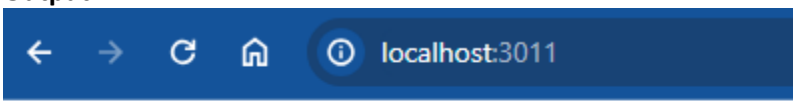
Slip 2: Create a Node.js file that will convert the output "Full Stack!" into reverse string

```
const http = require('http');

// Define a function to reverse a string
function reverseString(inputString) {
    // Split the string into an array of characters, reverse the array, and join
it back into a string
    return inputString.split('').reverse().join('');
}

// Create a server
const server = http.createServer((req, res) => {
    // String to reverse
    const inputString = "Full stack!";

    // Reverse the input string
    const reversedString = reverseString(inputString);

    // Set response headers
    res.writeHead(200, { 'Content-Type': 'text/plain' });

    // Send the reversed string as response
    res.end('Reversed string: ' + reversedString);
}).listen(3011, () => {
    console.log('Server is running on port 3011');
});
```

Output-

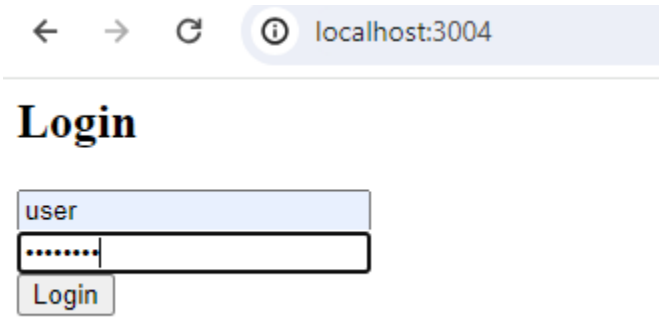


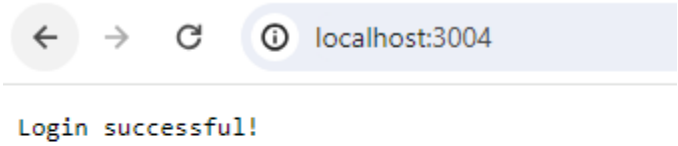
Reversed string: !kcats lluF

slip 3-Using node js create a User Login System.

```
const http = require('http');
const qs = require('querystring');
// Dummy user data (for simplicity)
const users = [
    { username: 'user', password: 'password' }
```

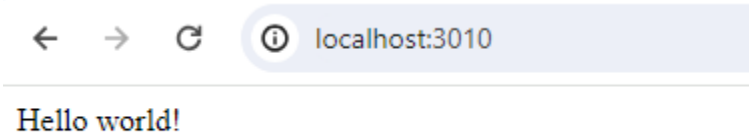
```
];
const server = http.createServer((req, res) => {
 if (req.method === 'GET') {
 res.writeHead(200, {'Content-Type': 'text/html'});
 res.write(`
            <h2>Login</h2>
            <form method="POST" action="/">
                <input type="text" name="username" placeholder="Username"
required><br>
                <input type="password" name="password" placeholder="Password"
required><br>
                <button type="submit">Login</button>
            </form>
        `);
        res.end();
 } else if (req.method === 'POST') {
    let body = '';
    req.on('data', chunk => {
        body += chunk.toString();
    });
    req.on('end', () => {
        const { username, password } = qs.parse(body);
        const user = users.find(u => u.username === username && u.password ===
password);
        if (user) {
            res.writeHead(200, { 'Content-Type': 'text/plain' });
            res.write('Login successful!');
            res.end();
        } else {
            res.writeHead(401, { 'Content-Type': 'text/plain' });
            res.write('Invalid username or password');
            res.end();
        }
    });
}
}).listen(3004, () => {
 console.log('Server is running on port 3004');
})
```





Slip14: Create a Simple Web Server using node js.

```
var http=require('http');
http.createServer(function(req,res){
    res.writeHead(200,{'Content-Type':'text/html'});
    res.write('Hello world!');
    res.end();
}).listen(3010,()=>{
    console.log('Active port 3010');
});
```



Slip13 Create a Node.js file that will convert the output "HELLO WORLD!" into lower-case letters.

```
var http= require('http');
http.createServer(function (req, res) {
    var str = "Hello World";
    res.write(str.toLowerCase());
    res.end();
}).listen(8089, function () {
    console.log('port 8089 is started');
});
```

