For node

npm init -y
npm install express body-parser mongoose cors

For creating virtual environment

python3 -m venv venv
source venv/bin/activate
deactivate
Or
python -m venv venv
venv\Scripts\activate
deactivate

Calculator

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>TS Calculator</title>
</head>
<body>
 <h2>Simple Calculator</h2>
 <input type="number" id="num1" placeholder="First Number">
 <select id="operator">
   <option value="+">+</option>
   <option value="-">-</option>
   <option value="*">*</option>
   <option value="/">/</option>
 </select>
 <input type="number" id="num2" placeholder="Second Number">
 <button onclick="calculate()">Calculate</button>
 <p id="result"></p>

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

script.ts

```
function calculate(): void {
 const input1 = document.getElementById("num1") as HTMLInputElement;
 const input2 = document.getElementById("num2") as HTMLInputElement;
 const operator = (document.getElementById("operator") as HTMLSelectElement).value;
```

```
const result = document.getElementById("result") as HTMLParagraphElement;

const num1 = parseFloat(input1.value);
const num2 = parseFloat(input2.value);

if (isNaN(num1) || isNaN(num2)) {
    result.textContent = "Please enter valid numbers.";
    return;
}

let calculation: number;

if (operator === "+") {
    calculation = num1 + num2;
} else if (operator === "-") {
    calculation = num1 - num2;
} else if (operator === "*") {
    calculation = num1 * num2;
} else if (operator === "/") {
    if (num2 === 0) {
        result.textContent = "Cannot divide by zero.";
        return;
    }
    calculation = num1 / num2;
} else {
    result.textContent = "Invalid operator.";
    return;
}

result.textContent = `Result: ${calculation}`;
}

tsc script.ts
```

Catalog

index.html

```
<!DOCTYPE html>
<html ng-app="productApp">
<head>
 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="ProductController">
```

```
<h2>Product Catalog</h2>
<table border="1">
  <tr>
    <th>ID</th>
    <th>Name (Uppercase)</th>
    <th>Info</th>
    <th>Price (Currency)</th>
  </tr>
  <tr ng-repeat="p in products | orderBy:'price'">
    <td>{{p.id}}</td>
    <td>{{p.name | uppercase}}</td>
    <td>{{p.info}}</td>
    <td>{{p.price | currency}}</td>
  </tr>
</table>

<script>
  angular.module('productApp', []).controller('ProductController', function($scope) {
    $scope.products = [
      { id: 1, name: 'laptop', info: 'Good for work', price: 45000 },
      { id: 2, name: 'mouse', info: 'Wireless', price: 700 },
      { id: 3, name: 'keyboard', info: 'Mechanical', price: 2500 }
    ];
  });
</script>

</body>
</html>
```

Dynamic search

\Templates\index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dynamic Product Search</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <h1>Search Products</h1>
  <input type="text", id="search" placeholder="Type product name...">
  <ul id="results"></ul>
```

```
  <!--runs the function after the page is fully loaded-->
  <script>
    $(document).ready(function(){
      $("#search").on("input", function(){
        var query = $(this).val(); //gets the current value typed by the user
        $.ajax({
          url: "/search",
          data: {q: query},
          success: function(data){
            $('#results').empty();
            data.forEach(function(product){
              $('#results').append("<li>" + product.name + "</li>");
            })
          }
        })

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

app,py

```
#Dynamic Search using AJAX from product catalog

from flask import Flask, render_template, request, jsonify
#flask to create the app, render_template to load HTML, request to read user input, jsonify to return
jsonify

app = Flask(__name__)

products = [
  {"id": 1, "name": "Laptop"},
  {"id": 2, "name": "Smartphone"},
  {"id": 3, "name": "Headphones"},
  {"id": 4, "name": "Keyboard"},
  {"id": 5, "name": "Mouse"},
  {"id": 6, "name": "Speaker"},
  {"id": 7, "name": "Charger"}
]

@app.route('/')
def index():
  return render_template('index.html')
```

```
@app.route('/search')
def search():
  query = request.args.get('q', '').lower()
  results = [p for p in products if query in p['name'].lower()]
  return jsonify(results)

if __name__ == '__main__':
  app.run(debug=True)
```

