Slip 1_1 Create an HTML form for Login and write a JavaScript to validate email ID and Password using Regular Expression.

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Form</title>
  <style>
    body {
      font-family: Arial;
      margin: 50px;
    }
    input {
      margin: 8px 0;
      padding: 8px;
      width: 250px;
    }
    .error {
      color: red;
    }
  </style>
</head>
<body>
  <h2>Login Form</h2>
  <form onsubmit="return validateForm()">
    <label>Email:</label><br>
    <input type="text" id="email"><br>
        <label>Password:</label><br>
    <input type="password" id="password"><br>
    <span id="error" class="error"></span><br>
    <input type="submit" value="Login">
  </form>
  <script>
    function validateForm() {
      let email =
document.getElementById("email").value;
      let password =
document.getElementById("password").value;
      let errorMsg = document.getElementById("error");
      let emailPattern = /^[a-zA-Z0-9._%+-]+@[a-z0-9.-
]+\.[a-z]{2,4}$/;
      let passwordPattern = /^(?=.*\d)(?=.*[a-z])(?=.*[A-
Z]).{6,}$/;
      if (!emailPattern.test(email)) {
        errorMsg.textContent = "Invalid email format.";
        return false;
     if (!passwordPattern.test(password)) {
        errorMsg.textContent = "Password must be at least
6 characters and include 1 digit, 1 uppercase, and 1
lowercase.";
        return false;
     }    errorMsg.textContent = "";
      alert("Login successful!");
      return true;
    }
  </script>
</body><html>
```

Slip 1_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.

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Registration Form</title>
  <script>
    function validateForm() {
      const firstName =
document.getElementById("firstName").value.trim();
      const lastName =
document.getElementById("lastName").value.trim();
      const age =
parseInt(document.getElementById("age").value.trim());

      const nameRegex = /^[A-Za-z]+$/;

      if (!nameRegex.test(firstName)) {
        alert("First name should contain alphabets only.");
        return false;
      }

      if (!nameRegex.test(lastName)) {
        alert("Last name should contain alphabets only.");
        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>
</head>
<body>
  <h2>Student Registration Form</h2>
  <form onsubmit="return validateForm()">
    <label>First Name:</label>
    <input type="text" id="firstName" required><br><br>

    <label>Last Name:</label>
    <input type="text" id="lastName" required><br><br>

    <label>Age:</label>
    <input type="number" id="age" required><br><br>

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

Slip 2_1 Create a Node.js file that will convert the output &quot;Full Stack!&quot; into reverse string.

```
//reverseString.js
const original = "Full Stack!";
const reversed = original.split("").reverse().join("");

console.log("Original:", original);
console.log("Reversed:", reversed);

//run
node reverseString.js
```

Slip 2_2 Using node js create a web page to read two file names from the user and append contents of the first file into the second file.

```
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
  <title>Append File Content</title>
</head>
<body>
  <h2>Append File 1 into File 2</h2>
  <form action="/append" method="POST">
    <label>File 1 Name:</label>
    <input type="text" name="file1" required><br><br>

    <label>File 2 Name:</label>
    <input type="text" name="file2" required><br><br>

    <button type="submit">Append</button>
  </form>
</body>
</html>

// server.js
const express = require('express');
const bodyParser = require('body-parser');
const fs = require('fs');
const app = express();
const PORT = 3000;
app.use(bodyParser.urlencoded({ extended: true }));
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/index.html');
});

app.post('/append', (req, res) => {
  const file1 = req.body.file1;
  const file2 = req.body.file2;
  fs.readFile(file1, 'utf8', (err, data1) => {
    if (err) return res.send("Error reading File 1: " + err);

    fs.appendFile(file2, data1, (err) => {
      if (err) return res.send("Error appending to File 2: " + err);

      res.send(`Content from "${file1}" appended to "${file2}" successfully.`);
    });
  });
});
app.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}`);
});
```

Slip 3_1 Using node js create a User Login System.

login.html

```
<!DOCTYPE html>
<html>
<head>
  <title>User Login</title>
</head>
<body>
  <h2>User Login</h2>
  <form action="/login" method="POST">
    <label>Email:</label>
    <input type="email" name="email"
required><br><br>
    <label>Password:</label>
    <input type="password" name="password"
required><br><br>
    <button type="submit">Login</button>
  </form>
</body>
</html>
```

server.js

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const PORT = 3000;
// Dummy user (like from a database)
const user = {
  email: "admin@example.com",
  password: "12345"
};
app.use(bodyParser.urlencoded({ extended: true }));
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/login.html');
});

app.post('/login', (req, res) => {
  const { email, password } = req.body;
  if (email === user.email && password ===
user.password) {
    res.send("Login successful!");
  } else {
    res.send("Invalid credentials.");
  }
});
app.listen(PORT, () => {
  console.log(`Server running at
http://localhost:${PORT}`);
});
```

Slip 3_2 Create a node.js file that Select all records from the &quot;Teacher&quot; table, and find the Teachers whose salary is greater than 20,000.

selectTeachers.js

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

// Create connection
const conn = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",    // add your password if needed
  database: "school" // change DB name as needed
});

// Connect to DB
conn.connect(err => {
  if (err) throw err;
  console.log("Connected to MySQL!");

  const query = "SELECT * FROM Teacher WHERE salary
> 20000";

  conn.query(query, (err, result) => {
    if (err) throw err;
    console.log("Teachers with salary > 20000:");
    console.table(result);
  });
});

//sql
CREATE DATABASE school;
USE school;

CREATE TABLE Teacher (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  salary INT
);

INSERT INTO Teacher (name, salary) VALUES
('John Doe', 18000),
('Jane Smith', 25000),
('Ravi Kumar', 22000);

//run
npm install mysql
node selectTeachers.js
```

Slip 4_1

Using node js create an eLearning System.

// server.js

```
const express = require('express');
const fs = require('fs');
const bodyParser = require('body-parser');
const app = express();
const PORT = 3000;
app.use(bodyParser.json());
app.use(express.static(__dirname));
// Register student
app.post('/register', (req, res) => {
  const student = req.body;
  let data = fs.existsSync('students.json') ?
JSON.parse(fs.readFileSync('students.json')) : [];
  data.push(student);
  fs.writeFileSync('students.json', JSON.stringify(data,
null, 2));
  res.send("Student registered successfully!");
});
// Get list of courses
app.get('/courses', (req, res) => {
  const courses = fs.existsSync('courses.json') ?
JSON.parse(fs.readFileSync('courses.json')) : [];
  res.json(courses);
});
app.listen(PORT, () => {
  console.log(`eLearning System running at
http://localhost:${PORT}`);
});
//course.json
[
  { "id": 1, "title": "HTML & CSS" },
  { "id": 2, "title": "JavaScript" },
  { "id": 3, "title": "Node.js" }
]
```

//Run it:

```
Copy code
npm init -y
npm install express body-parser
node server.js
```

Slip 4_2 Create an HTML form using AngularJS that contain the Student Registration details and validate Student first and last name as it should not contain other than alphabets and age should be between 18 to 50 and display greeting message depending on current time using ng-show (e.g. Good Morning, Good Afternoon, etc.)(Use AJAX).

