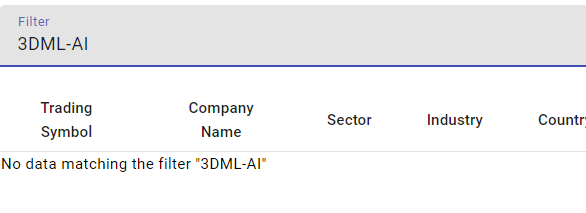
**Message Disappearance**: Note that the success message should automatically fade away or disappear after 2 seconds, ensuring that it doesn't obstruct the user interface or require any user interaction to close.

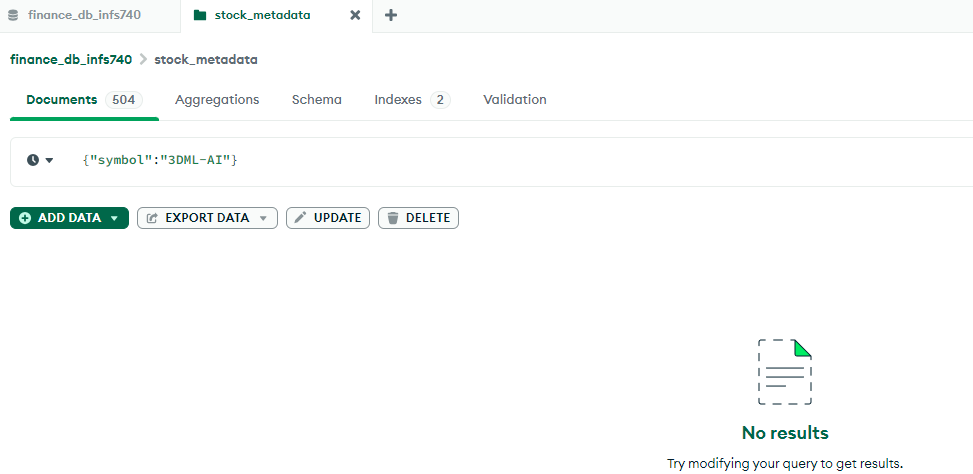
**Verify the Results**: After deleting, verify that the information for "3DML-AI" has been completely removed from all places in the system. This includes checking the UI, and backend MongoDB collections.

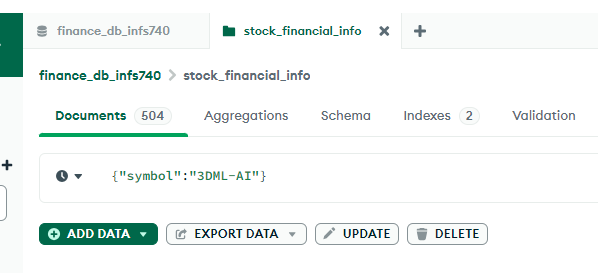
The following screenshots show that the symbol **3DML-AI** has been successfully deleted from all relevant parts of the system, including both the UI and the backend database.

