***** SLIP 1 *****

**** Q1) Write an AngularJS script for addition of two numbers using ng-init, ng-model & ng-bind. And also demonstrate ng-show, ng-disabled, ng-click directives on button component.**

==>

☑ STEP 1 — Create Project

```
ng new slip1_1
cd slip1_1
```

☑ STEP 2 — Update app.module.ts
(Path: src/app/app.module.ts)
Replace everything with:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, FormsModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP 3 — Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  num1: number = 0;
  num2: number = 0;
  result: number = 0;
  showResult: boolean = false;
  disableButton: boolean = false;

  add() {
    this.result = this.num1 + this.num2;
    this.showResult = true;
    this.disableButton = true;
  }
}
```

☑ STEP 4 — Update app.component.html
(Path: src/app/app.component.html)
Replace everything with:

```
<h2>Add Two Numbers</h2>

<input type="number" [(ngModel)]="num1">
<br><br>

<input type="number" [(ngModel)]="num2">
<br><br>

<button (click)="add()" [disabled]="disableButton">
  Compute Sum
</button>

<h3 *ngIf="showResult">
  Result: {{ result }}
</h3>
```

☑ STEP 5 — Run Project
ng serve -o

** Q2) Create a Node.js application that reads data from multiple files asynchronously using promises and async/await.
==>
☑ STEP 1 — Create project folder
mkdir slip1_2
cd slip1_2

☑ STEP 2 — Create files
Create two sample files using Notepad:
file1.txt
Hello from file 1

file2.txt
Hello from file 2

Save both in slip1_2 folder.

☑ STEP 3 — Create main JS file
Create file: app.js

```
const fs = require('fs').promises;

async function readFiles() {
   try {
      const file1 = fs.readFile('file1.txt', 'utf8');
      const file2 = fs.readFile('file2.txt', 'utf8');

      const results = await Promise.all([file1, file2]);

      console.log("File 1 Content:\n", results[0]);
```

```
    console.log("File 2 Content:\n", results[1]);

  } catch (err) {
    console.error("Error:", err);
  }
}

readFiles();
```

☑ STEP 4 — Run program
In terminal:

```
node app.js
```

*** SLIP 2 ***

** Q1) Write an AngularJS script to print details of bank (bank name, MICR code, IFC code, address etc.) in tabular form using ng-repeat.
==>
☑ STEP-1 → Create Project

```
ng new slip2_1
cd slip2_1
```

☑ STEP-2 → Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  banks = [
    { name: 'HDFC Bank', micr: '400240002', ifsc: 'HDFC0000123', address: 'Mumbai' },
    { name: 'SBI Bank', micr: '400002003', ifsc: 'SBIN0000456', address: 'Pune' },
    { name: 'ICICI Bank', micr: '400229999', ifsc: 'ICIC0000789', address: 'Delhi' }
  ];

}
```

☑ STEP-3 → Update app.component.html
(Path: src/app/app.component.html)
Replace everything with:

```
<h2>Bank Details</h2>

<table border="1" cellpadding="8">
  <tr>
    <th>Bank Name</th>
    <th>MICR Code</th>
```

```
    <th>IFSC Code</th>
    <th>Address</th>
  </tr>

  <tr *ngFor="let b of banks">
    <td>{{ b.name }}</td>
    <td>{{ b.micr }}</td>
    <td>{{ b.ifsc }}</td>
    <td>{{ b.address }}</td>
  </tr>
</table>
```

☑ STEP-4 → app.module.ts
(Only if not modified earlier, otherwise skip)
(Path: src/app/app.module.ts)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-5 → Run in Terminal

```
ng serve -o
```

** Q2) Create a simple Angular application that fetches data from an API using HttpClient. Implement an Observable to fetch data from an API endpoint.
==>

☑ STEP-1 → Create Project

```
ng new slip2_2
cd slip2_2
```

☑ STEP-2 → Add HttpClientModule
Open src/app/app.module.ts
Replace with:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    HttpClientModule
  ],
```

```
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-3 → Create Service

In terminal:

```
ng generate service api
```

Open: src/app/api.service.ts

Replace with:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ApiService {

  constructor(private http: HttpClient) { }

  getData(): Observable<any> {
    return this.http.get('https://jsonplaceholder.typicode.com/users');
  }
}
```

☑ STEP-4 → Update app.component.ts

Open: src/app/app.component.ts

Replace with:

```
import { Component, OnInit } from '@angular/core';
import { ApiService } from './api.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {

  users: any[] = [];

  constructor(private api: ApiService) { }

  ngOnInit(): void {
    this.api.getData().subscribe((data) => {
      this.users = data;
    });
  }
}
```

☑ STEP-5 → Update app.component.html

Open: src/app/app.component.html
Replace with:

```
<h2>API Data</h2>

<ul>
  <li *ngFor="let user of users">
    {{ user.name }} - {{ user.email }}
  </li>
</ul>
```

☑ STEP-6 → Run Project

```
ng serve -o
```

***** SLIP 3 *****

**** Q1) Write an AngularJS script to display list of games stored in an array on click of button using ng-click and also demonstrate ng-init, ng-bind directive of AngularJS.**
==>

☑ STEP-1 → Create Project

```
ng new slip3_1
cd slip3_1
```

☑ STEP-2 → Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  // ng-init equivalent
  games: string[] = ["Cricket", "Football", "Hockey", "Badminton"];
  showList: boolean = false;

  // ng-click equivalent
  displayGames() {
    this.showList = true;
  }
}
```

☑ STEP-3 → Update app.component.html
(Path: src/app/app.component.html)
Replace everything with:

```
<h2>Games List</h2>

<button (click)="displayGames()">
```

```
  Show Games
</button>

<ul *ngIf="showList">
  <li *ngFor="let g of games">
    {{ g }}   <!-- ng-bind equivalent -->
  </li>
</ul>
```

☑ STEP-4 → app.module.ts
(Only if not added already)
(Path: src/app/app.module.ts)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-5 → Run Project

```
ng serve -o
```

** Q2) Find a company with a workforce greater than 30 in the array (use find by id method)
==>

☑ STEP-1 → Create Project

```
ng new slip3_2
cd slip3_2
```

☑ STEP-2 → Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  companies = [
    { id: 1, name: "Google", workforce: 25 },
    { id: 2, name: "Amazon", workforce: 45 },
    { id: 3, name: "Microsoft", workforce: 32 },
    { id: 4, name: "Infosys", workforce: 18 }
  ];
```

```
  result: any = null;

  findCompany() {
    this.result = this.companies.find(c => c.workforce > 30);
  }
}
```

☑ STEP-3 → Update app.component.html
(Path: src/app/app.component.html)
Replace with:

```
<h2>Find Company With Workforce > 30</h2>

<button (click)="findCompany()">
  Find Company
</button>

<div *ngIf="result">
  <h3>Company Found:</h3>
  <p>Name: {{ result.name }}</p>
  <p>Workforce: {{ result.workforce }}</p>
</div>

<div *ngIf="!result">
  <p>No company found yet.</p>
</div>
```

☑ STEP-4 → app.module.ts
(if already same, no need to change)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-5 → Run Project

```
ng serve -o
```

*** SLIP 4 ***

** Q1) Fetch the details using ng-repeat in AngularJS
==>
☑ STEP-1 → Create Project

```
ng new slip4_1
cd slip4_1
```

☑ STEP-2 → Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  // Sample details to fetch (ng-init equivalent)
  details = [
    { id: 1, name: 'Rahul', city: 'Mumbai' },
    { id: 2, name: 'Sneha', city: 'Pune' },
    { id: 3, name: 'Amit', city: 'Delhi' }
  ];
}
```

☑ STEP-3 → Update app.component.html
(Path: src/app/app.component.html)
Replace with:

```
<h2>Details List</h2>

<table border="1" cellpadding="8">
  <tr>
    <th>ID</th>
    <th>Name</th>
    <th>City</th>
  </tr>

  <tr *ngFor="let d of details">
    <td>{{ d.id }}</td>
    <td>{{ d.name }}</td>
    <td>{{ d.city }}</td>
  </tr>
</table>
```

☑ STEP-4 → app.module.ts
(if already same, no changes needed)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
```

```
})
export class AppModule { }
```

☑ STEP-5 → Run

```
ng serve -o
```

** Q2) Express.js application to include middleware for parsing request bodies (e.g., JSON, form data) and validating input data.

==>

☑ STEP-1 → Create folder

```
mkdir slip4_2
cd slip4_2
```

☑ STEP-2 → Initialize project

```
npm init -y
```

☑ STEP-3 → Install Express

```
npm install express
```

☑ STEP-4 → Create server file

Create file: server.js

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

// ------ MIDDLEWARE FOR JSON + FORM DATA ------
app.use(express.json());          // parse JSON
app.use(express.urlencoded({extended: true})); // parse form data

// ------ VALIDATION MIDDLEWARE ------
function validateUser(req, res, next) {
    const { name, age } = req.body;

    if (!name || !age) {
        return res.status(400).json({ error: "Name and Age are required" });
    }

    next(); // allow request to continue
}

// ------ ROUTE WITH VALIDATION ------
app.post('/user', validateUser, (req, res) => {
    res.json({
        message: "User data received successfully",
        data: req.body
    });
});

// ------ START SERVER ------
app.listen(3000, () => {
    console.log("Server running on port 3000");
});
```