```
<!DOCTYPE html>
<html ng-app="studentApp">
<head>
  <title>Student Registration</title>
  <script src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.8.3/angular.min.js"></script>
</head>
<body ng-controller="studentCtrl">
  <h2>Student Registration</h2>
  <form name="regForm" ng-submit="registerStudent()"
novalidate>
    <label>First Name:</label>
    <input type="text" name="fname" ng-
model="student.fname" ng-pattern="/^[A-Za-z]+$/"
required>
    <span ng-show="regForm.fname.$error.pattern &&
regForm.fname.$touched">Only alphabets
allowed</span><br><br>
    <label>Last Name:</label>
    <input type="text" name="lname" ng-
model="student.lname" ng-pattern="/^[A-Za-z]+$/"
required>
    <span ng-show="regForm.lname.$error.pattern &&
regForm.lname.$touched">Only alphabets
allowed</span><br><br>
    <label>Age:</label>
    <input type="number" name="age" ng-
model="student.age" min="18" max="50" required>
    <span ng-show="(regForm.age.$error.min ||
 regForm.age.$error.max) && regForm.age.$touched">
      Age must be between 18 and 50
    </span><br><br>
    <button type="submit" ng-
disabled="regForm.$invalid">Register</button>
  </form>
  <h3 ng-show="greeting">{{greeting}}, {{student.fname
|| 'Student'}}!</h3>
  <script>
    const app = angular.module("studentApp", []);
    app.controller("studentCtrl", function ($scope, $http)
{
      $scope.student = {};
          const hour = new Date().getHours();
      if (hour < 12) $scope.greeting = "Good Morning";
      else if (hour < 18) $scope.greeting = "Good
Afternoon";
      else $scope.greeting = "Good Evening";
      $scope.registerStudent = function () {
        $http.post('/register', $scope.student)
          .then(function (response) {
            alert(response.data);
          }, function (error) {
            alert("Error registering student");
          });
      };
    });
  </script>
/body>
</html>
```

Slip 5_1

Create a Node.js file that writes an HTML form, with an upload field.

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

const formHTML = `
<!DOCTYPE html>
<html>
<head><title>File Upload</title></head>
<body>
  <h2>Upload a File</h2>
  <form action="/upload" method="POST"
enctype="multipart/form-data">
    <input type="file" name="myFile" required />
    <br><br>
    <input type="submit" value="Upload" />
  </form>
</body>
</html>
`;

const server = http.createServer((req, res) => {
  if (req.url === '/' && req.method === 'GET') {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(formHTML);
  } else if (req.url === '/upload' && req.method ===
'POST') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('File uploaded successfully (but not saved,
demo only)');
  }
});

server.listen(3000, () => {
  console.log('Server running at http://localhost:3000');
});
```

Slip 5_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 the value entered is appropriate the print “Valid username” and password should be a minimum of 8 characters.

```
<!DOCTYPE html>
<html ng-app="userApp">
<head>
  <title>User Validation</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs
/1.8.3/angular.min.js"></script>
</head>
<body ng-controller="userCtrl">
  <h2>User Validation Form</h2>

  <form name="userForm" novalidate ng-
submit="validateUser()">

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

    <label>Password:</label>
    <input type="password" ng-model="password"
required>
    <br><br>
    <button type="submit">Submit</button>

    <h3 style="color:green" ng-
show="validMessage">{{validMessage}}</h3>
  </form>
  <script>
    const app = angular.module('userApp', []);

    app.controller('userCtrl', function ($scope) {
      $scope.validateUser = function () {
        const uname = $scope.username;
        const pwd = $scope.password;
        if (!uname) {
          alert("Enter username");
        } else if (uname.length < 3) {
          alert("Username is too short");
        } else if (!pwd || pwd.length < 8) {
          alert("Password must be at least 8 characters");
        } else {
          $scope.validMessage = "Valid username";
        }
      };
    });
  </script>

</body>
</html>
```

Slip 6_1

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

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
  <title>ng-click Example</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs
/1.8.3/angular.min.js"></script>
</head>
<body ng-controller="myCtrl">

  <h2>Click the button to see an alert</h2>

  <button ng-click="showAlert()">Click Me!</button>

  <script>
    const app = angular.module('myApp', []);

    app.controller('myCtrl', function($scope) {
      $scope.showAlert = function() {
        alert("Hello! You clicked the button.");
      };
    });
  </script>

</body>
</html>
```

Slip 6_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.

fileServer.js

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

const server = http.createServer((req, res) => {
  const q = url.parse(req.url, true);
  const filename = "." + q.pathname;

  fs.readFile(filename, (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();
  });
});

server.listen(3000, () => {
  console.log("Server running at
http://localhost:3000/");
});
```

Slip 7_1
Create angular JS Application that show the current Date and Time of the System (Use Interval Service)

```
<!DOCTYPE html>
<html ng-app="timeApp">
<head>
  <title>Current Date and Time</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.3/angular.min.js"></script>
</head>
<body ng-controller="timeCtrl">

  <h2>Current Date and Time:</h2>
  <p>{{ currentDateTime }}</p>

  <script>
    const app = angular.module('timeApp', []);

    app.controller('timeCtrl', function($scope, $interval) {
      // Function to update the current date and time
      function updateDateTime() {
        const now = new Date();
        $scope.currentDateTime = now.toLocaleString(); // Formats date and time
      }

      // Call updateDateTime immediately to set initial date and time
      updateDateTime();

      // Update the date and time every second (1000 milliseconds)
      $interval(updateDateTime, 1000);
    });
  </script>

</body>
</html>
```

Slip 7_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.

//main.js

```
const EventEmitter = require('events');

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

// Event listener: 'start' event
eventEmitter.on('start', () => {
  console.log("The event has started!");
});

// Event listener: 'end' event
eventEmitter.on('end', () => {
  console.log("The event has ended!");
});

// Main loop to listen for events
function mainLoop() {
  console.log("Listening for events...");

  // Simulate event occurrences
  setTimeout(() => {
    eventEmitter.emit('start');  // Emit 'start' event
  }, 2000); // Trigger after 2 seconds

  setTimeout(() => {
    eventEmitter.emit('end');    // Emit 'end' event
  }, 5000); // Trigger after 5 seconds
}

// Call the main loop to start the application
mainLoop();
```

