

INVESTO- A STOCK MARKET ANALYSIS APPLICATION

CASE STUDY



Prepared By- Saiom Pal

February 7, 2022

BaJAJ FINANCE

Contents

[1. Introduction 3](#_Toc95085323)

[2. Tech Stack 3](#_Toc95085324)

[3. Application architecture 3](#_Toc95085325)

[4. Angular Component Hierarchy 4](#_Toc95085326)

[5. Detailed Component Descriptions 4](#_Toc95085327)

[5.1. Registration and Login 4](#_Toc95085328)

[5.2. Dashboard 6](#_Toc95085329)

[5.3. Charts 7](#_Toc95085330)

[5.4. Data Metrics 7](#_Toc95085331)

[5.5. Watchlist 8](#_Toc95085332)

[5.6. Contact Us 8](#_Toc95085333)

[6. Integration with Backend 9](#_Toc95085334)

[7. Authentication and Authorization 10](#_Toc95085335)

[7.1. GENERATING TOKENS: 10](#_Toc95085336)

[7.2. STORING THE TOKENS IN LOCAL STORAGE: 10](#_Toc95085337)

[7.3. INTERCEPTING AND VERIFYING THE TOKEN: 10](#_Toc95085338)

[8. Testing APIs using Postman 11](#_Toc95085339)

[9. Conclusion 13](#_Toc95085340)

# Introduction

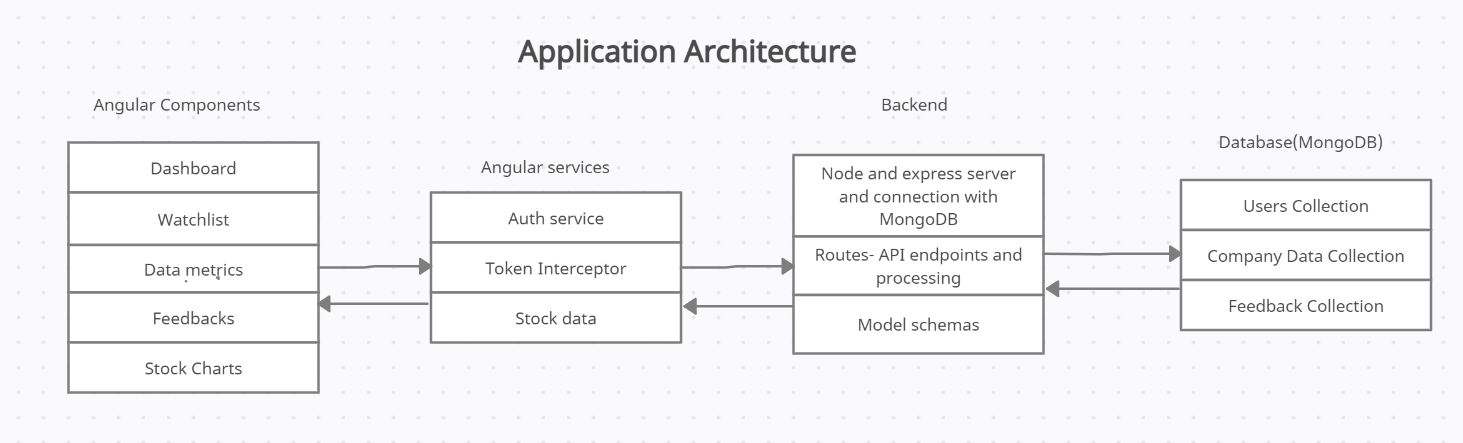
Investo is a stock market analysis application that allows users to view stock charts of any company listed on the New York Stock exchange. It provides key financial metrics on any company of choice along with analyst’s predictions. It allows users to create their own watchlists and track any stock’s performance. It highlights the stock price along with the percentage change in the last 24 hour during the recent recorded trading session. It can be edited according to the user’s needs. It provides a nice dark themed user experience as most of the trading websites and a very friendly interface that is easy to comprehend and play around with.

