20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by Iniyan R
CB.EN.U4CYS21023
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

AMRITA
VISHWA VIDYAPEETHAM

Outline

Principles of a reactive application

Asynchronous Event Processing

- In Asynchronous Event processing processing of one event doesn't block the processing of other event.
- So when an task takes long time for running and need to be started at first other tasks can be did simaltaneously without waiting for that task.
- Processing units don't block each other in asynchronous event processing.
- Working :
  - As Events arrive they are placed in the event queue.
  - Available processing units process the events from queue.
  - Each unit processes its event independently, emitting results as output streams.
  - The system remains responsive to new events even during processing.
  - Backpressure mechanisms manage event flow to maintain stability.
- Example - In an web application consider an user who uploads files or downloads so when at that time user can simaltaneously access other features of the application without any delay.

Data streams

- Data stream are a series of data items generated sequentially over time from a particular source.
- Data streams are coherent and cohesive collections of digital signals that are generated continual or near-continual basis.
- They encapsulate asynchronous or event-driven data, like :
  - user inputs
  - network requests
  - and sensor readings.
- Data streams can be either finite or infinite, continuing indefinitely.

Code

- Imperative Paradigm :

var y = 10
var z = 90
var x = y * z
y = 100
console.log(x)
In this value of x remains 900 even after y is updated.to update it the value of x should be updated again.

- Reactive paradigm :

In this consider an operation '$=' instead of = which updates the value of x when the value of y or z is updated.
var y = 10
var z = 90
var x $= y * z
y = 100
console.log(x)
In this the value of x will be 9000 as y is updated to 100 the operation is performed again and the value of y is updated.