Slip 8_1 Create a Simple Web Server using node js

```
// Import the built-in http module
const http = require('http');

// Define the hostname and port
const hostname = '127.0.0.1';
const port = 3000;

// Create the server
const server = http.createServer((req, res) => {
  // Set the response header
  res.statusCode = 200; // HTTP status code for successful response
  res.setHeader('Content-Type', 'text/plain'); // Type of content being sent

  // Write the response body
  res.end('Hello, World!'); // Send the message "Hello, World!" to the client
});

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

Slip 8_2 Using angular js display the 10 student details in Table format (using ng-repeat directive use Array to store data)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Details</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-app="studentApp">
<div ng-controller="studentCtrl">
    <table border="1">
        <thead>
            <tr>
                <th>Student ID</th>
                <th>Name</th>
                <th>Age</th>
                <th>Grade</th>
            </tr>
        </thead>
        <tbody>
            <tr ng-repeat="student in students">
                <td>{{ student.id }}</td>
                <td>{{ student.name }}</td>
                <td>{{ student.age }}</td>
                <td>{{ student.grade }}</td>
            </tr>
        </tbody>
    </table>
</div>
<script>
    var app = angular.module('studentApp', []);
    app.controller('studentCtrl', function($scope) {
        $scope.students = [
            {id: 1, name: 'John Doe', age: 18, grade: 'A'},
            {id: 2, name: 'Jane Smith', age: 19, grade: 'B'},
            {id: 3, name: 'Michael Brown', age: 18, grade: 'A'},
            {id: 4, name: 'Emily Davis', age: 20, grade: 'C'},
            {id: 5, name: 'Chris Wilson', age: 21, grade: 'B'},
            {id: 6, name: 'Sarah Moore', age: 22, grade: 'A'},
            {id: 7, name: 'David Taylor', age: 19, grade: 'B'},
            {id: 8, name: 'Laura Anderson', age: 20, grade: 'A'},
            {id: 9, name: 'James Thomas', age: 18, grade: 'C'},
            {id: 10, name: 'Sophie Martinez', age: 19, grade: 'A'}
        ];
    });
</script>
</body>
</html>
```

Slip 9_1
Create a Node.js file that writes an HTML form, with a concatenate two string.
//npm install express

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');

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

// Middleware to parse form data
app.use(bodyParser.urlencoded({ extended: true }));
// Serve the HTML form
app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'form.html'));
});
// Handle form submission
app.post('/concatenate', (req, res) => {
  const string1 = req.body.string1;
  const string2 = req.body.string2;
  const concatenatedString = string1 + ' ' + string2;

  // Display the concatenated result
  res.send(`<h1>Result: ${concatenatedString}</h1><a href="/">Go back</a>`);
});
// Start the server
app.listen(port, () => {
  console.log(`Server is running at http://localhost:${port}`);
});
//Form.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Concatenate Strings</title>
</head>
<body>
  <h1>Concatenate Two Strings</h1>
  <form action="/concatenate" method="POST">
    <label for="string1">String 1:</label>
    <input type="text" id="string1" name="string1" required>
    <br><br>
    <label for="string2">String 2:</label>
    <input type="text" id="string2" name="string2" required>
    <br><br>
    <button type="submit">Concatenate</button>
  </form>
</body>
</html>
```

Slip 9_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

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

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

// Middleware to serve static files (optional)
app.use(express.static('public'));

// Route to handle file requests
app.get('/file/:filename', (req, res) => {
  const filename = req.params.filename;
  const filePath = path.join(__dirname, 'files', filename);
// Assuming files are stored in a 'files' folder

  fs.readFile(filePath, 'utf8', (err, data) => {
    if (err) {
      // If the file doesn't exist or other errors, return 404
      res.status(404).send('File not found');
    } else {
      // If file exists, send the content
      res.send(data);
    }
  });
});

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

Slip 10_1

Create a Node.js file that demonstrate create college database and table in MySQL

```
//npm install mysql2
const mysql = require('mysql2');
// Create a connection to MySQL server
const connection = mysql.createConnection({
  host: 'localhost',    // MySQL server address
  user: 'root',         // MySQL username (default: 'root')
  password: '',         // MySQL password (leave empty for
no password)
});
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to the MySQL server:
' + err.stack);
    return;
  }
  console.log('Connected to MySQL server as ID ' +
connection.threadId);
  connection.query('CREATE DATABASE IF NOT EXISTS
college', (err, results) => {
    if (err) {
      console.error('Error creating database: ' + err.stack);
      return;
    }
    console.log('Database "college" created or already
exists.');
    connection.query('USE college', (err, results) => {
      if (err) {
        console.error('Error selecting database: ' +
err.stack);
        return;
      }
      console.log('Using "college" database.');
      const createTableQuery = `
        CREATE TABLE IF NOT EXISTS students (
          id INT AUTO_INCREMENT PRIMARY KEY,
          name VARCHAR(100),
          age INT,
          course VARCHAR(100)
        )
      `;
      connection.query(createTableQuery, (err, results) =>
{
        if (err) {
          console.error('Error creating table: ' + err.stack);
          return;
        }
        console.log('Table "students" created or already
exists.');
        connection.end();
      });
    });    });   });
```

Slip 10_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.

Create modules.js:

```
// modules.js
module.exports.getDateTime = function() {
    const date = new Date();
    return date.toString(); // You can format it however
you like
};
```

```
//Create app.js
// app.js
const http = require('http');
const myModule = require('./modules'); // Import your
custom module

const server = http.createServer((req, res) => {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Current Date and Time: ' +
myModule.getDateTime());
});

server.listen(3000, () => {
    console.log('Server running at
http://localhost:3000/');
});

//Run
node app.js
```

Slip 11_1

Create a Node.js file that demonstrate create college database and table in MySQL

```
//npm install mysql2
const mysql = require('mysql2');
// Create a connection to MySQL server
const connection = mysql.createConnection({
  host: 'localhost',    // MySQL server address
  user: 'root',         // MySQL username (default: 'root')
  password: '',         // MySQL password (leave empty for
no password)
});
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to the MySQL server:
' + err.stack);
    return;
  }
  console.log('Connected to MySQL server as ID ' +
connection.threadId);
  connection.query('CREATE DATABASE IF NOT EXISTS
college', (err, results) => {
    if (err) {
      console.error('Error creating database: ' + err.stack);
      return;
    }
    console.log('Database "college" created or already
exists.');
    connection.query('USE college', (err, results) => {
      if (err) {
        console.error('Error selecting database: ' +
err.stack);
        return;
      }
      console.log('Using "college" database.');
      const createTableQuery = `
        CREATE TABLE IF NOT EXISTS students (
          id INT AUTO_INCREMENT PRIMARY KEY,
          name VARCHAR(100),
          age INT,
          course VARCHAR(100)
        )
      `;
      connection.query(createTableQuery, (err, results) =>
{
        if (err) {
          console.error('Error creating table: ' + err.stack);
          return;
        }
        console.log('Table "students" created or already
exists.');
        connection.end();
      });
    });    });   });
```

Slip 11_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’

```
// npm init -y
npm install express

//app.js
// Importing express module
const express = require('express');
const path = require('path');
const app = express();

// Set up the route to download a file
app.get('/download', (req, res) => {
    // Path to the file you want to serve
    const filePath = path.join(__dirname, 'files',
'sample.txt');  // Replace with your file's path

    // Set the appropriate headers and download the file
    res.download(filePath, (err) => {
        if (err) {
            console.log("Error while downloading the file:",
err);
            res.status(500).send("Something went wrong!");
        }
    });
});

// Start the server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Server running at
http://localhost:${PORT}`);
});
```

Slip 12_1

Create a node.js file that Select all records from the &quot;customers&quot; table, and display the result object on console.

//npm init -y

//npm install mysql2

App.js