# Tech Stack

1. FRONTEND – ANGULAR (13.1)
2. BACKEND- NODEJS AND EXPRESSJS (17.0.14)
3. DATABASE MANAGEMENT SYSTEM- MONGODB

# Application architecture

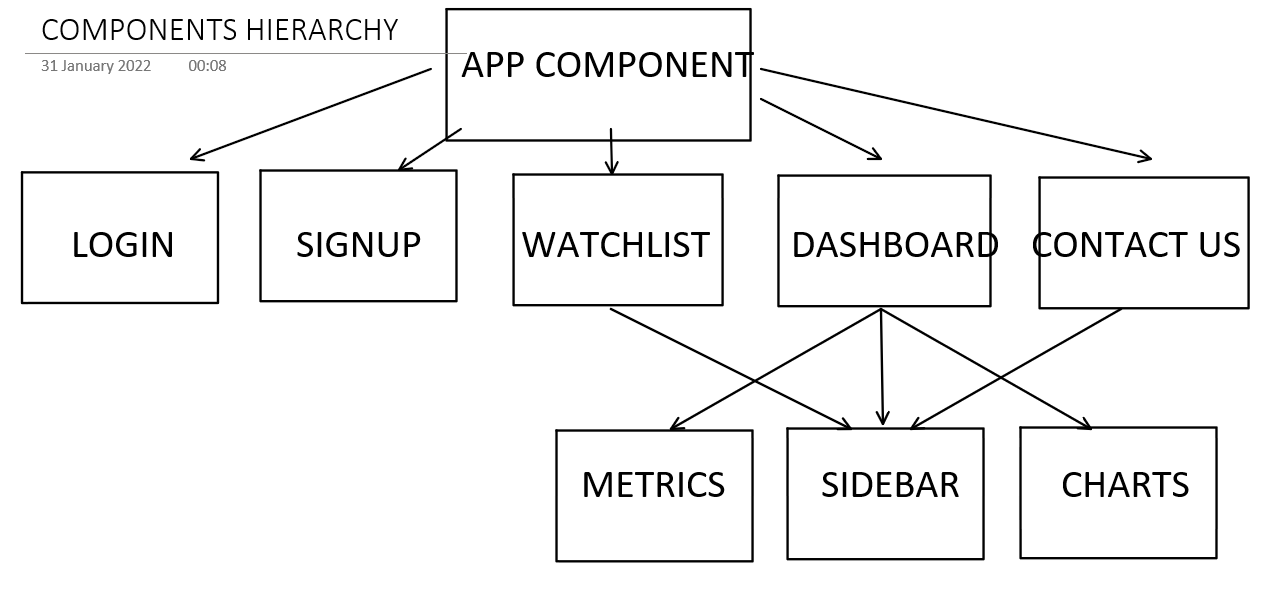
The below diagram shows the interaction between different parts of the application namely frontend- Angular components and services and the backend – Node server, API endpoints and MongoDB database.



Architecture diagram of the complete application

# Angular Component Hierarchy

The below diagram shows the component hierarchy i.e.. the parent child relationship among the components.



