Slip Solution

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 lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login Form</title>
<style>
    body {
        font-family: Arial, sans-serif;
    }
    form {
        max-width: 300px;
        margin: 0 auto;
    }
    input[type="text"],
    input[type="password"],
    input[type="submit"] {
        width: 100%;
        padding: 10px;
        margin: 5px 0;
        box-sizing: border-box;
    }
    input[type="submit"] {
        background-color: #4CAF50;
        color: white;
        border: none;
```

```
        cursor: pointer;
    }
    input[type="submit"]:hover {
        background-color: #45a049;
    }
    .error {
        color: red;
    }
</style>
</head>
<body>

<form id="loginForm" onsubmit="return validateForm()">
    <label for="email">Email:</label>
    <input type="text" id="email" name="email" placeholder="Enter your email" required>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" placeholder="Enter your password"
required>
    <input type="submit" value="Login">
</form>

<script>
    function validateForm() {
        var email = document.getElementById('email').value;
        var password = document.getElementById('password').value;
        var emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

        if (!emailPattern.test(email)) {
            document.getElementById('email').classList.add('error');
            return false;
        } else {
            document.getElementById('email').classList.remove('error');
        }
```

```
        // Password should be at least 8 characters long
        if (password.length < 8) {
            document.getElementById('password').classList.add('error');
            return false;
        } else {
            document.getElementById('password').classList.remove('error');
        }

        return true;
    }
</script>

</body>
</html>
```

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 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;
    }
    form {
        max-width: 300px;
```

```
        margin: 0 auto;
    }
    input[type="text"],
    input[type="number"],
    input[type="submit"] {
        width: 100%;
        padding: 10px;
        margin: 5px 0;
        box-sizing: border-box;
    }
    input[type="submit"] {
        background-color: #4CAF50;
        color: white;
        border: none;
        cursor: pointer;
    }
    input[type="submit"]:hover {
        background-color: #45a049;
    }
    .error {
        color: red;
    }
</style>
</head>
<body>

<form id="registrationForm" onsubmit="return validateForm()">
    <label for="firstName">First Name:</label>
    <input type="text" id="firstName" name="firstName" placeholder="Enter your first name"
required>
    <label for="lastName">Last Name:</label>
    <input type="text" id="lastName" name="lastName" placeholder="Enter your last name" required>
    <label for="age">Age:</label>
```

```
    <input type="number" id="age" name="age" placeholder="Enter your age" required>
    <input type="submit" value="Register">
</form>

<script>
    function validateForm() {
        var firstName = document.getElementById('firstName').value;
        var lastName = document.getElementById('lastName').value;
        var age = parseInt(document.getElementById('age').value);

        var namePattern = /^[a-zA-Z]+$/;

        if (!namePattern.test(firstName)) {
            document.getElementById('firstName').classList.add('error');
            return false;
        } else {
            document.getElementById('firstName').classList.remove('error');
        }

        if (!namePattern.test(lastName)) {
            document.getElementById('lastName').classList.add('error');
            return false;
        } else {
            document.getElementById('lastName').classList.remove('error');
        }

        // Age should be between 18 and 50
        if (age < 18 || age > 50 || isNaN(age)) {
            document.getElementById('age').classList.add('error');
            return false;
        } else {
            document.getElementById('age').classList.remove('error');
```

```
        }
        return true;
    }
</script>
</body>
</html>
```

Slip 2

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

```
// app.js

// Define the string
const originalString = "Full Stack!";

// Function to reverse a string
function reverseString(str) {
    return str.split('').reverse().join('');
}

// Convert the original string to reverse
const reversedString = reverseString(originalString);

console.log(reversedString);
```

To run this code:

1. Install Node.js if you haven't already.
2. Save the code above into a file named **app.js**.
3. Open your terminal or command prompt.
4. Navigate to the directory where **app.js** is saved.
5. Run the command **node app.js**.

6. You should see the output printed in the terminal as the reverse of "Full Stack!", which is "!kcatS lluF".

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.