```
const mysql = require('mysql2');
const connection = mysql.createConnection({
   host: 'localhost',      // Database host
   user: 'root',           // Your MySQL username
   password: '',           // Your MySQL password
   database: 'your_database_name' // Name of
the database
});
connection.connect((err) => {
   if (err) {
      console.error('Error connecting to the
database:', err.stack);
      return;
   }
   console.log('Connected to the database');
});
"connection.query('SELECT * FROM customers',
(err, results, fields) => {
   if (err) {
      console.error('Error querying the database:',
err.stack);
      return;
   }
   console.log('Records from customers table:',
results);
});
connection.end();
```

Slip 12_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>
<script>
      function validateForm() {
         const nameRegex = /^[A-Za-z\s]+$/;
         const emailRegex = /^[a-zA-Z0-9._-]+@[a-
zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
         const mobileRegex = /^[0-9]{10}$/;
          const name =
document.getElementById('name').value;
         const email =
document.getElementById('email').value;
         const mobile =
document.getElementById('mobile').value;
         const feedback =
document.querySelector('input[name="feedback"]:checked');
         if (!nameRegex.test(name)) {
            alert("Please enter a valid name (letters and spaces
only).");
            return false;
      }    if (!emailRegex.test(email)) {
            alert("Please enter a valid email address.");
            return false;
       }   if (!mobileRegex.test(mobile)) {
            alert("Please enter a valid 10-digit mobile
number.");
            return false;
       }  if (!feedback) {
            alert("Please select a feedback rating.");
            return false;
        }  alert("Form submitted successfully!");
        return true;
       }
   </script>
</head><body>
   <h2>Student Feedback Form</h2>
   <form onsubmit="return validateForm()">
      <label for="name">Name:</label><br>
      <input type="text" id="name" name="name"
required><br><br>
      <label for="email">Email ID:</label><br>
      <input type="email" id="email" name="email"
required><br><br>
      <label for="mobile">Mobile No.:</label><br>
      <input type="text" id="mobile" name="mobile"
required><br><br>
      <label>Feedback:</label><br>
      <input type="radio" id="notGood" name="feedback"
value="Not good">
      <label for="notGood">Not Good</label><br>
      <input type="radio" id="good" name="feedback"
value="Good">
      <label for="good">Good</label><br>
      <input type="radio" id="veryGood" name="feedback"
value="Very good">
      <label for="veryGood">Very Good</label><br>
      <input type="radio" id="excellent" name="feedback"
value="Excellent">
      <label for="excellent">Excellent</label><br><br>
      <input type="submit" value="Submit">
   </form>  </body>  </html>
```

Slip 13_1

Create a Node.js file that will convert the output &quot;HELLO WORLD!&quot; into lower-case letters.

```
// toLowerCase.js

const text = "HELLO WORLD!";
const lowerCaseText = text.toLowerCase();

console.log(lowerCaseText);
```

run

//node toLowerCase.js

Slip 13_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.

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Registration</title>
  <script>
    function validateForm() {
      const firstName =
document.forms["regForm"]["firstName"].value.trim();
      const lastName =
document.forms["regForm"]["lastName"].value.trim();
      const age =
parseInt(document.forms["regForm"]["age"].value.trim());

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

      if (!namePattern.test(firstName)) {
        alert("First name must contain only alphabets.");
        return false;
      }
      if (!namePattern.test(lastName)) {
        alert("Last name must contain only alphabets.");
        return false;
      }
      if (isNaN(age) || age < 18 || age > 50) {
        alert("Age must be a number between 18 and 50.");
        return false;
      }
      return true;
    }
  </script>
</head>
<body>
  <h2>Student Registration Form</h2>
  <form name="regForm" onsubmit="return
validateForm()">
    <label>First Name:</label>
    <input type="text" name="firstName"
required><br><br>

    <label>Last Name:</label>
    <input type="text" name="lastName"
required><br><br>
    <label>Age:</label>
    <input type="number" name="age" required><br><br>
    <input type="submit" value="Register">
  </form>
</body>
</html>
```

Slip 14_1

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

// simpleServer.js

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

const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/html' });
  res.write('<h2>Welcome to the Node.js Simple Web
Server!</h2>');
  res.end();
});

server.listen(3000, () => {
  console.log('Server is running at
http://localhost:3000');
});
```

Slip 14_2

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

```
<!DOCTYPE html>
<html>
<head>
  <title>Employee Registration</title>
  <script>
    function validateForm() {
      const dob = new
Date(document.forms["empForm"]["dob"].value);
      const joining = new
Date(document.forms["empForm"]["joining"].value);
      const salary =
parseFloat(document.forms["empForm"]["salary"].value);

      const today = new Date();
      if (dob >= today) {
        alert("DOB must be in the past.");
        return false;
      }

      if (joining > today || joining < dob) {
        alert("Joining date must be after DOB and not in the
future.");
        return false;
      }

      if (isNaN(salary) || salary <= 0) {
        alert("Salary must be a positive number.");
        return false;
      }
      return true;
    }
  </script>
</head>
<body>
  <h2>Employee Registration Form</h2>
  <form name="empForm" onsubmit="return
validateForm()">
    <label>Full Name:</label>
    <input type="text" name="name" required><br><br>
    <label>Date of Birth:</label>
    <input type="date" name="dob" required><br><br>

    <label>Joining Date:</label>
    <input type="date" name="joining" required><br><br>
    <label>Salary:</label>
    <input type="number" name="salary" step="0.01"
required><br><br>

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

Slip 15_1

Create a node.js file that Select all records from the &quot;students&quot; table, and display the result object on console.

```
// selectStudents.js
// selectStudents.js

const mysql = require('mysql');

// MySQL connection configuration
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',       // add your MySQL password if any
  database: 'your_database_name' // change to your
actual database
});

// Connect to MySQL
connection.connect(err => {
  if (err) throw err;
  console.log('Connected to MySQL!');

  const query = 'SELECT * FROM students';

  connection.query(query, (err, results) => {
    if (err) throw err;
    console.log('Student Records:', results);
    connection.end(); // Close connection
  });
});
```

Slip 15_2

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>
<head>
  <title>Employee Form</title>
  <script>
    function validateForm() {
      const name =
document.forms["empForm"]["name"].value.trim();
      const email =
document.forms["empForm"]["email"].value.trim();
      const mobile =
document.forms["empForm"]["mobile"].value.trim();
      const department =
document.forms["empForm"]["department"].value.trim();
      const joiningDate =
document.forms["empForm"]["joining"].value.trim();
      const namePattern = /^[A-Za-z\s]+$/;
      const emailPattern = /^[a-zA-Z0-9._%+-]+@[a-z0-9.-
]+\.[a-z]{2,4}$/;
      const mobilePattern = /^[6-9]\d{9}$/;
      const deptPattern = /^[A-Za-z\s]+$/;
      const datePattern = /^\d{4}-\d{2}-\d{2}$/; // yyyy-
mm-dd
      if (!namePattern.test(name)) {
        alert("Invalid name. Only alphabets and spaces
allowed.");
        return false;
      }
      if (!emailPattern.test(email)) {
        alert("Invalid email format.");
        return false;
      }
      if (!mobilePattern.test(mobile)) {
        alert("Invalid mobile number. Must be 10 digits
starting with 6-9.");
        return false;
      }
      if (!deptPattern.test(department)) {
        alert("Invalid department name.");
        return false;
      }
      if (!datePattern.test(joiningDate)) {
        alert("Invalid joining date format (yyyy-mm-
dd).");
        return false;
      }
      return true;
    }
  </script>
</head>
<body>
  <h2>Employee Registration Form</h2>
  <form name="empForm" onsubmit="return
validateForm()">
    <label>Full Name:</label>
    <input type="text" name="name"
required><br><br>
    <label>Email ID:</label>
    <input type="email" name="email"
required><br><br>
    <label>Mobile Number:</label>
    <input type="text" name="mobile"
required><br><br>
    <label>Department:</label>
    <input type="text" name="department"
required><br><br>
    <label>Joining Date:</label>
    <input type="date" name="joining"
required><br><br>

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

Slip 16_1
Using node js create a Recipe Book.

//recipes.json

