REST API

There is a difference between coding and programming and there is also difference between software engineer and developers. Developer is someone who develops applications based on certain pattern. Software Engineer have another set of availability that they know computers well. They know the approach; they know the inner workings of a computer.

The views folder has its own purpose. It has stuff related to handlebars and templates. A public folder on the other hand has files that can be shared with everyone. Let's say there is a test.txt file having content "This is some content". If we want to access that test.txt file then we need to create a GET request. So, to define GET request, we need to write,

```
app.get('/send', () => {
    res.send("Abc")
})
```

Now that we are sending "Abc", we can send the file as well.

```
app.get('/css, () => {
    res.sendFile(__dirname + 'views/layouts/style.css')
})
```

We can see the content of CSS being displayed. We are sharing the CSS file, but it is not being implemented. The first we have to do is send it to the browser. Now we need to send it in a way that the browser would implement it.

In the main.hbs then we just have to mention href='/css' then if everything works out then this route should give us CSS file. So, when browser comes to this line it will come to index.js file and then send the file or implement the file.

I would tell express in some way, to server some set of the files, which would always be served. Without specifying the route, itself, I should be able to server the simple files, i.e., browser-based CSS and JS file. If this is done, then all we have to do is relevant to the directory structure. We would have to type link in the handlebar and the server would know that it needs to server those files.

Conventionally developer's community suggest that we should create a public folder and put everything which is always going to be server. So, we will put the test.txt file in the public folder. We will also create a CSS folder in the public for all the different

CSS files that we will be creating. We'll put all the CSS files in the CSS folder. Now the style.css is inside the CSS folder. We now need to get rid of the route that are serving manually inside the code. We need to tell the express server to use, express.static and give folder name **('.public').**

```
app.use(express.static('public'));
```

Even without specifying a route, I'm able to get my test.txt file. So, if we put in **localhost:3000/css/style.css**, it will show the content of the style.css file.

use()

The use has multiple functions and definitions that specify what will the use() will be suing. This use() can expect n number of parameters and depending on the number of parameters, it would call the specific function. In the 'view engine' and as the second parameter we are giving hbs. In such case, the signature gets matched to the use() and it gets called.

use() function is defined on the application instance which the express creates for you and you can define application level or route level logic or route level configuration, such that it is defined once and the entire application is using it.

urlencoded – the form data that is shared as query parameter or some object (not as json object), but it come in URL encoded format. This is the data the comes to us as form from the user. We have to tell the static to use whatever **urlencoded** returns. Similarly, static as well returns a function. Meaning, a function and then it some magical way, our application creates specific routes for the files in the public folder and we can access them directly inside the URL without specifying that route inside my application logic.

If we use localhost:5000 → we would not get any information because we have not defined a route on what should happen when there is not route mentioned. Obviously, we have to specific test.txt, only then the browser will know that we trying to get the test.txt file. All the static assets are served after the '/' and we have to put in the file name.

handlebar:

The reason we put three braces for the body is contextual. Every request or route or template we are not specifying the entire body, but a bit of the html. What we have defined in the layout, it is quite relative the bank opening form or a sarkari form. These forms are like a template in which everything is written, all you specify is you

data and your photo. In the same way handlebar or template engine are the same thing. but using some special syntax we can define what data should be present here. Based on what we supply from backend, will always be rendered inside the paragraph tag which is going to be inside the paragraph tag. We have created a template and inside the template and inside the template, the data which is sent in some form, that data's value will be inserted in the p element. This combined html is getting sent to the front end.

So, {{{body}}} means we have defined a layout on top of a layout. One place we have defined the basic structure and then configured the handlebar in such a way that whenever we serve a particular template, it will take-up the layout from main.hbs and will put the entire body of features or home or profile inside the body tag of the main.hbs file for it to be rendered.

So, in order to pass some data to your handlebars, wherever you are calling the res.render, then you specify the name of the file that we have created here.

When we go to features route in the browser, we want to render features.hbs file in the browser. The problem is we have to specify the data as well. We are passing data by creating an object. The second parameter in the render is also an object. We can just copy the whole object or we can create an object and saved it into a variable featureObj. The parameter is getting passed internally to the template that we have sent. This entire object with all keys and values will be passed to the features.hbs fiel. If we need to access age property, then in the features.hbs we can use any html tag to display the age value and then we'll put the syntax {{age}} in the html tag and it will be rendered in the browser.

This template can be used for any value. We can put logic inside the index.js file and get the output on the browser using the handlebars. Our express will internally calculate what html it needs to create and that html will be sent to the client.

In the layouts folder main.hbs file is created conventionally. We have different routes for all the views. Similarly, if the html needs to be served, then it needs to be served. But, for partials we can create different