

Exp-5

Create a simple Node.js server with routes login, register, profile and logout.

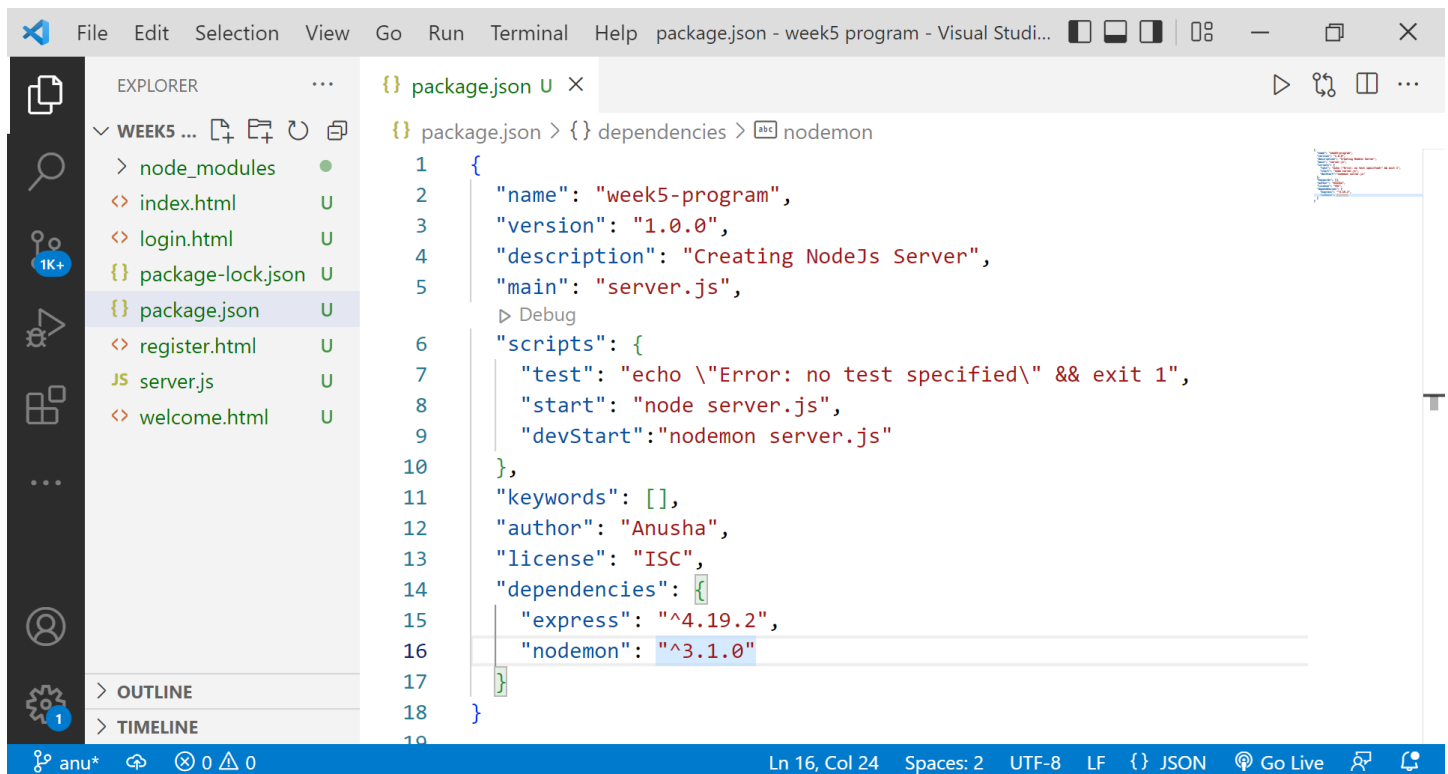


Fig.1. Visual Studio Week5 files and package.json

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

Code:-

index.html:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Week 3 Register and Login page</title>
  <style>
    a{
      background-color: blue;
      color: #f0e8e8;
      /*margin: 1px solid;*/
      font-size: 50px;
    }
    body{
```



