1. Create a user-friendly interface with a clean and proper design.

```
<html ng-app="login-form_Demo">
<head>
    <title>login form Demo Program</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>


    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
        }
        .welcome-message {
            font-size: 36px;
            color: #337ab7;
            text-align: center;
            padding: 20px;
        }
    </style>

    <script>
        var myvar = angular.module("login-form_Demo",[]);
        myvar.controller("myctrl",function($scope){

            $scope.message="";
            $scope.uservalidation="";
            $scope.passwordvalidation="";
            $scope.submit=function(){

                if($scope.username===""){
                    $scope.message="Please Enter the Username";
                }
                else if($scope.password===""){
                    $scope.message="Please Enter the Password";
                }
                else if($scope.username=="Jayesh" && $scope.password=="1256"){
                    $scope.message="Welcome Jayesh";
                }
                else{
                    $scope.message="User Invalid !! ";
                }
            };
        });
    </script>
</head>
<body>
    <div class="welcome-message" ng-controller="myctrl">

        <h2> User Interface</h2>
        <form ng-submit="submit()">
            Enter the Username:
                <input type="text" ng-model="username" ><br>
                <p>{{uservalidation}}</p>

            Enter the password:
                <input type="text" ng-model="password" ><br>
                <p>{{passwordvalidation}}</p>

             <button type="submit"> Submit </button>
              <p>{{ message }}</p>
            </form>
    </div>
</body>
</html>
```

User Interface

Enter the Username:

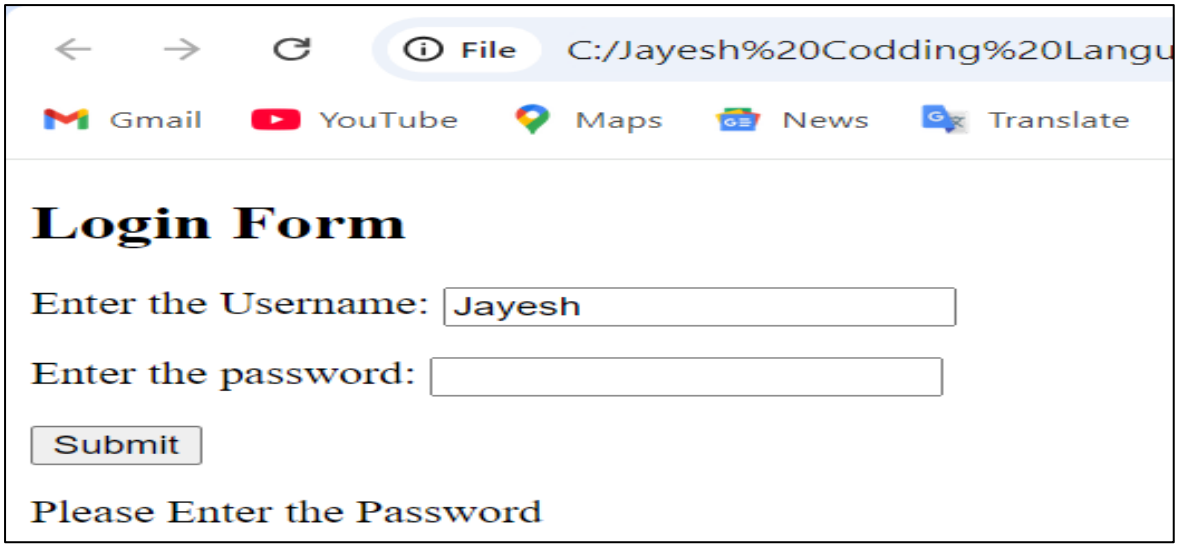
Enter the password:

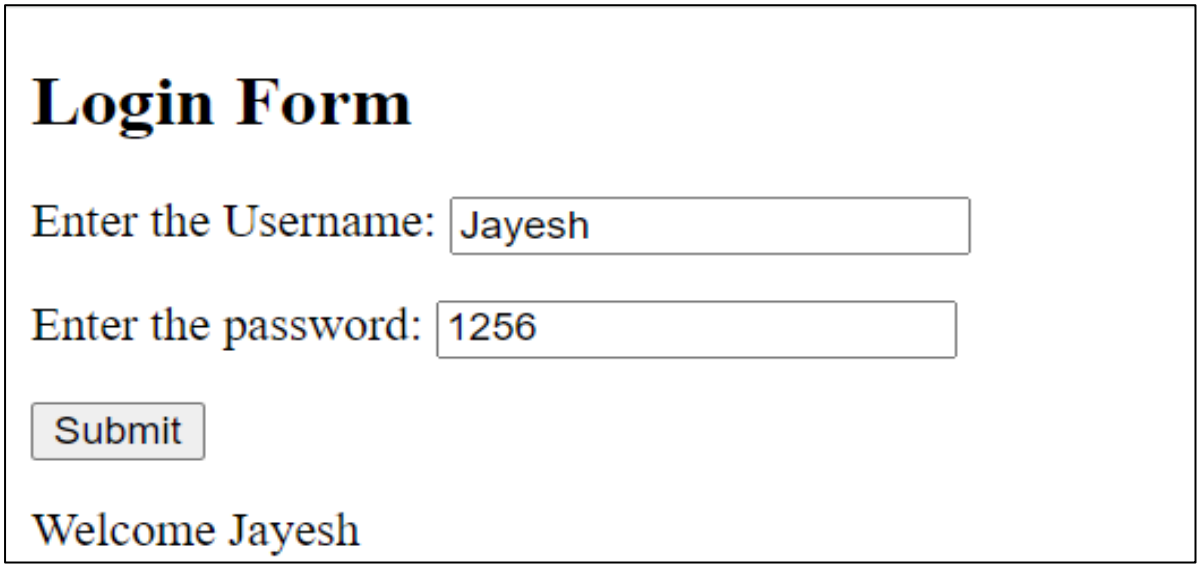
Submit

2. Develop AngularJS program to create a login form, with validation for the username and password fields.

```
<!DOCTYPE html>
<html ng-app="login-form_Demo">
<head>
    <title>login form Demo Program</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
    <script>
        var myvar = angular.module("login-form_Demo",[]);
        myvar.controller("myctrl",function($scope){

            $scope.message="";
            $scope.uservalidation="";
            $scope.passwordvalidation="";
            $scope.submit=function(){

                if($scope.username===""){
                    $scope.message="Please Enter the Username";
                }
                else if($scope.password===""){
                    $scope.message="Please Enter the Password";
                }
                else if($scope.username=="Jayesh" && $scope.password=="1256"){
                    $scope.message="Welcome Jayesh";
                }
                else{
                    $scope.message="User Invalid !! ";
                }
            };
        });
    </script>
</head>
<body>
    <div ng-controller="myctrl">

        <h2> Login Form </h2>
        <form ng-submit="submit()">
            Enter the Username:
                <input type="text" ng-model="username" ><br>
                <p>{{uservalidation}}</p>

            Enter the password:
                <input type="text" ng-model="password" ><br>
                <p>{{passwordvalidation}}</p>

             <button type="submit"> Submit </button>
              <p>{{ message }}</p>
            </form>
    </div>
</body>
</html>
```

