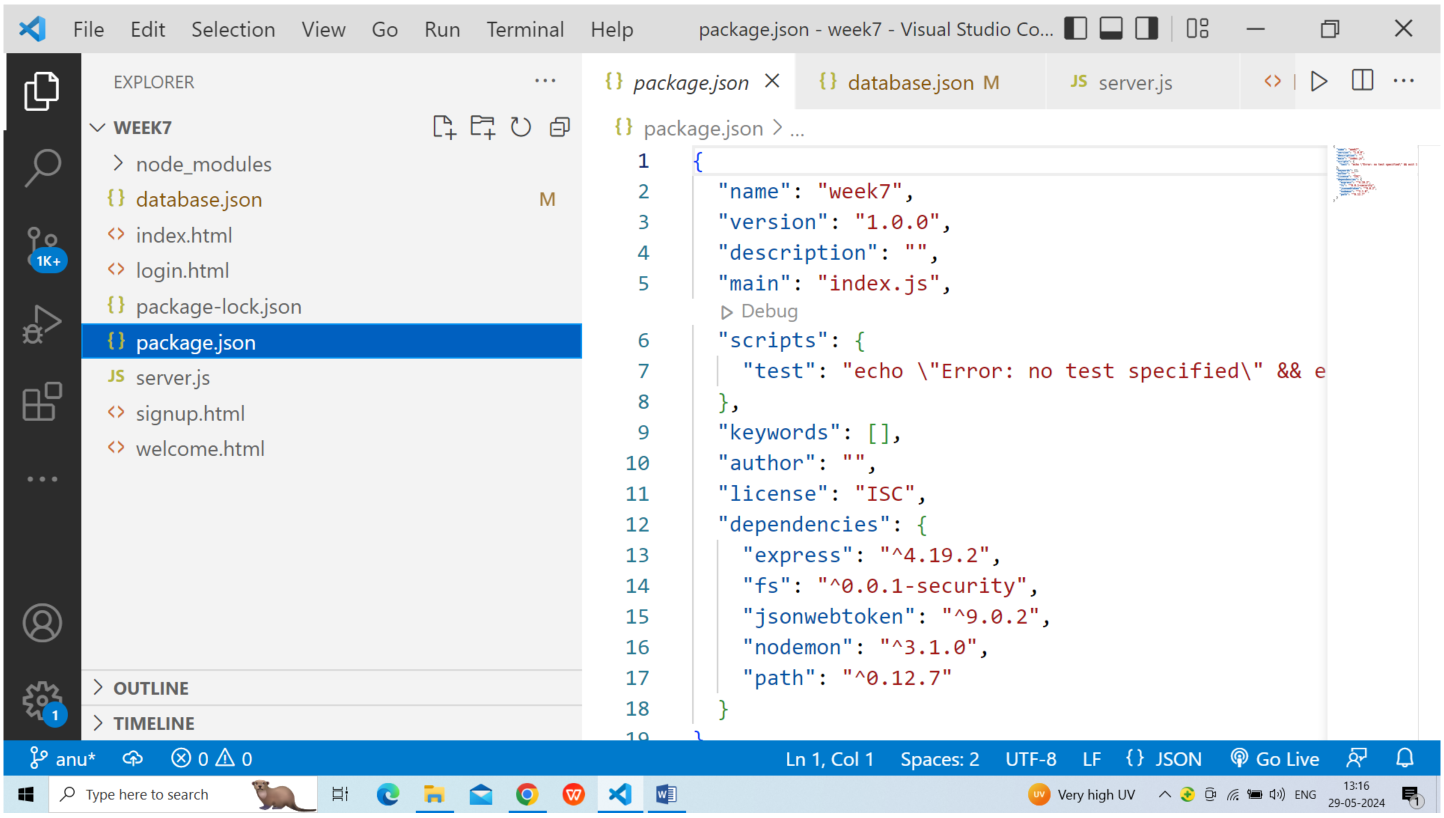
Exp-7

Write middleware to validate JWT and redirect to profile.



```
{
  "name": "week7",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && e
  },
  "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"
  }
```

Softwares:-Node Js 20 version, visual studio and Postman

Steps:-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.