```
[
  {
    "title": "Pasta",
    "ingredients": "Pasta, Tomato Sauce, Cheese",
    "instructions": "Boil pasta, add sauce, mix with cheese."
  }
]
```

//server .js

```
const http = require('http');
const fs = require('fs');
const url = require('url');
const qs = require('querystring');
const PORT = 3000;
function renderForm() {
  return `
    <h2>Add Recipe</h2>
    <form method="POST" action="/add">
      Title: <input name="title" required /><br><br>
      Ingredients: <input name="ingredients" required /><br><br>
      Instructions: <input name="instructions" required /><br><br>
      <input type="submit" value="Add Recipe" />
    </form>
    <br><a href="/">Back to Recipes</a>
  `;
}
function renderRecipes(recipes) {
  let content = `<h1>Recipe Book</h1><a href="/form">Add New Recipe</a><ul>`;
  recipes.forEach(r => {
    content += `<li><strong>${r.title}</strong><br>Ingredients: ${r.ingredients}<br>Instructions: ${r.instructions}</li><br>`;
  });
  content += '</ul>';
  return content;
}
const server = http.createServer((req, res) => {
  const parsedUrl = url.parse(req.url, true);
  if (req.method === 'GET' && parsedUrl.pathname === '/') {
    fs.readFile('recipes.json', (err, data) => {
      const recipes = JSON.parse(data || '[]');
      res.writeHead(200, { 'Content-Type': 'text/html' });
      res.end(renderRecipes(recipes));
    });

  } else if (req.method === 'GET' && parsedUrl.pathname === '/form') {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(renderForm());
  } else if (req.method === 'POST' && parsedUrl.pathname === '/add') {
    let body = '';
    req.on('data', chunk => { body += chunk; });
    req.on('end', () => {
      const formData = qs.parse(body);
      fs.readFile('recipes.json', (err, data) => {
        const recipes = JSON.parse(data || '[]');
        recipes.push(formData);
        fs.writeFile('recipes.json', JSON.stringify(recipes, null, 2), err => {
          res.writeHead(302, { Location: '/' });
          res.end();
        });
      });
    });
  } else {
    res.writeHead(404, { 'Content-Type': 'text/html' });
    res.end('<h2>404 - Page Not Found</h2>');
  }
});
server.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}`);
});
```

Slip 16_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 call-back function when one of those events is detected.

```
// main.js
const events = require('events');
// Create an EventEmitter object
const eventEmitter = new events.EventEmitter();
// Define callback functions
function onGreet() {
  console.log('Hello! Event "greet" was triggered.');
}
function onExit() {
  console.log('Exiting... Event "exit" was triggered.');
  process.exit();
}
// Register events with their callbacks
eventEmitter.on('greet', onGreet);
eventEmitter.on('exit', onExit);
// Simulate a main loop (event check every 2 seconds)
setInterval(() => {
  const possibleEvents = ['greet', 'exit'];
  const randomEvent = possibleEvents[Math.floor(Math.random() * possibleEvents.length)];
    console.log(`\n[Main Loop] Detected event: ${randomEvent}`);
  eventEmitter.emit(randomEvent);
}, 2000);
```

Slip 17_1

Q.1) Using angular js Create a SPA that show Syllabus content of all subjects of M.Sc (CS) Sem-II (use ng-view)

```
spa-syllabus/
 ├── index.html
 ├── app.js
 ├── home.html
 ├── subject1.html
 ├── subject2.html
 ├── subject3.html

<!DOCTYPE html>
<html ng-app="syllabusApp">
<head>
  <title>M.Sc (CS) Sem-II Syllabus</title>
  <script src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.8.2/angular.min.js"></script>
  <script src="https://ajax.googleapis.com/
ajax/libs/angularjs/1.8.2/angular-route.js"></script>
  <script src="app.js"></script>
</head>
<body>
  <h2>M.Sc (CS) Sem-II Syllabus</h2>
  <a href="#!/">Home</a> |
  <a href="#!/subject1">Advanced Java</a> |
  <a href="#!/subject2">Data Mining</a> |
  <a href="#!/subject3">Machine Learning</a>
  <div ng-view></div>
</body>
</html>
//  app.js
var app = angular.module("syllabusApp", ["ngRoute"]);
app.config(function($routeProvider) {
  $routeProvider
    .when("/", {
      templateUrl: "home.html"
    })
    .when("/subject1", {
      templateUrl: "subject1.html"
    })
    .when("/subject2", {
      templateUrl: "subject2.html"
    })
    .when("/subject3", {
      templateUrl: "subject3.html"   });});
//home.html
<h3>Welcome to M.Sc (CS) Sem-II Syllabus</h3>
<p>Select a subject to view its syllabus.</p>
//subject1.html
<h3>Advanced Java</h3>
<ul>
  <li>Servlets & JSP</li>
  <li>Spring Framework</li>
  <li>Hibernate ORM</li>
</ul>
```

Slip 17_2

Q.2) Using angular js display the 10 student details in Table format (using ng-repeat directive use Array to store data) (same as slip8 Q2)

```
<!DOCTYPE html>
<html ng-app="studentApp">
<head>
  <title>Student Details</title>
  <script src="https://ajax.googleapis.com
/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
  <div ng-controller="StudentController">
    <h2>Student Details</h2>
    <table border="1" cellpadding="10">
      <tr>
        <th>Roll No</th>
        <th>Name</th>
        <th>Age</th>
        <th>City</th>
      </tr>
      <tr ng-repeat="student in students">
        <td>{{student.roll}}</td>
        <td>{{student.name}}</td>
        <td>{{student.age}}</td>
        <td>{{student.city}}</td>
      </tr>
    </table>
  </div>
  <script>
    angular.module("studentApp", [])
      .controller("StudentController", function($scope) {
        $scope.students = [
          { roll: 1, name: "Amit", age: 22, city: "Pune" },
          { roll: 2, name: "Sneha", age: 21, city: "Mumbai" },
          { roll: 3, name: "Rahul", age: 23, city: "Nashik" },
          { roll: 4, name: "Priya", age: 22, city: "Satara" },
          { roll: 5, name: "Sagar", age: 24, city: "Kolhapur" },
          { roll: 6, name: "Neha", age: 20, city: "Nagpur" },
          { roll: 7, name: "Rohit", age: 21, city: "Solapur" },
          { roll: 8, name: "Kiran", age: 22, city: "Jalgaon" },
          { roll: 9, name: "Deepa", age: 23, city:
"Aurangabad" },
          { roll: 10, name: "Yogesh", age: 22, city:
"Ahmednagar" }
        ];
      });
  </script>