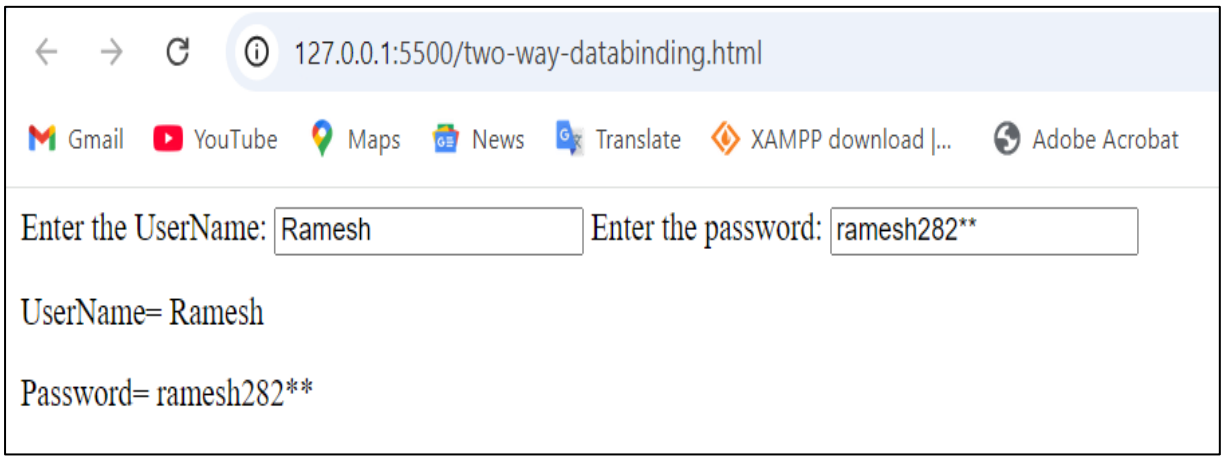
Output :


File
C:/Jayesh%20Codding%20Langu
Gmail
YouTube
Maps
News
Translate
Login Form
Enter the Username:
Jayesh
Enter the password:
Submit
Please Enter the Password


Login Form
Enter the Username:
Jayesh
Enter the password:
1256
Submit
Welcome Jayesh

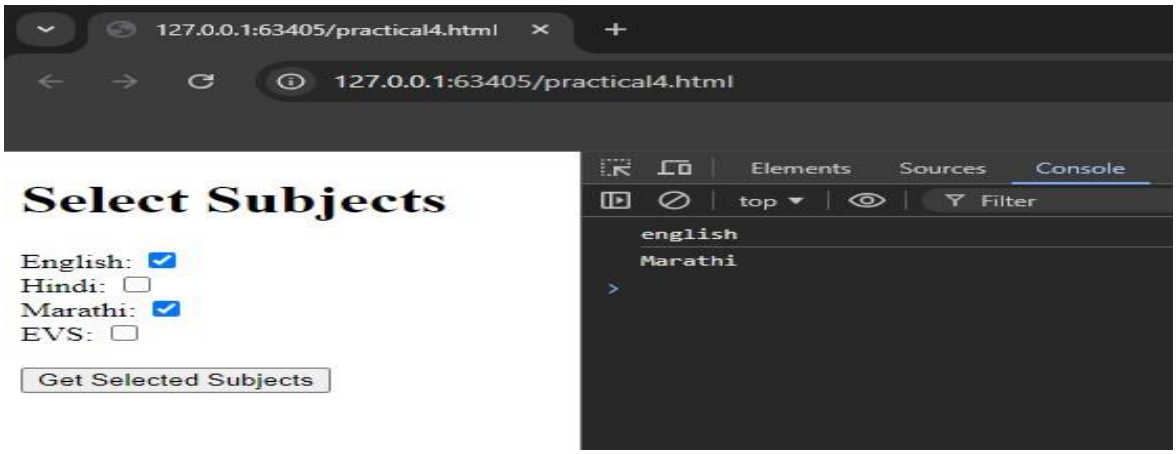
3. To demonstrate AngularJS services and data bind.

```
<!DOCTYPE html>
<html ng-app="two-way-databinding">
<head>
    <title>Two way databinding Demo Example</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
    <script>
        var myvar = angular.module("two-way-databinding",[]);
        myvar.controller("myctrl",function($scope){
                $scope.user_name="";
                $scope.password="";
        });
    </script>
</head>
<body>
    <div ng-controller="myctrl">
        Enter the UserName:
                 <input type="text" ng-model="user_name">
        Enter the password:
                  <input type="text" ng-model="password">

        <p>UserName= {{user_name}}</p>
        <p>Password= {{password}}</p>
    </div>
</body>
</html>
```