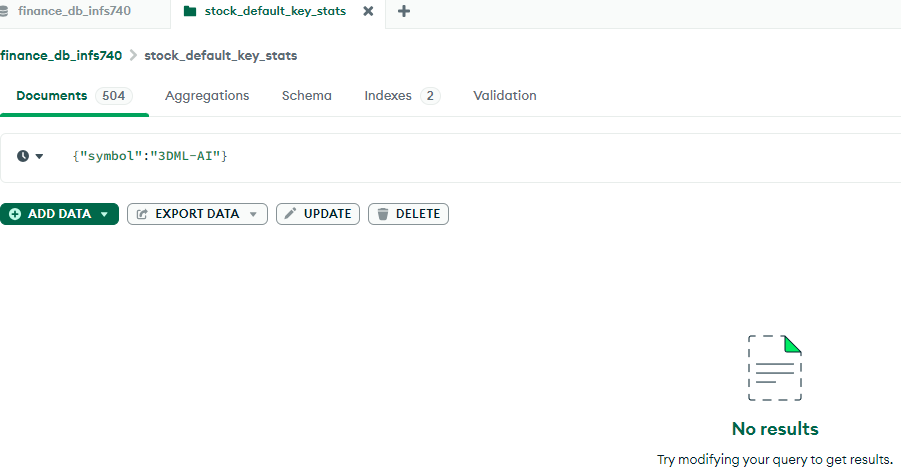
A screenshot of a phone

Description automatically generated









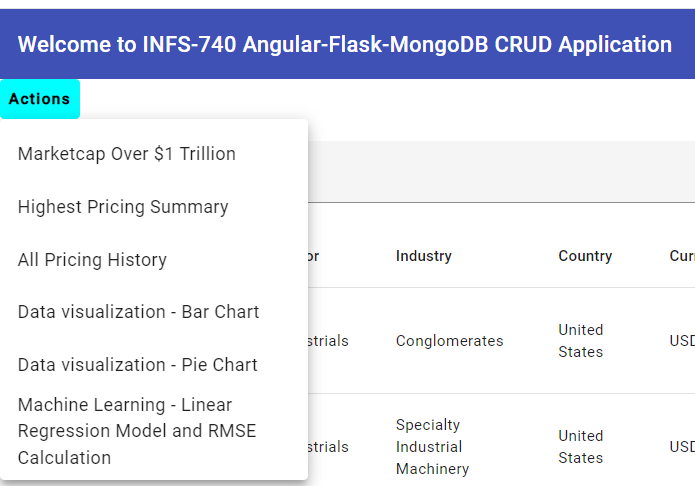
Please note that after every successful **Add, Update, or Delete** operation, the application automatically displays a message confirming the action. This message is designed to disappear after 2 seconds without requiring any user interaction. This is a useful feature for enhancing user experience by providing immediate feedback while keeping the interface clean and uncluttered.

#### Actions from UI:

Click on the **Actions** button to see a menu that lists different actions:

A blue and white sign

Description automatically generated



##### Market Capitalization Over $1 Trillion Dollars

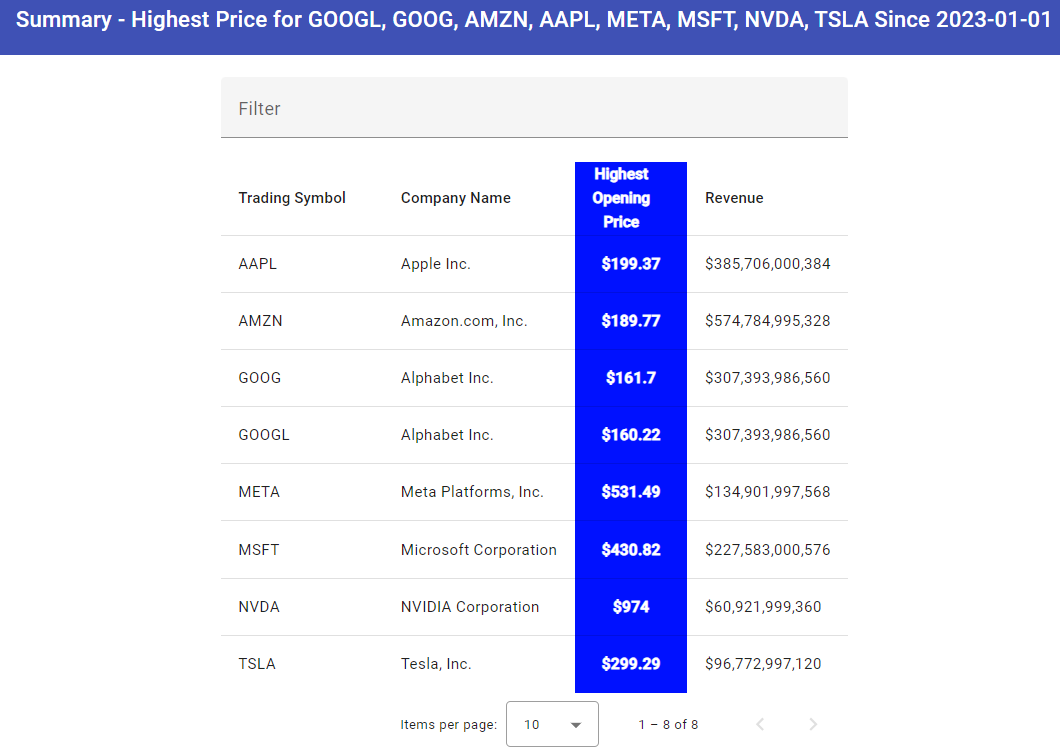
Click on the **'MarketCap Over $1 Trillion**' action item. The application will then query the 'stock\_financial\_info' collection, filter the results to include only those companies with a market capitalization over $1 trillion, and join this data with information from the 'stock\_metadata' and 'stock\_default\_key\_stats' collections to populate the page.