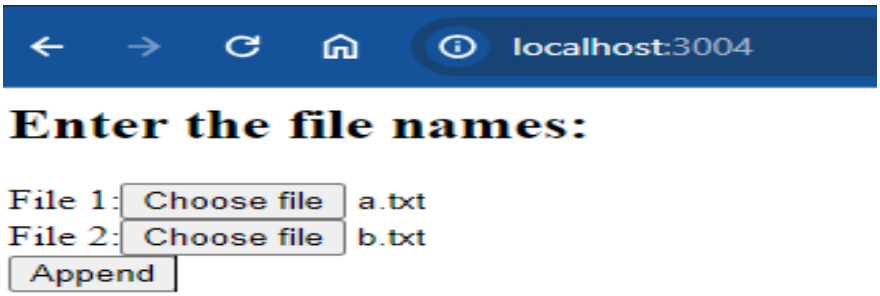
localhost:8089

hello world

Slip2-
Using node js create a web page to read two file names from user and append contents of first file into second file.

```
const http = require('http');
const fs = require('fs');
const querystring = require('querystring');

const server = http.createServer((req, res) => {
  if (req.method === 'GET') {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('<html><body>');
    res.write('<h2>Enter the file names:</h2>');
    res.write('<form action="/" method="post">');
    res.write('<label for="file1">File 1:</label>');
    res.write('<input type="file" id="file1" name="file1"><br>');
    res.write('<label for="file2">File 2:</label>');
    res.write('<input type="file" id="file2" name="file2"><br>');
    res.write('<input type="submit" value="Append">');
    res.write('</form>');
    res.write('</body></html>');
    res.end();
  } else if (req.method === 'POST') {
    let body = '';
    req.on('data', chunk => {
      body += chunk.toString();
    });
    req.on('end', () => {
      const postData = querystring.parse(body);
      const file1 = postData['file1'];
      const file2 = postData['file2'];

      fs.readFile(file1, 'utf8', (err, data) => {
        if (err) {
          res.writeHead(404, {'Content-Type': 'text/html'});
          res.write('<html><body><h1>File not found!</h1></body></html>');
          res.end();
        } else {
          fs.appendFile(file2, data, 'utf8', err => {
            if (err) {
              res.writeHead(500, {'Content-Type': 'text/html'});
              res.write('<html><body><h1>Error appending file!</h1></body></html>');
              res.end();
            } else {
              res.writeHead(200, {'Content-Type': 'text/html'});
              res.write('<html><body><h1>File appended successfully!</h1></body></html>');
              res.end();
```

```
                }
            });
        }
    });
  });
}
}).listen(3004, () => {
  console.log('Server is running on port 3004');
});
```

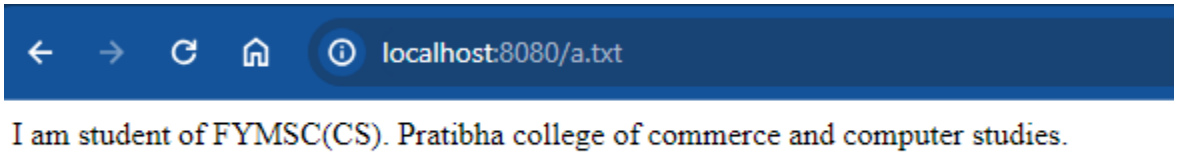


Slip6

Create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong, throw a 404 error.

```
var http = require('http');
var url = require('url');
var fs = require('fs');

http.createServer(function(req,res){
  var q = url.parse(req.url,true);
  var filename = "." +q.pathname;
  fs.readFile(filename,function(err,data){
    if(err){
      res.writeHead(404,{'content-type':'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200,{'content-type':'text/html'});
    res.write(data);
```

```
        return res.end();
    });
}).listen(8080);
```

