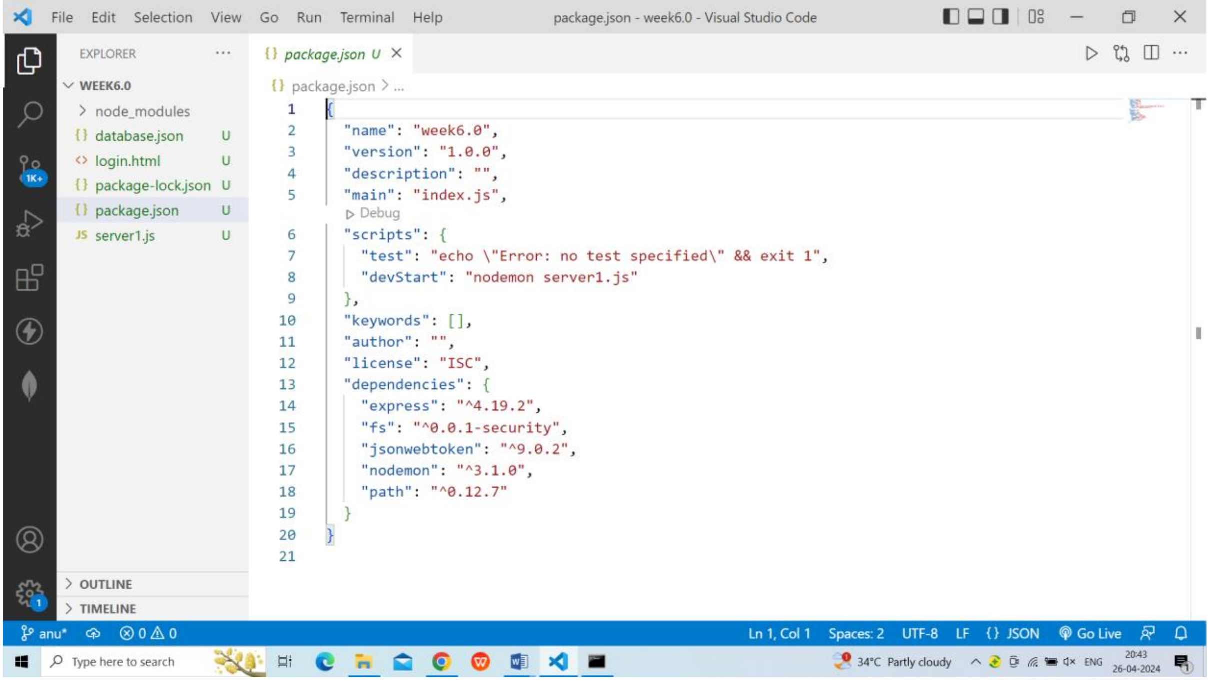
Exp-6

Write middleware to validate users and generate JWT.



```json
{
  "name": "week6.0",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "devStart": "nodemon server1.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.19.2",
    "fs": "^0.0.1-security",
    "jsonwebtoken": "^9.0.2",
    "nodemon": "^3.1.0",
    "path": "^0.12.7"
  }
}
```

Fig.1.Visual Studio Week6 files and package.json

Softwares:-Node Js 20 version and visual studio

Step 1:-Create package.json file initializing project using command npm(node package manager)

- **npm init (or) npm init –y**
  package.json file created show above fig.1.
- Install express and nodemon use command
  npm i express nodemon fs path jsonwebtoken

Note:

Step 1) Download Postman. Go to https://www.postman.com/downloads/ and choose your desired platform among Mac, Windows or Linux. ...

Step 2) Click on Run. ...

Step 3) Postman Installation Start. ...

Step 4) Signup for Postman Account. ...

Step 5) Click on Save My Preferences. ...

Step 6) Congratulation!

Code:-

database.json:-

```
[   {"name":"anusha","work":"Assistant Professor","password":"anu"},
    {"name":"radha","work":"Assistant Professor","password":"radha12"},
    {"name":"jaya","work":"Assistant Professor","password":"jaya12"}

]
```