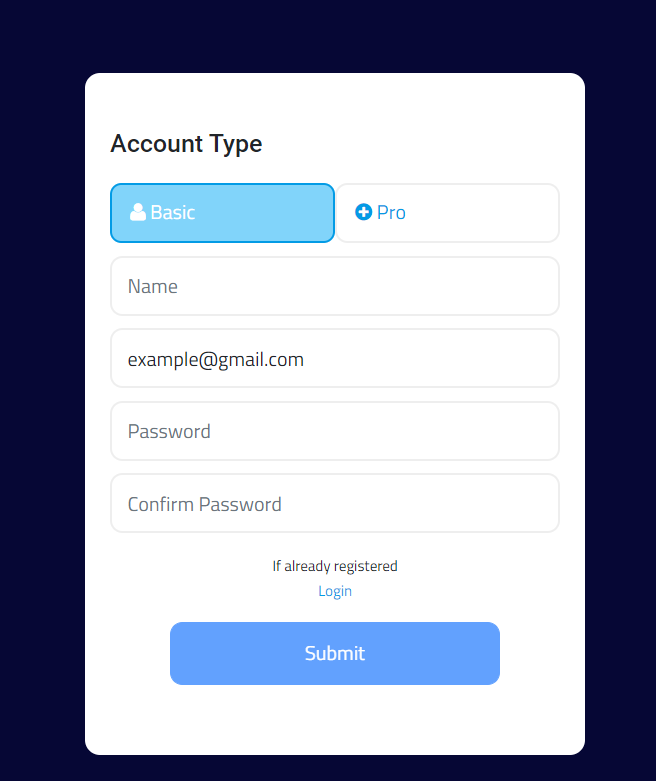
Component Hierarchy

# Detailed Component Descriptions

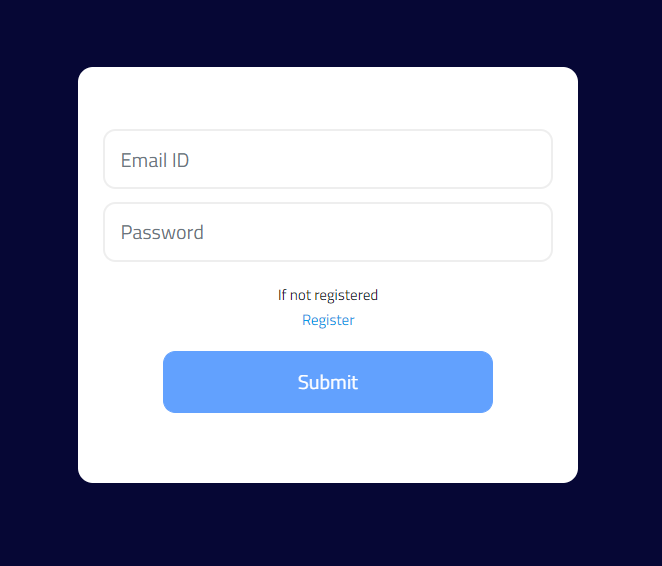
## Registration and Login

Registration component provides the user with the functionality to create an account by providing username, email, password and can also the type of account they want (basic or premium). The registration form is developed using angular’s template driven form and all the input fields are validated with visual feedback on being invalid.

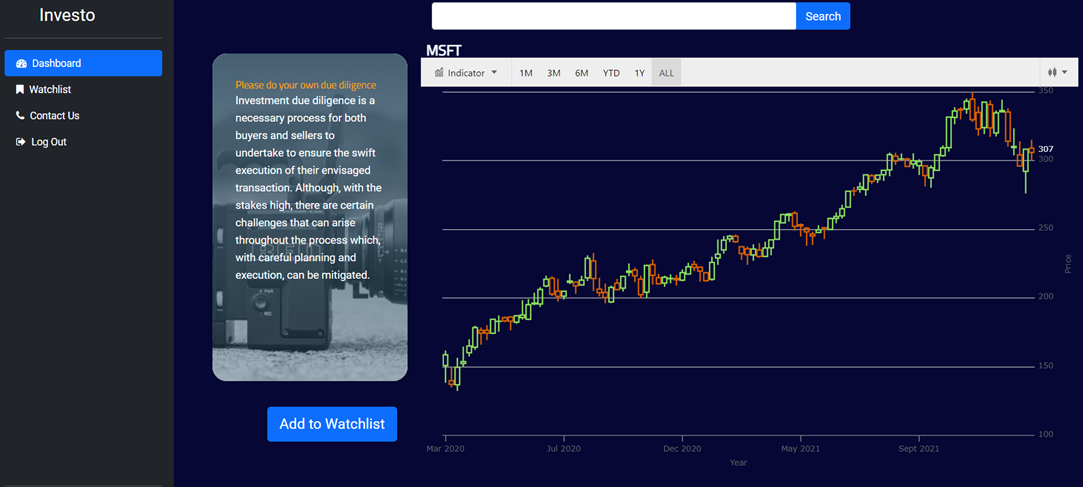
Once the user provides the valid inputs, a model is created from it and sent to the backend via a service using a post request. Now, the backend creates the user using mongoose a NodeJS library for integrating MongoDB.