Result

```
\models\Result.js
const mongoose = require('mongoose');

const resultSchema = new mongoose.Schema({
  class: String,
  prn: String,
  name: String,
  email: String,
  marks:{
     subject1: Number,
     subject2: Number,
     subject3: Number,
     subject4: Number,
     subject5: Number,
  }
});
module.exports = mongoose.model('Result', resultSchema);

\public\index.html
<!DOCTYPE html>
<html>
<head>
 <title>SEM V Result</title>
</head>
<body>
 <h2>SEM V Result Form</h2>
 <form id="resultForm">
  Name: <input type="text" id="name"><br>
  PRN: <input type="text" id="prn"><br>
  Email: <input type="email" id="email"><br>
  Class: <input type="text" id="class"><br>
  Subject1: <input type="number" id="sub1"><br>
```

```
  Subject2: <input type="number" id="sub2"><br>
  Subject3: <input type="number" id="sub3"><br>
  Subject4: <input type="number" id="sub4"><br>
  Subject5: <input type="number" id="sub5"><br>
  <button type="submit">Submit</button>
 </form>

 <h2>All Results</h2>
 <div id="results"></div>

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

\public\script.js

```
document.getElementById('resultForm').addEventListener('submit', async function (e) {
      e.preventDefault();
         const data = {
            name: document.getElementById('name').value,
            prn: document.getElementById('prn').value,
            email: document.getElementById('email').value,
            class: document.getElementById('class').value,
            marks: {
               subject1: +document.getElementById('sub1').value,
               subject2: +document.getElementById('sub2').value,
               subject3: +document.getElementById('sub3').value,
               subject4: +document.getElementById('sub4').value,
               subject5: +document.getElementById('sub5').value
            }
         };
         await fetch('/add', {
            method: 'POST',
            headers: {
               'Content-Type': 'application/json'
            },
            body: JSON.stringify(data)
         });
            alert('Result Saved!');
   location.reload();
});

async function loadResults() {
   const res = await fetch('/results');
   const data = await res.json();
```

```
  const container = document.getElementById('results');
  container.innerHTML = '';
    data.forEach(item => {
    const div = document.createElement('div');
    div.innerHTML = `<b>${item.name}</b> (${item.prn}) - ${item.class} - ${item.email} - Marks:
${Object.values(item.marks).join(', ')}`;
    container.appendChild(div);
  });
}
loadResults();
```

server.js

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

const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const Result = require('./models/result');
const app = express();

mongoose.connect('mongodb://localhost:27017/semv');

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

app.post('/add', async (req, res) => {
  const data = new Result(req.body);
  await data.save();
  res.send('Data saved successfully');
});

app.get('/results', async (req, res) => {
  const results = await Result.find();
  res.json(results);
});

app.put('/update/:id', async (req, res) => {
  await Result.findByIdAndUpdate(req.params.id, req.body);
  res.send('Data updated');
});

app.delete('/delete/:id', async (req, res) => {
  await Result.findByIdAndDelete(req.params.id);
  res.send('Data deleted');
});
```

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

Employee

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Employee Salary Calculator</title>
</head>
<body>
 <h2>Employee Final Salary Calculator</h2>

 <input type="text" id="empName" placeholder="Employee Name"><br><br>
 <input type="number" id="baseSalary" placeholder="Base Salary"><br><br>
 <select id="rating">
  <option value="10">Performance Rating: 10</option>
  <option value="5">Performance Rating: 5</option>
 </select><br><br>

 <button onclick="calculateSalary()">Calculate Salary</button>

 <p id="output"></p>

 <script src="script.js"></script> <!-- compiled TS file -->
</body>
</html>
```

script.ts

```
function calculateSalary(): void {
 const nameInput = document.getElementById("empName") as HTMLInputElement;
 const salaryInput = document.getElementById("baseSalary") as HTMLInputElement;
 const ratingSelect = document.getElementById("rating") as HTMLSelectElement;
 const output = document.getElementById("output") as HTMLParagraphElement;

 const name: string = nameInput.value;
 const baseSalary: number = parseFloat(salaryInput.value);
 const rating: number = parseInt(ratingSelect.value);