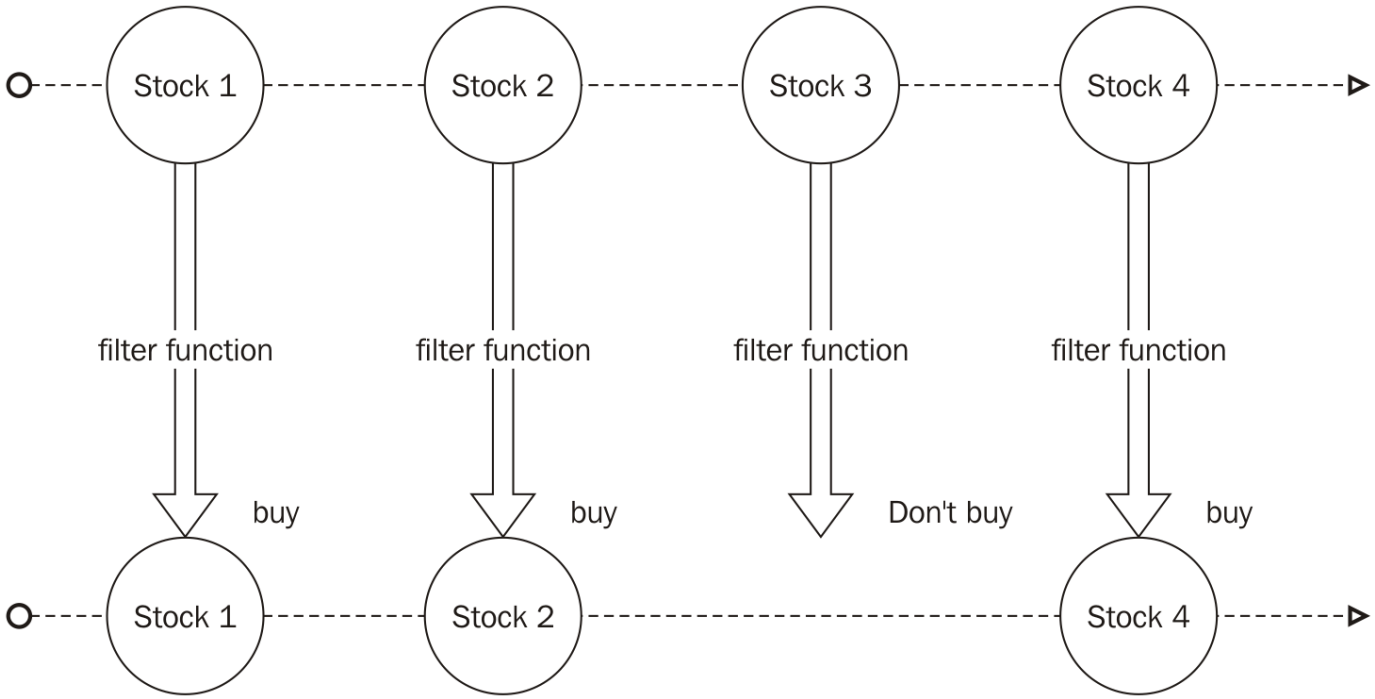
Producer-Consumer problem

Producer - Producer is an entity that emits data or events to one or more consumer.It can be a source of data like databases, web services, sensors, or user inputs. Producers are classified as hot or cold based on whether they start emitting data before or after consumer subscribes for data.
A producer can be unicast, emitting the same data to each consumer, or multicast, delivering different data to each consumer.
Consumer - Consumer is an object designed to receive and process data from a source. Consumers can subscribe to one or more sources, engaging in actions on the data they receive. These actions may include printing, filtering, transforming, storing, or forwarding the data to another consumer.

Implementation

->Pull - The consumer regularly checks for values and reacts whenever a relavant value is available.This process of regularly checking for values is known as polling.
->Push - Whenever there the value contain all information and no further quering is needed the consumer pushes.
->Push-Pull - So,if there is a change in value the a change notification is received by the consumer(eg.some value changed).This is push.So the change notification wont contain all information so the consumer should query again from the producer.This is pull.

Trading


Stock 1
Stock 2
Stock 3
Stock 4
filter function
filter function
filter function
filter function
buy
buy
Don't buy
buy
Stock 1
Stock 2
Stock 4

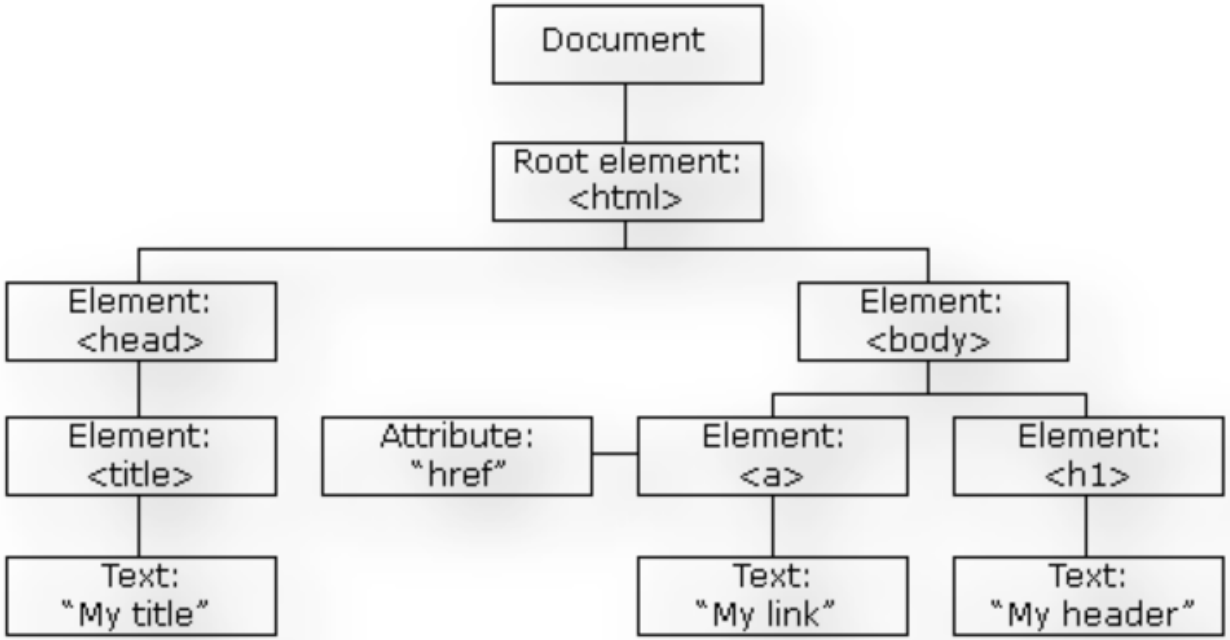
Introduction

- Angular is a MVVM (Model-View-ViewModel) framework.
- Angular is used for building Single-Web applications using html and typescript.
- Angular was written in typescript
- AngularJs version 1 was released in 2012.AngularJs was developed by Miško Hevery from 2009.And this was fully rewritten in typescript and and was named Angular 2.0,as of now angular version 17 is the latest.
- Angular is based on declarative and reactive programming.