4. To display tasks in a list with checkboxes for marking completion as per user choice.

```
<!DOCTYPE HTML>
<html>
<head>
    <script src="//ajax.googleapis.com/ajax/libs/angularjs/1.2.13/angular.min.js"></script>
    <script>
        var myApp = angular.module("app", []);
        myApp.controller("controller", function ($scope) {
            $scope.english = false;
            $scope.hindi = false;
            $scope.marathi = false;
            $scope.evs = false;

            $scope.getSelectedSubjects = function () {
                var selectedSubjects = [];
                if ($scope.english) console.log("english");
                if ($scope.hindi) console.log("hindi");
                if ($scope.marathi) console.log('Marathi');
                if ($scope.evs) console.log('EVS');
                return selectedSubjects;
            }
        });
    </script>
</head>
<body>
    <h1 >Select Subjects</h1>
    <div ng-app="app">
        <div ng-controller="controller">
            <form>
                English: <input type="checkbox" ng-model="english">
                <br>
                Hindi: <input type="checkbox" ng-model="hindi">
                <br>
                Marathi: <input type="checkbox" ng-model="marathi">
                <br>
                EVS: <input type="checkbox" ng-model="evs">
                <br>
                <br>
                <button ng-click="getSelectedSubjects()">Get Selected Subjects</button>


            </form>
        </div>
    </div>
</body>
</html>
```



5. To use ng-if directive to display tasks in the UI.

```
<!DOCTYPE html>
<html ng-app="ifdemo">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <scrip src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
    <script>
        var myvar = angular.module("ifdemo",[]);
        myvar.controller("ifctrl",function($scope){
            $scope.status = true;
            $scope.show = function(){
                $scope.status = true;
            }
            $scope.hide = function() {
                $scope.status = false;
            }
        });
    </script>
</head>
<body>
    <div ng-controller="ifctrl">
        <p ng-if="status">This is my message</p>
        <button ng-click="show()">Show</button><br><br>
        <button ng-click="hide()">Hide</button>
    </div>
</body>
</html>
```

6. create database and structure in MongoDB

1. Create or Switch to the Database:

```
> use imrddb
```

2. Insert 5 Student Records into imrdcollection:

```
> db.imrdcollection.insertMany([
    { "name": "John Doe", "age": 20, "course": "BCA", "email":
"john@example.com", "fees": 50000 },
    { "name": "Jane Smith", "age": 21, "course": "BCA", "email":
"jane@example.com", "fees": 52000 },
    { "name": "Robert Brown", "age": 22, "course": "BCA", "email":
"robert@example.com", "fees": 53000 },
    { "name": "Emily Davis", "age": 20, "course": "BCA", "email":
"emily@example.com", "fees": 51000 },
    { "name": "Michael Wilson", "age": 21, "course": "BCA", "email":
"michael@example.com", "fees": 55000 }
])
```

3. View the Inserted Data:

```
> db.imrdcollection.find().pretty()
```

Output -

```
{
    "acknowledged": true,
    "insertedIds": {
        "0": ObjectId("653123a7e45b38c1d8f1a1e1"),
        "1": ObjectId("653123a7e45b38c1d8f1a1e2"),
        "2": ObjectId("653123a7e45b38c1d8f1a1e3"),
        "3": ObjectId("653123a7e45b38c1d8f1a1e4"),
        "4": ObjectId("653123a7e45b38c1d8f1a1e5")
    }
}
```

Explanation:
use imrddb: Switches to or creates the database imrddb.
db.imrdcollection.insertMany([...]): Inserts multiple student records into the collection imrdcollection.
db.imrdcollection.find().pretty(): Displays all records in the collection in a readable format.

7. To demonstrate insert, update, delete, select operations in MongoDB / To create collection and insert records in collections.