1. First, make sure you have Node.js installed on your system.
2. Initialize a new Node.js project by running **npm init -y** in your project directory.
3. Install Express.js by running **npm install express**.
4. Create a new file named **app.js** and paste the following code:

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

const fs = require('fs');

const path = require('path');



const app = express();

const port = 3000;



app.use(express.urlencoded({ extended: true }));

app.use(express.static('public'));



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

    res.sendFile(path.join(__dirname, 'index.html'));

});



app.post('/mergeFiles', (req, res) => {
```

```
    const { file1, file2 } = req.body;

    // Read content from the first file
    fs.readFile(file1, 'utf8', (err, data) => {
        if (err) {
            return res.status(500).send(err);
        }

        // Append content of first file to the second file
        fs.appendFile(file2, data, (err) => {
            if (err) {
                return res.status(500).send(err);
            }
            res.send('Contents appended successfully!');
        });
    });
});

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

5. Create a directory named **public** in the same directory where **app.js** is located.
6. Create a new file named **index.html** inside the **public** directory and paste the following HTML code:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Merge Files</title>

</head>

<body>

    <h1>Merge Files</h1>

    <form action="/mergeFiles" method="post">

        <label for="file1">File 1:</label>

        <input type="text" id="file1" name="file1" required><br>

        <label for="file2">File 2:</label>

        <input type="text" id="file2" name="file2" required><br>

        <button type="submit">Merge Files</button>

    </form>

</body>

</html>
```

7. Run the Node.js application by executing **node app.js**.

8. Open your web browser and navigate to **http://localhost:3000**.
9. Enter the filenames of the two files you want to merge into the respective input fields and click the "Merge Files" button.

Slip 3

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

```
const express = require('express');
const mongoose = require('mongoose');
const bcrypt = require('bcrypt');

const app = express();
const PORT = process.env.PORT || 3000;

// Connect to MongoDB
mongoose.connect('mongodb://localhost/user_login_system', {
useNewUrlParser: true, useUnifiedTopology: true });
const db = mongoose.connection;

db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', () => {
    console.log('Connected to MongoDB');
```

```
});

// Define User Schema
const userSchema = new mongoose.Schema({
    username: String,
    password: String
});

const User = mongoose.model('User', userSchema);

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

// Routes
app.post('/register', async (req, res) => {
    try {
        const hashedPassword = await bcrypt.hash(req.body.password, 10);
        const user = new User({
            username: req.body.username,
            password: hashedPassword
        });
```

```
        await user.save();
        res.status(201).send('User registered successfully');
    } catch (error) {
        res.status(500).send('Error registering user');
    }
});

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

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**.**

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

// Create a connection to the database
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'your_database_user',
  password: 'your_database_password',
  database: 'your_database_name'
```

```
});

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

  // Select all records from the "Teacher" table where salary > 20000
  const query = 'SELECT * FROM Teacher WHERE salary > 20000';

  connection.query(query, (err, results) => {
    if (err) {
      console.error('Error executing query:', err);
      return;
    }
    console.log('Teachers with salary greater than 20000:');
    console.log(results);
  });
```

```
});


// Close the connection after executing the query

connection.end();
```

Slip 5

Q.1) Create a Node.js file that writes an HTML form, with an upload field.

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

const app = express();
const PORT = process.env.PORT || 3000;

// Set up multer for handling file uploads
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, 'uploads/')
  },
  filename: function (req, file, cb) {
    cb(null, file.fieldname + '-' + Date.now() + path.extname(file.originalname))
  }
});

const upload = multer({ storage: storage });

// Set up the route to serve the form
```

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

// Handle file upload
app.post('/upload', upload.single('file'), (req, res) => {
  if (!req.file) {
    return res.status(400).send('No files were uploaded.');
  }
  res.send('File uploaded successfully.');
});

// Serve static files from the 'uploads' directory
app.use('/uploads', express.static('uploads'));

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

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 Upload</title>
</head>
<body>
```

```
  <h2>Upload File</h2>
  <form action="/upload" method="post" enctype="multipart/form-data">
    <input type="file" name="file" id="file">
    <button type="submit">Upload</button>
  </form>
</body>
</html>
```

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.