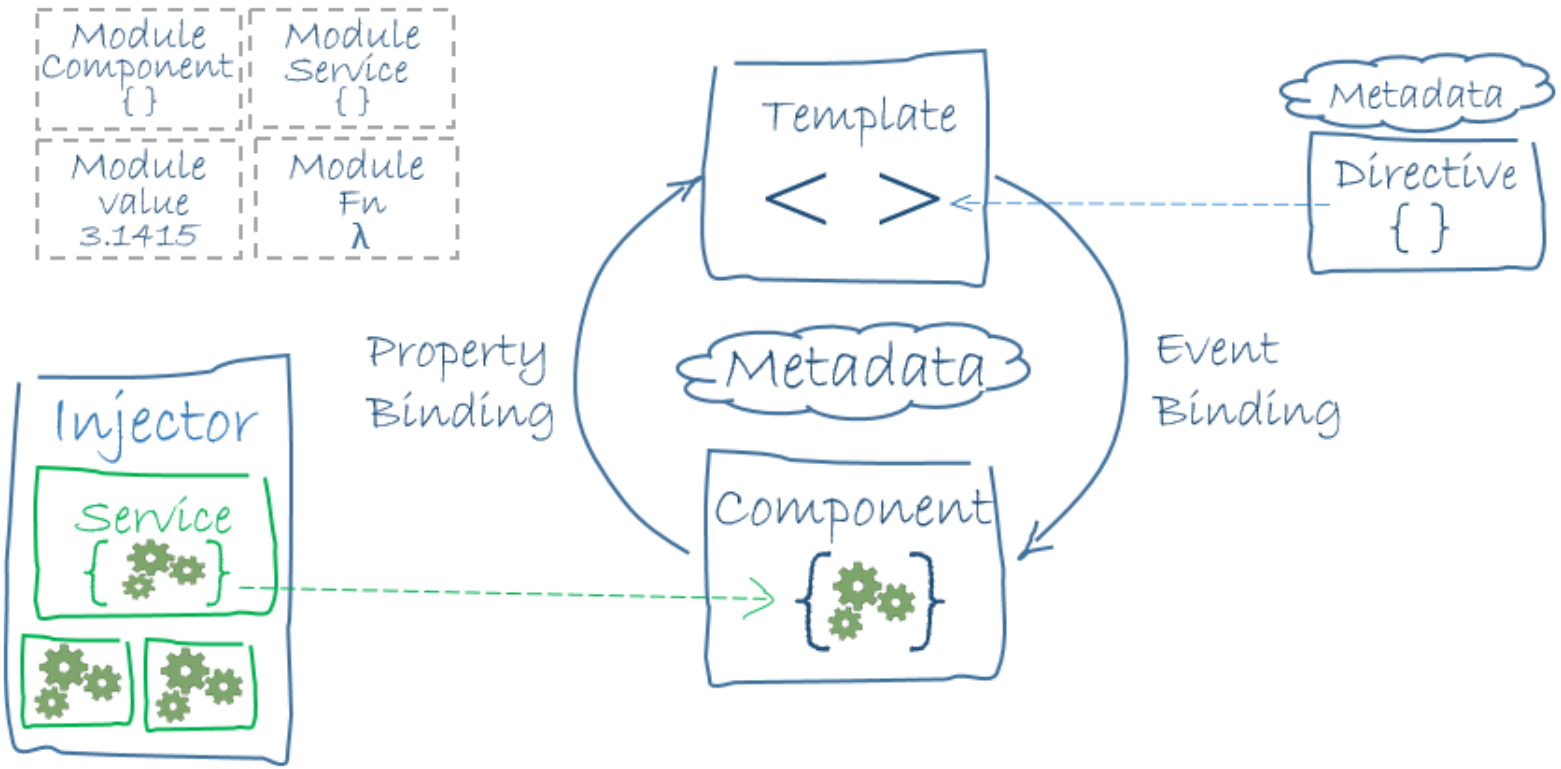
MVVM

- Model - Itis application's data and business logic. It is used for retrieving, storing, and manipulating data.Model notifies the ViewModel of any changes in the data.
- View - It is used for displaying UI to the user. It is used for displaying the visual elements, like buttons, textboxes, that users interact with. View observes the ViewModel for changes in the data it needs to display.
- ViewModel - It is intermediary between Model and the View. It transforms data from Model into format that is easily consumable by View. It also exposes commands and methods that the View can bind to for handling user interactions. It does not have direct knowledge of View.
- Data Binding: MVVM frameworks provide mechanism for automatic data binding between ViewModel and View. So, changes in ViewModel will automatically update the View, and vice versa.So this makes,it an Reactive paradigm.
- The components can be tested seperately because Model,View,ViewModel are seperate components.