A screenshot of a computer

Description automatically generated

##### Highest Pricing Summary

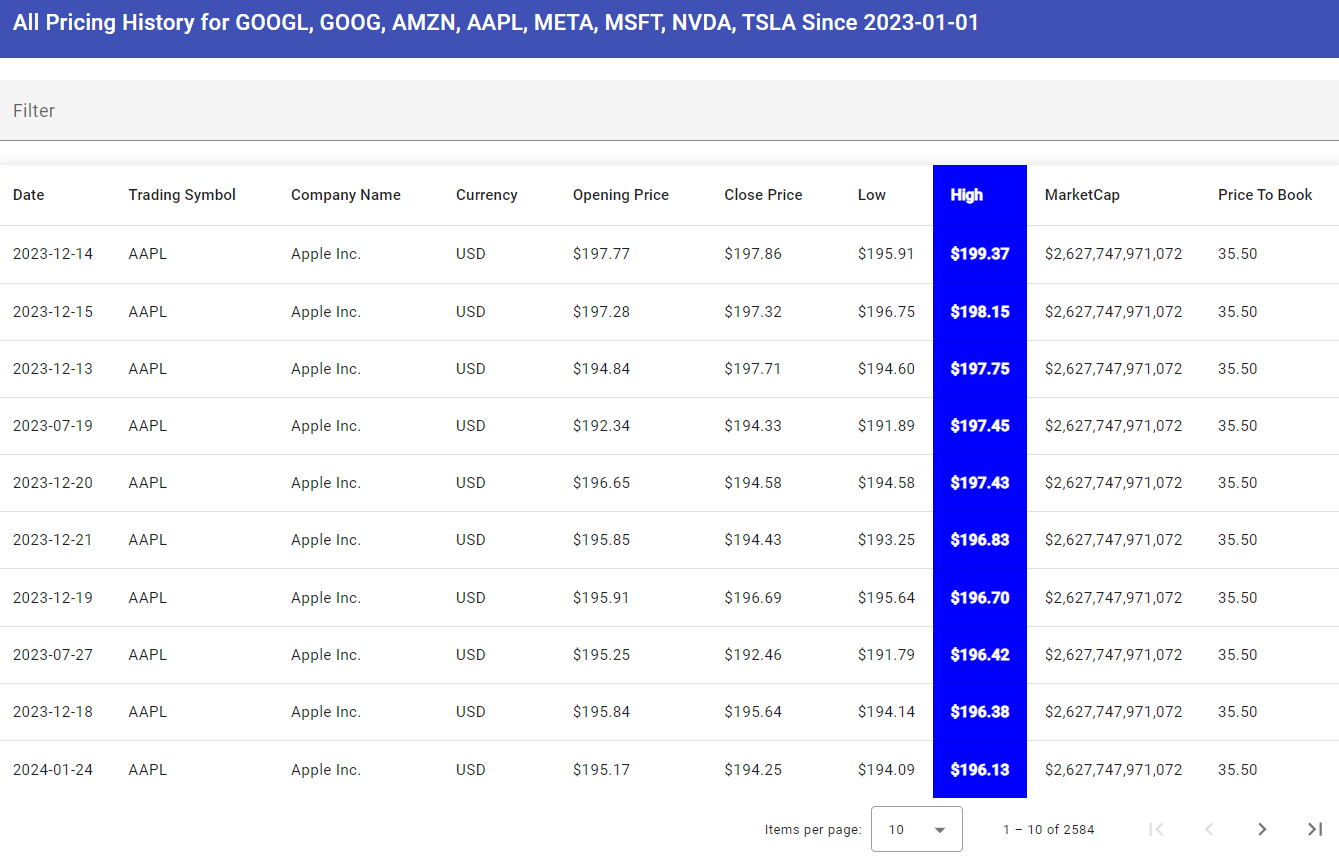
Select the **'Highest Pricing Summary'** menu item. The application will query the 'stock\_history\_data' collection, aggregate the data, perform grouping by symbol, and get the maximum price for each trading symbol over the past year. It will then join this data with 'stock\_metadata' and 'stock\_financial\_info' collections to compile and display the relevant information on the page.



##### All Pricing History

Select the ‘**All Pricing History’** menu item, the application will bring data from four collections stock\_history\_data, stock\_metadata, stock\_financial\_info, and stock\_default\_key\_stats.

This feature displays the complete pricing history for a selected symbol, sorted in descending order by the High price. For example, if you filter for 'AAPL' on this page, you will see that Apple has 323 records. The highest price listed will match the numbers from the **'Highest Pricing Summary**' page.