 if (!name || isNaN(baseSalary) || (rating !== 5 && rating !== 10)) {
  output.textContent = "Please enter all fields correctly.";
  return;
```

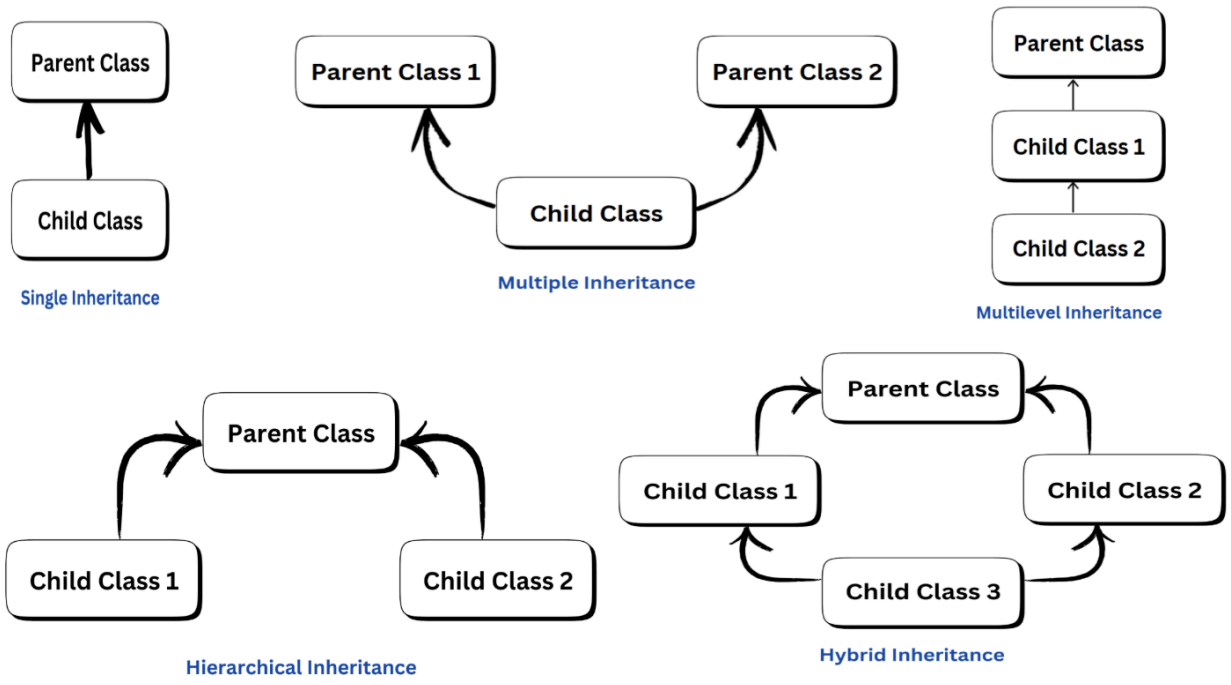
```
}

let bonusPercent: number;
if (rating === 10) {
  bonusPercent = 0.2;
} else {
  bonusPercent = 0.1;
}

const finalSalary: number = baseSalary + (baseSalary * bonusPercent);

output.textContent = `${name}'s final salary is $${finalSalary.toFixed(2)}`;
}
```

Inheritance



Single level inheritance

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <title>Single Inheritance</title>
</head>
<body>
```

```
 <h1>Single Inheritance: Dog 🐶</h1>
 <button onclick="show()">Show Output</button>
 <p id="output"></p>
 <script src="script.js"></script>
</body>
</html>
```

script.ts

```
class Animal {
 walk(): string {
   return "The animal walks.";
 }
}

class Dog extends Animal {
 make_sound(): string {
   return "The dog barks! 🐶";
 }
}

function show() {
 const dog = new Dog();
 document.getElementById("output")!.innerText =
   `${dog.walk()} ${dog.make_sound()}`;
}
```

Multiple inheritance

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <title>Multiple Inheritance</title>
</head>
<body>
 <h1>Multiple Inheritance: Penguin 🐧</h1>
 <button onclick="show()">Show Output</button>
 <p id="output"></p>
 <script src="script.js"></script>
</body>
</html>
```

script.ts

```
class Flyable {
```

```
 fly(): string {
   return "It can fly!";
 }
}

class Swimmable {
 swim(): string {
   return "It can swim!";
 }
}

class Penguin implements Flyable, Swimmable {
 fly!: () => string;
 swim!: () => string;
}

function applyMixins(derived: any, bases: any[]) {
 bases.forEach(base => {
   Object.getOwnPropertyNames(base.prototype).forEach(name => {
     derived.prototype[name] = base.prototype[name];
   });
 });
}

applyMixins(Penguin, [Flyable, Swimmable]);

function show() {
 const penguin = new Penguin();
 document.getElementById("output")!.innerText =
   `${penguin.fly()} ${penguin.swim()}`;
}
```

Multi level inheritance

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <title>Multilevel Inheritance</title>
</head>
<body>
 <h1>Multilevel Inheritance: Family 👨‍👩‍👧</h1>
 <button onclick="show()">Show Output</button>
 <p id="output"></p>
```