- Installing require modules
  npm i express nodemon fs path jsonwebtoken

Note:
Download Postman or thunder client for middleware.

<u>Database. Json:</u>

```
[
{"name":"jyotsna","work":"Assistant Professor","password":"abc"},
{"name":"Sarvesh","work":"Assistant Professor","password":"ambala"},
{"name":"ram","work":"Assistant Professor","password":"priya"},
{"name":"Lakshmi","work":"Asst Prof","password":"Cmrit@123"}
]
```

Server.js:

```
// Import express for creating API's endpoints
const express = require("express");
const path = require('path');
const fs = require("fs");
const users = require("./database.json");

// Import jwt for API's endpoints authentication
const jwt = require("jsonwebtoken");

// Creates an Express application, initiate
// express top level function
const app = express();

// A port for serving API's
const port = 3000;

// Allow json data
app.use(express.json());

app.get('/',
   (req, res) => {
      res.sendFile(__dirname + '/index.html');
   });
app.get('/welcome',
   (req, res) => {
      res.sendFile(__dirname + '/welcome.html');
   });
app.get('/signup',
   (req, res) => {
      res.sendFile(__dirname + '/signup.html');
   });
app.get('/login',
   (req, res) => {
        res.sendFile(__dirname + '/login.html');
    });
var database;
var token;


// SignUp route
app.post("/register", (req, res) => {
    // Get the name to the json body data
    const name = req.body.name;

    // Get the password to the json body data
    const password = req.body.password;

    // Get the profession to the json body data
    const work = req.body.work;

    // Make two variable for further use
    let isPresent = false;
    let isPresentIndex = null;

    // 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);

    // iterate a loop to the data items and
    // check what data are matched.
    for (let i = 0; i < database.length; i++) {
```

```
    // If data name are matched so check
    // the password are correct or not
    if (database[i].name === name
        && database[i].password === password) {

        // If both are correct so make
        // isPresent variable true
        isPresent = true;

        // And store the data index
        isPresentIndex = i;

        // Break the loop after matching successfully
        break;
    }
}

// If isPresent is true, then create a
// token and pass to the response
if (isPresent) {
    res.json({
        signup: false,
        token: null,
        error: "Already registered",
    });
} else {
    let user =
{
    name: name,
    work: work,
    password: password,
    token: token
};
users.push(user);
```

```
        fs.writeFile(
            "database.json",
            JSON.stringify(users),
            err => {
                // Checking for errors
                if (err) throw err;

                // Success
                res.json({
                    signup: true,
                    token: "generated",
                    data: "Successfully Registered",
                });
                console.log("Done writing");
            });
    }
});
});


// Login route
app.post("/auth", (req, res) => {
    // Get the name to the json body data

    const name = req.body.name;
    console.log(name);

    // Get the password to the json body data
    const password = req.body.password;
    console.log(password)
    // Make two variable for further use
    let isPresent = false;
    let isPresentIndex = null;

    // 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);

// iterate a loop to the data items and
// check what data are matched.
