Purpose Coding Challenge Instructions

1. Git repository: https://github.com/Syndicate555/purpose-coding-challenge
2. Dependencies
   These following packages need to be installed in the machine to run the project
   - Nodejs https://nodejs.org/en/download/
   - Npm
3. Installation & usage instructions

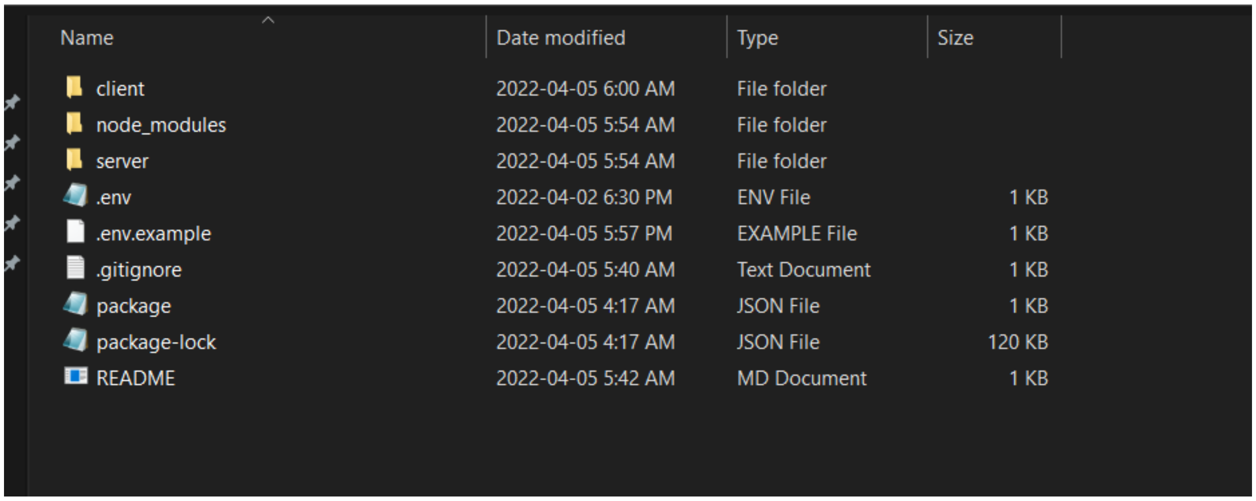
   After cloning the repo, on the root directory, enter the following commands to install and run the project

   - *npm run configure* (this will install all the necessary packages)
   - *npm run dev* (this will start the app and a browser window will open automatically)

     *Note: Make sure nothing is running on the localhost port 4545 or port 3000

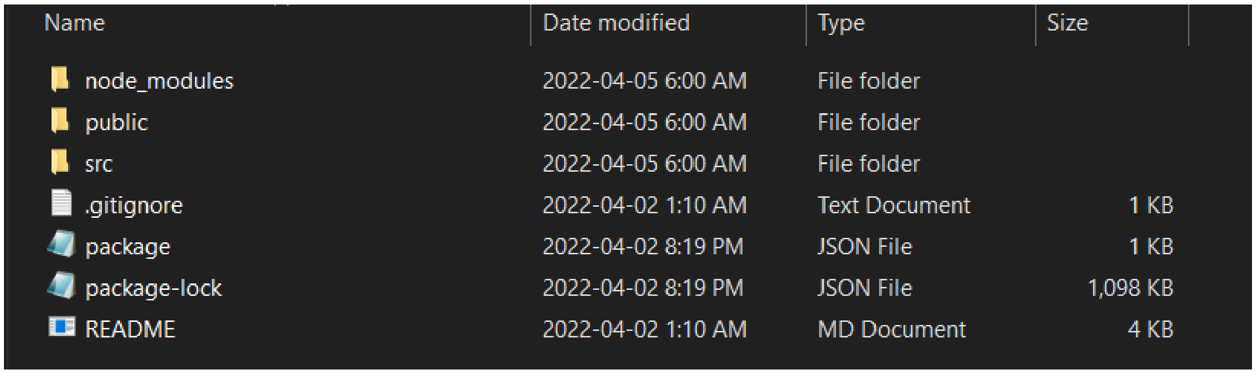
   Upon successful installation, there should a new folder called 'node_modules' created in the root directory and the client directory. It should look like this

   Root directory:



Name	Date modified	Type	Size
client	2022-04-05 6:00 AM	File folder	
node_modules	2022-04-05 5:54 AM	File folder	
server	2022-04-05 5:54 AM	File folder	
.env	2022-04-02 6:30 PM	ENV File	1 KB
.env.example	2022-04-05 5:57 PM	EXAMPLE File	1 KB
.gitignore	2022-04-05 5:40 AM	Text Document	1 KB
package	2022-04-05 4:17 AM	JSON File	1 KB
package-lock	2022-04-05 4:17 AM	JSON File	120 KB
README	2022-04-05 5:42 AM	MD Document	1 KB