☑ STEP-5 → Run
node server.js

*** SLIP 5 ***

** Q1) Create a simple Angular component that takes input data and displays it.
==>
☑ STEP-1 → Create Project
ng new slip5_1
cd slip5_1

☑ STEP-2 → Update app.module.ts
(Path: src/app/app.module.ts)
Replace everything with:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-3 → Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  userInput: string = "";
  displayText: string = "";

  showData() {
    this.displayText = this.userInput;
  }
}
```

☑ STEP-4 → Update app.component.html

(Path: src/app/app.component.html)
Replace everything with:

```
<h2>Input Display Component</h2>

<input type="text" [(ngModel)]="userInput" placeholder="Enter something">
<br><br>

<button (click)="showData()">Show</button>

<h3 *ngIf="displayText">
  You Entered: {{ displayText }}
</h3>
```

☑ STEP-5 → Run

```
ng serve -o
```

**** Q2) Implement a simple server using Node.js.**
==>

☑ STEP-1 → Create folder

```
mkdir slip5_2
cd slip5_2
```

☑ STEP-2 → Initialize project

```
npm init -y
```

☑ STEP-3 → Create server.js
Create a file named server.js and paste this:

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

const server = http.createServer((req, res) => {
   res.writeHead(200, { "Content-Type": "text/plain" });
   res.end("Node.js Simple Server Running");
});

server.listen(3000, () => {
   console.log("Server running on port 3000");
});
```

☑ STEP-4 → Run

```
node server.js
```

***** SLIP 6 *****

**** Q1) Develop an Express.js application that defines routes for Create and Read operations on a resource (products).**
==>

☑ STEP-1 → Create folder

```
mkdir slip6_1
cd slip6_1
```

☑ STEP-2 → Init project
npm init -y

☑ STEP-3 → Install Express
npm install express

☑ STEP-4 → Create server.js
Create file: server.js
Paste this:

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

app.use(express.json()); // parse JSON body

// TEMP PRODUCT STORAGE
let products = [];

// ---------------- CREATE (POST) ----------------
app.post('/products', (req, res) => {
   const product = req.body;
   products.push(product);

   res.json({
      message: "Product created",
      data: product
   });
});

// ---------------- READ (GET) ----------------
app.get('/products', (req, res) => {
   res.json(products);
});

// ---------------- START SERVER ----------------
app.listen(3000, () => {
   console.log("Server running on port 3000");
});
```

☑ STEP-5 → Run
node server.js

✔ Create Product
```
curl -X POST http://localhost:3000/products \
-H "Content-Type: application/json" \
-d "{\"id\":1, \"name\":\"Mobile\", \"price\":20000}"
```

✔ Read Products
```
curl http://localhost:3000/products
```

** Q2) Find a company with a workforce greater than 30 in the array. (Using find by id method)
==>

☑ STEP-1 → Create Project

```
ng new slip6_2
cd slip6_2
```

☑ STEP-2 → Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  companies = [
    { id: 1, name: "Google", workforce: 25 },
    { id: 2, name: "Amazon", workforce: 45 },
    { id: 3, name: "Microsoft", workforce: 32 },
    { id: 4, name: "Infosys", workforce: 18 }
  ];

  result: any = null;

  findCompanyById() {
    // Using .find() but checking workforce > 30
    this.result = this.companies.find(c => c.workforce > 30);
  }
}
```

☑ STEP-3 → Update app.component.html
(Path: src/app/app.component.html)
Replace everything with:

```
<h2>Find Company With Workforce > 30</h2>

<button (click)="findCompanyById()">
  Find Company
</button>

<div *ngIf="result">
  <h3>Company Found:</h3>
  <p>Name: {{ result.name }}</p>
  <p>Workforce: {{ result.workforce }}</p>
</div>

<div *ngIf="!result">
  <p>No company found yet.</p>
</div>
```

☑ STEP-4 → app.module.ts
(If already same, no changes needed)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-5 → Run Project

```
ng serve -o
```

*** SLIP 7 ***

** Q1) Create a Node.js application that reads data from multiple files asynchronously using promises and async/await.

==>

☑ STEP 1 — Create project folder

```
mkdir slip7_1
cd slip7_1
```

☑ STEP 2 — Create files

Create two sample files using Notepad:

```
file1.txt
Hello from file 1

file2.txt
Hello from file 2
```

Save both in slip1_2 folder.