for (let i = 0; i < database.length; i++) {

    // If data name are matched so check
    // the password are correct or not
    if (database[i].name === name
        && database[i].password === password) {

        // If both are correct so make
        // isPresent variable true
        isPresent = true;

        // And store the data index
        isPresentIndex = i;

        // Break the loop after matching successfully
        break;
    }
}

// If isPresent is true, then create a
// token and pass to the response
if (isPresent) {

    // The jwt.sign method are used
```

```
        // to create token
        const token = jwt.sign(database[isPresentIndex], "secret");

        // Pass the data or token in response
        res.json({
            login: true,
            token: token,
            data: database[isPresentIndex],
        });
    } else {

        // If isPresent is false return the error
        res.json({
            login: false,
            error: "please check name and password.",
        });
    }
});
});

// Verify route
app.post("/verifyToken", (req, res) => {

    // Get token value to the json body
    const token = req.body.token;

    // If the token is present
    if (token) {

        // Verify the token using jwt.verify method
        const decode = jwt.verify(token, "secret");

        // Return response with decode data
        res.json({
            login: true,
            data: decode,
        });
    } else {

        // Return response with error
        res.json({
            login: false,
            data: "error",
        });
    }
});

app.post('/welcome',(req, res) => {
    res.redirect("/welcome")
    });
app.post('/login',(req, res) => {
        res.redirect("/login")
        });
app.post('/signup',(req, res) => {
    res.redirect("/signup")
    });

// Listen the server
app.listen(port, () => {
    console.log(`Server is running :
    http://localhost:${port}/`);
});
```

<u>**Login.html:**</u>

```
<!DOCTYPE html>
<html>
 <head>
  <title>Login Page</title>
 <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();