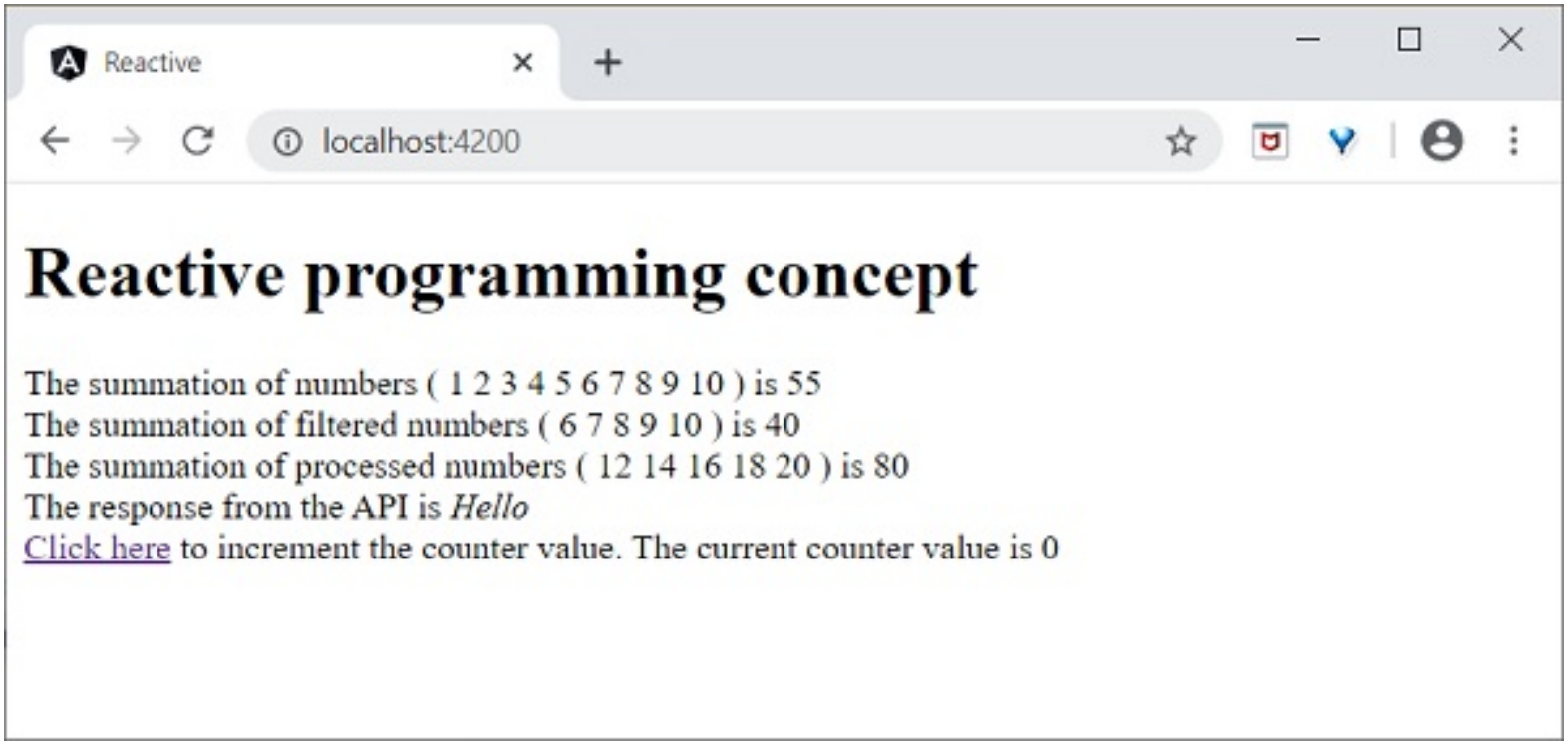
DOM


Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"