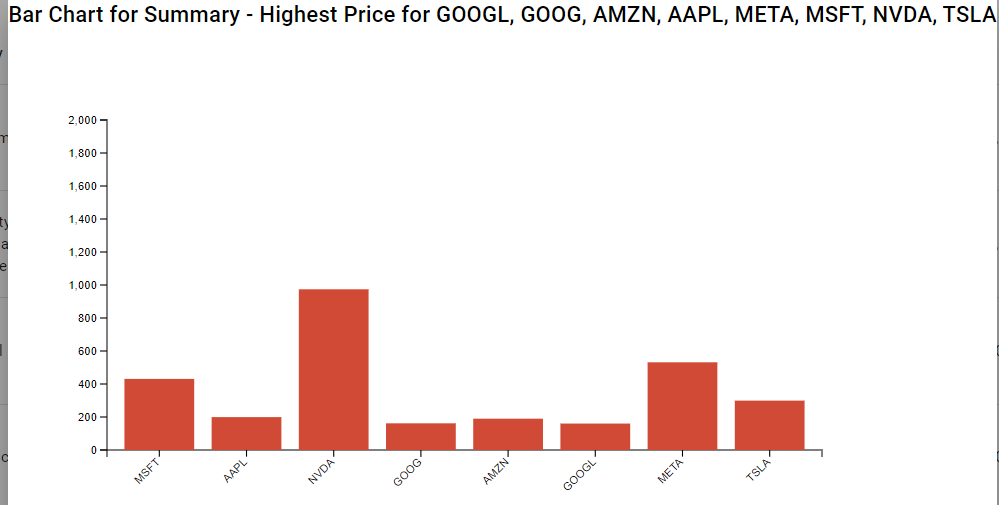
Filter for AAPL on this screen, you will see 323 records for pricing history.



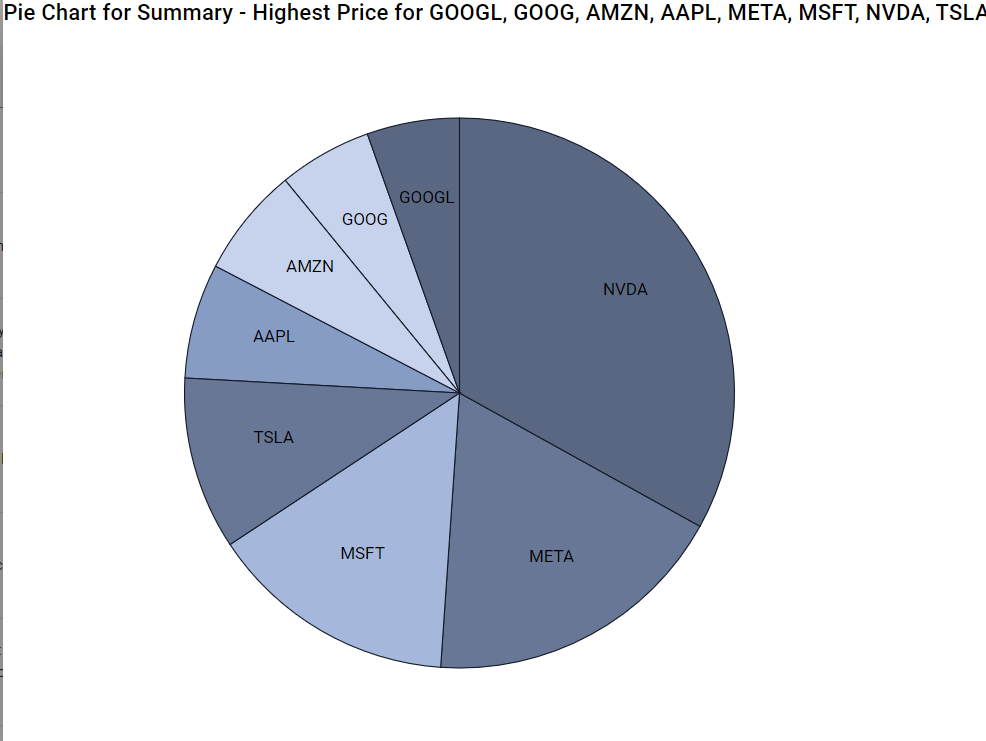
##### Data Visualization

Data Visualization is performed on the Highest Pricing Data.

Click on **Data Visualization - Bar Chart** to view a comparison of the highest pricing standings of various companies. This visualization helps illustrate the highest prices reached by each company, providing a clear and comparative insight into their financial performance.



Click on the **Data Visualization – Pie Chart** to view the highest pricing standings of various companies in a different visual format. This pie chart offers a marketing representation of each company's highest price relative to others, providing an intuitive and comparative insight into their financial performance.

