SLIP:-1

Q.1) Create an HTML form for Login and write a JavaScript to validate email ID and Password using Regular Expression

SOL:-

```
<!DOCTYPE html>
<html>

<head>
   <title>Simple Login Form</title>
   <style>
     .container {
   width: 300px;
   margin: 0 auto;
   }
  </style>
</head>

<body>
<div class="container">
   <h2>Login</h2>
   <class="container">
   <form onsubmit="return validateLogin()">
      <label>Email:</label><br>
      <input type="text" id="email"><br><br>

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

      <input type="submit" value="Login">
      <p id="error" style="color: red;"></p>
   </form>
 </div>
   <script>
      function validateLogin() {
         var email = document.getElementById("email").value;
         var password = document.getElementById("password").value;
         var error = document.getElementById("error");

         // Basic email pattern
         var emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
         // Password: at least 6 characters
         var passwordPattern = /^.{6,}$/;

         if (!emailPattern.test(email)) {
            error.textContent = "Invalid email format.";
            return false;
         }
```

```
        if (!passwordPattern.test(password)) {
            error.textContent = "Password must be at least 6 characters.";
            return false;
        }

        error.textContent = ""; // Clear error
        alert("Login successful!");
        return true;
    }
  </script>

</body>

</html>
```

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.

SOL:-

```
<!DOCTYPE html>
<html>
<head>
 <title>Student Registration</title>
 <style>
  body {
  font-family: Arial, sans-serif;
  }
  .container {
  width: 300px;
  margin: 0 auto;
  }
  input[type="text"], input[type="number"] {
  width: 100%;
  padding: 10px;
  margin: 10px 0;
  box-sizing: border-box;
  }
  input[type="submit"] {
  width: 100%;
  padding: 10px;
  margin-top: 10px;
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
  }
```

```
    input[type="submit"]:hover {
    background-color: #45a049;
    }

   </style>
</head>
<body>
  <div class="container">

  <h2>Student Registration</h2>

  <form onsubmit="return validateForm()">
    First Name: <input type="text" id="firstName"><br><br>
    Last Name: <input type="text" id="lastName"><br><br>
    Age: <input type="number" id="age"><br><br>
    <input type="submit" value="Submit">
  </form>
</div>
  <script>
    function validateForm() {
      let firstName = document.getElementById("firstName").value;
      let lastName = document.getElementById("lastName").value;
      let age = parseInt(document.getElementById("age").value);

      let namePattern = /^[A-Za-z]+$/;

      if (!namePattern.test(firstName)) {
        alert("First name must contain only letters.");
        return false;
      }

      if (!namePattern.test(lastName)) {
        alert("Last name must contain only letters.");
        return false;
      }

      if (isNaN(age) || age < 18 || age > 50) {
        alert("Age must be between 18 and 50.");
        return false;
      }

      alert("Form submitted successfully!");
      return true;
    }
  </script>

</body>
</html>
```


SLIP2

Q.1) Create a Node.js file that will convert the output "Full Stack!" into reverse string.

sol:-

```
const str = "Full Stack!";
const reversed = str.split('').reverse().join('');
console.log("Reversed String:", reversed);
```


Q.2) Using node js create a web page to read two file names from user and append contents of first file into second file.

(not run node is not install)

solution:-

```
// Import the filesystem module
const fs = require('fs');
// Get the file contents before the append operation
console.log("\nFile Contents of file before append:",
fs.readFileSync("programming.txt","utf8"));
fs.appendFile("programming.txt","Welcome to node js",
(err) => {
if (err) {
console.log(err);
}
else {
// Get the file contents after the append operation
console.log("\nFile Contents of file after append:",
fs.readFileSync("programming.txt","utf8"));
}
});
```

Programming.txt

node

SLIP3

Q.1) Using node js create a User Login System.

solution:-

```
// login.js

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

// Hardcoded user data
```

```
const users = [
  { username: 'admin', password: 'admin123' },
  { username: 'user', password: '12345' }
];

// Create server
http.createServer((req, res) => {
  if (req.method === 'GET') {
    // Display login form
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(`
      <h2>User Login</h2>
      <form method="POST">
        Username: <input type="text" name="username"><br><br>
        Password: <input type="password" name="password"><br><br>
        <input type="submit" value="Login">
      </form>
    `);
  } else if (req.method === 'POST') {
    let body = '';

    req.on('data', chunk => body += chunk);
    req.on('end', () => {
      const data = querystring.parse(body);
      const { username, password } = data;

      const found = users.find(user => user.username === username &&
user.password === password);

      res.writeHead(200, { 'Content-Type': 'text/html' });
      if (found) {
        res.end(`<h3>Welcome, ${username}!</h3>`);
      } else {
        res.end(`<h3>Login failed. Try again.</h3>`);
      }
    });
  }
}).listen(3000, () => {
  console.log('Server running at http://localhost:3000');
});
```

Q.2) Create a node.js file that Select all records from the "Teacher" table, and find the Teachers whose salary is greater than 20,000.

