Module 3

What to learn

Built in Directives in Angular
Component->directive with a template
Attribute directive
NgClass
NgStyle
NgModel
Structural directives
NgIf
NgFor
NgSwitch

Practice Exercise

Practice 1
Do the hands on from following link https://angular.io/guide/built-in-directives
Practice 2
Dynamic Welcome Message

Use *ngIf to show a "Welcome, [user's name]!" message only when a name is entered in an input field using two-way binding.

Practice 3
Editable Product List
Use *ngFor to display a list of product names, and allow editing each name using an input field with two-way binding.

Practice 4
Highlight Active User
Use [ngClass] to highlight a user as "active" in a list of users when selected via two-way binding on a dropdown menu.

Practice 5
Character Counter
Display a character count below a text area using two-way binding, and hide it using *ngIf if no text is entered.

Practice 6

Dynamic Style Application
Change the background color of a div based on a dropdown selection using two-way binding and [ngStyle].

Practice 7
Real-time Discount Calculator
Use two-way binding to accept a product price input and display the discounted price dynamically using *ngIf for values greater than zero.

Practice 8
Task Completion Status
Display a list of tasks with checkboxes using *ngFor, and dynamically update a "completed tasks" count using two-way binding.

Practice 9
Search Filter for a Table
Bind an input field to a search term using two-way binding and display filtered rows of a table using *ngFor and *ngIf.

Practice 10
Dynamic Tab Activation
Use *ngSwitch and two-way binding to toggle between content for tabs (e.g., Home, About, Contact).

Practice 11
Password Strength Indicator
Bind a password input field using two-way binding and show a "Weak", "Medium", or "Strong" indicator using *ngIf based on password length.

Practice 12
Dynamic Role Display
Use two-way binding to select a role (Admin, User) from a dropdown, and show a message like "Welcome, Admin!" using *ngIf.

Practice 13
Dynamic Pricing Based on Quantity
Create a quantity input field with two-way binding and display the total price dynamically using [ngStyle] to highlight it when the quantity exceeds 10.

Practice 14
Product Stock Status
Bind an input field to a product's stock quantity using two-way binding, and use [ngClass] to show "In Stock" or "Out of Stock" styles dynamically.

Practice 15
Real-time Word Counter
Use two-way binding with a textarea and display the total word count dynamically

using *ngIf when there is text entered.

Practice 16

Toggle Description Visibility

Bind a checkbox using two-way binding, and toggle the visibility of a product description using *ngIf.

Practice 17

Temperature Converter

Bind an input field for temperature in Celsius using two-way binding and display the equivalent Fahrenheit value dynamically using a formula.

Practice 18

Attendance Tracker

Use *ngFor to display a list of students, and toggle their attendance status (Present/Absent) using a checkbox with two-way binding.

Practice 19

User Profile Updation

Allow a user to edit their name and email using input fields with two-way binding, and display the updated values dynamically using *ngIf.

Practice 20

Dynamic Background Changer

Use a color input with two-way binding and [ngStyle] to update the background color of a div in real time.

Practice 21

Shopping Cart Quantity

Use two-way binding to adjust the quantity of items in a shopping cart and update the total price dynamically using *ngFor and [ngClass].

Assignment Exercise

Assignment 1 (65cdfdab7ba36e06448762a4)

Create a student Model interface with field(ID,Name,Age,Average,grade,Active) StudentList Component will create an array of student type.Display an array with NGFor in table. Grade wise color should be given using ngSwitch and ngClass or ngStyle. Only active should displayed using ngIf.

Student Table: control ID -> #student-table

* Use Below json

```
students=[
  { ID: 1, Name: 'John', Age: 20, Average: 85, Grade: 'A', Active: true },
```

```
  { ID: 2, Name: 'Alice', Age: 22, Average: 75, Grade: 'B', Active: false }
];
```

Assignment 2

Use Case: Employee Leave Management System

Domain: **HR Recruitment**

Scenario: **An** Employee Leave Management System **where employees can apply for leave. The system will allow employees to select the** type of leave, dates of leave, **and provide a reason for the leave. Based on the selected leave type, the system will calculate whether it is a** paid **or** unpaid **leave and display the appropriate message in real time. It will also show the** available leave balance **for each leave type.**

UI Control IDs:

Employee Name Input

Control ID: employeeName
Type: text
Label: Employee Name
Required: Yes

Leave Type Dropdown

Control ID: leaveType
Type: select
Options: Paid Leave, Sick Leave, Casual Leave
Label: Leave Type
Required: Yes

Leave Start Date

Control ID: startDate
Type: date
Label: Start Date
Required: Yes

Leave End Date

Control ID: endDate
Type: date
Label: End Date

Required: Yes

Leave Reason Input

Control ID: leaveReason
Type: textarea
Label: Reason for Leave
Required: Yes

Leave Balance Display

Control ID: leaveBalance
Type: text
Label: Available Leave Balance
Read-Only: Yes

Leave Approval Status

Control ID: leaveApproval
Type: text
Label: Leave Approval Status
Read-Only: Yes

Submit Leave Application Button

Control ID: submitLeave
Type: button
Label: Submit Leave Application
Action: Submit leave application

Functional Flow:

Employee enters their name:

Employee types their **name** into the employeeName input field.
This will automatically store the employee's name in the system.