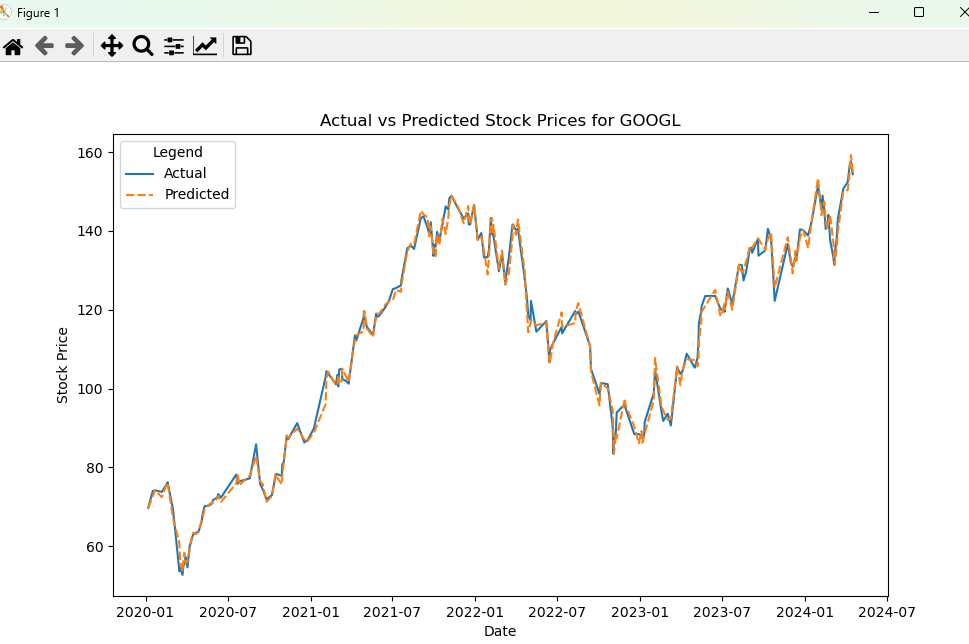


##### Machine Learning – Linear Regression Model and RMSE Calculation

Execute the python application file **infs\_740\_ml\_linear\_regression.py** that is included with the submission to see the Root Mean Squared Error calculation, Linear Regression model training, predictions and plotting using matplotlib and seaborn libraries.