1. Insert Operation:
Inserting a single new student into the imrdcollection.

```
db.imrdcollection.insertOne({
    "name": "Alice Johnson",
    "age": 22,
    "course": "BCA",
    "email": "alice@example.com",
    "fees": 54000
})
```

Expected Output:

```
{
    "acknowledged": true,
    "insertedId": ObjectId("653124b8e45b38c1d8f1b2e6")
}
```

2. Select Operation:
To retrieve all student records from the imrdcollection:

```
db.imrdcollection.find().pretty()
```

Expected Output (shows all records):

```
{
    "_id": ObjectId("653123a7e45b38c1d8f1a1e1"),
    "name": "John Doe",
    "age": 20,
    "course": "BCA",
    "email": "john@example.com",
    "fees": 50000
}
{
    "_id": ObjectId("653123a7e45b38c1d8f1a1e2"),
    "name": "Jane Smith",
    "age": 21,
    "course": "BCA",
    "email": "jane@example.com",
    "fees": 52000
}
{
    "_id": ObjectId("653123a7e45b38c1d8f1a1e3"),
    "name": "Robert Brown",
    "age": 22,
    "course": "BCA",
    "email": "robert@example.com",
    "fees": 53000
}
{
```

```
    "_id": ObjectId("653123a7e45b38c1d8f1a1e4"),
    "name": "Emily Davis",
    "age": 20,
    "course": "BCA",
    "email": "emily@example.com",
    "fees": 51000
}
{
    "_id": ObjectId("653123a7e45b38c1d8f1a1e5"),
    "name": "Michael Wilson",
    "age": 21,
    "course": "BCA",
    "email": "michael@example.com",
    "fees": 55000
}
{
    "_id": ObjectId("653124b8e45b38c1d8f1b2e6"),
    "name": "Alice Johnson",
    "age": 22,
    "course": "BCA",
    "email": "alice@example.com",
    "fees": 54000
}
```

3. Update Operation:
To update the fees of the student John Doe to 60000:

```
db.imrdcollection.updateOne(
    { "name": "John Doe" },  // Find the student by name
    { $set: { "fees": 60000 } }  // Update the fees field
)
```

Expected Output:

```
{
    "acknowledged": true,
    "matchedCount": 1,
    "modifiedCount": 1
}
```

To verify the update:

```
db.imrdcollection.find({ "name": "John Doe" }).pretty()
```

Expected Output:

```
{
    "_id": ObjectId("653123a7e45b38c1d8f1a1e1"),
    "name": "John Doe",
    "age": 20,
    "course": "BCA",
    "email": "john@example.com",
    "fees": 60000
```

```
}
```

4. Delete Operation:

To delete the student record of Alice Johnson:

```
db.imrdcollection.deleteOne({ "name": "Alice Johnson" })
```

Expected Output:

```
{
    "acknowledged": true,
    "deletedCount": 1
}
```

// 10. To Create Student interface to stored and update the information.

// 11. To display students information reports(All, Parametized)

// 12. Implement a user interface where users can view, add, update, and delete tasks with MongoDB Database.

1. Backend - Node.js + Express (server.js)

Install the required packages:

```
npm install express mongoose body-parser cors
```

server. js

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');
const app = express();
app.use(cors());
app.use(bodyParser.json());

// MongoDB Connection
mongoose.connect('mongodb://localhost:27017/studentdb', { useNewUrlParser: true })
    .then(() => console.log('MongoDB Connected'))
    .catch((error) => console.error('MongoDB connection error:', error));

// Schema for Student
const studentSchema = new mongoose.Schema({
    name: String,
    age: Number,
    course: String,
    email: String
});
const Student = mongoose.model('Student', studentSchema);