☑ STEP 3 — Create main JS file

Create file: app.js

```
const fs = require('fs').promises;

async function readFiles() {
   try {
      const file1 = fs.readFile('file1.txt', 'utf8');
      const file2 = fs.readFile('file2.txt', 'utf8');

      const results = await Promise.all([file1, file2]);

      console.log("File 1 Content:\n", results[0]);
      console.log("File 2 Content:\n", results[1]);

   } catch (err) {
      console.error("Error:", err);
   }
```

```
}

readFiles();
```

☑ STEP 4 — Run program
In terminal:

```
node app.js
```

** Q2) Develop an Express.js application that defines routes for Create and Read operations on a resource (User).
==>
☑ STEP-1 → Create folder

```
mkdir slip7_2
cd slip7_2
```

☑ STEP-2 → Init project

```
npm init -y
```

☑ STEP-3 → Install Express

```
npm install express
```

☑ STEP-4 → Create server.js
Create file: server.js
Paste this exact code:

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

app.use(express.json());   // parse JSON body

// TEMP USER STORAGE
let users = [];

// ---------------- CREATE (POST) ----------------
app.post('/users', (req, res) => {
   const user = req.body;
   users.push(user);

   res.json({
      message: "User created",
      data: user
   });
});

// ---------------- READ (GET) ----------------
app.get('/users', (req, res) => {
   res.json(users);
});

// ---------------- START SERVER ----------------
app.listen(3000, () => {
   console.log("Server running on port 3000");
});
```

☑ STEP-5 → Run

```
node server.js
```

✔ Create User

```
curl -X POST http://localhost:3000/users \
-H "Content-Type: application/json" \
-d "{\"id\":1, \"name\":\"Rahul\", \"email\":\"rahul@gmail.com\"}"
```

✔ Read Users

```
curl http://localhost:3000/users
```

*** SLIP 8 ***

** Q1) Create a simple Angular application that fetches data from an API using HttpClient. Implement an Observable to fetch data from an API endpoint.
==>

☑ STEP-1 → Create Project

```
ng new slip2_2
cd slip2_2
```

☑ STEP-2 → Add HttpClientModule

Open src/app/app.module.ts
Replace with:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-3 → Create Service

In terminal:

```
ng generate service api
```

Open: src/app/api.service.ts
Replace with:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
```

```
  providedIn: 'root'
})
export class ApiService {

  constructor(private http: HttpClient) { }

  getData(): Observable<any> {
    return this.http.get('https://jsonplaceholder.typicode.com/users');
  }
}
```

☑ STEP-4 → Update app.component.ts
Open: src/app/app.component.ts
Replace with:

```
import { Component, OnInit } from '@angular/core';
import { ApiService } from './api.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {

  users: any[] = [];

  constructor(private api: ApiService) { }

  ngOnInit(): void {
    this.api.getData().subscribe((data) => {
      this.users = data;
    });
  }
}
```

☑ STEP-5 → Update app.component.html
Open: src/app/app.component.html
Replace with:

```
<h2>API Data</h2>

<ul>
  <li *ngFor="let user of users">
    {{ user.name }} - {{ user.email }}
  </li>
</ul>
```

☑ STEP-6 → Run Project

```
ng serve -o
```

** Q2) Develop an Express.js application that defines routes for Create, Update operations on a resource (Employee).

==>

☑ STEP-1 → Create folder

```
mkdir slip8_2
cd slip8_2
```

☑ STEP-2 → Init project

```
npm init -y
```

☑ STEP-3 → Install Express

```
npm install express
```

☑ STEP-4 → Create server.js

Create file: server.js

Paste this exact code:

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

app.use(express.json());  // parse JSON body

// TEMP EMPLOYEE STORAGE
let employees = [];

// ---------------- CREATE (POST) ----------------
app.post('/employees', (req, res) => {
   const emp = req.body;
   employees.push(emp);

   res.json({
      message: "Employee created",
      data: emp
   });
});

// ---------------- UPDATE (PUT) ----------------
app.put('/employees/:id', (req, res) => {
   const empId = parseInt(req.params.id);
   const newData = req.body;

   const index = employees.findIndex(e => e.id === empId);

   if (index === -1) {
      return res.status(404).json({ error: "Employee not found" });
   }

   employees[index] = { ...employees[index], ...newData };

   res.json({
      message: "Employee updated",
      data: employees[index]
   });
});