```
var fn = frm.fname.value;
localStorage.setItem("firstname",fn)
for(x in fn)
{
    ch=fn.charCodeAt(x);
    if(ch<65||ch>90 && ch<97||ch>122)
    {
        alert("Invalid firstname");
        return false;
    }
}
var ln = frm.lname.value;
localStorage.setItem("lastname",ln)
for(y in ln)
{
    ch=ln.charCodeAt(y);
    if(ch<65||ch>90 && ch<97||ch>122)
    {
        alert("Invalid lastname");
        return false;
    }
}
var phn = frm.phone.value;
localStorage.setItem("phone",phn)
var lenp = phn.length;
if (lenp!==10)
{    alert("Phone no should be exactly 10 digits");
    return false;
}

var pwd1=frm.pwd.value;
localStorage.setItem("password",pwd1)
var pwdl=pwd1.length;
if(pwdl%2===1)
{
    alert("Password should contain even number of characters");
    return false;
}
if(pwdl > 8)
{
    alert("Password should not exceed 8 digits");
```

```

        return false;
    }

    var reg = /^[a-z0-9]+@[a-z]+\.[a-z]{2,3}$/;
    var mail=frm.mailid.value;
    localStorage.setItem("email",mail)
    if (reg.test(mail)){
        alert("Registration Successful!!");
    }
    else{
        alert("Invalid email");
    }
    return false;
}

return true;

}

</script>
<div id="container">
<body>
<center>
    <form name="frm" method="POST" action="/login" onSubmit="return
validate()">

        <fieldset align="center">

            <table align="center">

                <tr><td>First Name: </td><td><input type="text"
name="fname" value="" size="50" required/></td></tr>
                <tr><td>Last Name: </td><td> <input type="text" name="lname"
value="" size="50" required/></td></tr>
                <tr><td>Phone No: </td><td> <input type="text" name="phone"
value="" size="50" required/></td></tr>
                <tr><td>Mail id:</td><td><input type="email" name="mailid"
value="" size="50" required/></td></tr>
                <tr><td>Gender:</td>
                    <td>Male: <input type="radio" name="gender" value="male" >
                        Female: <input type="radio" name="gender"
value="female"></td></tr>
                <tr><td>DOB :</td><td><input type="date" name="dob" size="50"
required/></td></tr>

```

```

        <tr><td>Username:</td><td><input type="text" name="uname"
value="" size="50" required/></td></tr>
        <tr><td>Password:</td><td><input type="password" name="pwd"
value="" size="50" required/></td></tr>

        <tr><td>Age:</td><td><input type="text" name="age" value=""
size="50" required/></td></tr>
    </table>
    <input type="submit" value="SUBMIT" name="submit" />

</fieldset>
</form>
</div>
</center>
</body>
<div id="footer">
    Copyright © CMRIT_2024
</div>
</div>
</html>

```

login.html:-

```

<!DOCTYPE html>
<html>
<head>
    <title>Login Form</title>
</head>
<script>
    function LoginValidate(){

        var enteremail=document.getElementById('email').value;
        var enterpassword=document.getElementById('pwd').value;

        var getEmail=localStorage.getItem('email')
        var getPwd=localStorage.getItem('password')
        if(enteremail==getEmail)
        {
            if(enterpassword==getPwd)
            {

```

```

    alert("login success");
    window.location= "/welcome" ;
    return false;

}
else
{
    alert("wrong password");

}
}
else
{
    alert("invalid details");
}
}

</script>
<body style="text-align: center;">
    <div class="main">
        <h1>Login Form</h1>
        <h3>Enter your login credentials</h3>
        <form >
            <label for="first">
                Email:
            </label>
            <input type="text" name="first" id="email"
                placeholder="Enter your Username" required>
            <br> <br>
            <label for="password">
                Password:
            </label>
            <input type="password" name="password" id="pwd"
                placeholder="Enter your Password" required>
            <br> <br>
            <button type="submit" onclick="return LoginValidate()">Submit</button>
            <br>
        </div>
    </form>
    <p>Not registered?
        <a href="/register"
            style="text-decoration: none;">

```

