# Services & Injectable

By Nikunj Soni
JAVA FSD/MERN/MEAN (Expert & Instructor)

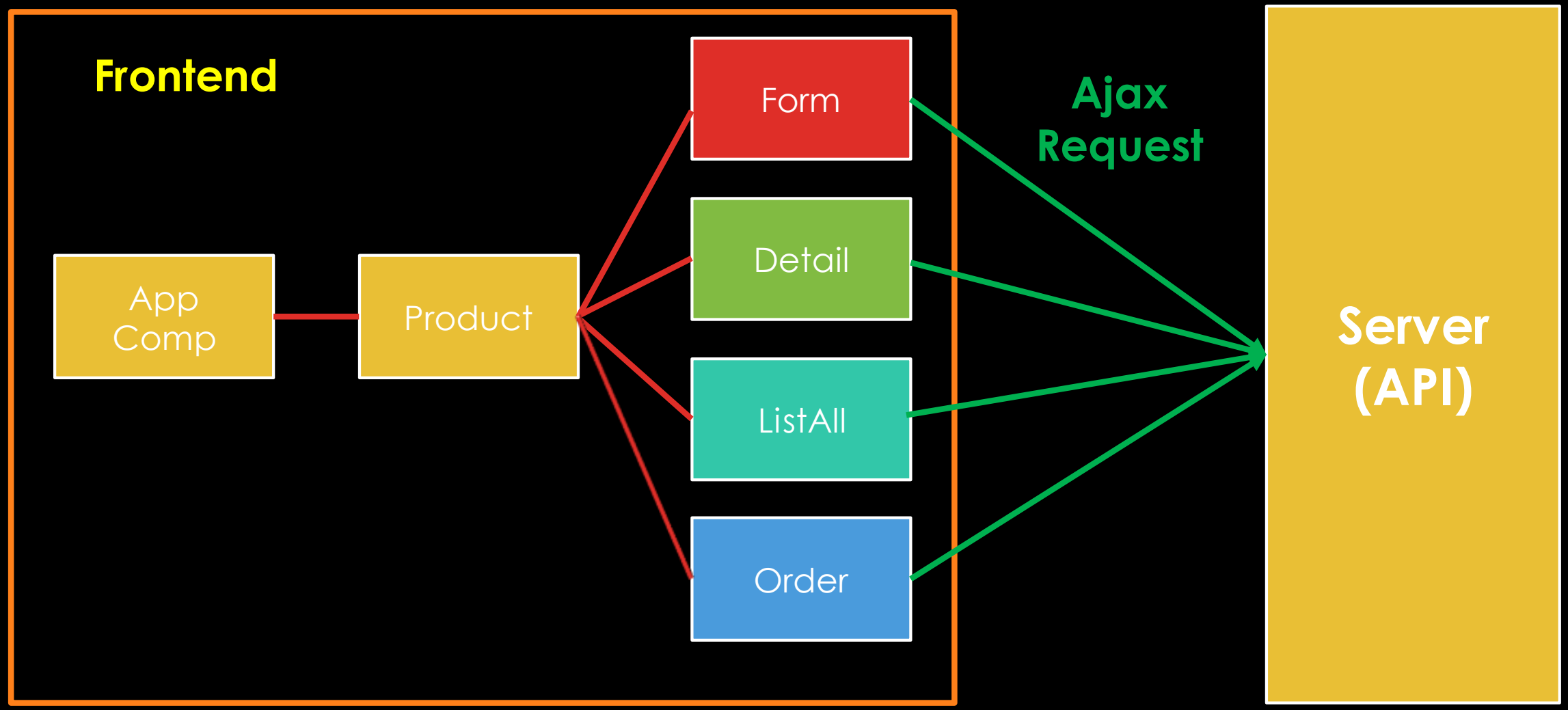
# SERVICES

- Service refer to a function or group of functions designed to do something specific.
- Used in many places like
  - **micro service architecture**
  - **service-oriented architecture**
  - **domain-driven design etc.**