```
<!DOCTYPE html>
<html lang="en" ng-app="validationApp">
<head>
    <meta charset="UTF-8">
    <title>Username Validation</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="MainController as vm">
    <h1>Username Validation</h1>

    <form ng-submit="vm.validateUsername()">
        <label for="username">Username:</label>
        <input type="text" id="username" ng-model="vm.username" required>

        <label for="password">Password:</label>
        <input type="password" id="password" ng-model="vm.password" required>
```

```
    <button type="submit">Submit</button>
  </form>

  <div ng-show="vm.message">
    {{ vm.message }}
  </div>

  <script src="app.js"></script>
</body>
</html>
```

App.js

```
angular.module('validationApp', [])
  .controller('MainController', function() {
    var vm = this;

    vm.validateUsername = function() {
      if (!vm.username) {
        vm.message = "Enter username";
      } else if (vm.username.length < 3) {
        vm.message = "Username is too short";
      } else if (!vm.password || vm.password.length < 8) {
        vm.message = "Password should be minimum 8 characters";
      } else {
        vm.message = "Valid username";
      }
    };
  });
```

Slip 6

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

```
<!DOCTYPE html>
<html lang="en" ng-app="alertApp">
<head>
  <meta charset="UTF-8">
  <title>ng-click Directive Example</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="MainController as vm">
  <h1>ng-click Directive Example</h1>

  <!-- Button with ng-click directive -->
  <button ng-click="vm.displayAlert()">Click Me</button>

  <script>
    angular.module('alertApp', [])
      .controller('MainController', function() {
        var vm = this;

        vm.displayAlert = function() {
          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.

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

const server = http.createServer((req, res) => {
  // Get the file path from the request URL
  const filePath = path.join(__dirname, req.url);

  // Check if the file exists
  fs.exists(filePath, (exists) => {
    if (exists) {
      // Read the file
      fs.readFile(filePath, (err, data) => {
        if (err) {
          // If there's an error reading the file, throw a 500 error
          res.writeHead(500, { 'Content-Type': 'text/plain' });
          res.end('Internal Server Error');
        } else {
          // Return the file content
          res.writeHead(200, { 'Content-Type': 'text/plain' });
          res.end(data);
        }
      });
    } else {
      // If the file doesn't exist, throw a 404 error
      res.writeHead(404, { 'Content-Type': 'text/plain' });
      res.end('404 Not Found');
```

```
    }
  });
});

const PORT = process.env.PORT || 3000;

server.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Slip 7

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

```
<!DOCTYPE html>
<html lang="en" ng-app="dateTimeApp">
<head>
  <meta charset="UTF-8">
  <title>Current Date and Time</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="DateTimeController as vm">
  <h1>Current Date and Time</h1>

  <p>{{ vm.currentDateTime }}</p>

  <script>
    angular.module('dateTimeApp', [])
      .controller('DateTimeController', function($interval) {
        var vm = this;
```

```
        // Function to update current date and time
        function updateDateTime() {
            vm.currentDateTime = new Date().toLocaleString();
        }

        // Call the updateDateTime function initially
        updateDateTime();

        // Update current date and time every second using $interval
        var intervalPromise = $interval(updateDateTime, 1000);

        // Cancel the interval when the controller is destroyed
        vm.$onDestroy = function() {
            $interval.cancel(intervalPromise);
        };
    });
  </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

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

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

// Define a callback function to handle the 'customEvent' event
```

```
function customEventHandler() {
    console.log('Custom event detected');
}

// Listen for the 'customEvent' event and trigger the callback function
eventEmitter.on('customEvent', customEventHandler);

// Main loop to listen for events
console.log('Event listener started. Waiting for events...');

// Simulate events being detected (for demonstration purposes)
setTimeout(() => {
    // Emit the 'customEvent' event after 3 seconds
    eventEmitter.emit('customEvent');
}, 3000);
```

Slip 8

Q.1) Create a Simple Web Server using node 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');
});
```

```
// Define the port on which the server will listen
const PORT = process.env.PORT || 3000;

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

Q2) 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" 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>
</head>
<body ng-controller="StudentController as vm">
    <h1>Student Details</h1>

    <!-- Table to display student details -->
    <table border="1">
        <thead>
            <tr>
```

```
            <th>Student ID</th>
            <th>Name</th>
            <th>Age</th>
            <th>Grade</th>
        </tr>
    </thead>
    <tbody>
        <!-- Use ng-repeat to iterate over the array of student objects -->
        <tr ng-repeat="student in vm.students">
            <td>{{ student.id }}</td>
            <td>{{ student.name }}</td>
            <td>{{ student.age }}</td>
            <td>{{ student.grade }}</td>
        </tr>
    </tbody>
</table>

<script>
    angular.module('studentApp', [])
        .controller('StudentController', function() {
            var vm = this;
```