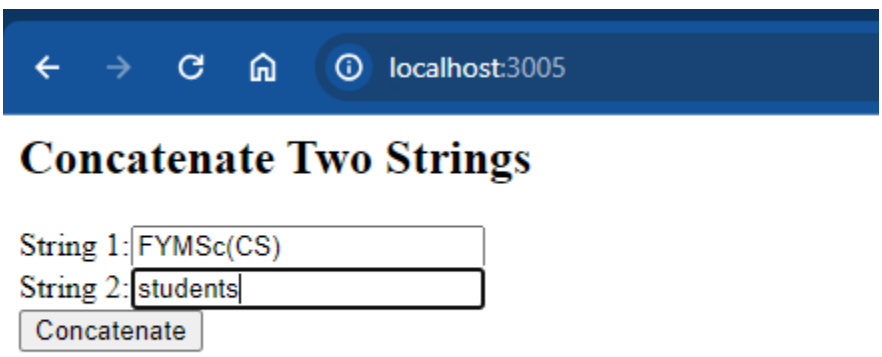


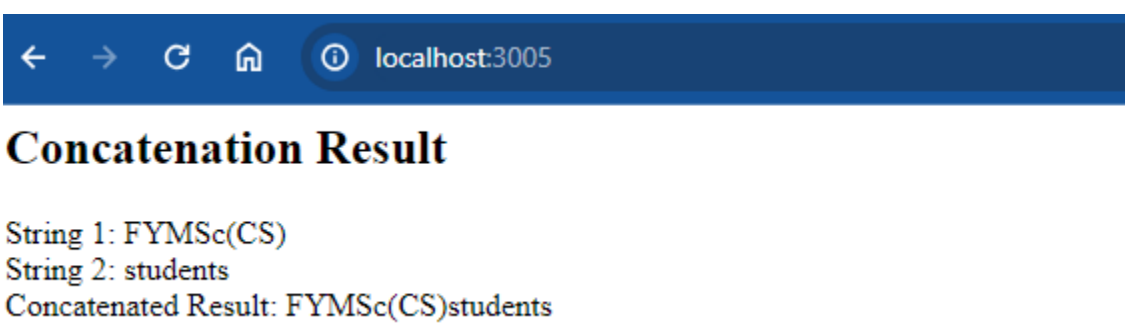
Slip9

Create a Node.js file that writes an HTML form, with a concatenate two string.

```
const http = require('http');
const querystring = require('querystring');

const server = http.createServer((req, res) => {
    if (req.method === 'GET') {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write('<html><body>');
        res.write('<h2>Concatenate Two Strings</h2>');
        res.write('<form action="/" method="post">');
        res.write('<label for="string1">String 1:</label>');
        res.write('<input type="text" id="string1" name="string1" required><br>');
        res.write('<label for="string2">String 2:</label>');
        res.write('<input type="text" id="string2" name="string2" required><br>');
        res.write('<input type="submit" value="Concatenate">');
        res.write('</form>');
        res.write('</body></html>');
        res.end();
    } else if (req.method === 'POST') {
        let body = '';
```

```
    req.on('data', chunk => {
      body += chunk.toString();
    });
    req.on('end', () => {
      const formData = querystring.parse(body);
      const string1 = formData['string1'];
      const string2 = formData['string2'];
      const result = string1 + string2;

      res.writeHead(200, { 'Content-Type': 'text/html' });
      res.write('<html><body>');
      res.write('<h2>Concatenation Result</h2>');
      res.write('<p>');
      res.write(`String 1: ${string1}<br>`);
      res.write(`String 2: ${string2}<br>`);
      res.write(`Concatenated Result: ${result}`);
      res.write('</p>');
      res.write('</body></html>');
      res.end();
    });
  }
}).listen(3005, () => {
  console.log('Server is running on port 3005');
});
```





Slip 23
Write node js script to interact with the file system, and serve a web page from a File.

a.txt
I am student of FYMSC(CS).
Pratibha college of commerce and computer studies.

Slip23.js

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile('a.txt', function(err, data) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8081,()=>{
  console.log("Port 8081 is active")
});
```

Slip 7 Create a node js file named main.js for event-driven application. There should be a main loop that listens for events, and then triggers a callback function when one of those events is detected.