login.html:-

```
<!DOCTYPE html>
<html>
    <head>
  <title>Login</title>
  <style>
      h2{
          display:none;
      }
  </style>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script>
      const form=$("#loginForm");
      $(document).ready(function(){
     $("#loginForm").submit(
         (event)=>{
event.preventDefault();
var name=$("#username").val();
var password=$("#pwd").val();
$.ajax({
    url:"/auth",
    type:"POST",
    contentType:"application/json",
    dataType:"json",
    data:JSON.stringify({
        name:name,
        password:password
    }),
    success:function(data){
        var tokenData=data;
        if(tokenData.login==true){
            if(verifyLogin(tokenData.token)){
                $("#verified").show();
                $("#not-verified").hide();
                alert("JWT Token: "+tokenData.token);
            }
        }
        else{
            $("#verified").hide();
                $("#not-verified").show();
                alert("Authentication Failed");
        }
    },
    error:function(data){
     console.log("Something went wrong");
    }
});
         });

      });
      function verifyLogin(token)
      {
          let result=true;
          $.ajax({
    url:"/verifyToken",
    type:"POST",
    contentType:"application/json",
    dataType:"json",
    data:JSON.stringify({
        token:token,

    }),
    success:function(data){
        if(data.login==true)
        {
           result=true;
        }
        else{
            result=false;
        }
    },
    error:function(data){
```

```
    console.log("wrong Token, Not Authenticated.");
   }
});
return result;
     }
  </script>
   </head>
   <body>
       <center>
  <h1>Login Page</h1>
  <form id="loginForm">
      <label for="username">UserName:</label><br>
      <input type="text" id="username" name="username"><br>
      <label for="pwd">Password:</label><br>
      <input type="password" id="pwd" name="pwd"><br>
      <input type="submit" value="Submit">
  </form>
  <h2 id="verified">You are a verified user</h2>
  <h2 id="not-verified">You are not a verified user</h2>
       </center>
   </body>
</html>
```

server1.js:-

```
//import express for creating APIs endpoints
const express=require("express");
const path=require("path");
const fs=require("fs");
const users=require("./database.json");

var database;
var token="wrong key";

//Read database.json file
fs.readFile("database.json",function(err,data){
//check for errors
if(err) throw err;
//Converting to JSON
database=JSON.parse(data);
});
```

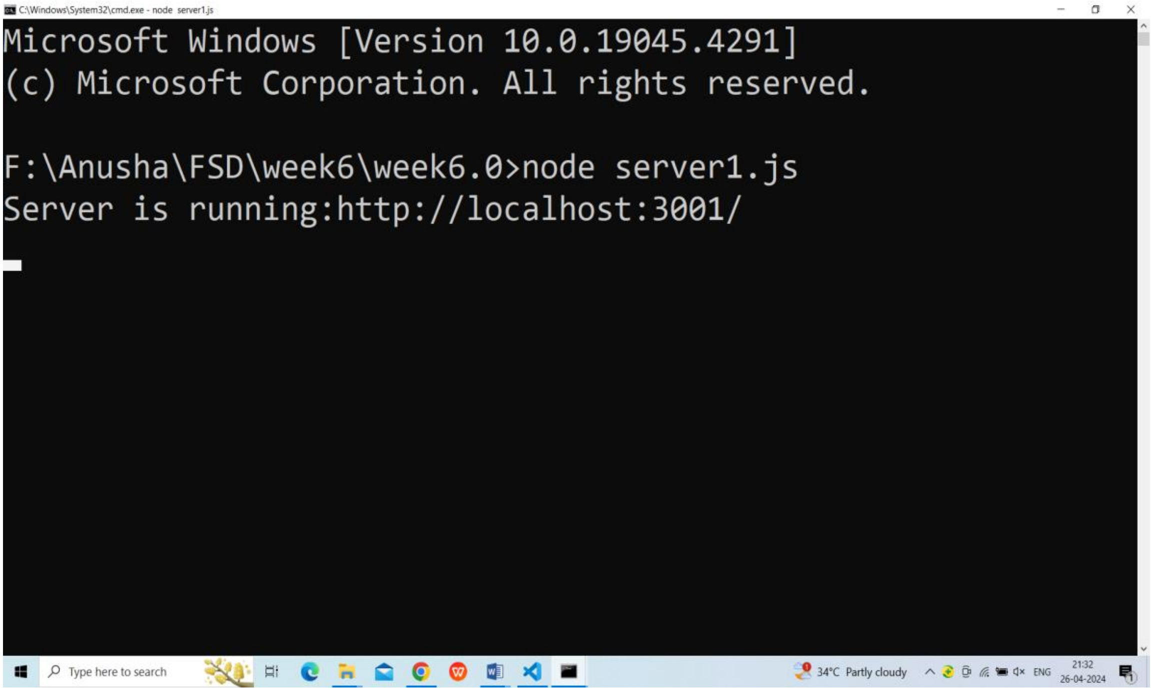
```
const jwt=require("jsonwebtoken");
const app=express();
const port=3001;
app.use(express.json());
app.get('/',(req, res)=>{
    res.sendFile(__dirname+'/login.html');
});
app.post("/auth",(req,res)=>{
const name=req.body.name;
console.log(name);
const password=req.body.password;
console.log(password);
let isPresent=false;
let isPresentIndex=null;
for(let i=0;i<database.length;i++){
    if(database[i].name===name && database[i].password===password){
        isPresent=true;
        isPresentIndex=i;
        break;
    }
}
if(isPresent){
    const token=jwt.sign(database[isPresentIndex],"secret");
    res.json({
login:true,
token:token,
data:database[isPresentIndex],
    });
}
else{
    res.json({
      login:false,
      token:token,
      error:"please check name and password"
    });
}
});

app.post("/verifyToken", (req, res)=>{
const token=req.body.token;
if(token){
    const decode=jwt.verify(token,"secret");
    res.json({
login:true,
data:decode
    });
}
else{
    res.json({
login:false,
data:"error"
    });
}
});
app.post('/login',(req, res)=>{
res.redirect("/login");
});
app.listen(port, ()=>{
    console.log(`Server is running:http://localhost:${port}/`);
});
```