Solution:-
Note:- first create a database with column 'salary'

```
// teacherQuery.js

const mysql = require('mysql');

// Step 1: Create connection
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',         // â†□ tuza MySQL cha password tak ani mysql
workbench nko
  database: 'school'    // â†□ ani tuzya database name
});

// Step 2: Connect to database
connection.connect(err => {
  if (err) throw err;
  console.log("Connected to database!");

  // Step 3: Query all teachers with salary > 20000
  const query = "SELECT * FROM Teacher WHERE salary > 20000";

  connection.query(query, (err, results) => {
    if (err) throw err;

    console.log("Teachers with salary > 20,000:");
    console.table(results);

    // Step 4: End connection
    connection.end();
  });
});
```

SLIP5

Q.1) Create a Node.js file that writes an HTML form, with an upload field.
-proper run nhi jhl

```
// uploadForm.js

const http = require('http');

http.createServer((req, res) => {
  // Serve the form on GET request
  if (req.method === 'GET') {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(`
      <h2>Upload File</h2>
      <form action="/upload" method="post" enctype="multipart/form-
data">
        Select file: <input type="file" name="myfile"><br><br>
        <input type="submit" value="Upload">
```

```
      </form>
    `);
  } else if (req.method === 'POST' && req.url === '/upload') {
    // For now, just respond without actual file handling
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end('<h3>File upload received (not saved in this
example).</h3>');
  }
}).listen(3000, () => {
  console.log('Server running at http://localhost:3000');
});
```

Q.2) Using angular js create a SPA to carry out validation for a username entered in a textbox. If the textbox is blank, alert â€œEnter usernameâ€. If the number of characters is less than three, alert â€Ÿ Username is too shortâ€. If value entered is appropriate the print â€œValid usernameâ€ and password should be minimum 8 characters.

Sol:-

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
  <title>Username Validation SPA</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min
.js"></script>
  <style>
    .message { font-weight: bold; color: green; margin-top: 10px; }
    .error { color: red; }
  </style>
</head>
<body ng-controller="myCtrl">

  <h2>Login Form</h2>

  Username:
  <input type="text" ng-model="username"><br><br>

  Password:
  <input type="password" ng-model="password"><br><br>

  <button ng-click="validate()">Check</button>

  <p class="message" ng-show="validUsername">Valid username</p>
  <p class="error" ng-show="passwordError">Password must be at least 8
characters</p>
```

```
  <script>
    angular.module('myApp', [])
      .controller('myCtrl', function($scope) {
        $scope.validate = function() {
          $scope.validUsername = false;
          $scope.passwordError = false;

          if (!$scope.username) {
            alert("Enter username");
            return;
          }

          if ($scope.username.length < 3) {
            alert("Username is too short");
            return;
          }

          $scope.validUsername = true;

          if (!$scope.password || $scope.password.length < 8) {
            $scope.passwordError = true;
          }
        };
      });
  </script>

</body>
</html>
```

SLIP-6

Q.1) Write angular JS by using ng-click directive to display an alert message after clicking the element.

```
<html lang="en" ng-app="clickApp">
<head>
<title>AngularJS Click Example</title>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="clickController">
<button ng-click="showAlert()">Click Me</button>
<script>
angular.module('clickApp', [])
.controller('clickController', ($scope) =>
$scope.showAlert = () => alert('Button Clicked!'));
</script>
```

```
</body>
</html>
```

Q.2) 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.

(error not run)

```
// fileServer.js

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 = "./text.txt" + 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(8081);
```

SLIP-7

Q.1) Create angular JS Application that show the current Date and Time of the System (Use Interval Service)

```
<!DOCTYPE html>
<html ng-app="dateTimeApp">
<head>
  <title>AngularJS Date & Time</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min
.js"></script>
</head>
<body ng-controller="dateCtrl">

  <h2>Current Date and Time</h2>
  <p>{{ currentTime }}</p>

  <script>
    // AngularJS Module
```

```
    angular.module('dateTimeApp', [])
      .controller('dateCtrl', function($scope, $interval) {

        // Function to update the time
        function updateTime() {
          $scope.currentTime = new Date().toLocaleString();
        }

        // Call updateTime every 1000ms (1 second)
        $interval(updateTime, 1000);

        // Initial call
        updateTime();
      });
  </script>

</body>
</html>
```

Q.2) 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.

Sol:-

```
// Import the events module
const EventEmitter = require('events');

// Create an instance of EventEmitter
const eventEmitter = new EventEmitter();

// Callback function for 'start' event
function onStart() {
    console.log('Start event triggered!');
}

// Callback function for 'stop' event
function onStop() {
    console.log('Stop event triggered!');
}

// Register event listeners
eventEmitter.on('start', onStart);
eventEmitter.on('stop', onStop);

// Simulating a main loop with setInterval
console.log('Event-driven main loop started.');

let counter = 0;
const interval = setInterval(() => {
```

```
    // Emit different events based on the counter
    if (counter === 2) {
        eventEmitter.emit('start');
    } else if (counter === 5) {
        eventEmitter.emit('stop');
        clearInterval(interval); // End the loop
        console.log('Event-driven main loop stopped.');
    }
    counter++;
}, 1000);
```

SLIP 8

Q.1) Create a Simple Web Server using node js (not run)

Sol:-

```
var http = require('http'); // 1 - Import Node.js core
module
var server = http.createServer(function (req, res) { // -creating
server
//handle incomming requests here..
});
server.listen(8081); //3 - listen for any incoming requests
console.log('Node.js web server at port 8081 is running..')
```

Q.2) Using angular js display the 10 student details in Table format (using ng-repeat directive use Array to store data)

Sol:-

```
<html lang="en" ng-app="studentApp">
<head>
<meta charset="UTF-8">
<title>Student Details</title>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/
angular.min.js"></script>
<style>
body {
margin: 2%;
font-size: 100%;
}
th,
td {
```