```
            // Array to store student details
            vm.students = [
                { id: 1, name: 'John Doe', age: 20, grade: 'A' },
                { id: 2, name: 'Jane Smith', age: 22, grade: 'B' },
                { id: 3, name: 'David Johnson', age: 21, grade: 'A' },
                { id: 4, name: 'Emily Brown', age: 19, grade: 'C' },
                { id: 5, name: 'Michael Wilson', age: 20, grade: 'B' },
                { id: 6, name: 'Emma Martinez', age: 23, grade: 'A' },
                { id: 7, name: 'Christopher Taylor', age: 21, grade: 'A' },
                { id: 8, name: 'Olivia Garcia', age: 22, grade: 'B' },
                { id: 9, name: 'James Rodriguez', age: 20, grade: 'A' },
                { id: 10, name: 'Sophia Hernandez', age: 19, grade: 'C' }
            ];
        });
    </script>
</body>
</html>
```

Slip 9

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

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

```
// Create an HTTP server
const server = http.createServer((req, res) => {
    // Check if the request method is POST
    if (req.method === 'POST') {
        let data = '';

        // Collect data from the request
        req.on('data', (chunk) => {
            data += chunk;
        });

        // When all data is received
        req.on('end', () => {
            // Parse the form data
            const formData = decodeURIComponent(data);
            const params = formData.split('&');
            const firstString = params[0].split('=')[1];
            const secondString = params[1].split('=')[1];

            // Concatenate the strings
            const result = firstString + secondString;
```

```
        // Send the response with the result
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(`
            <html>
            <head>
                <title>String Concatenation Result</title>
            </head>
            <body>
                <h1>String Concatenation Result</h1>
                <p>First String: ${firstString}</p>
                <p>Second String: ${secondString}</p>
                <p>Concatenated Result: ${result}</p>
            </body>
            </html>
        `);
    });
} else {
    // If request method is not POST, send the HTML form
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(`
        <html>
        <head>
```

```
            <title>String Concatenation Form</title>
        </head>
        <body>
            <h1>String Concatenation Form</h1>
            <form method="post" action="/">
                <label for="firstString">First String:</label>
                <input type="text" id="firstString" name="firstString"><br><br>
                <label for="secondString">Second String:</label>
                <input type="text" id="secondString" name="secondString"><br><br>
                <button type="submit">Concatenate</button>
            </form>
        </body>
        </html>
    `);
  }
});

// Define the port on which the server will listen
const PORT = process.env.PORT || 3000;

// Start the server
server.listen(PORT, () => {
```

```
    console.log(`Server is running on port ${PORT}`);
});
```

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

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

// Create an HTTP server
const server = http.createServer((req, res) => {
    // Get the file path from the request URL
    const filePath = path.join(__dirname, req.url);

    // Read the file
    fs.readFile(filePath, (err, data) => {
        if (err) {
            if (err.code === 'ENOENT') {
                // If the file doesn't exist, throw a 404 error
                res.writeHead(404, { 'Content-Type': 'text/plain' });
                res.end('404 Not Found');
            } else {
                // If there's any other error, throw a 500 error
```

```
            res.writeHead(500, { 'Content-Type': 'text/plain' });
            res.end('500 Internal Server Error');
        }
    } else {
        // If the file is successfully read, send its content back to the client
        res.writeHead(200, { 'Content-Type': 'text/plain' });
        res.end(data);
    }
  });
});

// Define the port on which the server will listen
const PORT = process.env.PORT || 3000;

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

Slip 10

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

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

```
// Create a connection to the MySQL server
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'your_mysql_username',
  password: 'your_mysql_password'
});

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

  // Create the college database
  connection.query('CREATE DATABASE IF NOT EXISTS college', (err) => {
    if (err) {
      console.error('Error creating database:', err);
      return;
    }
```

```
console.log('College database created');


// Switch to the college database
connection.query('USE college', (err) => {
  if (err) {
    console.error('Error switching to college database:', err);
    return;
  }


  // Create the students table
  connection.query(`
    CREATE TABLE IF NOT EXISTS students (
      id INT AUTO_INCREMENT PRIMARY KEY,
      name VARCHAR(255) NOT NULL,
      age INT,
      major VARCHAR(255)
    )
  `, (err) => {
    if (err) {
      console.error('Error creating students table:', err);
      return;
    }
```