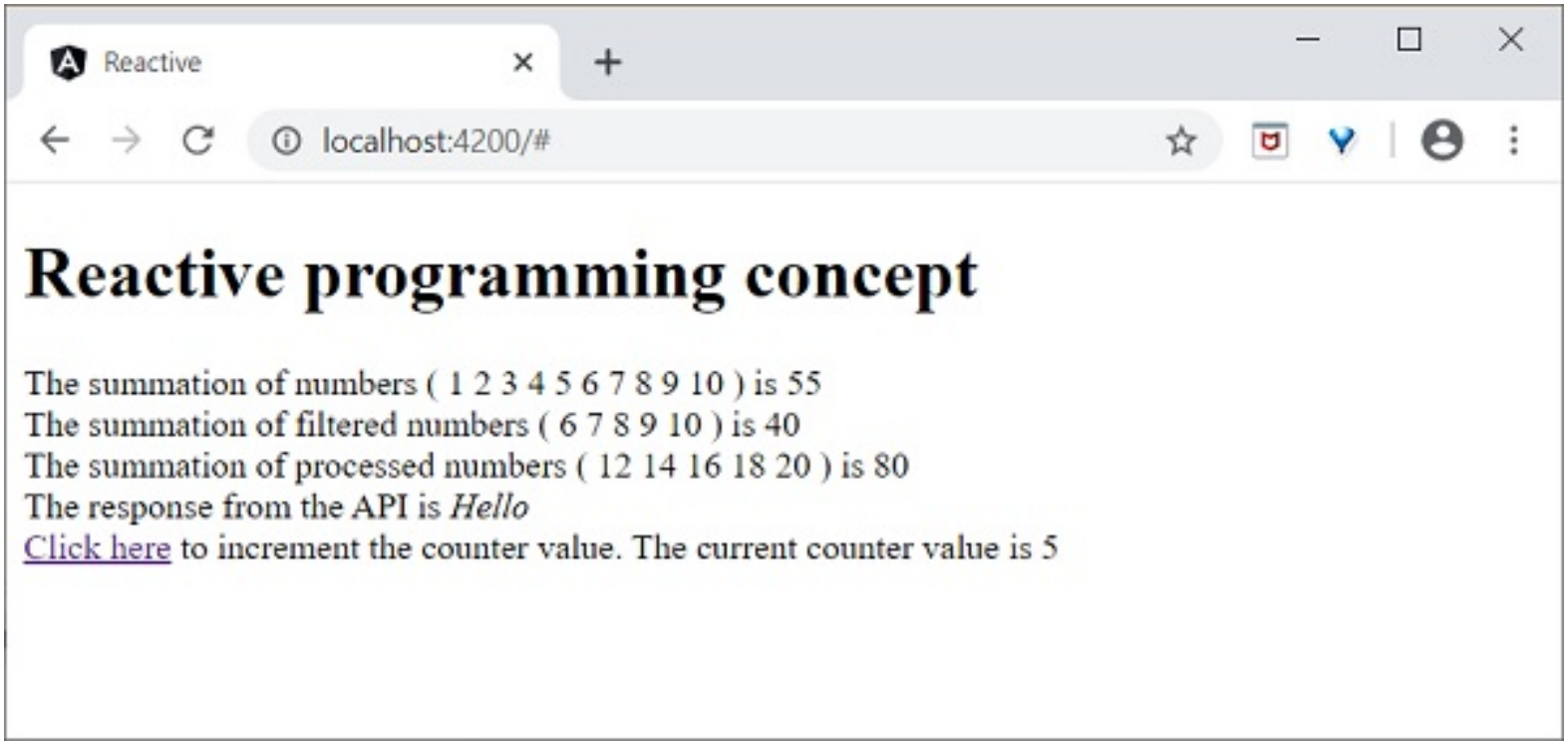
Architecture



Output


Reactive
localhost:4200
Reactive programming concept
The summation of numbers (1 2 3 4 5 6 7 8 9 10) is 55
The summation of filtered numbers (6 7 8 9 10) is 40
The summation of processed numbers (12 14 16 18 20) is 80
The response from the API is Hello
Click here to increment the counter value. The current counter value is 0

Output 2


Reactive
localhost:4200/#
Reactive programming concept
The summation of numbers (1 2 3 4 5 6 7 8 9 10) is 55
The summation of filtered numbers (6 7 8 9 10) is 40
The summation of processed numbers (12 14 16 18 20) is 80
The response from the API is Hello
Click here to increment the counter value. The current counter value is 5