        //generateToken by making a service call to /auth with post
        $.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){
            localStorage.setItem("username",tokenData.data.name);
            localStorage.setItem("job",tokenData.data.work);
            if(verifyLogin(tokenData.token))
            {
              window.location.href="/welcome";
            }
            else{
              alert("Authentication Failed");
            }
          }
```

```
            },
            error: function(data ){
                console.log("Something went wrong");
            }
            });
        });
    });

function verifyLogin(token)
{
    let result=true;
    //verifyToken by making a service call to /verifyToken with get
    $.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>
<body>
    <center>
        <h2>Sign In</h2>
        <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><br>
            <input type="submit" value="Submit">
        </form>
    </center>
</body>
</html>
```

<u>Index.html:</u>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>YOGO Solutions</title>
</head>
<body>
    <center>
        <a href="/signup">Click here to SignUp</a>
        <br>
        <a href="/login">Click here to Login</a>
        <br>
    </center>
</body>
</html>
```

SignUp.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SignUp</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script>
$(document).ready(function() {
  $('#signUpForm').submit(()=>{
  event.preventDefault();
  var name = $("#username").val();
  var password=$("#pwd").val();
  var work=$("#work").val();
  //generateToken by making a service call to /register with post
  $.ajax({
    url:"/register",
    type:"POST",
    contentType:"application/json",
    dataType:"json",
    data: JSON.stringify({
    name: name,
    password: password,
    work:work
    }),
    success: function(data) {
          var tokenData=data;
          if(tokenData.signup==false){
            $("#msg").html('<h6>You are Already registered with us!!!</h6><br><a