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

script.ts

```
class Grandparent {
 legacy(): string {
   return "Grandparent legacy passed.";
 }
}

class Parent extends Grandparent {
 guidance(): string {
   return "Parent gives guidance.";
 }
}

class Child extends Parent {
 learn(): string {
   return "Child is learning.";
 }
}

function show() {
 const child = new Child();
 document.getElementById("output")!.innerText =
   `${child.legacy()} ${child.guidance()} ${child.learn()}`;
}
```

Hierarchical inheritance

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <title>Hierarchical Inheritance</title>
</head>
<body>
 <h1>Hierarchical Inheritance: Vehicles 🚗🏍️</h1>
 <button onclick="show()">Show Output</button>
 <p id="output"></p>
 <script src="script.js"></script>
</body>
```

```
</html>
```

script.ts

```
class Vehicle {
 move(): string {
   return "Vehicle is moving.";
 }
}

class Car extends Vehicle {
 honk(): string {
   return "Car honks. 🚗";
 }
}

class Motorcycle extends Vehicle {
 rev(): string {
   return "Motorcycle revs. 🏍️";
 }
}

function show() {
 const car = new Car();
 const bike = new Motorcycle();
 document.getElementById("output")!.innerText =
   `${car.move()} ${car.honk()} | ${bike.move()} ${bike.rev()}`;
}
```

Hybrid inheritance

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <title>Hybrid Inheritance</title>
</head>
<body>
 <h1>Hybrid Inheritance: Electric Car ⚡🚗</h1>
 <button onclick="show()">Show Output</button>
 <p id="output"></p>
 <script src="script.js"></script>
</body>
</html>
```

script.ts

```
class Engine {
 startEngine(): string {
   return "Engine started.";
 }
}

class Wheels {
 roll(): string {
   return "Wheels are rolling.";
 }
}

class Car {
 drive(): string {
   return "Car is driving.";
 }
}

interface Car extends Engine, Wheels {}

applyMixins(Car, [Engine, Wheels]);

class ElectricCar extends Car {
 charge(): string {
   return "Charging battery. ⚡";
 }
}

function applyMixins(derivedCtor: any, baseCtors: any[]) {
 baseCtors.forEach(baseCtor => {
   Object.getOwnPropertyNames(baseCtor.prototype).forEach(name => {
     derivedCtor.prototype[name] = baseCtor.prototype[name];
   });
 });
}

function show() {
 const tesla = new ElectricCar();
 document.getElementById("output")!.innerText =
   `${tesla.startEngine()} ${tesla.roll()} ${tesla.drive()} ${tesla.charge()}`;
}
```

Interface

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Shapes Area Calculator</title>
 <script src="app.js" defer></script>
</head>
<body>
 <h1>Shapes Area Calculator</h1>
 <div>
   <p>Rectangle Area: <span id="rectangleArea"></span></p>
   <p>Circle Area: <span id="circleArea"></span></p>
   <p>Triangle Area: <span id="triangleArea"></span></p>
 </div>
</body>
</html>
```

app.ts

```
// Shape interface with getArea method
interface Shape {
 getArea(): number;
}

// Rectangle class implementing Shape interface
class Rectangle implements Shape {
 constructor(private width: number, private height: number) {}

 getArea(): number {
   return this.width * this.height;
 }
}

// Circle class implementing Shape interface
class Circle implements Shape {
 constructor(private radius: number) {}

 getArea(): number {
   return Math.PI * this.radius * this.radius;
 }
}
```

```
// Triangle class implementing Shape interface
class Triangle implements Shape {
 constructor(private base: number, private height: number) {}

 getArea(): number {
   return 0.5 * this.base * this.height;
 }
}

// Create instances of each shape
const rectangle = new Rectangle(10, 5);
const circle = new Circle(7);
const triangle = new Triangle(6, 4);

