# Explanation of Project

I approached this assignment as if I were building a real, maintainable Angular application, not just a quick prototype.

## Architecture & Structure

I followed a clean Angular architecture:  
- A core folder for models and services (the single source of truth).  
- A features folder for the main UI components: form, list, and chart.  
- The root AppComponent acts as the shell that wires everything together.  
  
This separation of concerns makes the project easy to scale and maintain.

## State Management

I centralized all state in a WeightTrackerService.  
- It uses RxJS BehaviorSubjects to hold the entries and the target weight.  
- Components subscribe with the async pipe, so the UI automatically updates when state changes.  
- This reactive pattern keeps components lean and ensures consistency across the app.

## Type Safety

I defined a WeightEntry interface, so every piece of data is strongly typed. This avoids runtime errors and makes the code self-documenting.

## Features

- Weight Form: add or update entries with date, weight, and optional note.  
- Weight List: displays history with options to remove entries.  
- Weight Chart: uses Chart.js v4 to visualize progress with smooth animations, hover tooltips, and a dashed target line if one is set.  
- Progress Calculation: the service computes difference from target and percentage progress, displayed clearly to the user.

## Styling & UX

I deliberately kept styling in plain CSS for simplicity, but still made it polished:  
- Cards with shadows and rounded corners for a modern look.  
- Consistent spacing and typography.  
- Color-coded buttons for clear user actions (blue for save, red for delete).  
- Responsive layout so form and list sit beside the chart neatly.

## Demo-Ready

I included sample data so that when the app starts, you immediately see the chart populated and the progress stats working. That makes the demo engaging right away.

## Scalability

The app is small, but the structure is future-proof:  
- Adding persistence (localStorage or API) would only require changes in the service, not the UI.  
- Components are reusable and independent.  
- State management is reactive and scalable.

# Executive Summary

The Weight Tracker application is a small but well-structured Angular project designed to demonstrate clean architecture, reactive state management, and user-friendly design. It allows users to record their weight history, set a target weight, and track progress visually through a dynamic chart and progress indicators.  
  
Key highlights:  
- Clean Angular architecture with clear separation of concerns.  
- Centralized, reactive state management using RxJS BehaviorSubjects.  
- Strong typing with interfaces for data safety and clarity.  
- Simple yet polished UI with plain CSS (cards, shadows, responsive layout).  
- Chart.js integration with animations, tooltips, and a target reference line.  
- Demo-ready with sample data preloaded.