id="login-link" href="\login">Login</a>');
          }
          else{
            $("#msg").html('<h6>Successfully registered with us!!!</h6><br><a
id="login-link" href="\login">Login</a>');
            $("login-link").show();
            //window.location.href="/login";
          }
          },
          error: function(data ){
            console.log("Something went wrong");

            }
          });
        });
});
    </script>
</head>
<body>
    <center>
        <h2>SignUp</h2>
        <form id="signUpForm">
            <label for="username">Username:</label><br>
            <input type="text" id="username" name="username"><br>
            <label for="work">Profession:</label><br>
            <input type="text" id="work" name="work"><br>
            <label for="pwd">Password:</label><br>
            <input type="password" id="pwd" name="pwd"><br><br>
            <input type="submit" value="Submit">
        </form>
        <div id="msg"></div>
    </center>
</body>
</html>
```

Welcome.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```html
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome Page</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script>
        var user=localStorage.getItem("username");
        var job=localStorage.getItem("job");
        $(document).ready(function(){
        $("#user").append(user);
        $("#job").append(job);
        });
    </script>
</head>
<body>
    <center>
        <h2>WELCOME TO YOGO SOLUTIONS</h2>
        <hr>
        <div style="border-style:solid; border-color:  blue; width:400px; height:200px">
            <h4 id="user">USER: </h4><br>
            <h4 id="job">PROFESSION: </h4>
        </div>
    </center>
</body>
</html>
```

Output:

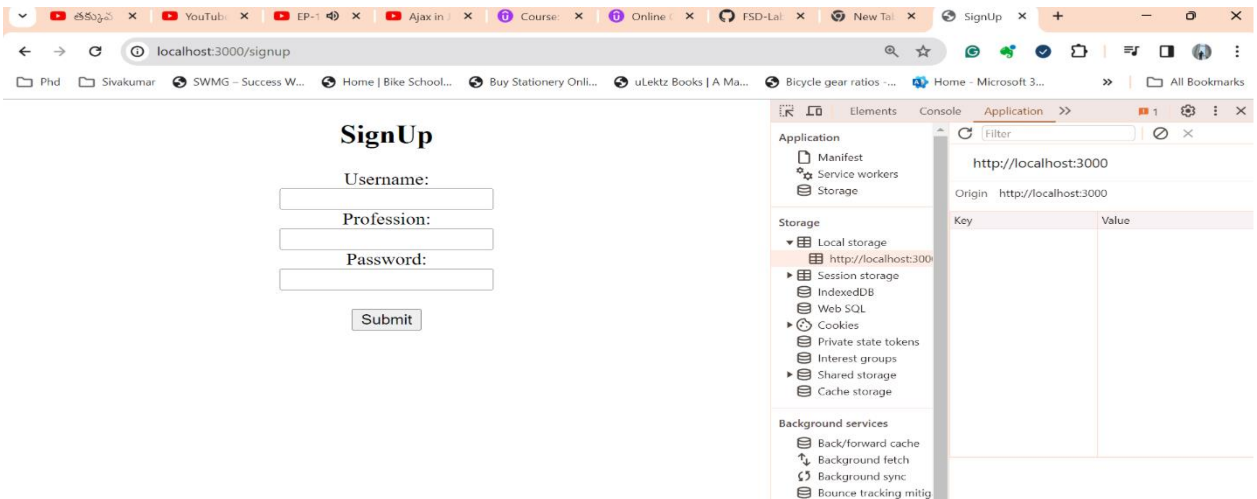
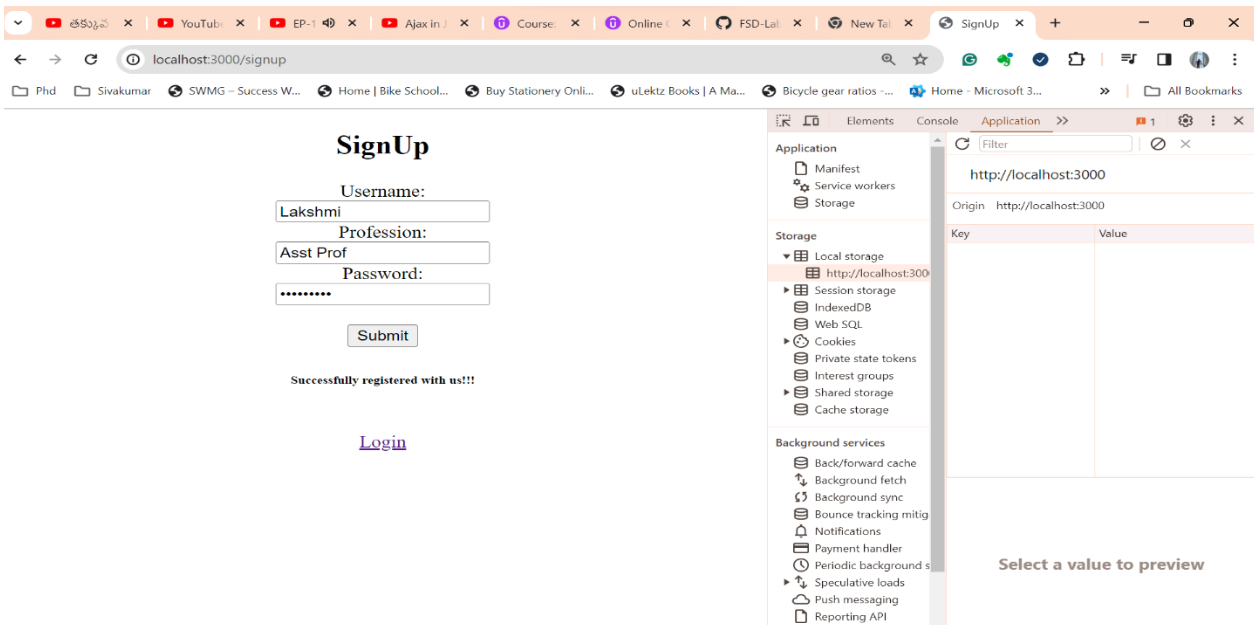


Fig.1. SignUp page



SignUp

Username: Lakshmi

Profession: Asst Prof

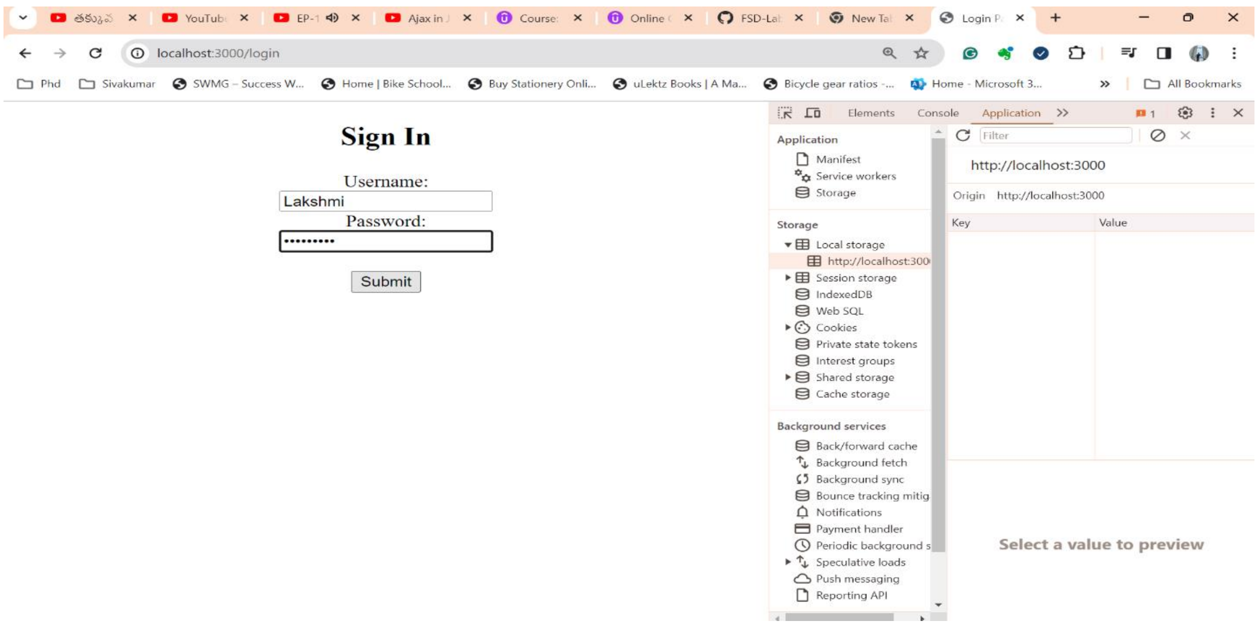
Password: •••••••••

Submit

Successfully registered with us!!!

Login

Fig.2. Successful Register Page