// Display the areas on the webpage
window.onload = () => {
 document.getElementById('rectangleArea')!.textContent = rectangle.getArea().toString();
 document.getElementById('circleArea')!.textContent = circle.getArea().toString();
 document.getElementById('triangleArea')!.textContent = triangle.getArea().toString();
};
```

Temperature

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Temperature Converter</title>
</head>
<body>
 <h2>Temperature Converter</h2>

 <label>Enter Temperature:</label>
 <input type="number" id="tempInput">

 <select id="conversionType">
   <option value="CtoF">Celsius to Fahrenheit</option>
   <option value="FtoC">Fahrenheit to Celsius</option>
 </select>

 <button onclick="convertTemp()">Convert</button>

 <p id="result"></p>
```

```
 <!-- Link to the compiled JS file -->
 <script src="converter.js"></script>
</body>
</html>
```

converter.js

```
function convertTemp() {
  var input = document.getElementById("tempInput");
  var type = document.getElementById("conversionType").value;
  var result = document.getElementById("result");
  var temp = parseFloat(input.value);
  if (isNaN(temp)) {
     result.textContent = "Please enter a valid number.";
     return;
  }
  if (type === "CtoF") {
     var fahrenheit = (temp * 9 / 5) + 32;
     result.textContent = "".concat(temp, "\u00B0C = ").concat(fahrenheit.toFixed(2), "\u00B0F");
  }
  else {
     var celsius = (temp - 32) * 5 / 9;
     result.textContent = "".concat(temp, "\u00B0F = ").concat(celsius.toFixed(2), "\u00B0C");
  }
}
```

Angular js form

Form

index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="registrationApp">
<head>
 <meta charset="UTF-8">
 <title>AngularJS Registration Form</title>
 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="FormController">

 <h2>Registration Form</h2>

 <form name="regForm" ng-submit="submitForm()" novalidate>

   <!-- Name -->
```

```
  <label>Name:</label><br>
  <input type="text" name="name" ng-model="user.name" ng-required="true" ng-minlength="3" />
  <div style="color:red" ng-show="regForm.name.$touched && regForm.name.$invalid">
    <div ng-if="regForm.name.$error.required">Name is required.</div>
    <div ng-if="regForm.name.$error.minlength">Name must be at least 3 characters.</div>
  </div>
  <br>

  <!-- Email -->
  <label>Email:</label><br>
  <input type="email" name="email" ng-model="user.email" ng-required="true" />
  <div style="color:red" ng-show="regForm.email.$touched && regForm.email.$invalid">
    <div ng-if="regForm.email.$error.required">Email is required.</div>
    <div ng-if="regForm.email.$error.email">Invalid email format.</div>
  </div>
  <br>

  <!-- Password -->
  <label>Password (6 digits):</label><br>
  <input type="password" name="password" ng-model="user.password"
         ng-required="true"
         ng-pattern="/^\d{6}$/"
         />
  <div style="color:red" ng-show="regForm.password.$touched && regForm.password.$invalid">
    <div ng-if="regForm.password.$error.required">Password is required.</div>
    <div ng-if="regForm.password.$error.pattern">Password must be exactly 6 digits.</div>
  </div>
  <br>

  <!-- Submit -->
  <button type="submit" ng-disabled="regForm.$invalid">Register</button>

</form>

<div ng-if="submitted">
  <h3 style="color:green;">Form submitted successfully!</h3>
  <p><strong>Name:</strong> {{ user.name }}</p>
  <p><strong>Email:</strong> {{ user.email }}</p>
  <p><strong>Password:</strong> {{ user.password }}</p>
</div>

<script>
  angular.module('registrationApp', [])
    .controller('FormController', function ($scope) {
```

```
    $scope.user = {};
    $scope.submitted = false;

    $scope.submitForm = function () {
      if ($scope.regForm.$valid) {
        $scope.submitted = true;
      }
    };
  });
</script>

</body>
</html>
```

Form mongo

index.html

```
<!DOCTYPE html>
<html ng-app="registrationApp">
<head>
 <meta charset="UTF-8">
 <title>AngularJS Registration</title>
 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="RegistrationController">

 <h2>Registration Form</h2>
 <form name="regForm" ng-submit="register()" novalidate>
   <label>Name:</label>
   <input type="text" name="name" ng-model="user.name" ng-required="true" />
   <span ng-show="regForm.name.$touched && regForm.name.$invalid">Name is
required</span><br><br>

   <label>Email:</label>
   <input type="email" name="email" ng-model="user.email" ng-required="true" />
   <span ng-show="regForm.email.$touched && regForm.email.$invalid">Valid email
required</span><br><br>