```
        Create an account
    </a>
</p>
</div>
</body>
</html>
```

welcome.html:-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>welcome</title>
</head>
<script>
    function LogoutValidate(){
        localStorage.clear();
    }
    function getdetails(){
        var getfirstname=localStorage.getItem("firstname");
        var getemail=localStorage.getItem("email")
        document.getElementById("data").innerHTML="welcome to
"+getfirstname+" <br>" +getemail;
    }
</script>
<body onload="getdetails()">
    <h1 id="data">Welcome to Cmrit</h1>
    <form action="/" method="post">
        <button type="submit" onclick="LogoutValidate()">Logout</button>
    </form>
</body>
</html>
```

server.js:-

```
// Importing express module
const express = require('express');
const app = express();
```

```
//app.use(express.json());

app.get('/',
  (req, res) => {
    res.sendFile(__dirname + '/index.html');
  });

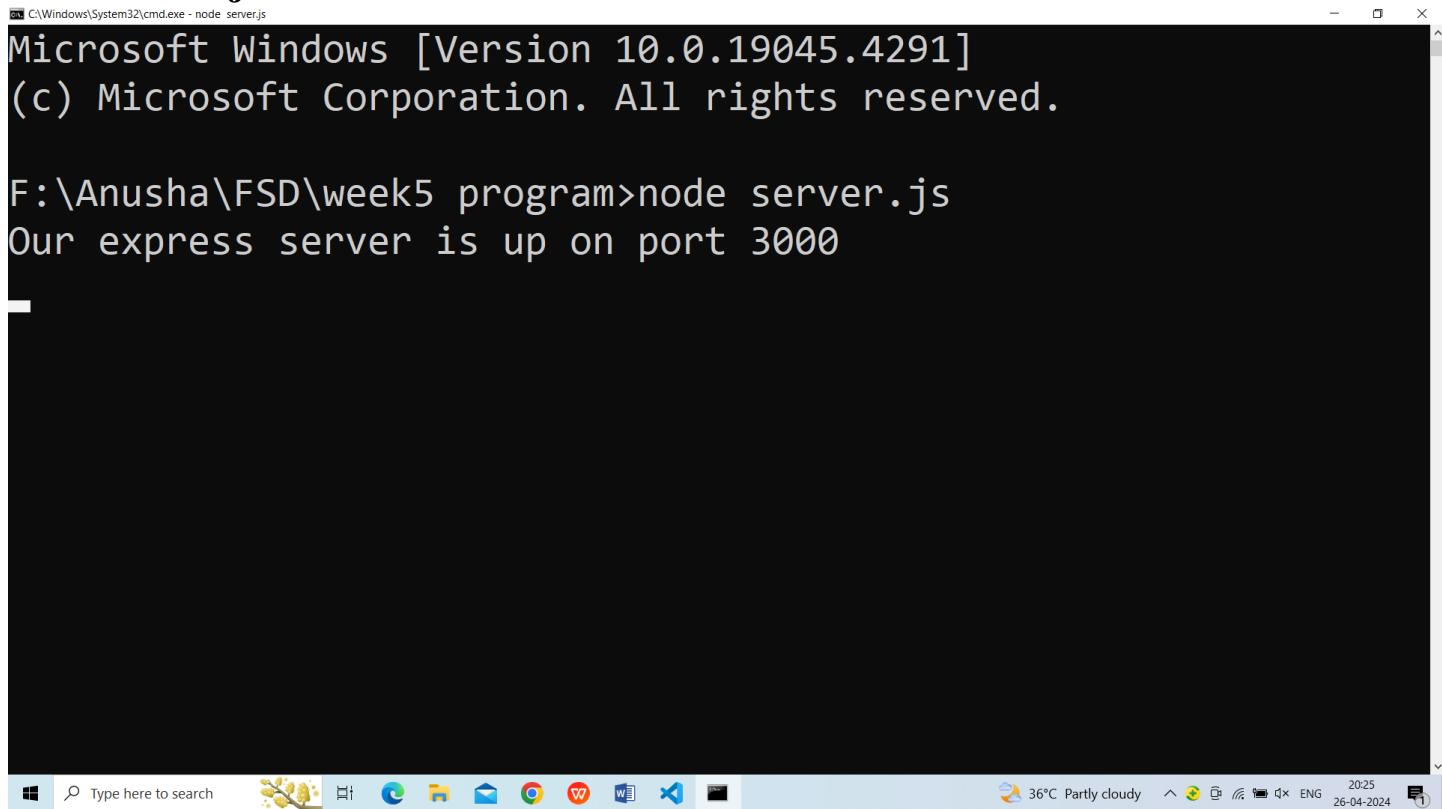
app.get('/login',
  (req, res) => {
    res.sendFile(__dirname + '/login.html');
  });
app.get('/register',
  (req, res) => {
    res.sendFile(__dirname + '/register.html');
  });
app.get('/welcome',
  (req, res) => {
    res.sendFile(__dirname + '/welcome.html');
  });
app.post('/',(req, res) => {
  res.redirect("/")
});
app.post('/login',(req, res) => {
  res.redirect("/login")
});
app.post('/register',(req, res) => {
  res.redirect("/register")
});
app.post('/welcome',(req, res) => {
  res.redirect("/welcome")
});

app.listen(3000,
  () => {
    console.log(
      'Our express server is up on port 3000'
    );
  });
```