</body>
</html>
```

Slip 18_1 *

Using node js create a User Login System

app.js

```
const express = require('express');
const bcrypt = require('bcryptjs');
const bodyParser = require('body-parser');
const session = require('express-session');
const app = express();
const port = 3000;
const users = [];
app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static('views'));
app.use(session({
  secret: 'secret-key',
  resave: false,
  saveUninitialized: true
}));
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/views/login.html');
});
app.get('/register', (req, res) => {
  res.sendFile(__dirname + '/views/register.html');
});
app.post('/register', async (req, res) => {
  const { username, password } = req.body;
  const hashedPassword = await bcrypt.hash(password, 10);
  users.push({ username, password: hashedPassword });
  res.redirect('/');
});
app.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const user = users.find(u => u.username === username);
  if (!user) {
    return res.send('User not found');
  }
  const match = await bcrypt.compare(password, user.password);
  if (match) {
    req.session.user = user;
    res.redirect('/dashboard');
  } else {
    res.send('Invalid credentials');
  }
});
app.get('/dashboard', (req, res) => {
  if (!req.session.user) {
    return res.redirect('/');
  }
  res.send(`<h1>Welcome, ${req.session.user.username}</h1><a href="/logout">Logout</a>`);
});
app.get('/logout', (req, res) => {
  req.session.destroy((err) => {
    res.redirect('/');
  });
});
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

// login.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
</head>
<body>
  <h2>Login</h2>
  <form action="/login" method="POST">
    <label for="username">Username:</label><br>
    <input type="text" id="username" name="username" required><br><br>
    <label for="password">Password:</label><br>
    <input type="password" id="password" name="password" required><br><br>
    <input type="submit" value="Login">
  </form>
  <br>
  <a href="/register">Don't have an account? Register here</a>
</body>
</html>
```

register.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Register</title>
</head>
<body>
  <h2>Register</h2>
  <form action="/register" method="POST">
    <label for="username">Username:</label><br>
    <input type="text" id="username" name="username" required><br><br>
    <label for="password">Password:</label><br>
    <input type="password" id="password" name="password" required><br><br>
    <input type="submit" value="Register">
  </form>
  <br>
  <a href="/">Already have an account? Login here</a>
</body>
</html>
```

Slip 18_2

Create a node.js file that Select all records from the &quot;customers&quot; table, and find the customers whose name starts from ‘A’.

//Create a customers table in MySQL

```
CREATE TABLE customers (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255),
  email VARCHAR(255),
  phone VARCHAR(15)
);
```

//Insert sample data into the customers table:

```
INSERT INTO customers (name, email, phone) VALUES
('Alice', 'alice@example.com', '1234567890'),
('Bob', 'bob@example.com', '2345678901'),
('Amanda', 'amanda@example.com', '3456789012'),
('Charlie', 'charlie@example.com', '4567890123');
```

// **Install the necessary packages**:

```
npm init -y
npm install mysql2
```

//app.js

```
const mysql = require('mysql2');
// Create a connection to the database
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '', // Replace with your MySQL password
  database: 'your_database_name' // Replace with your actual database name
});
// Connect to the database
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to the database:', err);
    return;
  }
  console.log('Connected to the database!');
});
// Query to select customers whose name starts with 'A'
const query = "SELECT * FROM customers WHERE name LIKE 'A%'";
// Execute the query
connection.query(query, (err, results) => {
  if (err) {
    console.error('Error executing the query:', err);
    return;
  }
  // Display the results
  console.log('Customers whose name starts with "A":');
  console.table(results);
});
// Close the database connection
connection.end();
```

Slip 19_1

Create a Node.js file that will convert the output &quot;Hello World!&quot; into upper-case letters.

app.js (Node.js file)

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

// Convert the string to upper-case
const upperCaseMessage = message.toUpperCase();

// Output the upper-case message
console.log(upperCaseMessage);
```

Slip 19_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.

HTML File (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Form Validation - AngularJS</title>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-app="validationApp" ng-controller="formController">
   <div class="container">
      <h2>Registration Form</h2>
      <form name="userForm" ng-submit="submitForm()">
         <div>
            <label for="name">Name:</label>
            <input type="text" id="name" name="name" ng-model="user.name" required
                  ng-pattern="/^[a-zA-Z\s]+$/" />
            <span ng-show="userForm.name.$touched && userForm.name.$invalid">Name should contain only characters.</span>
         </div>
         <div>
            <label for="mobile">Mobile Number:</label>
            <input type="text" id="mobile" name="mobile" ng-model="user.mobile" required
                  ng-pattern="/^\d{10}$/" />
            <span ng-show="userForm.mobile.$touched && userForm.mobile.$invalid">Mobile number should contain exactly 10 digits.</span>
         </div>
         <div>
            <label for="pinCode">Pin Code:</label>
            <input type="text" id="pinCode" name="pinCode" ng-model="user.pinCode" required
                  ng-pattern="/^\d{6}$/" />
            <span ng-show="userForm.pinCode.$touched && userForm.pinCode.$invalid">Pin code should contain exactly 6 digits.</span>
         </div>
         <div>
            <label for="email">Email:</label>
            <input type="email" id="email" name="email" ng-model="user.email" required
                  ng-pattern="/^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/" />
            <span ng-show="userForm.email.$touched && userForm.email.$invalid">Enter a valid email address with one @ and a dot symbol.</span>
         </div>
         <div>
            <label for="address">Address:</label>
            <textarea id="address" name="address" ng-model="user.address" required>
   SPPU M.Sc. Computer Science Syllabus 2023-24
            </textarea>
            <span ng-show="userForm.address.$touched && userForm.address.$invalid">Address should be SPPU M.Sc. Computer Science Syllabus 2023-24.</span>
         </div>
         <button type="submit" ng-disabled="userForm.$invalid">Submit</button>
      </form>
   </div>
   <script src="app.js"></script>
</body>
</html>
```

AngularJS Controller (app.js)

```
var app = angular.module('validationApp', []);
app.controller('formController', function($scope) {
   $scope.user = {
      name: '',
      mobile: '',
      pinCode: '',
      email: '',   address: 'SPPU M.Sc. Computer Science Syllabus 2023-24'
   };
   $scope.submitForm = function() {
      if ($scope.userForm.$valid) {
         alert('Form Submitted Successfully!');
         console.log($scope.user);
      } else {
         alert('Please fill in the form correctly.');
      }
   };
});
```

