COMP 3100 – Project Iteration 2 – Server-Side Code

Team 18

Mehul Kapoor, student number : 201917168

Jasjeet Singh, student number : 201904992

1) Functionalities

a) For search functionality. The API provides an endpoint '/search/:symbol', which as a response return a list of best matching stocks bases on symbol provided. The list is obtained from AlphaVantage API which provides this search data.

b) for the stocks, the endpoint '/stocks/:id' provides history of open, close, volume and other indicators for the stock symbol "id". The historic data about a stock is again obtained from a call to AlphaVantage API. This data then can me modelled to make a graph in UI.

c) AlphaVantage also provides company overview through their API. This is also used when the above endpoint is called. The overview about a company is stored as text in the Stock Model.

d) The User functionality is also added with endpoints provided for all functions related to a User. The CRUD functionality is present with 4 endpoints respectively which then call functions in the controller file for users. The user contains a unique name, email, and password. The update function allows the name and password to be changed. Email stays the same and acts like a unique identifier for a user.

e) Watchlists are also maintained under each user with their own CRUD endpoints. All CRUD calls for watchlists are performed on currently logged in user. This way users can make custom watchlist and this can be displayed in the home page, or the user dashboard.

Ticker Tape and Compare have not been implemented since they are UI based functionality and will use these endpoints above to gather data.

2) models designed are as follows :

a) User: The User model contains a class called User that models all the data related to user. The instance variables are name, email, password (hashed crypted using bcrypto for security) and a list called watchlists. The watchlists is an empty list at default and will hold objects of Watchlist model. The user model is important since it has functions that connect to the database. The class has functions get(), add(), update(), delete(), getbyemail(), which are used as end layer to do CRUD functionality with the database. The user model also has watchlist related functions that manipulates the list containing watchlists object.

b) Watchlist: The Watchlist model is rather simple. It contains a name and a list of strings representing stock symbols. The names of watchlists are unique for a user and can be updated. Watchlist model has its own functions that work on watchlist objects. The function add() adds a string symbol representing stock to the list. The get() function just returns the list inside the watchlist object. The getAll() function returns all the watchlists of currently logged in user.

c) Stock: The stock model contains a class that makes the API calls to AlphaVantage and stores the necessary data.

3) Routes and Connections.

app.get('/stock/:id', stock.getStock) this route connects to stock controller which later uses stock model to get data about a particular stock id

//these routes connect to functions in user controller which after validating inputs calls user model to perform the CRUD functionality.
app.get('/user/:id', user.getUser)
app.post('/user/', user.add)
app.put('/user/:id', user.updateUser)
app.delete('/user/:id', user.delete)

app.get('/watchlist/', watchlist.getAll) // this route calls the watchlist controller to return all watchlists of current signed in user.

app.get('/watchlist/:id', watchlist.getWatchlist) //get specific watchlist by their name. this function maps to getWatchlist() function in Watchlist controller which then calls User class object to return watchlists if it exists

app.post('/watchlist/', watchlist.addWatchlist) //adding a watchlist. The body contains name which will be used by controller to create a Watchlist object and add it to User

app.post('/watchlist/:id', watchlist.additem) //add stock name to a specific watchlist. Id represents the name of the watchlist

app.delete('/watchlist/:id', watchlist.deleteWatchlist) //deleting whole watchlist by name.

app.delete('/watchlist/:id/:name', watchlist.deleteitem) //deleting specific stock string from a specific watchlist of the current logged in user

app.put('watchlist/:id', function(req, res){ watchlist.update });
// this endpoint can be used to change the name of watchlist with token id.

app.post('/user/login/', user.dologin) //login function that when given an object containing email and password calls the dologin function in user controller which then uses bcrypt to compare password entered and hashed cryptic password. If they match, login is successful.

4) Data Model

We are using an embedded data model since we have several documents merged together creating one great object. This happened because all user data related to their details and their customs watchlists are combined in a single object.

5) Testing

Mocha is the tool that has been used for testing .

Different tests were performed for the API calls to make sure that the program is running as intended to run.

First, the Fail cases were covered and then the success cases.

Test cases for the user model are-:

POST – Used to update the existing resources

Fail Test 1- Invalid Name in the object

Success Test1- Updates the user name for the given user

Fail Test 2- Invalid E-Mail in the object

Success Test 2- Updates the E-mail for the given user

Fail Test-3 Invalid Password in the object

Success Test 3- Updates the password

GET- Used to retrieve data from the specified resource

Fail Test1- Where the user name is not found in the object

Success Test1- Retrieves data of the specified user

DELETE – Used to delete data from the specified resource

Fail Test1- Where the user with that given name does not exist

Success Test1- Deletes the user with the given name

PUT -Used to update e a resource

Fail Test1- Where the user with the given name does not exist

Success Test1- Where the user with the given name exists