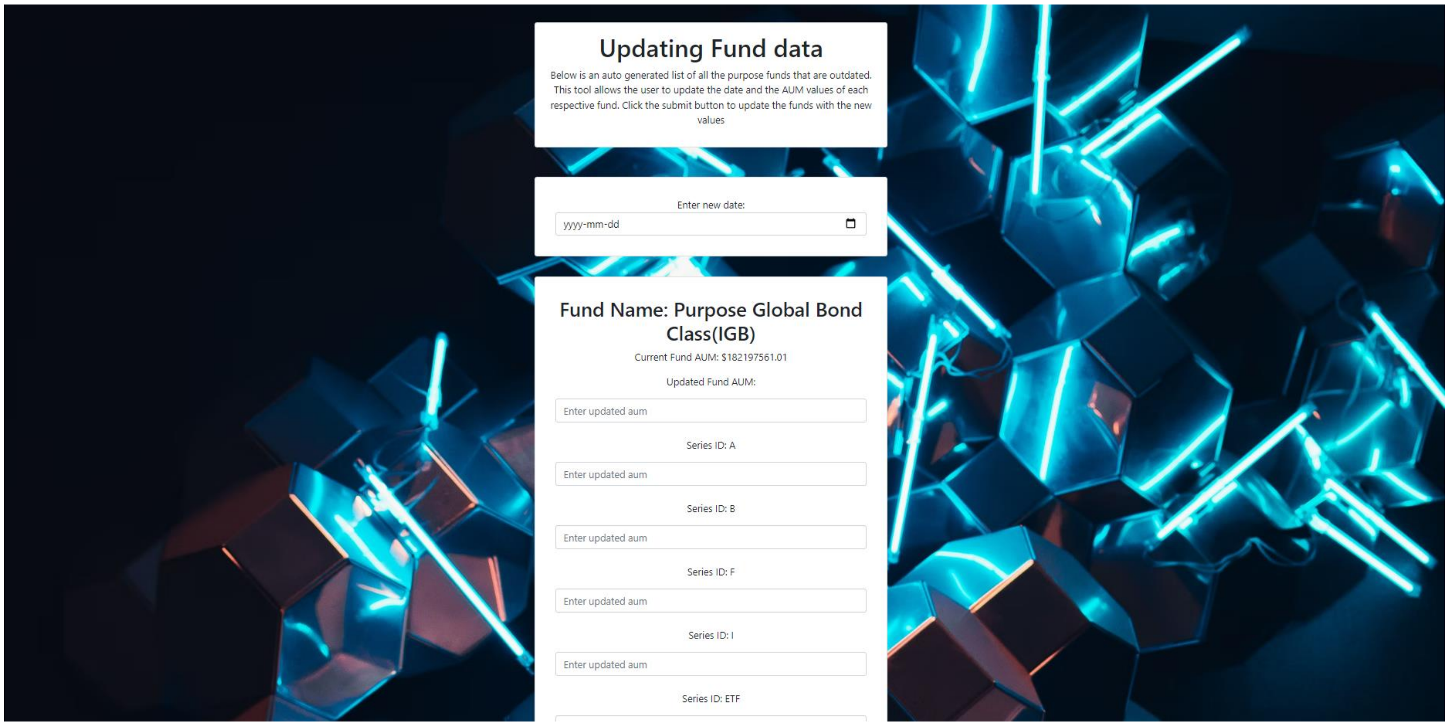
Client:



Name	Date modified	Type	Size
node_modules	2022-04-05 6:00 AM	File folder	
public	2022-04-05 6:00 AM	File folder	
src	2022-04-05 6:00 AM	File folder	
.gitignore	2022-04-02 1:10 AM	Text Document	1 KB
package	2022-04-02 8:19 PM	JSON File	1 KB
package-lock	2022-04-02 8:19 PM	JSON File	1,098 KB
README	2022-04-02 1:10 AM	MD Document	4 KB

Explanation:

1) I first broke down the requirements into two tiers – frontend + backend
2) I started by creating the server using nodejs and the express framework. I created two API endpoints. The GET endpoint fetches the source data from the URL provided, formats it and filters out the funds that have their date older than 1 day. It then returns the filtered data as json. This data will be used to generate the form in the frontend
3) The POST endpoint accepts the updated form data and does the following
    a. Writes the updated data into a csv file inside the csv folder. The file name is new-fund-data.csv
    b. It then fetches the entire fund data from the URL provided and overwrites all the data from the form.
    c. It then saves the updated data with the new date and aum values into a json file. The file will be saved in the root directory of the project. The name will be fundData.json
4) I used the CORS package to only allow the frontend (using localhost:3000) to fetch data using the APIs
5) The frontend will use the GET endpoint to fetch all outdated fund data and dynamically generate a form. It should look like this



Updating Fund data

Below is an auto generated list of all the purpose funds that are outdated. This tool allows the user to update the date and the AUM values of each respective fund. Click the submit button to update the funds with the new values

Enter new date: yyyy-mm-dd

Fund Name: Purpose Global Bond Class(IGB)

Current Fund AUM: $182197561.01

Updated Fund AUM: Enter updated aum

Series ID: A: Enter updated aum

Series ID: B: Enter updated aum

Series ID: F: Enter updated aum

Series ID: I: Enter updated aum

Series ID: ETF

6) I used some basic CSS and bootstrap to style the frontend

Testing:

The APIs can be tested in isolation using any HTTP client such as Postman or Insomnia.

1) GET endpoint: http://localhost:4545

   This should return all outdated fund data

2) POST endpoint: http://localhost:4545/submit
   This endpoint will take in a json body. I have included a sample json payload in the testing folder (insider server folder). Use this payload in the body of the request to test the POST endpoint.