Output:

Output:-

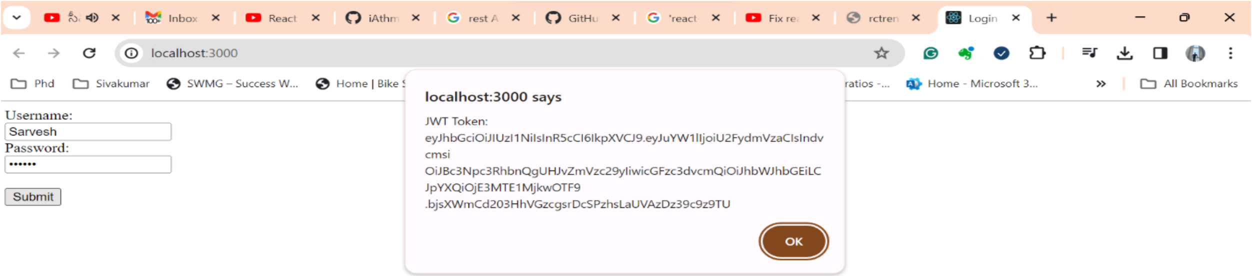
Run code use command

- **node server.js**



```
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

F:\Anusha\FSD\week6\week6.0>node server1.js
Server is running:http://localhost:3001/
```

Fig.2.Now server running successful



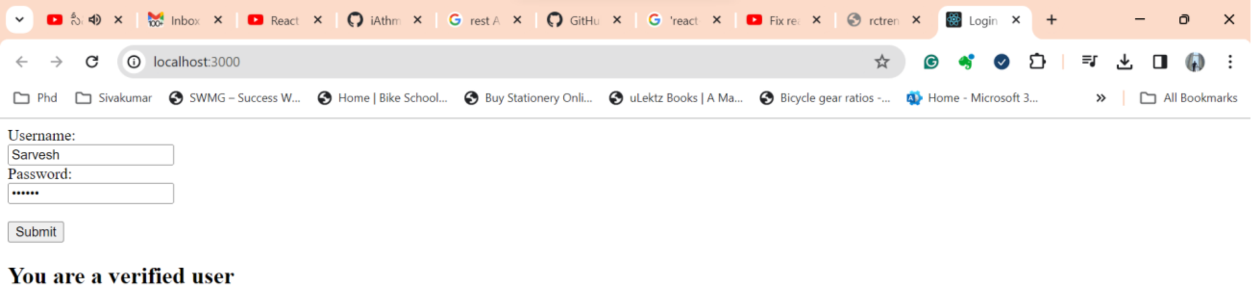
localhost:3000

Username:
Sarvesh
Password:
••••••

localhost:3000 says

JWT Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiU2FydmVzaCIsIndv
cmsi
OiJBc3Npc3RhbnQgUHJvZmVzc29yIiwicGFzc3dvcmQiOiJhbWJhbGEiLC
JpYXQiOjE3MTE1MjkwOTF9
.bjsXWmCd203HhVGzcgsrDcSPzhsLaUVAzDz39c9z9TU