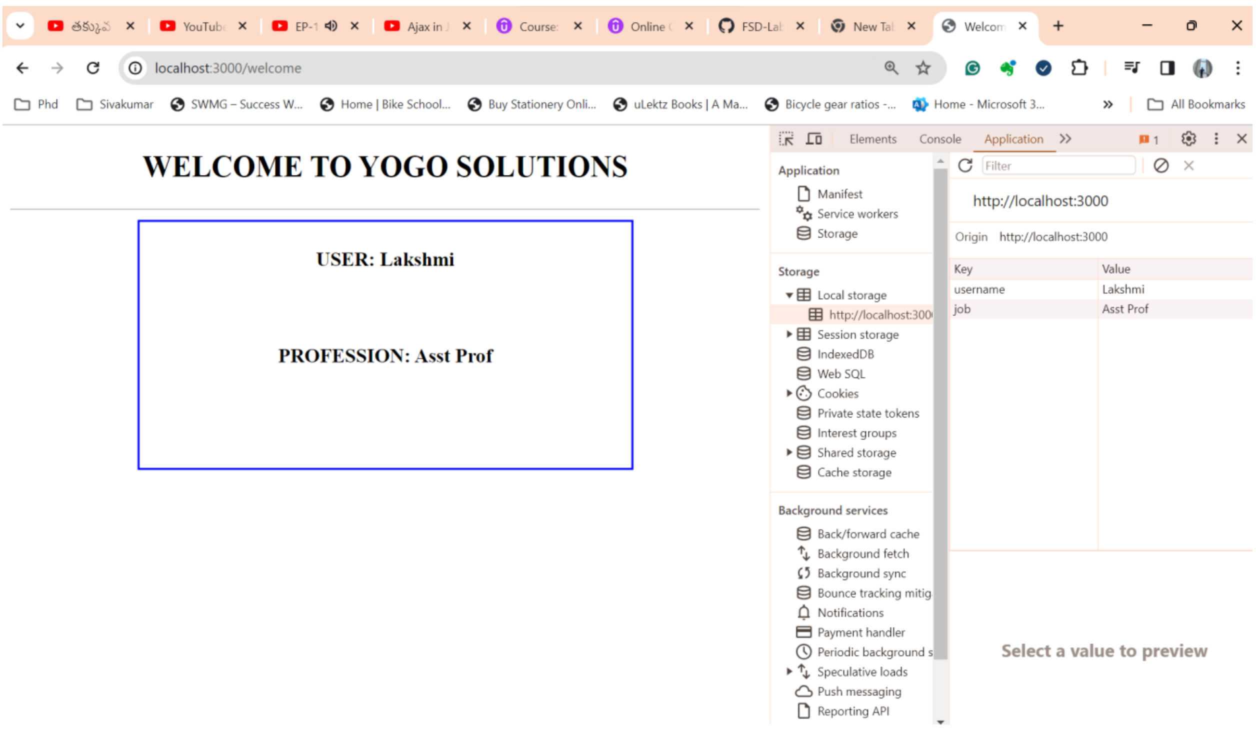
Sign In

Username: Lakshmi

Password: •••••••••

Submit

Fig.3. SignIn page



localhost:3000/welcome

WELCOME TO YOGO SOLUTIONS

USER: Lakshmi

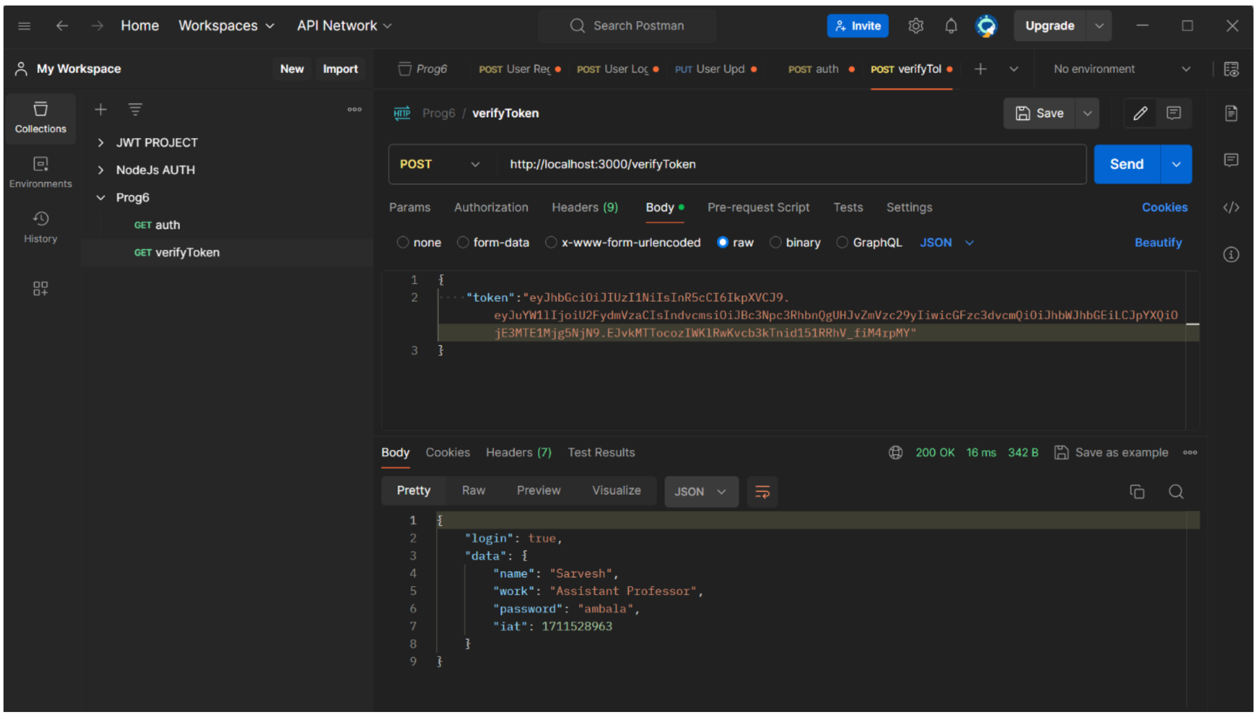
PROFESSION: Asst Prof

http://localhost:3000

Origin http://localhost:3000

Key	Value
username	Lakshmi
job	Asst Prof

Fig.4. Welcome page



POST http://localhost:3000/verifyToken

```
{
    "token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJuYW1lIjoiU2FydmVzaCIsIndvcmsiOiJBc3Npc3RhbnQgUHJvZmVzc29yIiwicGFzc3dvcmQiOiJhbWJhbGEiLCJpYXQiO
        jE3MTE1Mjg5NjN9.EJvkMTTocozIWKlRwKvcb3kTnid151RRhV_fiM4rpMY"
}
```

200 OK 16 ms 342 B

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

Fig.5. JWT Middleware page

Viva Questions:

1.What is the meant by bearer token ?

Ans: Security: Bearer Tokens should be protected from unauthorized access as they grant access rights to whoever holds them. Transport over secure channels (HTTPS) is mandatory to prevent interception.
Statelessness: Bearer Tokens are self-contained, carrying all the necessary information for authentication. This means the server does not need to keep a session state, making this method scalable and suitable for distributed systems.
Usage: They are typically included in the HTTP request's Authorization header with the prefix "Bearer", followed by the token itself. For example: Authorization: Bearer <token>.
Expiration: Bearer Tokens usually have a limited lifetime, after which they expire and can no longer be used. A new token needs to be obtained after expiration.

2.How the consensus problem is useful in web application ?

Ans: The consensus problem, while traditionally discussed in the context of distributed systems and databases, has significant implications for web applications, particularly those that operate on a distributed architecture or require high availability, fault tolerance, and data consistency across multiple nodes or services.

1. Data Consistency
2. High Availability and Fault Tolerance
3. Session Management
4. Distributed Locking