```
padding: 10px;
}
</style>
</head>
<body>
<div ng-controller="StudentController as ctrl">
<table>
<table border=1>
<tr>
<th>ID</th>
<th>Name</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr ng-repeat="student in ctrl.students">
<td>{{ student.id }}</td>
<td>{{ student.name }}</td>
<td>{{ student.class }}</td>
</tr>
</tbody>
</table>
</div>
<script>
angular.module('studentApp'
, [])
.controller('StudentController'
, function() {
var ctrl = this;
ctrl.students = [
{ id: 1, name: 'payal'
, class: 'FY' },
{ id: 2, name: 'ROHIT'
, class: 'SY' },
{ id: 3, name: 'RONIT'
, class: 'TY' },
{ id: 4, name: 'RAHUL'
, class: 'MCS' },
{ id: 5, name: 'RONIT'
, class: 'FY' },
{ id: 6, name: 'KARAN'
, class: 'SY' },
{ id: 7, name: 'VIRAT'
, class: 'SY' },
{ id: 8, name: 'RAHUL'
, class: 'FY' },
{ id: 9, name: 'SUMIT'
, class: 'MCS' },
```

```
{ id: 10, name: 'SUMEDH'
, class: 'TY' }
];
});
</script>
</body>
</html>
```

SLIP-9

Q.1) Create a Node.js file that writes an HTML form, with a concatenate two string. (not run)

SOL:-

```
const http = require('http');
http.createServer((req, res) => {
res.writeHead(200, { 'Content-Type': 'text/html' });
const str1 = 'Hello';
const str2 = 'World';
const result = str1 + ' ' + str2;
res.end(`
<form>
<label for="concatenatedString">${result}</label><br><br>
<input type="text" name="concatenatedString" value="${result}"
disabled>
</form>
`);
}).listen(3000, () => {
console.log('Server running on http://localhost:3000/');
});
```

Q.2) 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(not run)

SOL:-

```
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 = "./text.txt" + 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(8081);
```

SLIP-10Q.1) Create a Node.js file that demonstrate create college database and table in MySQL (not run)

SLIP:-

```
var mysql=require('mysql');
var con=mysql.createConnection({
host:"localhost",user:"root",password:"srv001"
});
con.connect(function(err){
if(err) throw err;
console.log("Connected!");
con.query("create database college",function(err,result){
if(err) throw err;
console.log("Database Created!");
});
});
```

```
var mysql=require('mysql');
var con=mysql.createConnection({
host:"localhost",user:"root",password:"srv001",database:"college"
});
con.connect(function(err){
if(err) throw err;
console.log("Connected!");
var sql="create table node(id int primary key,name varchar(20))";
con.query(sql,function(err,result){
if(err) throw err;
console.log("Table Created!");
});
});
```

Q.2) Write node js script to build Your Own Node.js Module. Use require ('http') module is a built-in 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. (not run)

modules.js

```
exports.mydatetime = function () {
return Date();
 }
```

node.js

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

```
var dt = require('./modules');
http.createServer(function (req, res) {
res.writeHead(200, {'Content-Type': 'text/html'});
res.write("The date and time is currently: " +
dt.mydatetime());
res.end();
}).listen(8082);
```

SLIP 11

Q.1) Create a Node.js file that demonstrates create Movie database and table in MySQL.

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

// Create a connection to the MySQL server
const connection = mysql.createConnection({
    host: 'localhost', // Your MySQL host
    user: 'root',      // Your MySQL username
    password: 'your_password' // Your MySQL password
});

// Connect to the MySQL server
connection.connect((err) => {
    if (err) {
        console.error('Error connecting to MySQL:', err);
        return;
    }
    console.log('Connected to MySQL server.');

    // Create a new database
    const createDatabaseQuery = 'CREATE DATABASE IF NOT EXISTS
MovieDB';
    connection.query(createDatabaseQuery, (err, result) => {
        if (err) {
            console.error('Error creating database:', err);
            return;
        }
        console.log('Database "MovieDB" created or already exists.');

        // Use the newly created database
        connection.changeUser ({ database: 'MovieDB' }, (err) => {
            if (err) {
                console.error('Error changing database:', err);
                return;
            }

            // Create a new table he mysql mdhe type krach ahe or
kuthl pan mysql compiler
```

```
            const createTableQuery = `
                CREATE TABLE IF NOT EXISTS movies (
                    id INT AUTO_INCREMENT PRIMARY KEY,
                    title VARCHAR(255) NOT NULL,
                    director VARCHAR(255) NOT NULL,
                    release_year INT NOT NULL,
                    genre VARCHAR(100) NOT NULL
                )
            `;
            connection.query(createTableQuery, (err, result) => {
                if (err) {
                    console.error('Error creating table:', err);
                    return;
                }
                console.log('Table "movies" created or already
exists.');

                // Close the connection
                connection.end((err) => {
                    if (err) {
                        console.error('Error closing the connection:',
err);
                    } else {
                        console.log('Connection closed.');
                    }
                });
            });
        });
    });
});
```

Q.2) 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

```
// app.js

const express = require('express');
const path = require('path');

const app = express();
const PORT = 3000;

// Route to download the file
app.get('/download', (req, res) => {
    const filePath = path.join(__dirname, 'sample.txt'); // Path to
the file
    res.download(filePath, 'downloaded_sample.txt', (err) => {
        if (err) {
```

```
            console.error('Error downloading the file:', err);
            res.status(500).send('Could not download the file.');
        }
    });
});

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

SLIP12

Q.1) Create a node.js file that Select all records from the "customers" table, and display the result object on console.