The Login component works the same way as the registration component, the only difference being the validation on the server side for the user credentials. Also, a JSON Web Token is generated using the user id and a secret key which is then sent as a response to the frontend and stored in the local storage.

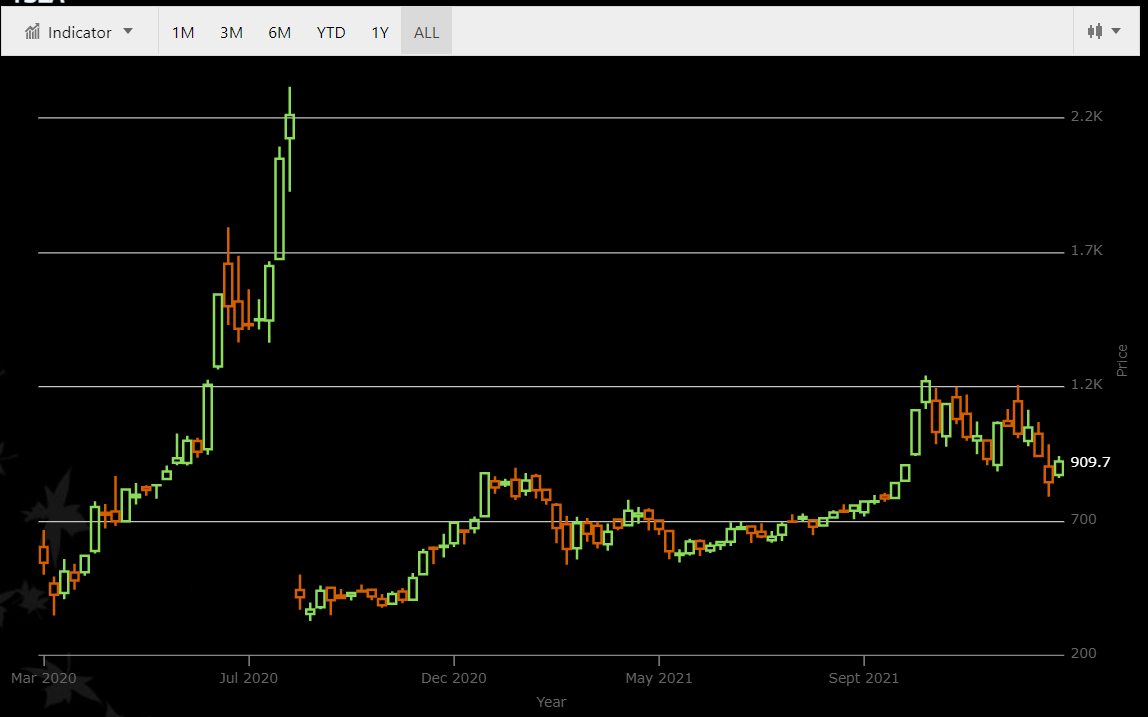


## Dashboard

The dashboard component provides the interface where the user can view the stock charts and key financial metrics of any particular stock. This is the main component(parent) where every other component(child) is rendered. The dashboards allow the users to navigate to other components of the application. This component provides the name of the company whose data is to be shown to all other child components using angular’s input decorator.

## Charts

This component is responsible to show the financial charts using IgniteUI library of angular. It takes the name of the company as input from the dashboard component and sends it to the service which fetches the stock data to be shown in the chart.



The data is retrieved using an API from Alpha Vantage. The chart has the functionality to show data according any time frame and technical indicators for professional traders.