A screenshot of a computer

Description automatically generated

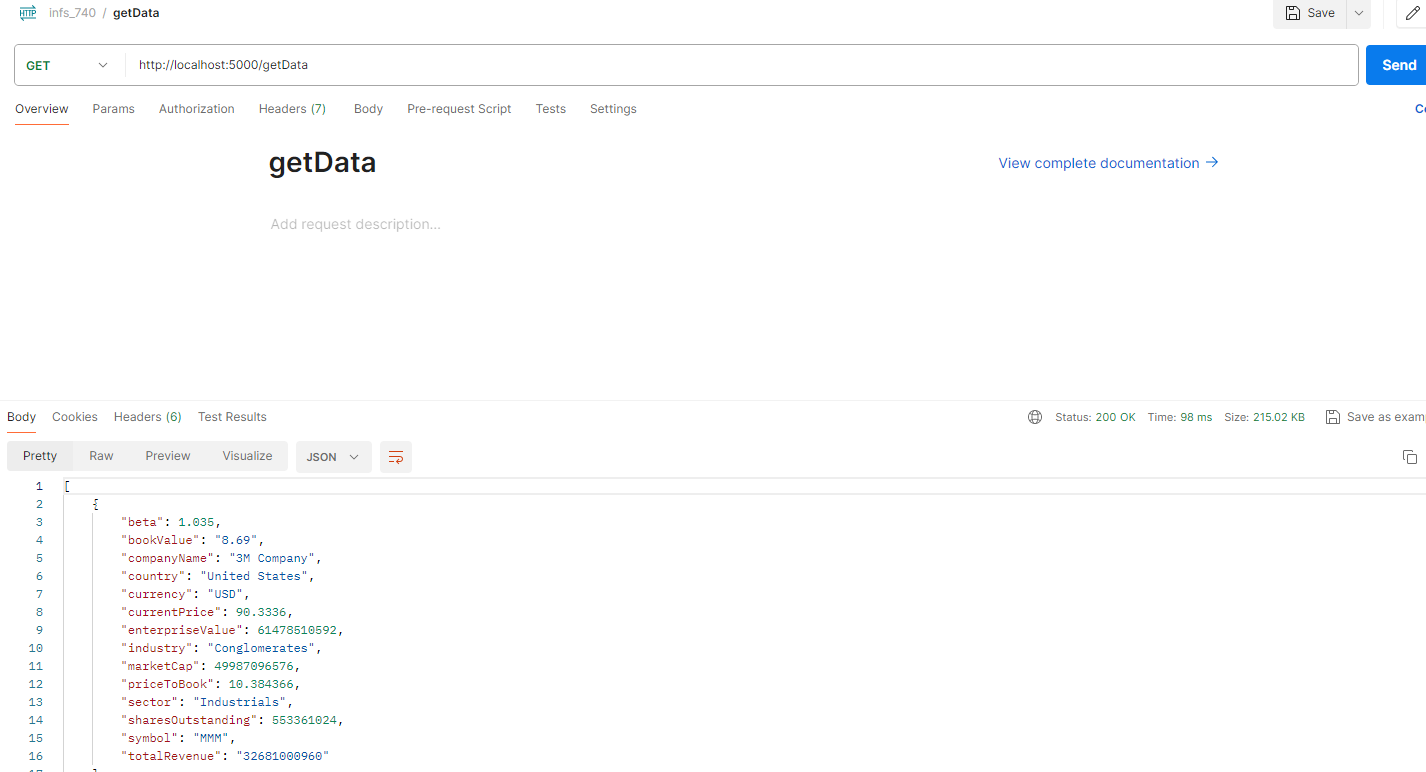


# Testing of APIs - CRUD Operations- get, post, put, delete via postman.

Postman API testing tool is used to verify the functionality of APIs for CRUD (Create, Read, Update, Delete) and other UI operations. The Flask REST API is deployed at <http://127.0.0.1:5000/>