```
// app.js

const mysql = require('mysql2');

// Create a connection to the database
const connection = mysql.createConnection({
    host: 'localhost', // Replace with your database host jo tuza asel toh
    user: 'your_username', // Replace with your database username (root takun bhg)
    password: 'your_password', // Replace with your database password(mysql cha password takaych)
    database: 'your_database_name' // Replace with your database name(mysql mdhe je database create kela ahe toch)
});

// Connect to the database
connection.connect((err) => {
    if (err) {
        console.error('Error connecting to the database:', err.stack);
        return;
    }
    console.log('Connected to the database.');
});

// Query to select all records from the customers table
connection.query('SELECT * FROM customers', (error, results) => {
    if (error) {
        console.error('Error fetching data:', error);
        return;
    }
    // Display the result object on console
    console.log('Customers:', results);
```

```
});

// Close the connection
connection.end();
```

Q.2) Create an HTML form for Student Feedback Form with Name, Email ID, Mobile No., feedback (Not good, good, very good, excellent) and write a JavaScript to validate all field using Regular Expression.

:-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Student Feedback Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .error {
            color: red;
            font-size: 0.9em;
        }
        .success {
            color: green;
            font-size: 1em;
        }
    </style>
</head>
<body>
    <h1>Student Feedback Form</h1>
    <form id="feedbackForm">
        <label for="name">Name:</label><br>
        <input type="text" id="name" name="name" required><br>
        <span class="error" id="nameError"></span><br>

        <label for="email">Email ID:</label><br>
        <input type="email" id="email" name="email" required><br>
        <span class="error" id="emailError"></span><br>

        <label for="mobile">Mobile No.:</label><br>
        <input type="text" id="mobile" name="mobile" required><br>
        <span class="error" id="mobileError"></span><br>
```

```
        <label for="feedback">Feedback:</label><br>
        <select id="feedback" name="feedback" required>
            <option value="">Select</option>
            <option value="not_good">Not Good</option>
            <option value="good">Good</option>
            <option value="very_good">Very Good</option>
            <option value="excellent">Excellent</option>
        </select><br>
        <span class="error" id="feedbackError"></span><br>

        <button type="submit">Submit</button>
        <div class="success" id="successMessage"></div>
    </form>

    <script>

document.getElementById('feedbackForm').addEventListener('submit',
function(event) {
            event.preventDefault(); // Prevent form submission

            // Clear previous error messages
            document.getElementById('nameError').textContent = '';
            document.getElementById('emailError').textContent = '';
            document.getElementById('mobileError').textContent = '';
            document.getElementById('feedbackError').textContent = '';
            document.getElementById('successMessage').textContent =
'';

            // Get form values
            const name = document.getElementById('name').value.trim();
            const email =
document.getElementById('email').value.trim();
            const mobile =
document.getElementById('mobile').value.trim();
            const feedback =
document.getElementById('feedback').value;

            let isValid = true;

            // Validate Name (only letters and spaces)
            const nameRegex = /^[A-Za-z\s]+$/;
            if (!nameRegex.test(name)) {
                document.getElementById('nameError').textContent =
'Please enter a valid name (letters and spaces only).';
                isValid = false;
            }

            // Validate Email
            const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
            if (!emailRegex.test(email)) {
```

```
                document.getElementById('emailError').textContent =
'Please enter a valid email address.';
                isValid = false;
            }

            // Validate Mobile No. (10 digits)
            const mobileRegex = /^\d{10}$/;
            if (!mobileRegex.test(mobile)) {
                document.getElementById('mobileError').textContent =
'Please enter a valid mobile number (10 digits).';
                isValid = false;
            }

            // Validate Feedback selection
            if (feedback === "") {
                document.getElementById('feedbackError').textContent =
'Please select your feedback.';
                isValid = false;
            }

            // If all fields are valid, display success message
            if (isValid) {
                document.getElementById('successMessage').textContent
= 'Feedback submitted successfully!';
                // Here you can submit the form or send the data to
the server
                // For example: this.submit(); // Uncomment to submit
the form
            }
        });
    </script>
</body>
</html>
```

SLIP:-13

Q.1) Create a Node.js file that will convert the output "HELLO WORLD!" into lower-case letters.

:- yavar run kr (https://onecompiler.com/nodejs/43ekcddjc)

```
// convertToLower.js

// Define the string
const originalString = "HELLO WORLD!";

// Convert the string to lower-case
const lowerCaseString = originalString.toLowerCase();
```

```
// Output the result
console.log(lowerCaseString);
```

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.

SOL:-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Student Registration Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .error {
            color: red;
            font-size: 0.9em;
        }
        .success {
            color: green;
            font-size: 1em;
        }
    </style>
</head>
<body>
    <h1>Student Registration Form</h1>
    <form id="registrationForm">
        <label for="firstName">First Name:</label><br>
        <input type="text" id="firstName" name="firstName"
required><br>
        <span class="error" id="firstNameError"></span><br>

        <label for="lastName">Last Name:</label><br>
        <input type="text" id="lastName" name="lastName" required><br>
        <span class="error" id="lastNameError"></span><br>

        <label for="age">Age:</label><br>
        <input type="number" id="age" name="age" required><br>
        <span class="error" id="ageError"></span><br>
```

```
        <button type="submit">Register</button>
        <div class="success" id="successMessage"></div>
    </form>

    <script>

document.getElementById('registrationForm').addEventListener('submit',
function(event) {
            event.preventDefault(); // Prevent form submission

            // Clear previous error messages
            document.getElementById('firstNameError').textContent =
'';
            document.getElementById('lastNameError').textContent = '';
            document.getElementById('ageError').textContent = '';
            document.getElementById('successMessage').textContent =
'';

