**Applications/ Services**

There is a piece of code which runs on client side and server side which communicate with each other and fulfill the purpose of system.

Request

Server

Client

Response

Front-End Apps Backend- Apps

Applications

Front end Backend

Languages: Java, C#, Kotlin, Dart, Swift, JS Java, Python, C#, Go, JS, PHP

Frameworks: Angular, React, WPF, Flutter Node, ASP.NET, SpringBoot

Applications can be developed in any language of framework but the method of communicating (APIs) cient-server is agnostic, not Lang specific.

**Responsibilities**

1. **Client Apps**

Render UI Element

Handle interactions, events

Collect Data

Communicate with Backend apps and fetch/store data

1. **Server Apps**

Expose API Endpoints

Housing business logic

Handle data transforming, modelling

Interact with data stores

Interact with other services

**Factors to consider for developing an application (Ex: Parking Lot System)**

1. **Requirement**

You should be able to display all parking slots, which are empty and which are taken

Finding the next empty slot

Given the car no., you should be able to find the slot No and vice versa.

1. **Layer**

Ask whether you need to just expose backend endpoints for a client app

Or build a full client-server application

Or will it be just CLI app

Ask if it’s a mobile, desktop or we app

1. **Tech Stack**

Depending on layer, we decide the tech stack.

Android App – Java/Kotlin/ Flutter

Desktop App- Java-Swing/ .NET-WPF

Web App- Spring Boot/ ASP.NET MVC/ MERN/MEAN

1. **Code structure/ design pattern**

What design patterns will be used?

How project will look like.

What libraries will be required?

How endpoints will be exposed.

1. **Data store interaction**

How it will interact with databases.

How to store cache, etc.

1. **Performance/ Cost**

We cannot write an app which eats a lot of memory or takes many CPU Cycles.

If this is not considered, we will end up writing an app which hogs on machine resources and slows down the app.

1. **Deployment**

Want to deploy on cloud.

Or just make available at certain port.

1. **Monitoring**

Measuring how much time and memory app takes.

How many request are served daily, per sec.

Logging the errors occurred.

1. **Operational excellence/ Reliability**

Application should not crash on giving wrong input or when load increases.

Should handle failures.

It should be reliable/ resilient.

There are certain applications which include multiple modules like, keeping notes, payments, user authentication etc, which are handled by only one service in backend.

Such backend applications are called monolithic applications.

Many startups still prefer **monolithic** architecture when applications are small and requests served are very less.

But as application scales and user base increases, the load on one application increases. Hence, we need to distribute the load to multiple services such that each service handles specific types of requests. Such architecture is called **micro service** architecture.