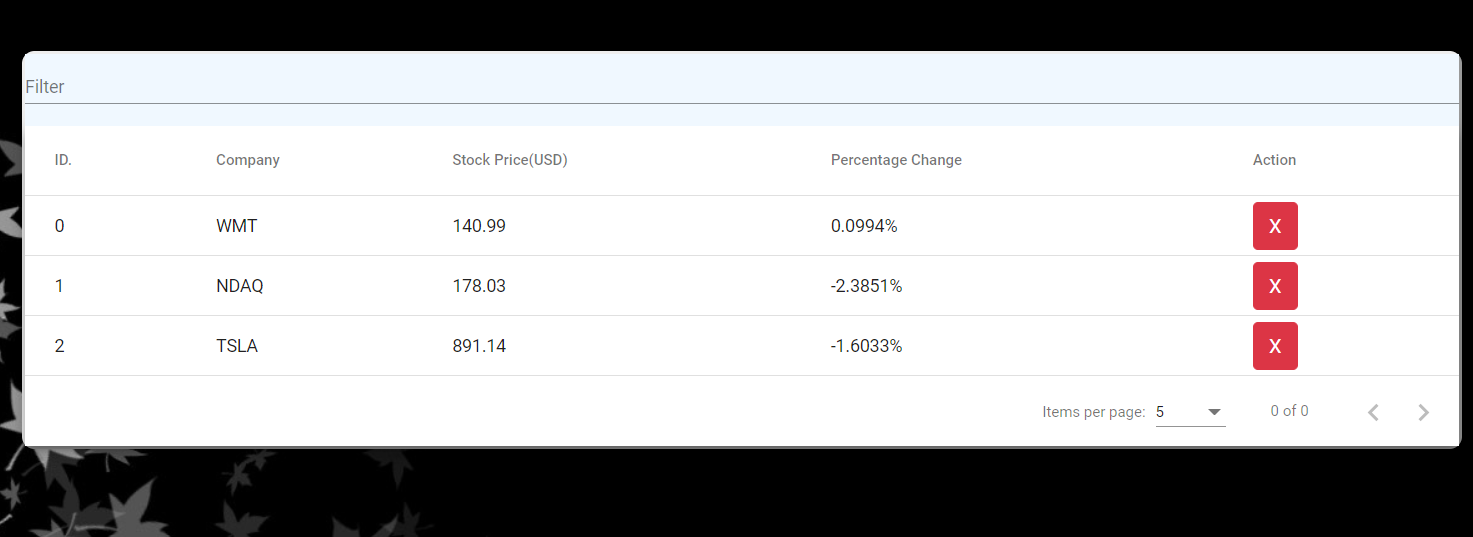
## Data Metrics

It takes the name of the company as input from the dashboard component and sends it to the service which fetches the data metrics from an API. This functionality is only provided to the users with premium accounts and are locked for the basic accounts.