Selects Leave Type:

Employee selects the type of leave from the **Leave Type Dropdown** (leaveType).
The available options are **Paid Leave**, **Sick Leave**, and **Casual Leave**.
Each leave type will have different **leave balance** and **approval criteria**.

Leave Dates:

The employee selects **start date** and **end date** of the leave using the date pickers (startDate, endDate).
The system calculates the **duration** of leave in days based on the difference between the selected start and end dates.

Enter Leave Reason:

Employee provides the **reason for leave** in the leaveReason textarea. This will be used to evaluate whether the leave is approved and for record-keeping purposes.

Check Leave Balance:

Once the employee selects the leave type, the system will display the **leave balance** for that type in the **Leave Balance Display** (leaveBalance).
For example:

Paid Leave: 12 days remaining
Sick Leave: 8 days remaining
Casual Leave: 5 days remaining

The system will validate if the **requested leave duration** is within the employee's available leave balance. If the leave balance is insufficient, an alert is shown.

Approval Status:

Based on the leave type and duration, the system will calculate whether the leave is **approved** or **rejected**:

Paid Leave: Can be applied for up to 12 days in a year.
Sick Leave: May require a doctor's note for approval.
Casual Leave: Requires manager approval.

The **Leave Approval Status** (leaveApproval) is updated in real time based on the employee's inputs and the available leave balance.

Submit Leave Application:

When the employee is ready to submit the leave application, they click the **Submit Leave Application** button (submitLeave).
The system validates the form, ensuring all fields are filled correctly, and processes the leave request based on availability and approval rules.
If the leave is **approved**, a confirmation message is shown; if not, the employee is prompted to correct the details or try again later.

Leave Application Confirmation:

Once the leave application is successfully submitted, the **Leave Approval Status** (leaveApproval) updates with a message like "Leave approved" or "Leave pending approval" based on the internal logic.

Business Logic:

Leave Type Logic:

Paid Leave: The employee can take paid leave from the available balance. If the requested days exceed the balance, the system will not allow

submission and display a message stating “Insufficient Paid Leave Balance.”

Sick Leave: Sick leave may require documentation (doctor’s note). If the employee selects Sick Leave, the system will show a field for uploading documents.

Casual Leave: Casual leave requires manager approval. The system will display a message “Pending Manager Approval” once the leave is requested.

Leave Duration Calculation:

The system calculates the number of days requested based on the difference between startDate and endDate. If the start date is after the end date, an error message will appear.

Leave Balance:

The system retrieves and displays the **remaining leave balance** based on the employee’s leave type. It deducts the requested leave days after approval.

Approval Flow:

For **Sick Leave** and **Casual Leave**, additional steps are added to request manager approval or upload supporting documentation.

For **Paid Leave**, it automatically checks the leave balance and allows submission if sufficient days are available.

Expected User Flow:

The **employee** fills in the necessary details in the leave form, including their **name**, **leave type**, **start and end dates**, and **leave reason**.

The system calculates the **leave duration** and displays the **leave balance** for the selected leave type.

The system checks if the employee has sufficient leave balance and approves or rejects the application based on the type of leave and duration.

The employee submits the leave application.

The employee receives an immediate notification on the **leave approval status**.

Time Estimate:

UI Design and Layout: 1-1.5 hours for setting up the form, labels, and controls.

Business Logic Implementation: 1 hour for implementing the leave validation, approval flow, and balance calculations.

Testing and Debugging: 30-45 minutes for ensuring the correct flow and handling errors gracefully.

This use case covers a **real-world scenario** in **HR Recruitment** where an employee applies for leave, and the system ensures that business logic is followed for **leave approval**, **balance calculation**, and **user interaction**. It leverages Angular's **two-way binding**, **built-in directives**, and **real-time validation** for a smooth user experience.

Online Reference

No online Reference

Introduction

the basics

course project-basics

debugging

components & databinding deep dive

course project – components & databinding

directives deep dive

Using Services & Dependency Injection

Course Project – Services & Dependency Injection

Changing Pages with Routing

Course Project – Routing

Handling Forms in Angular Apps

Course Project-Forms

Using Pipes to Transform Output

Making Http Requests