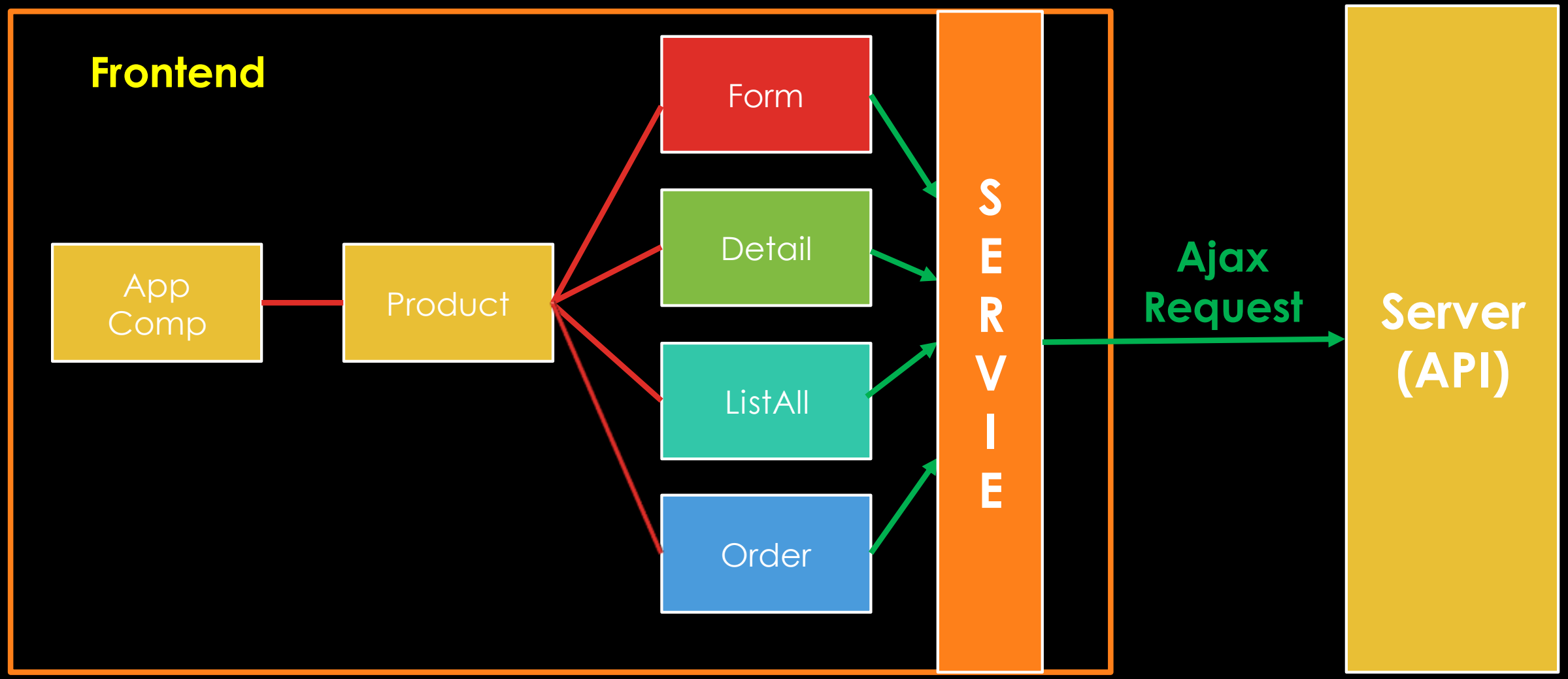
# HOW SERVICE WORKS

- In order to build a loosely-coupled application and reuse code, it is best if you design your components to be lean and efficient.
- This means that the component's job should be to focus on the user experience and nothing more.
- A component should contain properties and methods for data binding, and delegate tasks such as fetching data and input validation to another class (a service).
- Doing it this way, we can also reuse that code or service in other components.

# ANGULAR APP WITHOUT SERVICE



# ANGULAR APP WITH SERVICE



# SERVICE

- Services are classes with a narrow, well-defined purpose.
- Services act as a central repository or business unit.
- Injecting a service into a component gives the component access to that service.
- The Injectable decorator marks a class as a service
  - **@Injectable()**
- Service injection involves three main objects:
  - **Injector**
  - **Provider**
  - **Constructor**

# HOW TO CREATE SERVICE

- Use below command for service creation
- Ng generate service service-name
- Or you can use short cut: ng g s service-name
- E.g.- **ng g s user**
- So it will create **user.service.ts** and **user.service.spec.ts** file in app folder

# DEPENDENCY INJECTION

- Dependency Injection (DI) is a design pattern by which dependencies or services are passed to objects or clients that need them.
- The idea behind this pattern is to have a separate object create the required dependency, and pass it to the client.
- This makes a class or module to focus on the task it is designed for, and prevents side effects when replacing that dependency.
- DI is also at the core of Angular and can be used to provide components with the dependencies that they need.
- Just register the service with the Angular DI system so it knows how to inject it into components that need it.

# INJECTORS & PROVIDERS

- **Injector**:
- is responsible for creating the dependencies and maintains a container of dependency instances that it reuses if needed.
- The *injector* knows how to find and create dependencies through an object which is called **Provider**.