MVVM (Model-View-ViewModel)

MVVM Architecture in Angular

- **Model**
  Represents the application data and business logic (e.g., Services, Interfaces, APIs)

- **View**
  The UI elements (HTML templates + Angular bindings)

- **ViewModel**
  The component class (`.ts`) that connects View and Model

📊 ASCII Diagram

```
+----------------+        Bind Data       +------------------+        API/Data Logic       +--------------------+
|     View       | <--------------------> |   ViewModel        | <------------------------> |      Model         |
| (HTML, UI)     |   (Two-way binding)    | (Component .ts)     |     (Service, Interface)   | (API, Business
Logic)|
+----------------+                        +------------------+                            +--------------------+

Example:             home.component.html          home.component.ts              user.service.ts
                     -------------------          -----------------              -----------------
                     <input [(ngModel)]="name">   name: string = '';             getUser(), saveUser()
```

🔄 Flow Explanation

Layer	Role
Model	Talks to backend via services (`HttpClient`), handles business logic
ViewModel	Component class: holds UI data, methods, and handles events
View	HTML template: binds to component variables, emits events

---

View (HTML) – `app.component.html`

```
<h1>Hello {{ name }}</h1>
<input [(ngModel)]="name" />
<button (click)="save()">Save</button>
```

ViewModel (Component) – `app.component.ts`

```
export class AppComponent {
  name = '';

  constructor(private userService: UserService) {}

  save() {
    this.userService.saveUser(this.name);
  }
}
```

`Model (Service)` – `user.service.ts`

```
@Injectable({ providedIn: 'root' })
export class UserService {
  saveUser(name: string) {
    console.log("Saving user:", name);
    // HTTP logic goes here
  }
}
```

✅ Summary Table

MVVM Role	Angular Entity	Responsibility
Model	Service, Interface	Data fetching, persistence, business logic
ViewModel	Component Class (.ts)	Handles data & logic for the template
View	HTML Template	UI and user interactions