Fig.3.Generating JWT

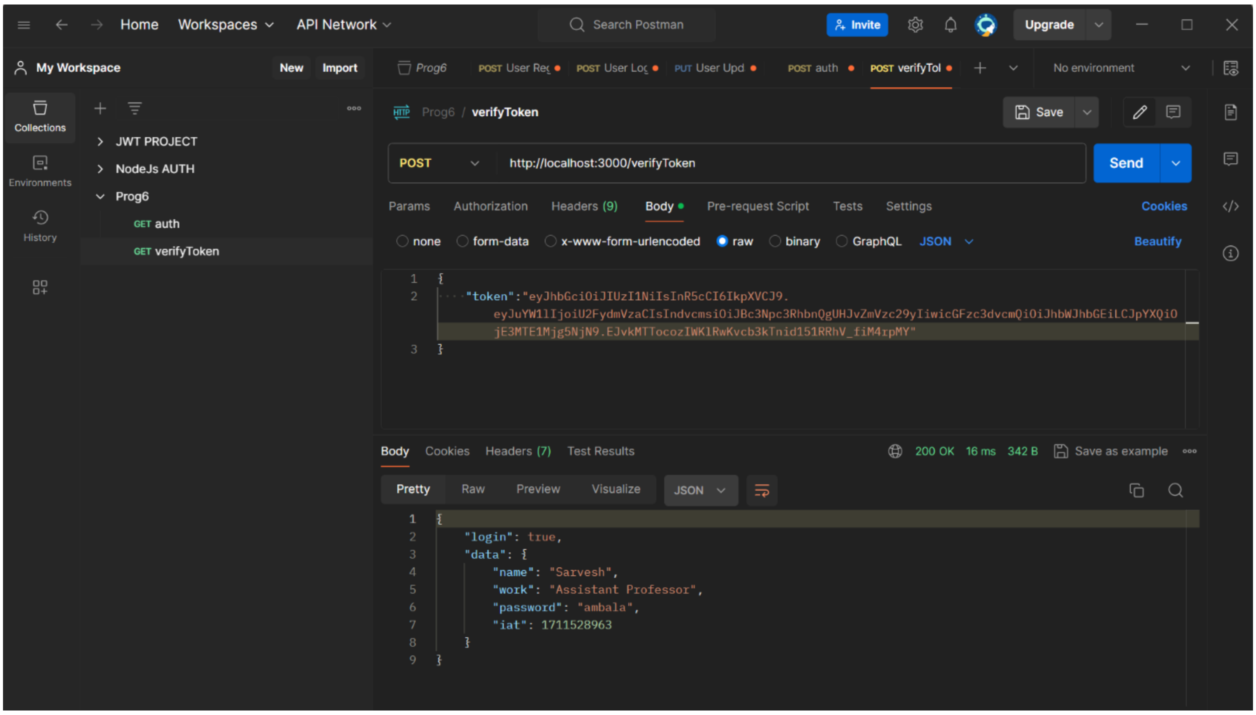


localhost:3000

Username:
Sarvesh
Password:
••••••

You are a verified user

Fig.4.Verified user



POST http://localhost:3000/verifyToken

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiU2FydmVzaCIsIndvcmsiOiJBc3Npc3RhbnQgUHJvZmVzc29yIiwicGFzc3dvcmQiOiJhbWJhbGEiLCJpYXQiOjE3MTE1Mjg5NjN9.EJvkMTTocozIWKlRwKvcb3kTnid151RRhV_fiM4rpMY"
}
```

200 OK 16 ms 342 B

```
{
    "login": true,
    "data": {
        "name": "Sarvesh",
        "work": "Assistant Professor",
        "password": "ambala",
        "iat": 1711528963
    }
}
```

Fig.5.Middleware JWT

Viva Questions:

1. **Why use jwt authentication ?**

Ans: JWT, or JSON Web Token, is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity-protected with a Message Authentication Code (MAC) and/or encrypted. JWT authentication has become popular for several reasons, particularly in web and mobile applications.

2. **why do we need the JSON Web Token (JWT) in the modern web?**

Ans: In the landscape of modern web development, JSON Web Tokens (JWT) fulfill critical needs for secure, flexible, and scalable user authentication and information exchange.

Their significance is magnified by the evolution of web applications towards more complex architectures, including microservices and single-page applications (SPAs), as well as the increasing emphasis on mobile and API-first development. Here's why

JWTs are particularly valuable in the modern web:

1.Decentralized Authentication and Authorization
2. Stateless Sessions for Scalability
3. Cross-Platform and Cross-Domain Communication
4. Microservices Architecture
5. Single Sign-On (SSO)
6. Enhanced Security Features
7. Simplicity and Speed
8. Compliance and Best Practices

3. **How it works for angular JS authentication ?**

Ans: AngularJS, a client-side JavaScript framework by Google, has been widely used for building dynamic single-page applications (SPAs). When it comes to implementing authentication in AngularJS applications, JSON Web Tokens (JWT) provide a robust and flexible method to secure the application while maintaining a user-friendly experience.

4. Give the necessary steps for JSON web tokens?

Ans: Implementing JSON Web Tokens (JWT) for authentication and authorization in a web application involves several key steps, from user authentication to token validation. Below is a generalized process that outlines the necessary steps to effectively use JWTs:

1. User Authentication
2. Token Generation
3. Token Transmission
4. Token Usage
5. Token Validation
6. Access Control
7. Token Expiry and Renewal

5.What do you mean by typescript ?

Ans: TypeScript is an open-source programming language developed and maintained by Microsoft. It is a superset of JavaScript, meaning that any valid JavaScript code is also valid TypeScript code. TypeScript extends JavaScript by adding static types to the language.

1.Static Type-Checking
2.Type Inference
3.Advanced Types
4.IDE Support
5.Compatibility with JavaScript
6.Compilation to JavaScript