            // Get form values
            const firstName =
document.getElementById('firstName').value.trim();
            const lastName =
document.getElementById('lastName').value.trim();
            const age =
parseInt(document.getElementById('age').value.trim(), 10);

            let isValid = true;

            // Validate First Name (only letters)
            const nameRegex = /^[A-Za-z]+$/;
            if (!nameRegex.test(firstName)) {
                document.getElementById('firstNameError').textContent
= 'First name should contain only alphabets.';
                isValid = false;
            }

            // Validate Last Name (only letters)
            if (!nameRegex.test(lastName)) {
                document.getElementById('lastNameError').textContent =
'Last name should contain only alphabets.';
                isValid = false;
            }

            // Validate Age (between 18 and 50)
            if (isNaN(age) || age < 18 || age > 50) {
                document.getElementById('ageError').textContent = 'Age
must be a number between 18 and 50.';
                isValid = false;
            }

            // If all fields are valid, display success message
```

```
            if (isValid) {
                document.getElementById('successMessage').textContent
= 'Registration successful!';
                // Here you can submit the form or send the data to
the server
                // For example: this.submit(); // Uncomment to submit
the form
            }
        });
    </script>
</body>
</html>
```

SLIP-14

Q.1) Create a Simple Web Server using node js.

SOL:-

```
// server.js

const http = require('http');

// Create a server
const server = http.createServer((req, res) => {
    // Set the response HTTP header with HTTP status and Content type
    res.writeHead(200, { 'Content-Type': 'text/plain' });

    // Send the response body "Hello, World!"
    res.end('Hello, World!\n');
});

// Server listens on port 3000
const PORT = 3000;
server.listen(PORT, () => {
    console.log(`Server running at http://localhost:${PORT}/`);
});
```

Q.2) Create an HTML form that contain the Employee Registration details and write a JavaScript to validate DOB, Joining Date, and Salary

SOL:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Employee Registration Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    .error {
      color: red;
      font-size: 0.9em;
    }
    .success {
      color: green;
      font-size: 1em;
    }
  </style>
</head>
<body>
  <h1>Employee Registration Form</h1>
  <form id="registrationForm">
    <label for="dob">Date of Birth:</label><br>
    <input type="date" id="dob" name="dob" required><br>
    <span class="error" id="dobError"></span><br>

    <label for="joiningDate">Joining Date:</label><br>
    <input type="date" id="joiningDate" name="joiningDate" required><br>
    <span class="error" id="joiningDateError"></span><br>

    <label for="salary">Salary:</label><br>
    <input type="number" id="salary" name="salary" required><br>
    <span class="error" id="salaryError"></span><br>

    <button type="submit">Register</button>
    <div class="success" id="successMessage"></div>
  </form>

  <script>
    document.getElementById('registrationForm').addEventListener('submit', function(event) {
      event.preventDefault(); // Prevent form submission

      // Clear previous error messages
      document.getElementById('dobError').textContent = '';
      document.getElementById('joiningDateError').textContent = '';
      document.getElementById('salaryError').textContent = '';
      document.getElementById('successMessage').textContent = '';

      // Get form values
      const dob = new Date(document.getElementById('dob').value);
      const joiningDate = new Date(document.getElementById('joiningDate').value);
```

```
        const salary = parseFloat(document.getElementById('salary').value);

        let isValid = true;

        // Validate Date of Birth (DOB)
        if (dob >= new Date()) {
            document.getElementById('dobError').textContent = 'DOB must be in the past.';
            isValid = false;
        }

        // Validate Joining Date (not in the future)
        if (joiningDate > new Date()) {
            document.getElementById('joiningDateError').textContent = 'Joining Date cannot be in the
future.';
            isValid = false;
        }

        // Validate Salary (must be a positive number)
        if (isNaN(salary) || salary <= 0) {
            document.getElementById('salaryError').textContent = 'Salary must be a positive number.';
            isValid = false;
        }

        // If all fields are valid, display success message
        if (isValid) {
            document.getElementById('successMessage').textContent = 'Registration successful!';
        }
    });
  </script>
</body>
</html>
```

SLIP:-15

Q.1) Create a node.js file that Select all records from the "students" table, and display the result object on console

```
// fetch_students.js

const mysql = require('mysql');

// Create a connection to the database
const connection = mysql.createConnection({
  host: 'localhost',      // Your database host
  user: 'your_username',  // Your database username
  password: 'your_password', // Your database password
  database: 'your_database' // Your database name
});
```

```
// Connect to the database
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to the database:', err.stack);
    return;
  }
  console.log('Connected to the database.');
});

// Query to select all records from the "students" table
const query = 'SELECT * FROM students';

connection.query(query, (error, results) => {
  if (error) {
    console.error('Error fetching records:', error);
    return;
  }

  // Display the result object on the console
  console.log('Records from students table:', results);
});

// Close the database connection
connection.end();
```

Q.2) Create an HTML form for Employee and write a JavaScript to validate name, email ID, mobile number, department, joining date using Regular Expression.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Employee Registration Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    .error {
      color: red;
      font-size: 0.9em;
    }
    .success {
```

```
        color: green;
        font-size: 1em;
      }
    </style>
  </head>
  <body>
    <h1>Employee Registration Form</h1>
    <form id="registrationForm">
      <label for="name">Name:</label><br>
      <input type="text" id="name" required><br>
      <span class="error" id="nameError"></span><br>

      <label for="email">Email ID:</label><br>
      <input type="email" id="email" required><br>
      <span class="error" id="emailError"></span><br>

      <label for="mobile">Mobile Number:</label><br>
      <input type="text" id="mobile" required><br>
      <span class="error" id="mobileError"></span><br>