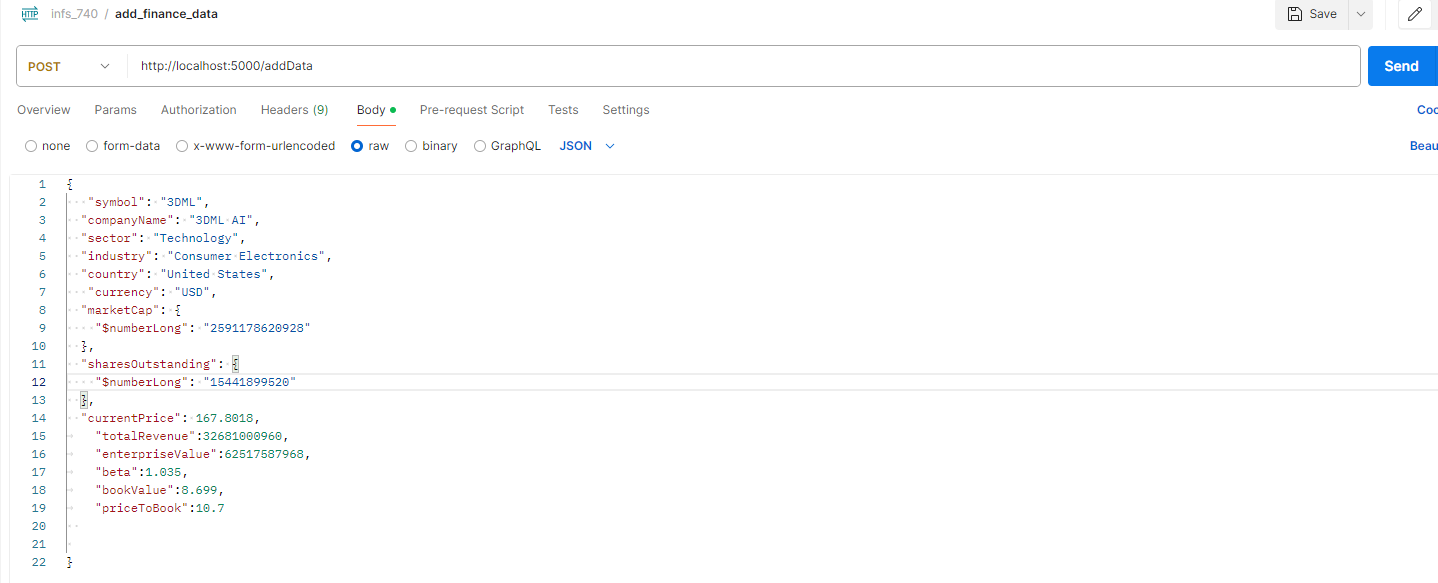
#### Get Data

<http://localhost:5000/getData>



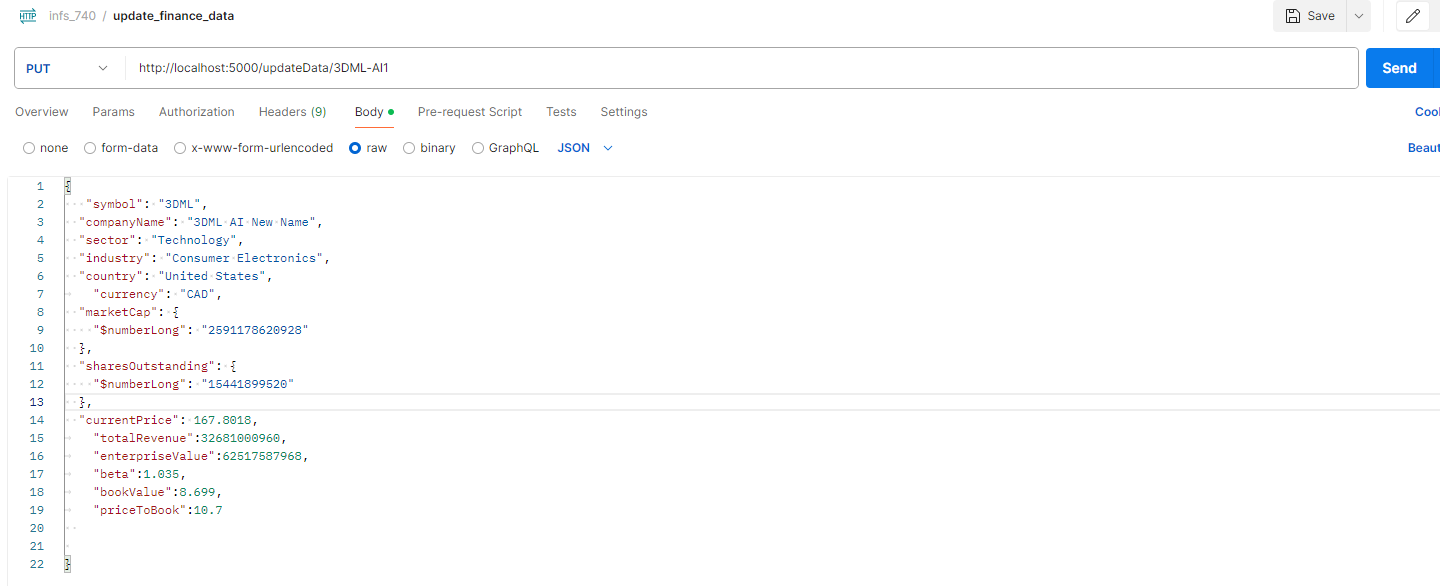
#### Add Data

<http://localhost:5000/addData>



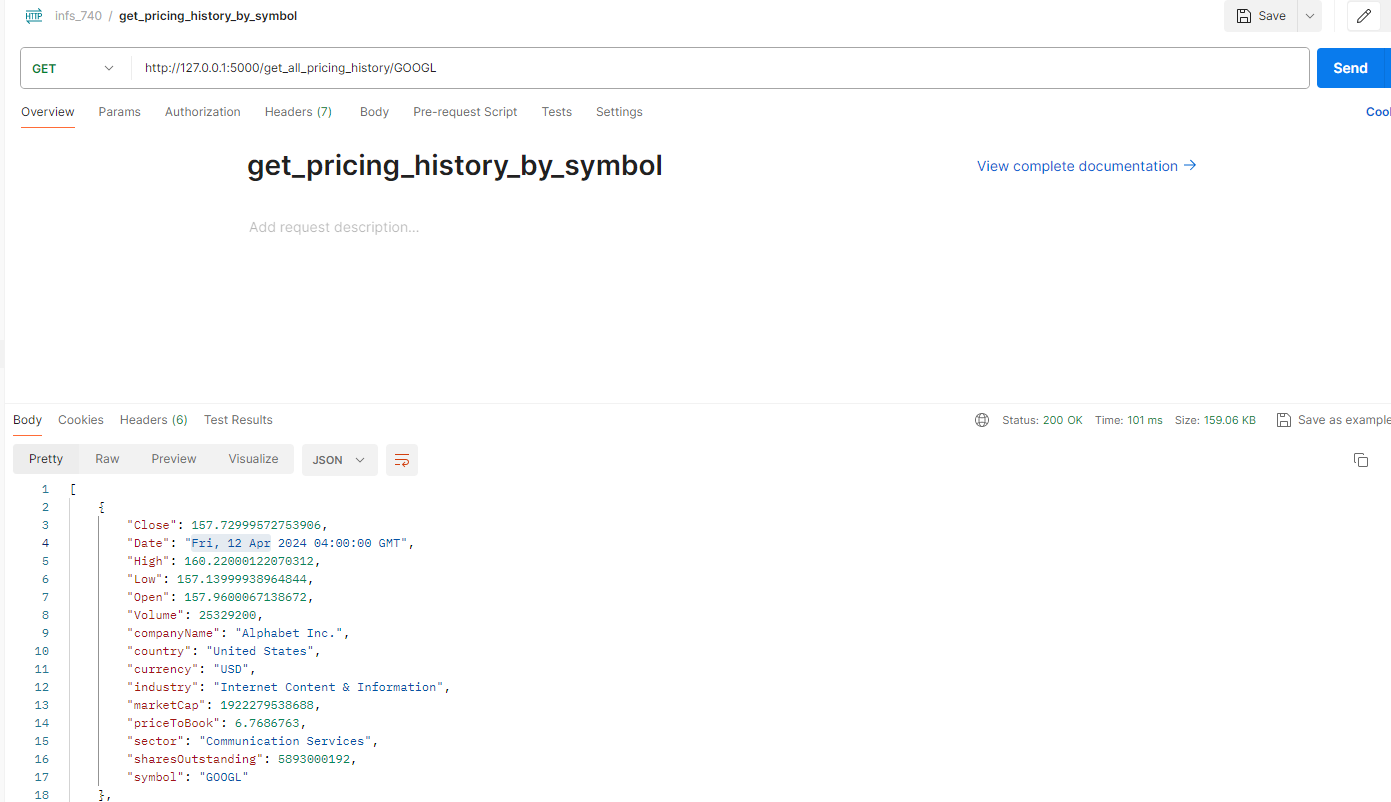
#### Update Data

<http://localhost:5000/updateData/3DML>



#### Get Pricing History for a given trading symbol

<http://127.0.0.1:5000/get_all_pricing_history/GOOGL>



#### Delete Data for a given trading symbol

<http://localhost:5000/deleteData/3DML> where 3DML is a trading symbol

# Integration of API with Frontend Angular Application

The Angular frontend application interacts with the backend by calling REST API methods (GET, POST, PUT, DELETE) mapped to specific URLs. Angular controllers utilize the global **StockService** service to perform CRUD operations—getting, posting, updating, and deleting data from/to the MongoDB database via RESTful Flask APIs. Below are several screenshots that illustrate these interactions within the Angular front-end application.

**StockService**

A screenshot of a computer code

Description automatically generated

**Components**:

A screenshot of a computer code

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen shot of a code

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

# Running the project

Please follow the steps below in the specified order to ensure all components of the project are operational:

1. **MongoDB Setup**: Ensure the Mongo database is set up using the archive included with the submission, **finance\_db\_infs740**. Refer to the MongoDB database setup instructions to restore the database with this archive.
2. **API Verification**: Ensure the API is actively running and serving endpoints at <http://localhost:5000/>.