   <label>Password:</label>
   <input type="password" name="password" ng-model="user.password" ng-required="true"
        ng-pattern="/^\d{6}$/" />
   <span ng-show="regForm.password.$touched && regForm.password.$error.required">Password is
required</span>
```

```
  <span ng-show="regForm.password.$error.pattern">Password must be exactly 6
digits</span><br><br>

  <button type="submit" ng-disabled="regForm.$invalid">Register</button>
  <p>{{ message }}</p>
 </form>

 <script>
  const app = angular.module('registrationApp', []);
  app.controller('RegistrationController', function ($scope, $http) {
    $scope.user = {};
    $scope.message = '';

    $scope.register = function () {
      $http.post('/register', $scope.user)
        .then(function (response) {
          $scope.message = '✅ Registration successful!';
          $scope.user = {};
          $scope.regForm.$setPristine();
          $scope.regForm.$setUntouched();
        }, function (error) {
          $scope.message = '❌ Registration failed!';
        });
    };
  });
 </script>
</body>
</html>

server.js
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const path = require('path');
const bodyParser = require('body-parser');

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

// Middleware
app.use(cors());
app.use(bodyParser.json());
app.use(express.static(path.join(__dirname, 'public')));
```

```
// MongoDB Connection
mongoose.connect('mongodb://127.0.0.1:27017/registrationDB', {
 useNewUrlParser: true,
 useUnifiedTopology: true
}).then(() => console.log('✅ Connected to MongoDB'))
 .catch(err => console.error('❌ MongoDB Error:', err));

// Mongoose Schema
const UserSchema = new mongoose.Schema({
 name: String,
 email: String,
 password: String
});

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

// API Endpoint
app.post('/register', async (req, res) => {
 const { name, email, password } = req.body;
 if (!name || !email || !password) return res.status(400).send('Missing fields');
 try {
   const user = new User({ name, email, password });
   await user.save();
   res.status(201).send('✅ Registered');
 } catch (err) {
   res.status(500).send('❌ Registration Failed');
 }
});

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

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

Library system

index.html

```
<!-- index.html -->
<!DOCTYPE html>
```

```
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <title>Library System</title>
</head>
<body>
 <h2>📚 Library Management System</h2>

 <label>User ID: <input type="number" id="userId" /></label><br><br>
 <label>Book ID: <input type="number" id="bookId" /></label><br><br>

 <button id="issueBtn">Issue Book</button>
 <button id="returnBtn">Return Book</button>
 <button id="listBtn">List Issued Books</button>

 <div id="output" style="margin-top: 20px;"></div>

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

app.ts

```
// app.ts

class Book {
 constructor(
   public id: number,
   public title: string,
   public author: string,
   public isIssued: boolean = false
 ) {}
}

class User {
 constructor(
   public id: number,
   public name: string,
   public borrowedBooks: number[] = []
 ) {}
}

class Transaction {
 constructor(private books: Book[], private users: User[]) {}
```

```
 issueBook(userId: number, bookId: number): string {
   const user = this.users.find(u => u.id === userId);
   const book = this.books.find(b => b.id === bookId);

   if (!user) return "Error: User not found.";
   if (!book) return "Error: Book not found.";
   if (book.isIssued) return "Error: Book is already issued.";

   book.isIssued = true;
   user.borrowedBooks.push(bookId);
   return `Book "${book.title}" issued to ${user.name}.`;
 }

 returnBook(userId: number, bookId: number): string {
   const user = this.users.find(u => u.id === userId);
   const book = this.books.find(b => b.id === bookId);

   if (!user) return "Error: User not found.";
   if (!book) return "Error: Book not found.";
   if (!book.isIssued) return "Error: Book was not issued.";

   book.isIssued = false;
   user.borrowedBooks = user.borrowedBooks.filter(id => id !== bookId);
   return `Book "${book.title}" returned by ${user.name}.`;
 }

 listIssuedBooks(): string[] {
   const issued = this.books.filter(book => book.isIssued);
   if (issued.length === 0) return ["📚 No books currently issued."];

   return ["📖 Issued Books:"].concat(
     issued.map(book => `- ${book.title} by ${book.author}`)
   );
 }
}