// ---------------- START SERVER ----------------
```

```
app.listen(3000, () => {
   console.log("Server running on port 3000");
});
```

☑ STEP-5 → Run

```
node server.js
```

🎯 OPTIONAL TEST COMMANDS (ONLY IF TEACHER ASKS)

✔ Create Employee

```
curl -X POST http://localhost:3000/employees \
-H "Content-Type: application/json" \
-d "{\"id\":1, \"name\":\"Rahul\", \"salary\":30000}"
```

✔ Update Employee

```
curl -X PUT http://localhost:3000/employees/1 \
-H "Content-Type: application/json" \
-d "{\"salary\":45000}"
```

*** SLIP 9 ***

** Q1) Find a company with a workforce greater than 30 in the array. (Using find by id method).

==>

☑ STEP-1 → Create Project

```
ng new slip9_1
cd slip9_1
```

☑ STEP-2 → Update app.component.ts

(Path: src/app/app.component.ts)

Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  companies = [
    { id: 1, name: "Google", workforce: 22 },
    { id: 2, name: "Amazon", workforce: 45 },
    { id: 3, name: "Microsoft", workforce: 31 },
    { id: 4, name: "Infosys", workforce: 18 }
  ];

  result: any = null;

  findCompany() {
    this.result = this.companies.find(c => c.workforce > 30);
  }
}
```

☑ STEP-3 → Update app.component.html
(Path: src/app/app.component.html)
Replace with:

```
<h2>Find Company With Workforce > 30</h2>

<button (click)="findCompany()">
  Find Company
</button>

<div *ngIf="result">
  <h3>Company Found:</h3>
  <p>Name: {{ result.name }}</p>
  <p>Workforce: {{ result.workforce }}</p>
</div>

<div *ngIf="!result">
  <p>No company found yet.</p>
</div>
```

☑ STEP-4 → app.module.ts
(No change needed if same already)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-5 → Run Project

```
ng serve -o
```

** Q2) Create Express.js application to include middleware for parsing request bodies (e.g., JSON, form data) and validating input data. Send appropriate JSON responses for success and error cases.
==>
☑ STEP-1 → Create Folder

```
mkdir slip9_2
cd slip9_2
```

☑ STEP-2 → Initialize Project

```
npm init -y
```

☑ STEP-3 → Install Express

```
npm install express
```

☑ STEP-4 → Create server.js
Create file server.js and paste exactly this:

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

// ------------ MIDDLEWARE (JSON + Form Data) ------------
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// ------------ VALIDATION MIDDLEWARE ------------
function validateInput(req, res, next) {
   const { name, age } = req.body;

   if (!name || !age) {
      return res.status(400).json({
         success: false,
         message: "Validation failed: name and age are required"
      });
   }

   next();
}

// ------------ ROUTE USING VALIDATION ------------
app.post('/user', validateInput, (req, res) => {
   res.json({
      success: true,
      message: "Data received successfully",
      data: req.body
   });
});

// ------------ SERVER START ------------
app.listen(3000, () => {
   console.log("Server running on port 3000");
});
```

☑ STEP-5 → Run

```
node server.js
```

➤ JSON Request

```
curl -X POST http://localhost:3000/user \
-H "Content-Type: application/json" \
-d "{\"name\":\"Rahul\", \"age\":22}"
```

➤ Form Data Request

```
curl -X POST http://localhost:3000/user \
-d "name=Rahul&age=22"
```

➤ Error Case

```
curl -X POST http://localhost:3000/user \
-H "Content-Type: application/json" \
-d "{\"name\":\"Rahul\"}"
```

***** SLIP 10 *****

**** Q1) Implement a simple server using Node.js.**
==>
☑ STEP-1 → Create folder
mkdir slip5_2
cd slip5_2

☑ STEP-2 → Initialize project
npm init -y

☑ STEP-3 → Create server.js
Create a file named server.js and paste this:

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

const server = http.createServer((req, res) => {
   res.writeHead(200, { "Content-Type": "text/plain" });
   res.end("Node.js Simple Server Running");
});

server.listen(3000, () => {
   console.log("Server running on port 3000");
});
```

☑ STEP-4 → Run
node server.js

🎯 To show output to teacher
Open browser →
http://localhost:3000

**** Q2) Extend the previous Express.js application to include middleware for parsing request bodies (e.g., JSON, form data) and validating input data. Send appropriate JSON responses for success and error cases.**
==>
☑ STEP-1 → Create folder
mkdir slip10_2
cd slip10_2

☑ STEP-2 → Initialize project
npm init -y

☑ STEP-3 → Install Express
npm install express

☑ STEP-4 → Create server.js
Create server.js and paste this EXACT code:

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

```
// ----------- MIDDLEWARE (JSON + FORM DATA) -----------
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// ----------- VALIDATION MIDDLEWARE -----------
function validateInput(req, res, next) {
   const { id, name, salary } = req.body;

   if (!id || !name || !salary) {
      return res.status(400).json({
         success: false,
         message: "Validation failed: id, name, salary are required"
      });
   }

   next();
}

// ----------- CREATE ROUTE (POST) -----------
app.post('/employee', validateInput, (req, res) => {
   res.json({
      success: true,
      message: "Employee created successfully",
      data: req.body
   });
});

// ----------- SERVER START -----------
app.listen(3000, () => {
   console.log("Server running on port 3000");
});
```

☑ STEP-5 → Run

```
node server.js
```

🎯 OPTIONAL TEST COMMANDS (Only if teacher asks)

✔ Success Case

```
curl -X POST http://localhost:3000/employee \
-H "Content-Type: application/json" \
-d "{\"id\":1, \"name\":\"Rahul\", \"salary\":30000}"
```

✔ Error Case (salary missing)

```
curl -X POST http://localhost:3000/employee \
-H "Content-Type: application/json" \
-d "{\"id\":1, \"name\":\"Rahul\"}"
```

***** SLIP 11 *****

** Q1) Develop an Express.js application that defines routes for Create operations on a resource (Movie).

==>

☑ STEP-1 → Create folder

mkdir slip11_1
cd slip11_1

☑ STEP-2 → Initialize project
npm init -y

☑ STEP-3 → Install Express
npm install express

☑ STEP-4 → Create server.js
Create file server.js and paste this EXACT code:

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

app.use(express.json());   // JSON parsing middleware

let movies = [];   // temporary in-memory array

// ----------- CREATE MOVIE (POST) -----------
app.post('/movies', (req, res) => {
   const movie = req.body;
   movies.push(movie);

   res.json({
      message: "Movie created successfully",
      data: movie
   });
});

// ----------- START SERVER -----------
app.listen(3000, () => {
   console.log("Server running on port 3000");
});
```

☑ STEP-5 → Run server
node server.js

🎯 OPTIONAL TEST (Only if teacher asks)
✔ Create Movie

```
curl -X POST http://localhost:3000/movies \
-H "Content-Type: application/json" \
-d "{\"id\":1, \"title\":\"Inception\", \"rating\":9}"
```

** Q2) Create Angular application that print the name of students who play basketball using filter and map method.
==>
☑ STEP-1 → Create Project
ng new slip11_2
cd slip11_2

☑ STEP-2 → Update app.module.ts
(Required for ngModel if needed later)

Open:

src/app/app.module.ts

Replace with:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-3 → Update app.component.ts

Open:

src/app/app.component.ts

Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  students = [
    { name: "Rahul", sport: "Cricket" },
    { name: "Amit", sport: "Basketball" },
    { name: "Sneha", sport: "Basketball" },
    { name: "Neha", sport: "Badminton" }
  ];

  basketballPlayers: string[] = [];

  showPlayers() {
    this.basketballPlayers = this.students
      .filter(s => s.sport === "Basketball")
      .map(s => s.name);
  }
}
```

☑ STEP-4 → Update app.component.html

Open:

src/app/app.component.html

Replace everything with:

```
<h2>Basketball Players</h2>

<button (click)="showPlayers()">Show Players</button>

<ul>
  <li *ngFor="let name of basketballPlayers">
    {{ name }}
  </li>
</ul>
```

☑ STEP-5 → Run Project

```
ng serve -o
```

*** SLIP 12 ***

** Q1) Write an AngularJS script to print details of Employee (employee name, employee Id,Pin code, address etc.) in tabular form using ng-repeat.

==>

☑ STEP-1 → Create Project

```
ng new slip12_1
cd slip12_1
```

☑ STEP-2 → Update app.component.ts

(Path: src/app/app.component.ts)

Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  employees = [
    { id: 101, name: 'Rahul', pincode: 411001, address: 'Pune' },
    { id: 102, name: 'Sneha', pincode: 400001, address: 'Mumbai' },
    { id: 103, name: 'Amit', pincode: 560001, address: 'Bangalore' }
  ];

}
```

☑ STEP-3 → Update app.component.html

(Path: src/app/app.component.html)

Replace everything with:

