VIRTUSA NODE.JS ASSIGNMENT

This was implemented using Node.js, TypeScript and Express framework.

Design/ Technical Approach

The first step was on setting up the environment for development. It was started by installing node v14.7.0. The required paths for Node were added to system environment variables.

Visual studio code was used all the way during development purposes.

The next set of steps were followed to set the project directory. I created a folder and inside the folder I ran

npm init -y , which created the base files for the project.

I made sure that the project was on typescript and installed the below mentioned packages.

```
npm install typescript@3.3.3
npm install tslint@5.12.1
```

Tslint was used for basic linting highlights.

Then, I created a src folder inside which the app.ts was created.

Since, the project demands API, and request bodies, I ran

```
npm install body-parser
npm install express
```

Once the packages were added, then I configured the tsconfig.json for linting and compiler options.

I also updated the package.json to update main path and added start command with a suitable script that fits my need.

Then, I started working with the app.ts

Here, I also added a strict type checking for the model. In a real-world app, we would use schema defined according to the database requirements. Here, I defined an interface which would be of the response type.

Then, in app.ts , I imported the interface. After, that I imported necessary packages and started the server listening at port 3000.

Once I had the app running on my server, I started working with the v1 route. So in the v1 post route, I got the “request.body” param and stored it in a property called requestData.

After that ,I declared firstName, lastName and clientId properties and from the strings I took the necessary substingd and stored them back. Once, I had the data for client, I wrote another object that was strictly of type from the interface and stored the values in the object. I then also made sure to display, the response codes depending on the res status and displayed along with the data.

I finally displayed the object using “res.json” with all the user details and status code for the user.

A similar approach was carried out for v2 route with a slight logic change in how the user data was carried out.

I have added necessary comments along with the project wherever necessary.

This was successfully tested using Postman and the screenshots are attached as well.