```
        console.log('Students table created');


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

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.

```
// modules.js


// Function to return today's date and time
exports.getCurrentDateTime = function() {
```

```
    const currentDate = new Date();

    const options = { weekday: 'long', year: 'numeric', month: 'long', day:
'numeric', hour: 'numeric', minute: 'numeric', second: 'numeric' };

    return currentDate.toLocaleDateString('en-US', options);

};


// app.js


// Require the built-in HTTP module

const http = require('http');


// Require your custom module

const myModule = require('./modules');


// Create an HTTP server

const server = http.createServer((req, res) => {

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


    // Call the function from your custom module to get current date and time

    const currentDateTime = myModule.getCurrentDateTime();


    // Send the current date and time as the response
```

```
    res.end(`Current Date and Time: ${currentDateTime}`);
});

// Define the port on which the server will listen
const PORT = process.env.PORT || 3000;

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

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',
  user: 'your_mysql_username',
  password: 'your_mysql_password',
  database: 'your_database_name'
```

```
});

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

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

  connection.query(createTableQuery, (err) => {
```

```
    if (err) {
      console.error('Error creating movies table:', err);
      return;
    }
    console.log('Movies table created');

    // Close the connection to the MySQL server
    connection.end((err) => {
      if (err) {
        console.error('Error closing connection:', err);
        return;
      }
      console.log('Connection to MySQL server 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

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

const app = express();
```

```
// Serve static files from the 'public' directory
app.use(express.static('public'));

// Define a route to handle file download
app.get('/download', (req, res) => {
    // Define the path to the file to be downloaded
    const filePath = path.join(__dirname, 'public', 'example.txt');

    // Check if the file exists
    fs.exists(filePath, (exists) => {
        if (exists) {
            // Set the content type and attachment header to prompt download
            res.setHeader('Content-Type', 'text/plain');
            res.setHeader('Content-Disposition', 'attachment; filename="example.txt"');

            // Create a read stream to read the file
            const fileStream = fs.createReadStream(filePath);

            // Pipe the file stream to the response object
            fileStream.pipe(res);
        } else {
            // If the file doesn't exist, send a 404 response
            res.status(404).send('File not found');
```

```
        }
    });
});

// Define the port on which the server will listen
const PORT = process.env.PORT || 3000;

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

Slip 12

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

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

// Create a connection to the MySQL server
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'your_mysql_username',
  password: 'your_mysql_password',
  database: 'your_database_name'
});
```

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

  // Select all records from the "customers" table
  connection.query('SELECT * FROM customers', (err, rows) => {
    if (err) {
      console.error('Error selecting records:', err);
      return;
    }

    // Display the result object on console
    console.log('Result:', rows);

    // Close the connection to the MySQL server
    connection.end((err) => {
      if (err) {
        console.error('Error closing connection:', err);
        return;
```

```
    }
    console.log('Connection to MySQL server closed');
  });
 });
});
```

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">
<title>Student Feedback Form</title>
</head>
<body>

<h2>Student Feedback Form</h2>

<form id="feedbackForm" onsubmit="return validateForm()">
  <label for="name">Name:</label><br>
  <input type="text" id="name" name="name" required><br>
```

```
    <label for="email">Email ID:</label><br>
    <input type="email" id="email" name="email" required><br>

    <label for="mobile">Mobile No.:</label><br>
    <input type="text" id="mobile" name="mobile" pattern="[0-9]{10}" required><br>

    <label for="feedback">Feedback:</label><br>
    <select id="feedback" name="feedback" required>
        <option value="">Select Feedback</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>

    <input type="submit" value="Submit">
</form>

<script>
function validateForm() {
    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.trim();
```

```
// Regular expression for email validation
const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

// Regular expression for mobile number validation (10 digits)
const mobilePattern = /^[0-9]{10}$/;

// Validate name
if (name === '') {
    alert('Please enter your name');
    return false;
}

// Validate email
if (!emailPattern.test(email)) {
    alert('Please enter a valid email address');
    return false;
}

// Validate mobile number
if (!mobilePattern.test(mobile)) {
    alert('Please enter a valid 10-digit mobile number');
    return false;
}
```

```
    // Validate feedback
    if (feedback === '') {
        alert('Please select your feedback');
        return false;
    }

    return true;
}
</script>

</body>
</html>
```