```
<h2>Employee Details</h2>

<table border="1" cellpadding="8">
  <tr>
    <th>ID</th>
    <th>Name</th>
```

```
    <th>Pin Code</th>
    <th>Address</th>
  </tr>

  <tr *ngFor="let e of employees">
    <td>{{ e.id }}</td>
    <td>{{ e.name }}</td>
    <td>{{ e.pincode }}</td>
    <td>{{ e.address }}</td>
  </tr>
</table>
```

☑ STEP-4 → app.module.ts
(No change needed if same)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-5 → Run

```
ng serve -o
```

** Q2) Develop an Express.js application that defines routes for Create operations on a resource (User).
==>

☑ STEP-1 → Create folder

```
mkdir slip12_2
cd slip12_2
```

☑ STEP-2 → Initialize project

```
npm init -y
```

☑ STEP-3 → Install Express

```
npm install express
```

☑ STEP-4 → Create server.js
Create server.js and paste this EXACT code:

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

app.use(express.json());   // JSON parsing middleware

let users = [];   // temporary in-memory array
```

```
// ---------- CREATE USER (POST) ----------
app.post('/users', (req, res) => {
   const user = req.body;
   users.push(user);

   res.json({
      message: "User created successfully",
      data: user
   });
});

// ---------- START SERVER ----------
app.listen(3000, () => {
   console.log("Server running on port 3000");
});
```

☑ STEP-5 → Run server

```
node server.js
```

✔ Create User

```
curl -X POST http://localhost:3000/users \
-H "Content-Type: application/json" \
-d "{\"id\":1, \"name\":\"Rahul\", \"email\":\"rahul@gmail.com\"}"
```

***** SLIP 13 *****

** Q1) Extend the previous Express.js application to include middleware for parsing request bodies (e.g., JSON, form data) and validating input data. Send appropriate JSON responses for success and error cases.

==>

☑ STEP-1 → Create folder

```
mkdir slip13_1
cd slip13_1
```

☑ STEP-2 → Initialize project

```
npm init -y
```

☑ STEP-3 → Install Express

```
npm install express
```

☑ STEP-4 → Create server.js

Create server.js and paste EXACTLY this:

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

// ---------- MIDDLEWARE ----------
app.use(express.json());                    // parse JSON
app.use(express.urlencoded({ extended: true })); // parse form data

// ---------- VALIDATION ----------
function validateUser(req, res, next) {
   const { id, name, email } = req.body;
```

```
    if (!id || !name || !email) {
        return res.status(400).json({
            success: false,
            message: "Validation failed: id, name, email are required"
        });
    }

    next();
}

// ---------------- CREATE ROUTE ----------------
app.post('/users', validateUser, (req, res) => {
    res.json({
        success: true,
        message: "User created successfully",
        data: req.body
    });
});

// ---------------- START SERVER ----------------
app.listen(3000, () => {
    console.log("Server running on port 3000");
});
```

☑ STEP-5 → Run

```
node server.js
```

🎯 OPTIONAL TEST COMMANDS (Only if teacher asks)

✔ Success Case

```
curl -X POST http://localhost:3000/users \
-H "Content-Type: application/json" \
-d "{\"id\":1, \"name\":\"Rahul\", \"email\":\"rahul@gmail.com\"}"
```

** Q2) Create a simple Angular component that takes input data and displays it.

==>

☑ STEP-1 → Create Project

```
ng new slip13_2
cd slip13_2
```

☑ STEP-2 → Update app.module.ts

(Path: src/app/app.module.ts)

Replace everything with:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
```

```
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-3 → Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything with:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  inputText: string = "";
  outputText: string = "";

  showValue() {
    this.outputText = this.inputText;
  }
}
```

☑ STEP-4 → Update app.component.html
(Path: src/app/app.component.html)
Replace everything:

```
<h2>Input Display Component</h2>

<input type="text" [(ngModel)]="inputText" placeholder="Enter something">
<br><br>

<button (click)="showValue()">Show</button>

<h3 *ngIf="outputText">
  You Entered: {{ outputText }}
</h3>
```

☑ STEP-5 → Run
ng serve -o

*** SLIP 14 ***

** Q1) Create Angular application that print the name of students who got 85% using filter and map method.
==>
☑ STEP-1 → Create Project
ng new slip14_1

```
cd slip14_1
```

☑ STEP-2 → Update app.module.ts
(Path: src/app/app.module.ts)
Replace everything:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-3 → Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  students = [
    { name: "Rahul", percent: 75 },
    { name: "Amit", percent: 85 },
    { name: "Sneha", percent: 90 },
    { name: "Riya", percent: 60 }
  ];

  resultNames: string[] = [];

  showToppers() {
    this.resultNames = this.students
      .filter(s => s.percent >= 85)
      .map(s => s.name);
  }
}
```

☑ STEP-4 → Update app.component.html
(Path: src/app/app.component.html)
Replace everything:

```
<h2>Students Who Got 85% or Above</h2>

<button (click)="showToppers()">Show Students</button>

<ul>
  <li *ngFor="let name of resultNames">
    {{ name }}
  </li>
</ul>
```

☑ STEP-5 → Run

```
ng serve -o
```

** Q2) Develop an Express.js application that defines routes for Create, Update operations on a resource (Employee).

==>

☑ STEP-1 → Create folder

```
mkdir slip14_2
cd slip14_2
```

☑ STEP-2 → Initialize project

```
npm init -y
```

☑ STEP-3 → Install Express

```
npm install express
```

☑ STEP-4 → Create server.js

Create file: server.js

Paste EXACT code:

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

app.use(express.json());   // parse JSON body

let employees = [];   // temporary in-memory array

// ---------------- CREATE EMPLOYEE (POST) ----------------
app.post('/employees', (req, res) => {
    const emp = req.body;
    employees.push(emp);

    res.json({
        message: "Employee created successfully",
        data: emp
    });
});

// ---------------- UPDATE EMPLOYEE (PUT) ----------------
app.put('/employees/:id', (req, res) => {
    const empId = parseInt(req.params.id);
    const newData = req.body;
```

```
  const index = employees.findIndex(e => e.id === empId);

  if (index === -1) {
    return res.status(404).json({
      error: "Employee not found"
    });
  }

  employees[index] = { ...employees[index], ...newData };

  res.json({
    message: "Employee updated successfully",
    data: employees[index]
  });
});

// ---------------- START SERVER ----------------
app.listen(3000, () => {
  console.log("Server running on port 3000");
});
```

☑ STEP-5 → Run server

```
node server.js
```

🎯 OPTIONAL TEST COMMANDS (Only if teacher asks)

✔ Create Employee

```
curl -X POST http://localhost:3000/employees \
-H "Content-Type: application/json" \
-d "{\"id\":1, \"name\":\"Rahul\", \"salary\":30000}"
```

✔ Update Employee

```
curl -X PUT http://localhost:3000/employees/1 \
-H "Content-Type: application/json" \
-d "{\"salary\":45000}"
```

*** SLIP 15 ***

** Q1) Find an emp with a Salary greater than 25000 in the array. (Using find by id method)
==>

☑ STEP-1 → Create Project

```
ng new slip15_1
cd slip15_1
```

☑ STEP-2 → Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
```

```
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  employees = [
    { id: 1, name: "Rahul", salary: 20000 },
    { id: 2, name: "Amit", salary: 30000 },
    { id: 3, name: "Sneha", salary: 45000 },
    { id: 4, name: "Neha", salary: 18000 }
  ];

  result: any = null;

  findEmployee() {
    this.result = this.employees.find(e => e.salary > 25000);
  }
}
```

☑ STEP-3 → Update app.component.html
(Path: src/app/app.component.html)
Replace everything:

```
<h2>Employee with Salary > 25000</h2>

<button (click)="findEmployee()">Find Employee</button>

<div *ngIf="result">
  <h3>Employee Found:</h3>
  <p>Name: {{ result.name }}</p>
  <p>Salary: {{ result.salary }}</p>
</div>

<div *ngIf="!result">
  <p>No employee found yet.</p>
</div>
```

☑ STEP-4 → app.module.ts
(No need to change if same)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-5 → Run

```
ng serve -o
```

** Q2) Create Angular application that print the name of students who got 85% using filter and map method.
==>
☑ STEP-1 → Create Project
ng new slip15_2
cd slip15_2

☑ STEP-2 → Update app.module.ts
(Path: src/app/app.module.ts)
Replace everything:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

☑ STEP-3 → Update app.component.ts
(Path: src/app/app.component.ts)
Replace everything:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  students = [
    { name: "Rahul", percent: 72 },
    { name: "Amit", percent: 85 },
    { name: "Sneha", percent: 90 },
    { name: "Riya", percent: 60 }
  ];

  result: string[] = [];

  showResult() {
    this.result = this.students
      .filter(s => s.percent >= 85)
      .map(s => s.name);
```

```
  }
}
```

☑ STEP-4 → Update app.component.html
(Path: src/app/app.component.html)
Replace everything:

```
<h2>Students Who Got 85% or Above</h2>

<button (click)="showResult()">Show Students</button>

<ul>
  <li *ngFor="let name of result">
    {{ name }}
  </li>
</ul>
```

☑ STEP-5 → Run

```
ng serve -o
```