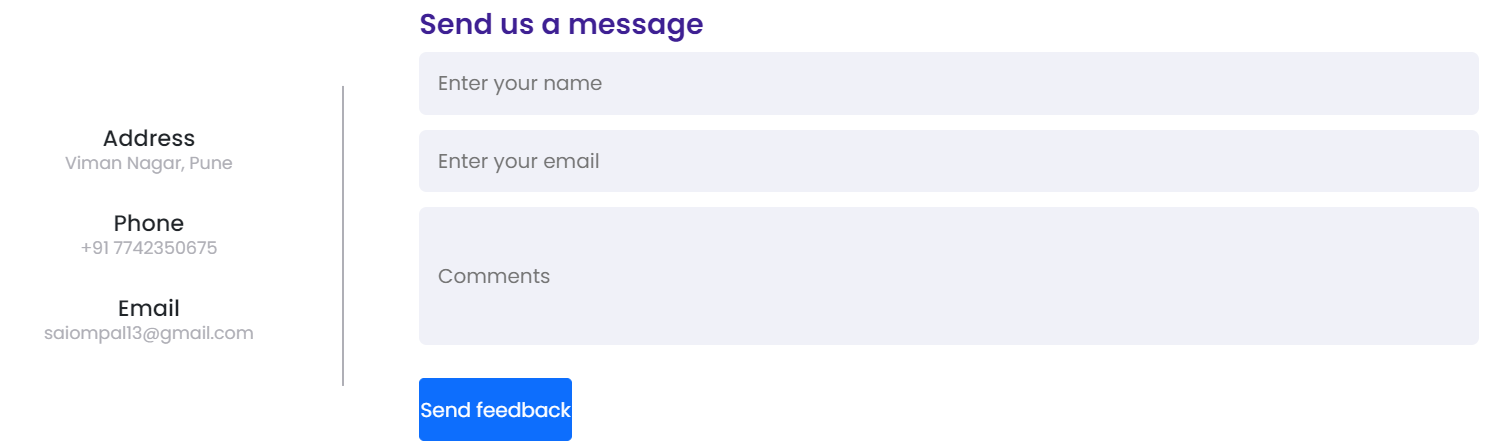
## Watchlist

Any of the stocks displayed in the dashboard component can be added to the watchlist for tracking it later. The user can edit the watchlist according to his/her needs. The watchlist shows the data of any stock from the last recorded trading session using the same API as used for the chart’s component.

For implementing this, whenever the user adds something to the watchlist, it gets saved in the database using the backend and fetched whenever the watchlist component loads via a service that gets the record back from the database.

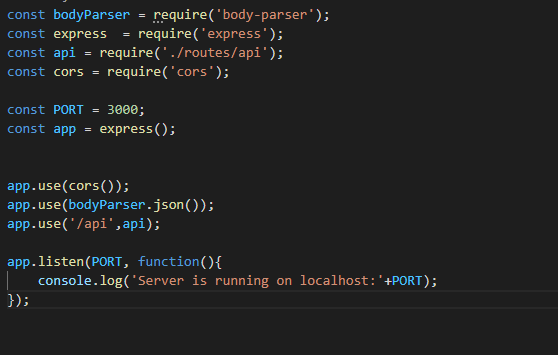
## Contact Us

This page is developed to receive feedback from the users and is implemented in the same way as the registration component. The data is sent to the backend using a service and is saved in the Feedback collections of the database.



# Integration with Backend

For integration of frontend with back-end, we use express and cors which can be installed from Node package manager. Then we create a server.js file where we listen to the port 3000.



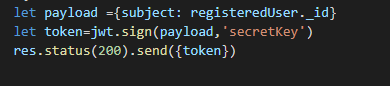
Also, we define routes in the api.js file where all the endpoints of the APIs are defined and integrated from the frontend. Also, in the same file we use mongoose to connect with the MongoDB database and keep all the models defined in the Models folder. In this case we have three models namely, Users, Company Data and Feedback which serves the schema for the database collections.



# Authentication and Authorization

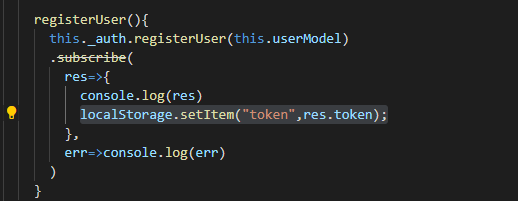
## GENERATING TOKENS:

1. Define the payload
2. Generate the token using jwt.sign() method with the id of the user data and a secret key.
3. Send the token object as a response whenever the login/register request is received.



## STORING THE TOKENS IN LOCAL STORAGE:

1. We store the token in the Local storage while subscribing to the login and registration service.

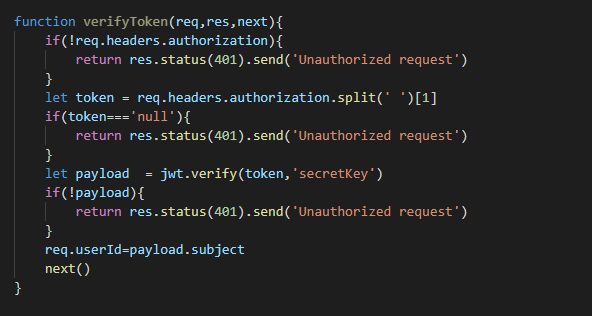


## INTERCEPTING AND VERIFYING THE TOKEN:

1. We create a token-interceptor service that implements the HttpInterceptor module.
2. Now, we clone the request and set the header: (authorization, intercepted token) and call the next method.