      <label for="department">Department:</label><br>
      <input type="text" id="department" required><br>
      <span class="error" id="departmentError"></span><br>

      <label for="joiningDate">Joining Date:</label><br>
      <input type="date" id="joiningDate" required><br>
      <span class="error" id="joiningDateError"></span><br>

      <button type="submit">Register</button>
      <div class="success" id="successMessage"></div>
    </form>

    <script>
      document.getElementById('registrationForm').addEventListener('submit', function(event) {
        event.preventDefault(); // Prevent form submission

        // Clear previous error messages
        document.querySelectorAll('.error').forEach(el => el.textContent = '');
        document.getElementById('successMessage').textContent = '';

        // Get form values
        const name = document.getElementById('name').value.trim();
        const email = document.getElementById('email').value.trim();
        const mobile = document.getElementById('mobile').value.trim();
        const department = document.getElementById('department').value.trim();
        const joiningDate = document.getElementById('joiningDate').value;

        let isValid = true;
```

```
        // Validate Name (only letters and spaces)
        if (!/^[A-Za-z\s]+$/.test(name)) {
            document.getElementById('nameError').textContent = 'Invalid name. Only letters and spaces
allowed.';
            isValid = false;
        }

        // Validate Email
        if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {
            document.getElementById('emailError').textContent

        }
    });
```

SLIP 18

Q.1) Using node js create a User Login System

SOL:-

```
// app.js

const express = require('express');
const session = require('express-session');
const bodyParser = require('body-parser');
const app = express();
const PORT = 3000;

// Middleware
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({
  secret: 'your_secret_key',
  resave: false,
  saveUninitialized: true
}));

// Set view engine to EJS
app.set('view engine', 'ejs');

// In-memory user store (for demonstration purposes)
const users = [
  { username: 'user1', password: 'password1' },
  { username: 'user2', password: 'password2' }
];

// Routes
app.get('/', (req, res) => {
```

```
  res.render('login', { error: null });
});

app.post('/login', (req, res) => {
  const { username, password } = req.body;
  const user = users.find(u => u.username === username && u.password === password);

  if (user) {
    req.session.user = user;
    res.redirect('/dashboard');
  } else {
    res.render('login', { error: 'Invalid username or password' });
  }
});

app.get('/dashboard', (req, res) => {
  if (!req.session.user) {
    return res.redirect('/');
  }
  res.render('dashboard', { user: req.session.user });
});

app.get('/logout', (req, res) => {
  req.session.destroy(err => {
    if (err) {
      return res.redirect('/dashboard');
    }
    res.redirect('/');
  });
});

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

Q.2) Create a node.js file that Select all records from the "customers" table, and find the customers whose name starts from 'A'

SOL:-

```
// app.js

const mysql = require('mysql');

// Create a connection to the database
const connection = mysql.createConnection({
```

```
  host: 'localhost', // Your database host
  user: 'your_username', // Your database username
  password: 'your_password', // Your database password
  database: 'your_database_name' // Your database name
});

// Connect to the database
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to the database:', err.stack);
    return;
  }
  console.log('Connected to the database.');
});

// Query to select all customers whose name starts with 'A'
const query = "SELECT * FROM customers WHERE name LIKE 'A%'";

// Execute the query
connection.query(query, (error, results) => {
  if (error) {
    console.error('Error executing query:', error.stack);
    return;
  }

  // Display the results
  console.log('Customers whose names start with "A":');
  results.forEach(customer => {
    console.log(`ID: ${customer.id}, Name: ${customer.name}, Email: ${customer.email}`);
  });
});

// Close the database connection
connection.end((err) => {
  if (err) {
    console.error('Error closing the connection:', err.stack);
    return;
  }
  console.log('Database connection closed.');
});
```

SLIP:-19

Q.1) Create a Node.js file that will convert the output "Hello World!" into upper-case letters.

SOL:-

```
// app.js
```

```
// Define the output string
const output = "Hello World!";

// Convert the string to upper-case
const upperCaseOutput = output.toUpperCase();

// Print the upper-case string to the console
console.log(upperCaseOutput);
```

Q.2) Using angular js create a SPA to accept the details such as name, mobile number, pin code and email address and make validation. Name should contain character only, address should contain SPPU M.Sc. Computer Science Syllabus 2023-24, mobile number should contain only 10 digit, Pin code should contain only 6 digit, email id should contain only one @, . Symbol.

SOl:-

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>User Details Form</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    .error {
      color: red;
    }
    .success {
      color: green;
    }
  </style>
</head>
<body ng-controller="FormController">

  <h1>User Details Form</h1>
  <form name="userForm" ng-submit="submitForm()" novalidate>
    <div>
      <label for="name">Name:</label>
      <input type="text" name="name" ng-model="user.name" required ng-pattern="/^[a-zA-Z\s]+$/">
      <span class="error" ng-show="userForm.name.$error.required &&
userForm.name.$touched">Name is required.</span>
      <span class="error" ng-show="userForm.name.$error.pattern &&
userForm.name.$touched">Name must contain characters only.</span>
    </div>

    <div>
      <label for="mobile">Mobile Number:</label>
```

```
        <input type="text" name="mobile" ng-model="user.mobile" required ng-pattern="/^\d{10}$/">
        <span class="error" ng-show="userForm.mobile.$error.required &&
userForm.mobile.$touched">Mobile number is required.</span>
        <span class="error" ng-show="userForm.mobile.$error.pattern &&
userForm.mobile.$touched">Mobile number must be 10 digits.</span>
      </div>