5. Blockchain and Cryptocurrencies
6. Configuration Management

3.What is the use of JWT digital signature ?

Ans:

The digital signature in a JSON Web Token (JWT) serves as a crucial mechanism for ensuring the integrity and authenticity of the token. JWTs are composed of three parts: the header, the payload, and the signature, and they are used in various authentication and authorization processes in web applications and APIs. Here’s why the digital signature is essential:

1.Ensuring Integrity
2.Authenticating the Source
3.Securing Communications
4.Non-repudiation

4. RS256 Vs HS256 ?

Ans: RS256 (RSA Signature with SHA-256) is an asymmetric algorithm, and it uses a public/private key pair: the identity provider has a private (secret) key used to generate the signature, and the consumer of the JWT gets a public key to validate the signature. Since the public key, as opposed to the private key, doesn't need to be kept secured, most identity providers make it easily available for consumers to obtain and use (usually through a metadata URL).

HS256 (HMAC with SHA-256), on the other hand, involves a combination of a hashing function and one (secret) key that is shared between the two parties used to generate the hash that will serve as the signature. Since the same key is used both to generate the signature and to validate it, care must be taken to ensure that the key is not compromised.

5.Why backend is prominent in any web application ?

Ans: The backend of a web application plays a crucial role in its functionality, performance, security, and scalability. It's where the core logic of the application resides,
handling data processing, storage, and communication between the frontend and various other services or databases. Here's why the backend is prominent and indispensable in any web application:

1. Data Management and Storage
2. Business Logic Implementation
3. Authentication and Authorization
4. Integration with Other Services
5. Performance and Scaling
6. Data Security and Compliance
7. Serving Dynamic Content
8. Asynchronous Processing