// Routes for CRUD Operations
app.post('/api/students', async (req, res) => {
    const newStudent = new Student(req.body);
    try {
        await newStudent.save();
        res.status(201).send(newStudent);
    } catch (error) {
        res.status(500).send(error);
    }
});

app.get('/api/students', async (req, res) => {
    try {
        const students = await Student.find();
        res.status(200).send(students);
    } catch (error) {
        res.status(500).send(error);
    }
});
```

```
app.put('/api/students/:id', async (req, res) => {
    try {
        const student = await Student.findByIdAndUpdate(req.params.id, req.body, { new: true
});
        res.status(200).send(student);
    } catch (error) {
        res.status(500).send(error);
    }
});

app.delete('/api/students/:id', async (req, res) => {
    try {
        await Student.findByIdAndDelete(req.params.id);
        res.status(200).send({ message: "Student deleted" });
    } catch (error) {
        res.status(500).send(error);
    }
});

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

2. Frontend - Angular

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Management</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></scrip
t>
    <script>
        var app = angular.module('studentApp', []);

        app.controller('StudentController', function ($scope, $http) {
            $scope.students = [];
            $scope.student = {}; // Object to hold form data

            // Function to load students from the server
            $scope.loadStudents = function () {
                $http.get('http://localhost:3000/api/students')
                    .then(function (response) {
                        $scope.students = response.data;
                    }, function (error) {
                        console.error("Error loading students:", error);
                    });
```

```
            };

            // Function to add or update a student
            $scope.addStudent = function () {
                if ($scope.student._id) {
                    // If _id exists, update the student
                    $http.put(`http://localhost:3000/api/students/${$scope.student
._id}`, $scope.student)
                        .then(function (response) {
                            $scope.loadStudents();
                            $scope.student = {}; // Clear the form
                        });
                } else {
                    // Add new student
                    $http.post('http://localhost:3000/api/students',
$scope.student)
                        .then(function (response) {
                            $scope.loadStudents();
                            $scope.student = {}; // Clear the form
                        });
                }
            };

            // Function to delete a student
            $scope.deleteStudent = function (id) {
                $http.delete(`http://localhost:3000/api/students/${id}`)
                    .then(function (response) {
                        $scope.loadStudents();
                    });
            };

            // Function to edit student (populate form)
            $scope.editStudent = function (student) {
                $scope.student = angular.copy(student);
            };

            // Load students when the page loads
            $scope.loadStudents();
        });
    </script>
</head>
<body ng-app="studentApp" ng-controller="StudentController">
    <h1>Student Management</h1>

    <!-- Student Form -->
    <form ng-submit="addStudent()">
        <label>Name:</label>
        <input type="text" ng-model="student.name" placeholder="Name" required>
        <br>
        <label>Age:</label>
        <input type="number" ng-model="student.age" placeholder="Age" required>
        <br>
        <label>Course:</label>
```

```
            <input type="text" ng-model="student.course" placeholder="Course"
        required>
            <br>
            <label>Email:</label>
            <input type="email" ng-model="student.email" placeholder="Email" required>
            <br>
            <button type="submit">Add/Update Student</button>
        </form>

        <!-- Display Students -->
        <ul>
            <li ng-repeat="student in students">
                {{ student.name }} - {{ student.age }} - {{ student.course }} - {{
        student.email }}
                <button ng-click="editStudent(student)">Edit</button>
                <button ng-click="deleteStudent(student._id)">Delete</button>
            </li>
        </ul>
        </body>
        </html>
```

```
        <button type="submit">Add/Update Student</button>
    </form>

    <!-- Display Students -->
    <ul>
        <li ng-repeat="student in students">
            {{ student.name }} - {{ student.age }} - {{ student.course }} - {{ student.email
}}
            <button ng-click="editStudent(student)">Edit</button>
            <button ng-click="deleteStudent(student._id)">Delete</button>
        </li>
    </ul>
</body>
</html>
```

Run the App:

1. Open `index.html` in your browser (`http://127.0.0.1:5500/index.html` if using Live Server).
2. Ensure your backend (`server.js`) is running on `http://localhost:3000`.