3. After successfully starting and verifying MongoDB database and API (steps #1 and #2), perform the steps below.
4. **Web Project Application Setup**:

* Navigate to the 'web' folder within the **project-app** directory.
* Open a terminal or VS Code and run the following commands:
  + cd project-app/web — Changes the directory to the web folder.
  + npm install — Installs all required dependencies first.
  + ng serve — Compiles the application and provides a URL to access it.

A computer screen shot of a black screen

Description automatically generated

* **Access the Application**: After successful compilation, follow the URL provided to access the home page of the application. This page will display financial information. <http://localhost:4200/>

A screenshot of a computer

Description automatically generated

##### Validation

* **CRUD Operations**: Perform Add, Update, and Delete operations within the web application to ensure these functionalities are working correctly.
* **Menu Items**: Click on the various menu items under the **'Actions'** menu to verify that different queries are executed properly, and that the application is serving respective pages as expected.

# Project Overview Video Recording

Play the video **video1133150885.mp4** included in the submission folder to view a demonstration of all components of the INFS-740 web application. The video will start after a few seconds and provide a comprehensive explanation of how each part of the application functions.

# References

* INFS-740 Lectures and Support Material
* Angular documentation: [Angular Official Site](https://angular.io/)
* Angular Material Getting Started Guide: [Material Design for Angular](https://material.angular.io/guide/getting-started)
* MongoDB Documentation: [MongoDB Official Documentation](https://www.mongodb.com/docs)