      <div>
        <label for="pin">Pin Code:</label>
        <input type="text" name="pin" ng-model="user.pin" required ng-pattern="/^\d{6}$/">
        <span class="error" ng-show="userForm.pin.$error.required && userForm.pin.$touched">Pin
code is required.</span>
        <span class="error" ng-show="userForm.pin.$error.pattern && userForm.pin.$touched">Pin code
must be 6 digits.</span>
      </div>

      <div>
        <label for="email">Email:</label>
        <input type="email" name="email" ng-model="user.email" required ng-pattern="/^[a-zA-Z0-
9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/">
        <span class="error" ng-show="userForm.email.$error.required &&
userForm.email.$touched">Email is required.</span>
        <span class="error" ng-show="userForm.email.$error.pattern &&
userForm.email.$touched">Invalid email format.</span>
      </div>

      <button type="submit" ng-disabled="userForm.$invalid">Submit</button>
    </form>

    <div ng-show="successMessage" class="success">
      {{ successMessage }}
    </div>

    <script>
      // AngularJS Application
      const app = angular.module("myApp", []);

      app.controller("FormController", function($scope) {
        $scope.user = {};
        $scope.successMessage = '';

        $scope.submitForm = function() {
          if ($scope.userForm.$valid) {
            $scope.successMessage = "Form submitted successfully!";
            // You can also send the data to a server here
            console.log($scope.user);
          }
        };
```

```
    });
  </script>
</body>
</html>
```

SLIP-20

Q.1) Create a Node.js file that demonstrate create student database and table in MySQL

```
// app.js

const mysql = require('mysql');

// Create a connection to the MySQL server
const connection = mysql.createConnection({
  host: 'localhost', // Your MySQL server host
  user: 'your_username', // Your MySQL username
  password: 'your_password' // Your MySQL password
});

// Connect to the MySQL server
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to the database:', err.stack);
    return;
  }
  console.log('Connected to the MySQL server.');

  // Create a new database
  const createDatabaseQuery = 'CREATE DATABASE IF NOT EXISTS studentDB';
  connection.query(createDatabaseQuery, (err, result) => {
    if (err) {
      console.error('Error creating database:', err.stack);
      return;
    }
    console.log('Database created or already exists.');

    // Use the newly created database
    connection.query('USE studentDB', (err) => {
      if (err) {
        console.error('Error selecting database:', err.stack);
        return;
      }

      // Create a new table
      const createTableQuery = `
        CREATE TABLE IF NOT EXISTS students (
          id INT AUTO_INCREMENT PRIMARY KEY,
          name VARCHAR(100) NOT NULL,
```

```
            age INT NOT NULL,
            email VARCHAR(100) NOT NULL UNIQUE
        )
    `;
    connection.query(createTableQuery, (err, result) => {
        if (err) {
            console.error('Error creating table:', err.stack);
            return;
        }
        console.log('Table "students" created or already exists.');

        // Close the connection
        connection.end((err) => {
            if (err) {
                console.error('Error closing the connection:', err.stack);
                return;
            }
            console.log('Database connection closed.');
        });
    });
  });
});
```

Q.2) Using angular js create a SPA to carry out validation for a username entered in a textbox. If the textbox is blank, alert "Enter username". If the number of characters is less than three, alert " Username is too short". If value entered is appropriate the print "Valid username" and password should be minimum 8 characters

SOL:-

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Username and Password Validation</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    .error {
      color: red;
    }
    .success {
      color: green;
    }
  </style>
</head>
```

```
<body ng-controller="ValidationController">

  <h1>Username and Password Validation</h1>
  <form name="validationForm" ng-submit="validateUser ()" novalidate>
    <div>
      <label for="username">Username:</label>
      <input type="text" name="username" ng-model="user.username" required>
    </div>
    <div>
      <label for="password">Password:</label>
      <input type="password" name="password" ng-model="user.password" required>
    </div>
    <button type="submit">Validate</button>
  </form>

  <div ng-show="message" class="success">
    {{ message }}
  </div>

  <script>
    // AngularJS Application
    const app = angular.module("myApp", []);

    app.controller("ValidationController", function($scope) {
      $scope.user = {};
      $scope.message = '';

      $scope.validateUser  = function() {
        const username = $scope.user.username;
        const password = $scope.user.password;

        if (!username) {
          alert("Enter username");
        } else if (username.length < 3) {
          alert("Username is too short");
        } else if (!password || password.length < 8) {
          alert("Password must be at least 8 characters long");
        } else {
          $scope.message = "Valid username and password.";
        }
      };
    });
  </script>
</body>
</html>
```


SLIP 21

Q.1) Create a Node.js file that demonstrate create Movie database and table in MySQL

SOL:-

```
// app.js

const mysql = require('mysql');

// Create a connection to the MySQL server
const connection = mysql.createConnection({
  host: 'localhost', // Your MySQL server host
  user: 'your_username', // Your MySQL username
  password: 'your_password' // Your MySQL password
});

// Connect to the MySQL server
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to the database:', err.stack);
    return;
  }
  console.log('Connected to the MySQL server.');

  // Create a new database
  const createDatabaseQuery = 'CREATE DATABASE IF NOT EXISTS movieDB';
  connection.query(createDatabaseQuery, (err, result) => {
    if (err) {
      console.error('Error creating database:', err.stack);
      return;
    }
    console.log('Database created or already exists.');

    // Use the newly created database
    connection.query('USE movieDB', (err) => {
      if (err) {
        console.error('Error selecting database:', err.stack);
        return;
      }

      // Create a new table
      const createTableQuery = `
        CREATE TABLE IF NOT EXISTS movies (
          id INT AUTO_INCREMENT PRIMARY KEY,
          title VARCHAR(255) NOT NULL,
          director VARCHAR(100) NOT NULL,
          release_year INT NOT NULL,
          genre VARCHAR(100)
        )
      `;
```

```
        connection.query(createTableQuery, (err, result) => {
          if (err) {
            console.error('Error creating table:', err.stack);
            return;
          }
          console.log('Table "movies" created or already exists.');

          // Close the connection
          connection.end((err) => {
            if (err) {
              console.error('Error closing the connection:', err.stack);
              return;
            }
            console.log('Database connection closed.');
          });
        });
      });
    });
});
```

Q.2) 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.