```
//import event modules
var events = require('events');

//create an eventEmitter object
var eventEmitter = new events.EventEmitter();

//create an event handler
var connectHandler = function(s){
   console.log('Its',s);
}

//Bind the connection event with the Handler
eventEmitter.on('data_received',function(name){
   console.log(name,"Understood event -Driven");
});
eventEmitter.emit('data_received',"FYMSC(CS)");
```

```
eventEmitter.on('connection',connectHandler);
eventEmitter.emit('connection',"SIMPLE SOLUTION")

console.log("program Ended");
```

output-

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\ADMIN\Documents\angular> cd NodeJS
PS C:\Users\ADMIN\Documents\angular\NodeJS> node slip7Event.js
FYMSC(CS) Understood event -Driven
Its SIMPLE SOLUTION
program Ended
PS C:\Users\ADMIN\Documents\angular\NodeJS>
```

Slip 10 Write node js script to build Your Own Node.js Module. Use require ('http') module is a builtin Node module that invokes the functionality of the HTTP library to create a local server. Also use the export statement to make functions in your module available externally. Create a new text file to contain the functions in your module called, "modules.js" and add this function to return today's date and time.

Modules.js

```
function datetime()
{
    let dt = new Date();
    //current date
    let date = ("0"+dt.getDate()).slice(-2);

    //current month
    let month = ("0"+ (dt.getMonth()+1)).slice(-2);

    //current year
    let year = dt.getFullYear();

    //current hours
    let hours = dt.getHours();

    //current minutes
    let minutes = dt.getMinutes();
```

```
    //current seconds
    let seconds = dt.getSeconds();

    var output = year + "-" +month + "-" + date + " " + hours
+":"+minutes+":"+seconds;
    return output;
}
module.exports = {datetime}
```

slip10.js

```
var http = require('http');
var dt = require('./modules');

var server = http.createServer(function(req,res){
   res.writeHead(200,{'content-type':'text/html'});
   const result = dt.datetime();
   res.write('current date and time is ');
   res.write(result);
   res.end();

});
server.listen(1234);
```

output-

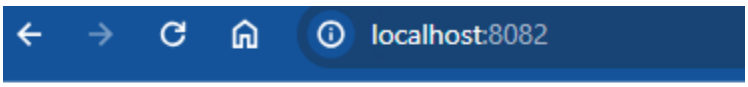
current date and time is 2024-04-05 10:39:31

Slip 11: Write node js application that transfer a file as an attachment on web and enables browser to prompt the user to download file using express js.

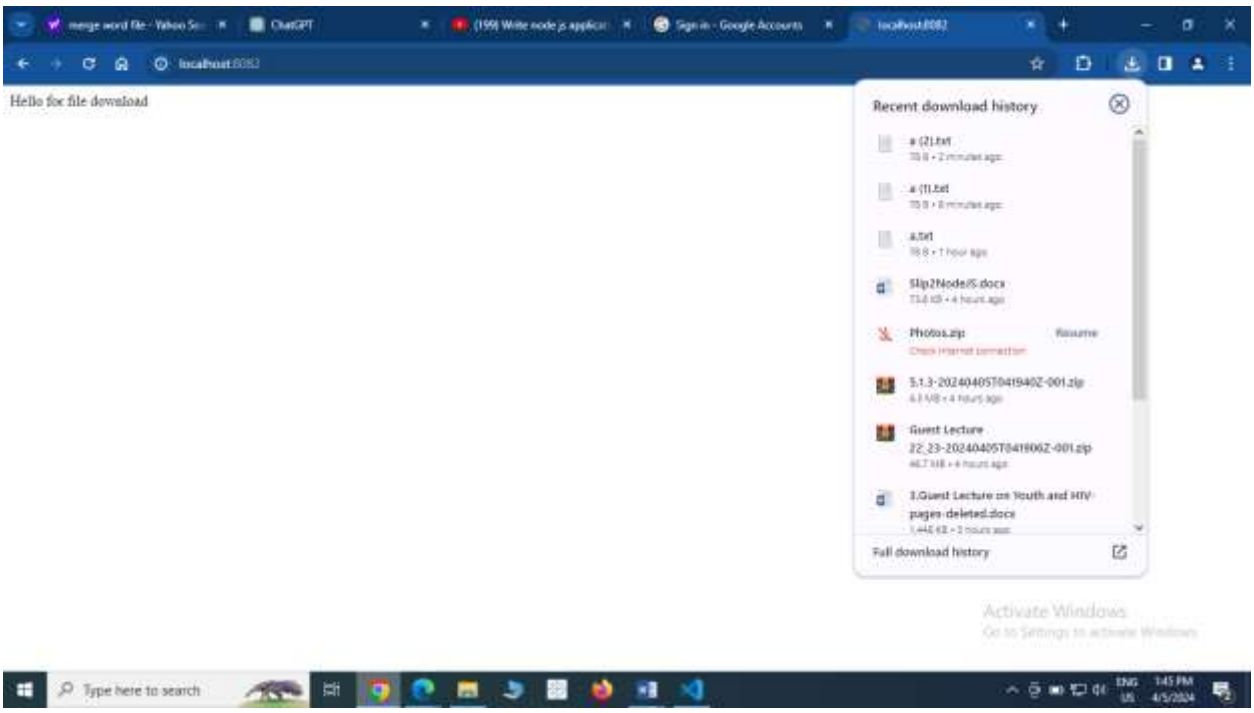
```
const express = require('express');
const app = express();

app.get('/',function(req,res){
```

```
    res.send("Hello for download")
})
app.get('/download',function(req,res){
    res.download('a.txt')
})

// Start the server
app.listen(8082, () => {
    console.log('Server is running on port 8082');
});
```