Slip 20_1

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

//App.js

```
const mysql = require('mysql2');
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
});
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to the database:',
err);
    return;
  }
  console.log('Connected to MySQL server!');

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

    connection.changeUser({ database: 'school_db' },
(err) => {
      if (err) {
        console.error('Error changing database:', err);
        return;
      }
      console.log('Now using "school_db" database.');
          const createTableQuery = `
        CREATE TABLE IF NOT EXISTS students (
          id INT AUTO_INCREMENT PRIMARY KEY,
          name VARCHAR(100),
          age INT,
          email VARCHAR(100),
          phone VARCHAR(15)
        )
      `;
      connection.query(createTableQuery, (err, result) => {
        if (err) {
          console.error('Error creating students table:', err);
          return;
        }
        console.log('Table "students" created or already
exists.');
        connection.end();
      });
    });
  });
});
```

Slip 20_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

```
<!DOCTYPE html>
<html lang="en">
<head> <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>Username and Password Validation -
AngularJS</title>
    <script src="https://ajax.googleapis.com/ajax/libs
/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-app="validationApp" ng-
controller="formController">
    <div class="container">
        <h2>Login Form</h2>
        <form name="loginForm" ng-
submit="submitForm()">
    <div>  <label for="username">Username:</label>
                <input type="text" id="username"
name="username" ng-model="user.username" required
/>
     </div>  <div>
                <label for="password">Password:</label>
     <input type="password"
id="password"name="password" ng-
model="user.password" required />
            </div>
            <button type="submit">Submit</button>
        </form>
    </div>
    <script src="app.js"></script>
</body>
</html>
var app = angular.module('validationApp', []);
app.controller('formController', function($scope) {
    $scope.user = {
        username: '',
        password: ''
    };
    $scope.submitForm = function() {
        if (!$scope.user.username) {
            alert("Enter username");
        } else if ($scope.user.username.length < 3) {
            alert("Username is too short");
        } else if ($scope.user.password.length < 8) {
            alert("Password should be at least 8 characters");
        } else {
            alert("Valid username");
        }
    };  });
```

Slip 21_1 Create a Node.js file that demonstrate create Movie database and table in MySQL

```
const mysql = require('mysql2');

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',  // Replace with your MySQL password
});

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

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

    connection.changeUser({ database: 'movie_db' },
(err) => {
      if (err) {
        console.error('Error changing database:', err);
        return;
      }
      console.log('Now using "movie_db" database.');

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

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

Slip 21_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.

```
npm init -y
npm install express

const express = require('express');
const path = require('path');
const app = express();
const port = 3000;

app.get('/download', (req, res) => {
  const filePath = path.join(__dirname, 'files',
'sample.txt'); // Path to the file you want to send

  res.download(filePath, 'sample.txt', (err) => {
    if (err) {
      console.error('Error sending file:', err);
      res.status(500).send('Error sending file.');
    }
  });
});

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

Slip 22_1

Using node js create an Employee Registration Form validation.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Employee Registration</title>
</head>
<body>
  <h2>Employee Registration Form</h2>
  <form action="/register" method="POST">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>
    <br><br>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <br><br>
    <label for="phone">Phone:</label>
    <input type="text" id="phone" name="phone" required>
    <br><br>
    <label for="salary">Salary:</label>
    <input type="number" id="salary" name="salary" required>
    <br><br>
    <label for="joining_date">Joining Date:</label>
    <input type="date" id="joining_date" name="joining_date" required>
    <br><br>
    <button type="submit">Register</button>
  </form>
</body>
</html>
```

//Node.js

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const port = 3000;
app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static('views'));
app.get('/', (req, res) => {
 res.sendFile(__dirname + '/views/index.html');
});
app.post('/register', (req, res) => {
 const { name, email, phone, salary, joining_date } = req.body;
 const nameRegex = /^[A-Za-z ]+$/;
 if (!nameRegex.test(name)) {
   return res.send('Invalid name. Only alphabets are allowed.');
 }
 const emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
 if (!emailRegex.test(email)) {
   return res.send('Invalid email format.');
 }
 const phoneRegex = /^[0-9]{10}$/;
 if (!phoneRegex.test(phone)) {
   return res.send('Phone number must be exactly 10 digits.');
} if (isNaN(salary) || salary <= 0) {
   return res.send('Salary must be a valid number greater than 0.');
 }
 const dateRegex = /^\d{4}-\d{2}-\d{2}$/;
 if (!dateRegex.test(joining_date)) {
   return res.send('Invalid joining date. Use the format YYYY-MM-DD.');
 }
 res.send('Employee Registered Successfully!');
});
app.listen(port, () => {
 console.log(`Server running at http://localhost:${port}`);
});
```

Slip 22_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.

```
const EventEmitter = require('events');
const eventEmitter = new EventEmitter();
const startEventHandler = () => {
 console.log("The event 'start' was triggered!");
};
const endEventHandler = () => {
 console.log("The event 'end' was triggered!");
};
eventEmitter.on('start', startEventHandler);
eventEmitter.on('end', endEventHandler);
const mainLoop = () => {
 let counter = 0;
   const interval = setInterval(() => {
   counter++;
   console.log(`Main loop is running... Cycle ${counter}`);
     if (counter === 2) {
    eventEmitter.emit('start');
   }
     if (counter === 5) {
    eventEmitter.emit('end');
    clearInterval(interval);
   } }, 1000);
};
mainLoop();
```

Slip 23_1

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

//Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-
width, initial-scale=1.0">
   <title>File System Web Page</title>
</head>
<body>
   <h1>Welcome to the Web Page served from a
File</h1>
   <p>This web page was read from the file system using
Node.js.</p>
</body>
</html>
```

App.js

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

const port = 3000;

http.createServer((req, res) => {
   const filePath = path.join(__dirname, 'views',
'index.html');

   fs.readFile(filePath, 'utf-8', (err, data) => {
      if (err) {
         res.statusCode = 500;
         res.end('Error reading the HTML file.');
         return;
      }

      res.statusCode = 200;
      res.setHeader('Content-Type', 'text/html');
      res.end(data);
   });
}).listen(port, () => {
   console.log(`Server is running at
http://localhost:${port}`);
});
```

Slip 23_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.

// modules.js

```
// Function to return today's date and time
function getCurrentDateTime() {
   const currentDate = new Date();
   return currentDate.toString();
}

// Export the function to make it available externally
module.exports.getCurrentDateTime =
getCurrentDateTime;
```

//app.js

```
const http = require('http'); // Built-in HTTP module
const myModule = require('./modules'); // Import the
custom module

const port = 3000;

// Create an HTTP server
const server = http.createServer((req, res) => {
   // Get the current date and time from the custom
module
   const currentDateTime =
myModule.getCurrentDateTime();

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

   // Send the current date and time as a response
   res.end(`<h1>Current Date and Time:
${currentDateTime}</h1>`);
});

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

Slip 24_1

Using node js create an eLearning System

db.js

```
const sqlite3 = require('sqlite3').verbose();
const db = new sqlite3.Database(':memory:');
db.serialize(() => {
   db.run("CREATE TABLE users (id INTEGER PRIMARY
KEY, username TEXT, password TEXT)");
   db.run("CREATE TABLE courses (id INTEGER PRIMARY
KEY, name TEXT, description TEXT)");
   db.run("CREATE TABLE enrollments (user_id INTEGER,
course_id INTEGER, PRIMARY KEY (user_id,
course_id))");
});
module.exports = db;
```

//app.js