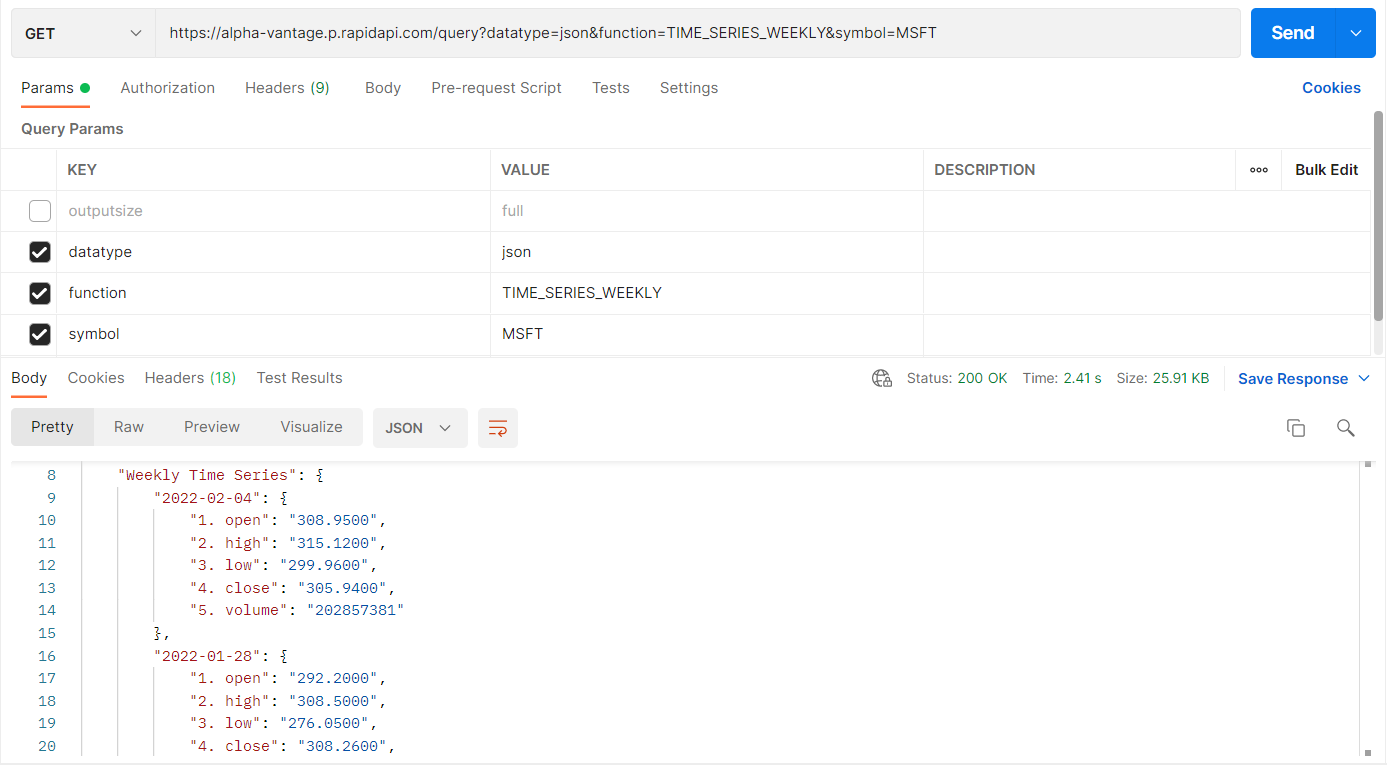
1. Now, on the server side we create a verifytoken() method that parses the header for the token and verify it using the secret key which was used while generating the token.
2. If, the token is valid then we call the next method.