// Setup sample data
const books: Book[] = [
 new Book(1, "1984", "George Orwell"),
 new Book(2, "The Great Gatsby", "F. Scott Fitzgerald"),
 new Book(3, "To Kill a Mockingbird", "Harper Lee")
];

const users: User[] = [
```

```
 new User(1, "Alice"),
 new User(2, "Bob")
];

const library = new Transaction(books, users);

// DOM Elements
const userIdInput = document.getElementById("userId") as HTMLInputElement;
const bookIdInput = document.getElementById("bookId") as HTMLInputElement;
const outputDiv = document.getElementById("output") as HTMLDivElement;

(document.getElementById("issueBtn") as HTMLButtonElement).onclick = () => {
 const userId = parseInt(userIdInput.value);
 const bookId = parseInt(bookIdInput.value);
 outputDiv.textContent = library.issueBook(userId, bookId);
};

(document.getElementById("returnBtn") as HTMLButtonElement).onclick = () => {
 const userId = parseInt(userIdInput.value);
 const bookId = parseInt(bookIdInput.value);
 outputDiv.textContent = library.returnBook(userId, bookId);
};

(document.getElementById("listBtn") as HTMLButtonElement).onclick = () => {
 const issuedBooks = library.listIssuedBooks();
 outputDiv.innerHTML = issuedBooks.map(line => `<div>${line}</div>`).join("");
};
```

tsconfig.json

```
{
 "compilerOptions": {
   "target": "ES6",
   "module": "none",
   "outFile": "app.js",
   "strict": true
 },
 "include": ["app.ts"]
}
```

SPA

index.html

```
<!DOCTYPE html>
```

```
<html ng-app="myApp">
<head>
 <meta charset="utf-8">
 <title>Simple AngularJS SPA</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>
  var app = angular.module("myApp", ["ngRoute"]);

  app.config(function($routeProvider) {
    $routeProvider
      .when("/", {
        template: "<h2>Home</h2><p>Welcome to the Home page!</p>"
      })
      .when("/about", {
        template: "<h2>About</h2><p>This is the About page.</p>"
      })
      .when("/contact", {
        template: "<h2>Contact</h2><p>Contact us at contact@example.com</p>"
      })
      .otherwise({
        redirectTo: "/"
      });
  });
 </script>

 <style>
  nav a {
    margin-right: 15px;
    text-decoration: none;
    color: blue;
  }
 </style>
</head>
<body>

 <nav>
  <a href="#!/">Home</a>
  <a href="#!/about">About</a>
  <a href="#!/contact">Contact</a>
 </nav>

 <hr>
```

```
 <div ng-view></div>

</body>
</html>
```

Task Manager

index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Task Manager</title>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.8.2/angular.min.js"></script>
 </head>
 <body ng-app="taskApp" ng-controller="taskController">
  <div class="container">
   <h1>Task Manager</h1>

   <input
    type="text"
    ng-model="taskTitle"
    placeholder="Enter task"
   />
   <button ng-click="addTask()">Add Task</button>

   <ul>
    <li ng-repeat="task in tasks">
     <span
      ng-click="toggleCompletion(task)"
      ng-style="{'text-decoration': task.completed ? 'line-through' : 'none'}"
     >
      {{ task.title }}
     </span>
     <button ng-click="deleteTask(task)">Delete</button>
    </li>
   </ul>
  </div>

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

app.ts

```
/// <reference path="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.8.2/angular.min.js" />

var app = angular.module("taskApp", []);

// Task service
app.service('taskService', function () {
 var tasks = [];

 this.getTasks = function () {
  return tasks;
 };

 this.addTask = function (title) {
  var newTask = {
   id: Date.now(),
   title: title,
   completed: false,
  };
  tasks.push(newTask);
 };

 this.toggleTaskCompletion = function (task) {
  task.completed = !task.completed;
 };

 this.deleteTask = function (task) {
  var index = tasks.indexOf(task);
  if (index !== -1) {
   tasks.splice(index, 1);
  }
 };
});

// Task manager controller
app.controller("taskController", function ($scope, taskService) {
 $scope.tasks = taskService.getTasks();
 $scope.taskTitle = '';

 $scope.addTask = function () {
  if ($scope.taskTitle) {
   taskService.addTask($scope.taskTitle);
   $scope.taskTitle = '';
  }
```

```
  };

  $scope.toggleCompletion = function (task) {
    taskService.toggleTaskCompletion(task);
  };

  $scope.deleteTask = function (task) {
    taskService.deleteTask(task);
  };
});
```