Output-



Hello for file download



Slip22

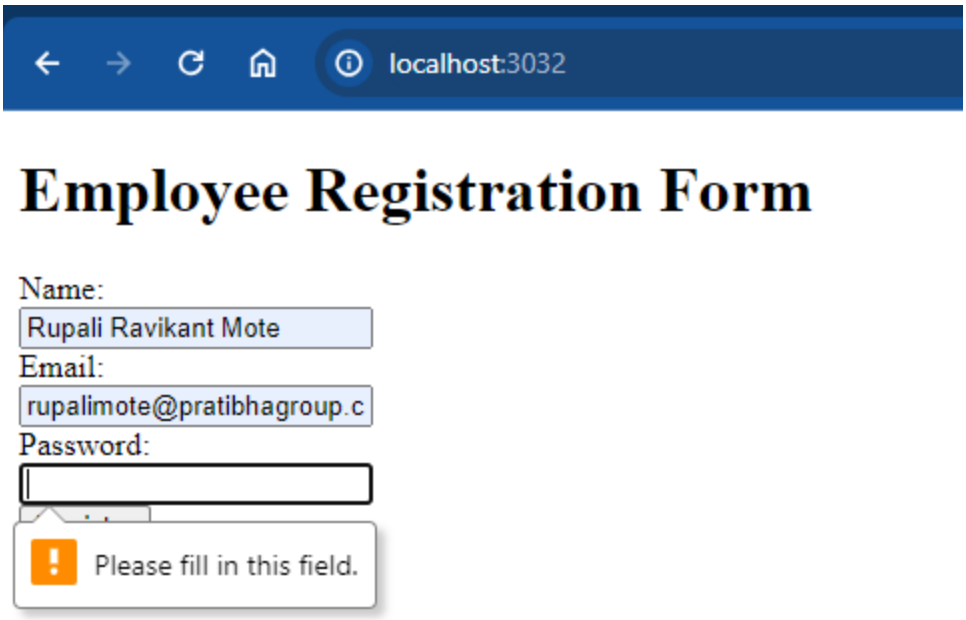
Using node js create an Employee Registration Form validation.

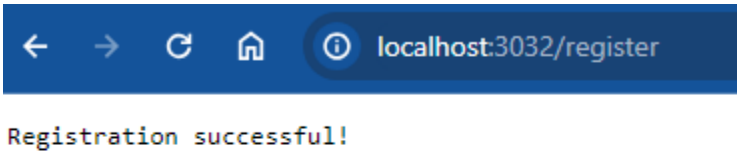
Index.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Employee Registration</title>
</head>
<body>
    <h1>Employee Registration Form</h1>
    <form action="/register" method="post">
        <label>Name:</label><br>
        <input type="text" name="name" required><br>

        <label>Email:</label><br>
        <input type="email" name="email" required><br>

        <label>Password:</label><br>
        <input type="password" name="password" required><br>

        <input type="submit" value="Register">
    </form>
</body>
</html>
```

```javascript
const http = require('http');
const fs = require('fs');


const server = http.createServer((req, res) => {
    if (req.method === 'GET' && req.url === '/') {
        // Serve HTML form
        fs.readFile('index.html', (err, data) => {
            if (err) {
                res.writeHead(500);
```