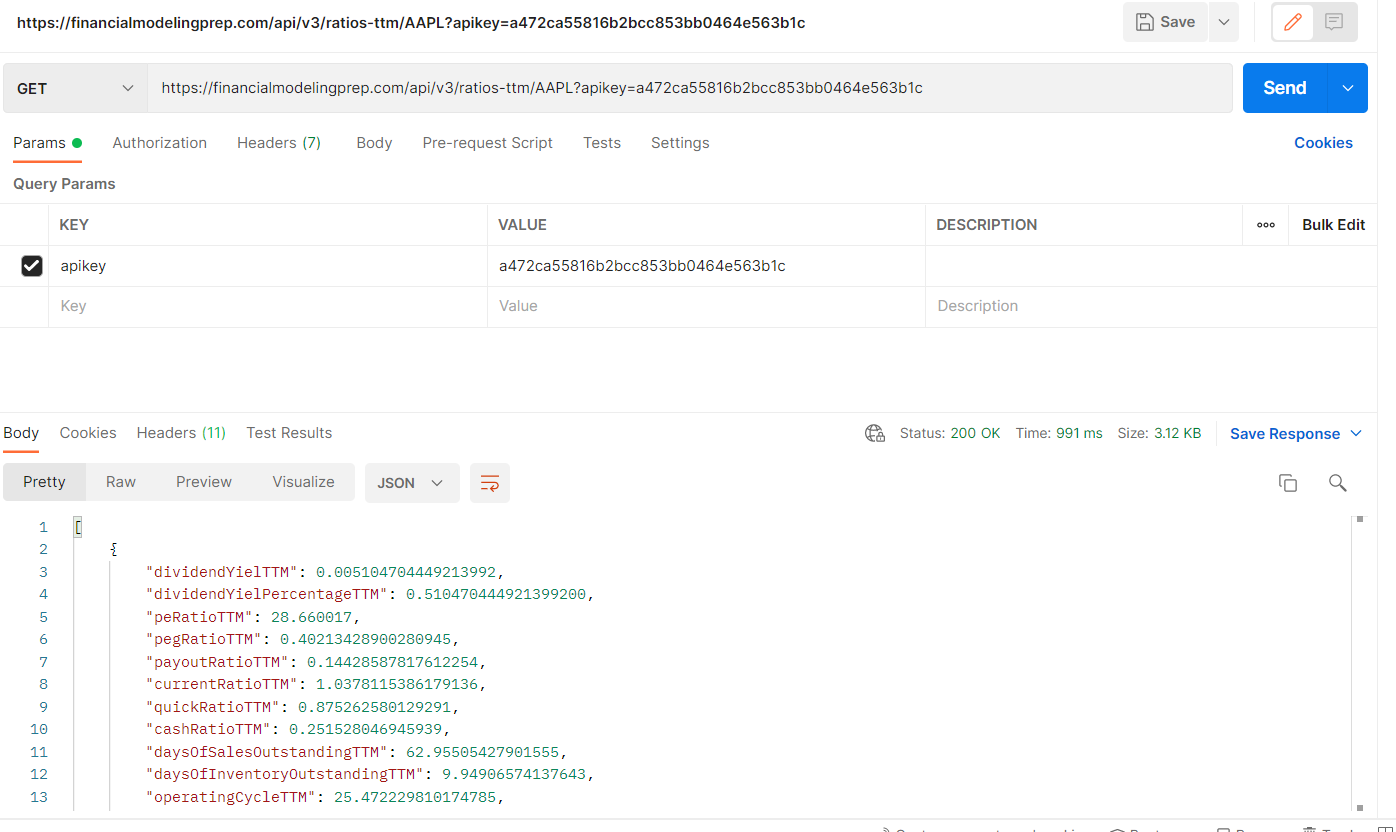
1. We apply the verifytoken() method on the routes we want to protect from un-authorized users.

# Testing APIs using Postman

Endpoint url (Stock data): <https://alpha-vantage.p.rapidapi.com/query>'

Endpointurl:

(keyfinancialmetrics)-https://financialmodelingprep.com/api/v3/ratios-ttm/AAPL?apikey=a472ca55816b2bcc853bb0464e563b1c



# Conclusion

This application was simply based on the idea of showing a website where users can analyse stock market and track their investments. During the development of said application, countless concepts were implemented from the frontend-Angular and backend-Node, Express and MongoDB technologies. All the planned functionalities were implemented but this can also be improved upon in the future by adding more responsiveness to the application and some other quality of life features.