Introduction

- Aspect-oriented programming was developed by Gregor Kickzales and Xerox PARC in 2001.
- It was developed based on subject-oriented programming,adaptive programming.
- The aim of aspect-oriented programming is increasing the usage of modularity(modularisation).It is a programming paradigm that aims to modularize crosscutting concerns, such as logging, transaction management security,and error hadlling by separating them from main business logic
- It allows developers to separate concerns into different aspects rather than mixing them all in one module
- So for example in a banking system Security is a cross-cutting concern,so it should be applied in many part of the application so here we define the functionality in a code instead of implementing in every method and is called when needed by the application

How AOP Works

STEP 01 Identify concerns and define aspects

STEP 02 Determine join points

STEP 03 Define pointcuts

STEP 04 Define advice

STEP 05 Weave aspects

Introduction

- Django is a python-based web framework that was created in 2003 and maintained by Django Softwre Foundation(DSF)
- Aim of the project was reusability and pluggability of components.
- It offers tools for data validation,caching, logging, pagination, authentication, and static file management in web apps.
- Django follows DRY(Dont Repeat Yourself) principle.
- Django provides administrative create, read, update and delete interface which is generated dynamically through introspection and configured via admin models.

How Django is Aspect-Oriented paradigm?

Django is aspect-oriented by using the futures of Middleware,Decorators and Signls.

- Middleware - Middleware components are used to process requests and responses globally before they reach the view or after the view has processed the request. Middleware functions are applied in a chain-like fashion, so now cross-cutting concerns such as authentication, logging, or caching are performed.
- Decorators - Decorators are metaprogramming and are used to implement aspects of AOP.

Example: `login_required` decorator is a cross-cutting concern that ensures that a user is authenticated before accessing a particular view.

- Signals - Django signals are used to allow decoupled applications to get notified when certain actions occur elsewhere in the application. By connecting signal handlers to specific signals, the code can be executed in response to certain events without directly coupling the code to the code that triggers the event.

Comparison

- The aim of reactive programming is to handle asynchronous and event-driven systems while the aim of aspect-oriented programming is modularizing cross-cutting concerns.
- The main concept of reactive programming is observer and observable state(producer and consumer) whereas in aop it is aspects.
- Aop doesnt concern about asyncornous properties.
- AOP is used mainly for logging,authentication and authorization whereas reactive is used for responsiveness and scalability

Similarities

- Both languages have a steep learning curve
- Although reactive paradigm mainly foucses on asynchronous property but it also uses modularity for data streams.
- Both paradigm use declarative approach meaning expressing the desired outcome without specifying step-by-step procedure for achieving it.
- AOP supports dynamic adaptation by allowing aspects to be woven into code during runtime, providing flexibility in changing behavior of a system without modifying code while reactive systems use dynamic adaptation by reacting to changes in the environment or data streams, adjusting their behavior in realtime.Django implements using middleware while angular implements in client side making components and services to react to changes in the application state.
- AOP implements aspects related to event-driven architecture by capturing events and handling them in separate aspects whereas in reactice programmming components react to changes or events in system giving real-time updates.
- In Django event-driven architecture is implement through its support for signals whereas in angular it is implemented in client side like HTTP respones.
- Django implements cross-cutting concerns in backend(authentication, security, and database access) while angular implements it in front end(routing and HTTP request

References

- Paradigm 1: Reactive
  - https://www.baeldung.com/cs/reactive-programming
  - https://en.wikipedia.org/wiki/Reactive_programming
  - https://www.techtarget.com/searchapparchitecture/definition/reactive-programming
  - https://subscription.packtpub.com/book/web-development/9781786463388/1/ch01lvl1sec0/the-reactive-paradigm
- Language for Paradigm 1: Angular
  - https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular
  - https://www.tutorialspoint.com/angular8/angular8_reactive_programming.html
  - https://v2.angular.io/docs/ts/latest/guide/architecture.html
- Paradigm 2 : Aspect-Oriented
  - https://en.wikipedia.org/wiki/Aspect-oriented_programming#Implementation
  - https://www.spiceworks.com/tech/devops/articles/what-is-aop/
  - https://medium.com/hprog99/aspect-oriented-programming-b9a06ca256db
- Language for Paradigm 2 : Django
  - https://en.wikipedia.org/wiki/Django_(web_framework)
  - https://www.djangoproject.com/
  - https://www.tutorialspoint.com/python_web_development_libraries/python_web_development_libraries_django_framework.html
  - Chatgpt for code.Prompt : give an code in django which authenticates an user which is base on aspect oriented programming.