SOL:-

```
// app.js

const express = require('express');
const path = require('path');

const app = express();
const PORT = 3000;

// Route to download the file
app.get('/download', (req, res) => {
  const filePath = path.join(__dirname, 'sample.txt'); // Path to the file
  res.download(filePath, 'sample.txt', (err) => {
    if (err) {
      console.error('Error downloading the file:', err);
      res.status(500).send('Could not download the file.');
    }
  });
});

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

SLIP 23

Q.1) Write node js script to interact with the file system, and serve a web page from a File

SOL:-

```
// server.js

const http = require('http');
const fs = require('fs');
const path = require('path');

// Create an HTTP server
const server = http.createServer((req, res) => {
  // Define the file path to the HTML file
  const filePath = path.join(__dirname, 'index.html');

  // Read the HTML file from the file system
  fs.readFile(filePath, 'utf8', (err, data) => {
    if (err) {
      // If there's an error reading the file, send a 500 response
      res.writeHead(500, { 'Content-Type': 'text/plain' });
      res.end('Error reading the file.');
      return;
    }

    // If the file is read successfully, send a 200 response with the HTML content
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(data);
  });
});

// Define the port to listen on
const PORT = 3000;

// Start the server
server.listen(PORT, () => {
  console.log(`Server is running at http://localhost:${PORT}`);
});
```

Q.2) Write node js script to build Your Own Node.js Module. Use require (“http”) module is a built-in 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.

SOL:-

```
// app.js

const http = require('http');

// Function to return today's date and time
const getCurrentDateTime = () => {
   const now = new Date();
   return now.toString(); // Returns the current date and time as a string
};

// Create an HTTP server
const server = http.createServer((req, res) => {
   // Set the response header
   res.writeHead(200, { 'Content-Type': 'text/plain' });

   // Get the current date and time using the function
   const currentDateTime = getCurrentDateTime();

   // Send the current date and time as the response
   res.end(`Current Date and Time: ${currentDateTime}`);
});

// Define the port to listen on
const PORT = 3000;

// Start the server
server.listen(PORT, () => {
   console.log(`Server is running at http://localhost:${PORT}`);
});
```

SLIP 25

Q.1) Create an angular JS Application that shows the location of the current web page

```
<!DOCTYPE html>
<html lang="en" ng-app="locationApp">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Current Location App</title>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 20px;
```

```
    }
    h1 {
      color: #333;
    }
    .url {
      font-weight: bold;
      color: #007BFF;
    }
  </style>
</head>
<body ng-controller="LocationController">

  <h1>Current Web Page Location</h1>
  <p>The current URL is: <span class="url">{{ currentUrl }}</span></p>

  <h2>Navigate to:</h2>
  <ul>
    <li><a href="#/home" ng-click="navigate('home')">Home</a></li>
    <li><a href="#/about" ng-click="navigate('about')">About</a></li>
    <li><a href="#/contact" ng-click="navigate('contact')">Contact</a></li>
  </ul>

  <div ng-view></div>

  <script>
    // AngularJS application
    var app = angular.module('locationApp', ['ngRoute']);

    // Configure routes
    app.config(['$routeProvider', function($routeProvider) {
      $routeProvider
        .when('/home', {
          template: '<h2>Welcome to the Home Page!</h2>'
        })
        .when('/about', {
          template: '<h2>About Us</h2><p>This is the about page.</p>'
        })
        .when('/contact', {
          template: '<h2>Contact Us</h2><p>This is the contact page.</p>'
        })
        .otherwise({
          redirectTo: '/home'
        });
    }]);

    // Controller to manage the current URL
    app.controller('LocationController', ['$scope', '$location', function($scope, $location) {
      $scope.currentUrl = $location.absUrl(); // Get the current URL
```

```
        // Function to navigate to different routes
        $scope.navigate = function(page) {
            $location.path('/' + page);
            $scope.currentUrl = $location.absUrl(); // Update the current URL
        };
    }]);
  </script>

</body>
</html>
```

Q.2) Create a 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.

SOL:-

```
// main.js

const EventEmitter = require('events');

// Create an instance of EventEmitter
const eventEmitter = new EventEmitter();

// Define a callback function for the 'eventOccurred' event
const eventCallback = (message) => {
  console.log(`Event occurred: ${message}`);
};

// Register the callback function for the 'eventOccurred' event
eventEmitter.on('eventOccurred', eventCallback);

// Main loop that listens for events
const mainLoop = () => {
  let count = 0;

  // Simulate an event occurring every 2 seconds
  const intervalId = setInterval(() => {
    count++;
    const message = `This is event number ${count}`;

    // Emit the event
    eventEmitter.emit('eventOccurred', message);

    // Stop the loop after 5 events
    if (count >= 5) {
      clearInterval(intervalId);
      console.log('No more events will be emitted.');
    }
```

```
    }, 2000);
};

// Start the main loop
mainLoop();
```