```
                res.end('Internal Server Error');
                return;
            }
            res.writeHead(200, { 'Content-Type': 'text/html' });
            res.end(data);
        });
    } else if (req.method === 'POST' && req.url === '/register') {
        // Handle form submission
        let body = '';
        req.on('data', chunk => {
            body += chunk.toString();
        });
        req.on('end', () => {
            const formData = new URLSearchParams(body);
            const name = formData.get('name');
            const email = formData.get('email');
            const password = formData.get('password');

            // Basic form validation
            if (!name || !email || !password) {
                res.writeHead(400);
                res.end('Please fill out all fields');
            } else {
                // If validation passes, process registration
                res.writeHead(200);
                res.end('Registration successful!');
            }
        });
    } else {
        // Handle other routes
        res.writeHead(404);
        res.end('Not Found');
    }
}).listen(3032, () => {
    console.log('Server is running on port 3032');
});
```


localhost:3032
Employee Registration Form
Name:
Rupali Ravikant Mote
Email:
rupalimote@pratibhagroup.c
Password:
Please fill in this field.


localhost:3032/register
Registration successful!

Slip22

Using node js create an Employee Registration Form validation.

```html
<!DOCTYPE html>
<html>
<head>
<title>Employee Registration</title>
</head>
<body>
<h1>Employee Registration Form</h1>
<form action="/register" method="post">
<label>Name:</label><br>
<input type="text" name="name" required><br>
<label>Email:</label><br>
<input type="email" name="email" required><br>
<label>Password:</label><br>
<input type="password" name="password" required><br>
<input type="submit" value="Register">
</form>
</body>
</html>
```

Slip22.js

```javascript
const http = require('http');
const fs = require('fs');
const qs =require('querystring');
const server = http.createServer((req, res) => {
if (req.method === 'GET' && req.url === '/') {
// Serve HTML form
fs.readFile('index.html', (err, data) => {
if (err) {
res.writeHead(500);
res.end('Internal Server Error');
return;
}
res.writeHead(200, { 'Content-Type': 'text/html' });
res.end(data);
});
} else if (req.method === 'POST' && req.url === '/register') {
// Handle form submission
let body = '';
req.on('data', chunk => {
body += chunk.toString();
});
req.on('end', () => {
const formData =qs.parse(body);
const name = formData['name'];
const email = formData['email'];
const password = formData['password'];
```