```
const express = require('express');
const bcrypt = require('bcryptjs');
const bodyParser = require('body-parser');
const session = require('express-session');
const db = require('./db');
const app = express();
const port = 3000;
app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static('public'));
app.use(session({
   secret: 'secret-key',
   resave: false,
   saveUninitialized: true
}));
app.get('/', (req, res) => {
   if (!req.session.userId) {
     return res.redirect('/login');   }
   res.redirect('/dashboard');
});
app.get('/register', (req, res) => {
   res.send(`
      <form action="/register" method="POST">
         <input type="text" name="username"
placeholder="Username" required />
         <input type="password" name="password"
placeholder="Password" required />
         <button type="submit">Register</button>
      </form>   `);  });
app.post('/register', (req, res) => {
   const { username, password } = req.body;
   const hashedPassword = bcrypt.hashSync(password,
10);
   db.run("INSERT INTO users (username, password)
VALUES (?, ?)", [username, hashedPassword], (err) => {
      if (err) {
         return res.send("Error registering user.");
      }
      res.redirect('/login');
}); });
  app.get('/login', (req, res) => {
 res.send(`
      <form action="/login" method="POST">
         <input type="text" name="username"
placeholder="Username" required />
         <input type="password" name="password"
placeholder="Password" required />
         <button type="submit">Login</button>
      </form> `);  });
app.post('/login', (req, res) => {
   const { username, password } = req.body;
   db.get("SELECT * FROM users WHERE username = ?",
[username], (err, user) => {
      if (!user || !bcrypt.compareSync(password,
user.password)) {
         return res.send("Invalid username or
password.");
      }
      req.session.userId = user.id;
      res.redirect('/dashboard');
  }); });
app.get('/dashboard', (req, res) => {
   if (!req.session.userId) {
      return res.redirect('/login');  }
   db.all("SELECT * FROM courses", (err, courses) => {
      res.send(`
         <h1>Welcome to the Dashboard</h1>
         <h2>Available Courses:</h2> <ul>
            ${courses.map(course => `  <li>
  <a href="/enroll/${course.id}">${course.name}</a>:
${course.description}
               </li>
            `).join('')}
         </ul>
         <a href="/logout">Logout</a>  `);   });  });
app.get('/enroll/:courseId', (req, res) => {
   if (!req.session.userId) {
    return res.redirect('/login'); }
   const courseId = req.params.courseId;
   const userId = req.session.userId;
   db.run("INSERT INTO enrollments (user_id,
course_id) VALUES (?, ?)", [userId, courseId], (err) => {
      if (err) {
   return res.send("Error enrolling in course.");  }
      res.send("Successfully enrolled! <a
href='/dashboard'>Back to Dashboard</a>");
   });  });
app.get('/logout', (req, res) => {
   req.session.destroy(() => {
      res.redirect('/');
  });   });
app.listen(port, () => {
   console.log(`Server is running at
http://localhost:${port}`);db.run("INSERT INTO courses
(name, description) VALUES ('Math 101', 'Basic
Mathematics'), ('CS 101', 'Introduction to Computer
Science')");
});
```

Slip 24_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

//index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="validationApp">
<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://code.angularjs.org/1.8.2/angular.min.js"></script>
  <script src="app.js"></script>
</head>
<body ng-controller="validationController">
  <div>
    <h2>User Registration</h2>
    <form name="validationForm">
      <label for="username">Username:</label>
      <input type="text" id="username" name="username" ng-model="username" ng-change="validateUsername()" required>
      <br>
      <span ng-show="usernameAlert" style="color:red">{{ usernameAlert }}</span>
      <br><br>
      <label for="password">Password:</label>
      <input type="password" id="password" name="password" ng-model="password" ng-change="validatePassword()" required>
      <br>
      <span ng-show="passwordAlert" style="color:red">{{ passwordAlert }}</span>
      <br><br>
      <button type="submit" ng-disabled="usernameAlert || passwordAlert || !username || !password">Submit</button>
    </form>
  </div>
</body>
</html>
```

App.js

```
angular.module('validationApp', [])
 .controller('validationController', ['$scope', function($scope) {
  $scope.username = '';
  $scope.password = '';

  $scope.usernameAlert = '';
  $scope.passwordAlert = '';

  $scope.validateUsername = function() {
    if (!$scope.username) {
      $scope.usernameAlert = 'Enter username';
    } else if ($scope.username.length < 3) {
      $scope.usernameAlert = 'Username is too short';
    } else {
      $scope.usernameAlert = '';
    }
  };

  $scope.validatePassword = function() {
    if ($scope.password && $scope.password.length < 8) {
      $scope.passwordAlert = 'Password must be at least 8 characters';
    } else {
      $scope.passwordAlert = '';
    }
  };
}]);
```

Slip 25_1
Create an angular JS Application that shows the location of the current web page.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>AngularJS Geolocation</title>
  <script src="https://ajax.googleapis.com/ajax/libs
/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-app="locationApp">

  <div ng-controller="LocationController">
    <h1>Your Current Location</h1>
    <p>Latitude: {{latitude}}</p>
    <p>Longitude: {{longitude}}</p>
    <button ng-click="getLocation()">Get Current
Location</button>
  </div>

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

    app.controller('LocationController', ['$scope',
function($scope) {
      $scope.latitude = 'Not available';
      $scope.longitude = 'Not available';

      // Function to get the current location using
geolocation API
      $scope.getLocation = function() {
        if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(function(position)
{
            $scope.latitude = position.coords.latitude;
            $scope.longitude = position.coords.longitude;
            $scope.$apply();  // Apply scope changes
          }, function() {
            alert('Geolocation failed or is not supported
by this browser.');
          });
        } else {
          alert('Geolocation is not supported by this
browser.');
        }
      };
    }]);
  </script>

</body>
</html>
```

Slip 25_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.

// main.js

```
// EventEmitter class for managing events
class EventEmitter {
  constructor() {
    this.events = {};
  }
  // Register an event and its corresponding callback
  on(event, listener) {
    if (!this.events[event]) {
      this.events[event] = [];
    }
    this.events[event].push(listener);
  }
  // Emit an event and trigger all its listeners
  emit(event, ...args) {
    if (this.events[event]) {
      this.events[event].forEach(listener =>
listener(...args));
    }
  }
}
// Instantiate the EventEmitter
const eventEmitter = new EventEmitter();
// Example callback functions
function onUserLogin(username) {
  console.log(`${username} has logged in.`);
}
function onDataReceived(data) {
  console.log(`Data received: ${data}`);
}
// Register events and their callbacks
eventEmitter.on('userLogin', onUserLogin);
eventEmitter.on('dataReceived', onDataReceived);
// Simulating event triggering in the main loop
function mainLoop() {
  // Simulate user login event
  setTimeout(() => {
    eventEmitter.emit('userLogin', 'RohanJadhav');
  }, 2000);
  // Simulate data received event
  setTimeout(() => {
    eventEmitter.emit('dataReceived', 'Sample
Data');
  }, 4000);

  // More events can be added here
}
// Start the main loop
mainLoop();
```