Output:-

Run code use command

- **node server.js**



```
C:\Windows\System32\cmd.exe - node server.js
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

F:\Anusha\FSD\week5 program>node server.js
Our express server is up on port 3000
```

Fig.2.Now server running successfull

Open server in any browser like below format my port number is 3000

<http://localhost:3000/>



[Login](#) [Register](#)



Fig.3.index page

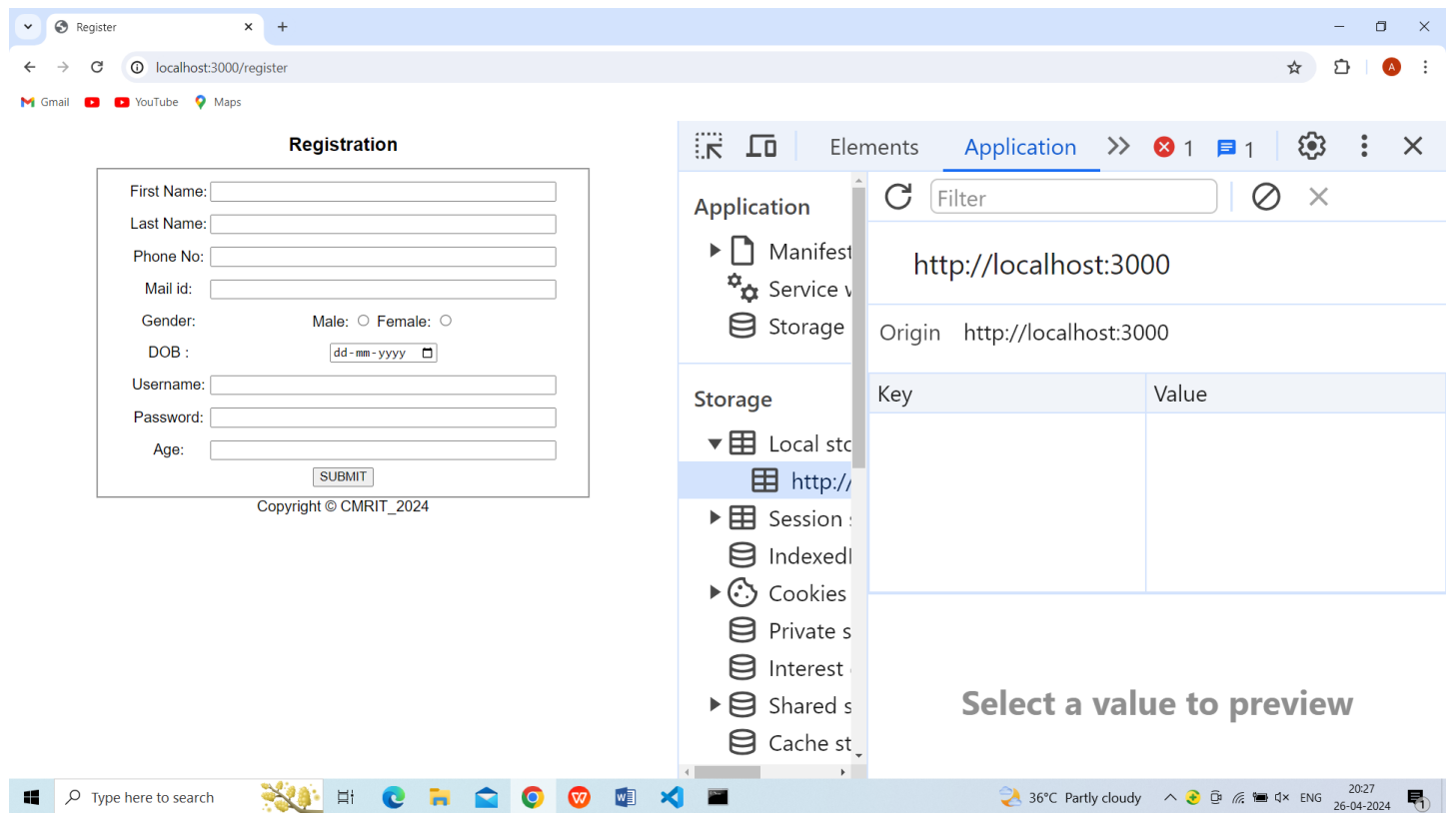


Fig.4.Registration Page

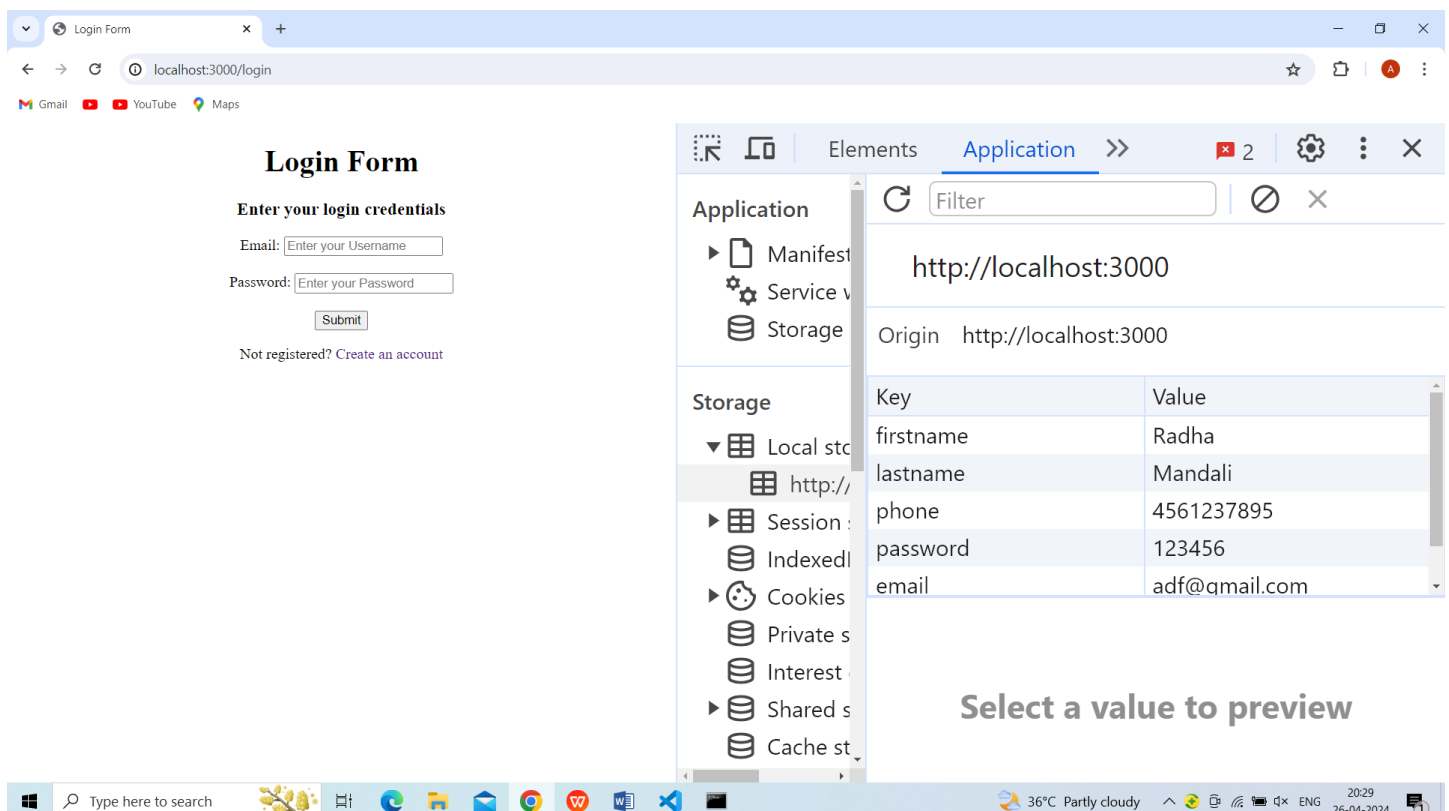


Fig.5.Login page

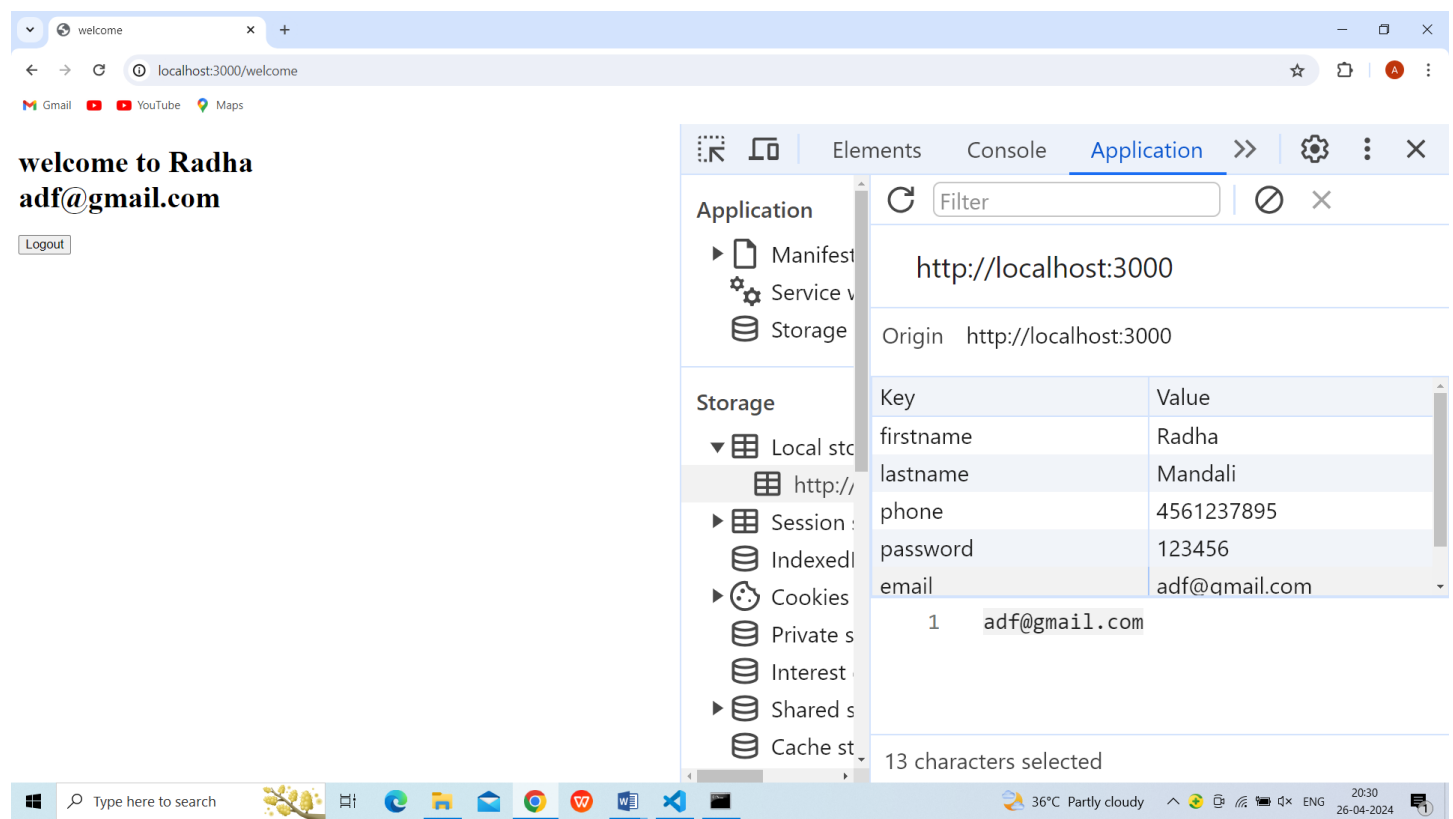


Fig.6.Profile Page

1. What is AJAX?

Ans: AJAX stands for Asynchronous JavaScript and XML. It's a set of web development techniques that allows web applications to send and retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Despite the name, AJAX applications can use any kind of data, not just XML, and JSON (JavaScript Object Notation) has become the more popular format due to its lightweight nature and easy integration with JavaScript.

2. What are the security issues with AJAX ?

Ans: AJAX, while powerful and instrumental in building dynamic and responsive web applications, introduces several security considerations that developers must address. These issues stem from the same complexities that make AJAX so useful: dynamic content loading, asynchronous server communication, and client-side processing. Here are some of the primary security issues associated with AJAX

- 1.Cross-Site Scripting (XSS)
- 2.Cross-Site Request Forgery (CSRF)
- 3.Security Misconfiguration
- 4.Insecure Direct Object References (IDOR)
- 5.Data Leakage
- 6.Same-Origin Policy and CORS Issues

3. What are the tools for debugging AJAX applications ?

Ans: Debugging AJAX applications can be intricate due to their asynchronous nature and the dynamic interaction between the client and server. Fortunately, there are several tools and features within modern web development environments designed to help with debugging AJAX requests and responses, as well as the client-side JavaScript that manages them. Here's an overview of some key tools and techniques:

- 1.Browser Developer Tools
- 2.JavaScript Debugging
- 3.Proxy Tools
- 4.AJAX Debugging Libraries
5. Testing Frameworks

4. What are the common AJAX frameworks ?

Ans:

AJAX (Asynchronous JavaScript and XML) frameworks and libraries play a crucial role in simplifying the development of web applications that require asynchronous data fetching and updating of web pages without a full reload. Over the years, numerous frameworks have been developed to abstract the complexities of making AJAX calls, handling responses, and manipulating the DOM based on those responses

5. What is jquery filter method ?

Ans:

The jQuery `.filter()` method is a powerful tool that allows developers to reduce the set of matched elements to those that match a specific selector, pass a test provided by a function, or match the elements against a specific jQuery object. This method is particularly useful when you need to refine search results from a broad selection to a more targeted subset based on certain criteria.