```
// Basic form validation
if (!name || !email || !password) {
res.writeHead(400);
res.end('Please fill out all fields');
} else {
// If validation passes, process registration
res.writeHead(200);
res.end('Registration successful!');
}
});
} else {
// Handle other routes
res.writeHead(404);
res.end('Not Found');
}
}).listen(3032, () => {
console.log('Server is running on port 3032');
});
```

Slip4

Using node js create an eLearning System.

Slip4.html

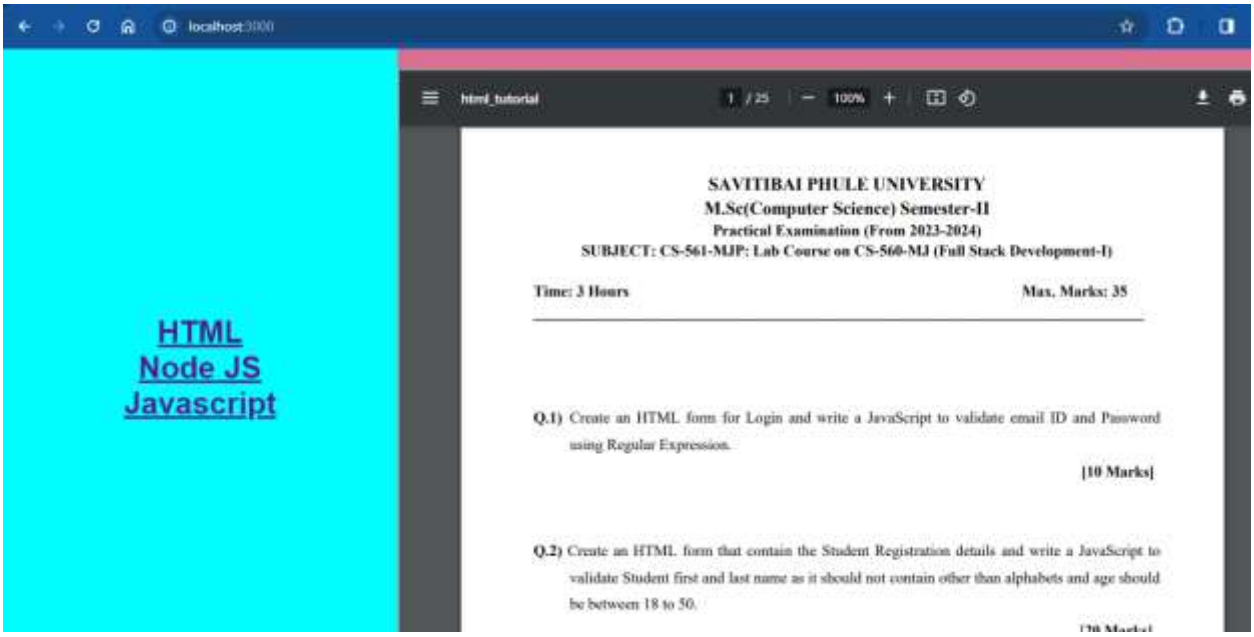
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        body{ font-family:Arial;
            color: white;
        }
        .split1{
            height: 100%;
            width: 30%;
            position: fixed;
            z-index:1;
            top:0;
            overflow-x: hidden;
            padding-top: 20px;
        }
        .split{
            height: 100%;
            width: 70%;
            position: fixed;
            z-index:1;
            top:0;
            overflow-x: hidden;
            padding-top: 20px;
        }
        .left{
            left:0;
            background-color: aqua;
        }
        .right{
            right:0;
            background-color:palevioletred;
        }
        .centered{
            position: absolute;
            transform: translate(-50%,-50%);
            top:50%;
```

```
            left:50%;
            text-align: center;
        }
        .link{
            position: absolute;
            height: 100%;
            width: 100%;
            top:0;
            z-index: 1;
        }
    </style>
</head>
<body>
    <h1>eLearning System</h1>
    <div class="split1 left">
        <div class="centered">
            <h1>
            <a href="/html_tutorial" target="a">HTML</a><br>
            <a href="/nodejs_tutorial" target="a">Node JS</a><br>
            <a href="/javascript_tutorial" target="a">Javascript</a>
            </h1>
        </div>
    </div>
    <div class="split right">
        <iframe name="a" height="100%" width="100%"></iframe>
    </div>
</body>
</html>
```

Slip4.js

```
var http=require('http');
var fs=require('fs');
http.createServer(function(req,res){
    if(req.url=='/')
    {
        fs.readFile('slip4.html',function(err,data){
            res.writeHead(200,{'Content-type':'text/html'});
            res.write(data);
            res.end();
        });
    }
   else if(req.url=='/html_tutorial')
    {
        fs.readFile('html_tutorial.pdf',function(err,data){
```

```
            res.writeHead(200,{'Content-type':'application/pdf'});
            res.write(data);
            res.end();
        });
    }
    else if(req.url=='/nodejs_tutorial')
    {
        fs.readFile('nodejs_tutorial.pdf',function(err,data){
            res.writeHead(200,{'Content-type':'application/pdf'});
            res.write(data);
            res.end();
        });

    }
   else if(req.url=='/javascript_tutorial')
    {
        fs.readFile('javascript_tutorial.pdf',function(err,data){
            res.writeHead(200,{'Content-type':'application/pdf'});
            res.write(data);
            res.end();
        });
    }
    else{
        res.end('end');
    }
}).listen(3000,()=>{
    console.log('port is active')
})
```


localhost:3000
HTML
Node JS
Javascript
html_tutorial
1 / 25
100%
SAVITIBAI PHULE UNIVERSITY
M.Sc(Computer Science) Semester-II
Practical Examination (From 2023-2024)
SUBJECT: CS-561-MJP: Lab Course on CS-560-MJ (Full Stack Development-I)
Time: 3 Hours
Max. Marks: 35
Q.1) Create an HTML form for Login and write a JavaScript to validate email ID and Password using Regular Expression.
[10 Marks]
Q.2) Create an HTML form that contain the Student Registration details and write a JavaScript to validate Student first and last name as it should not contain other than alphabets and age should be between 18 to 50.