The Storyline of System Design's Genesis

educative

1960s

Single Unit Monolithic Systems

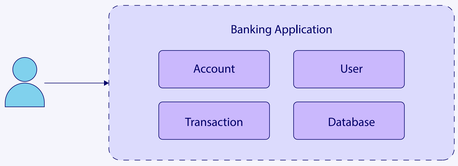
Software architecture where all components of an application are tightly integrated into a single processing unit.

Simplicity | Single Deployment | Low Overhead | Synchronous Communication

There are two major types of monolithic architecture

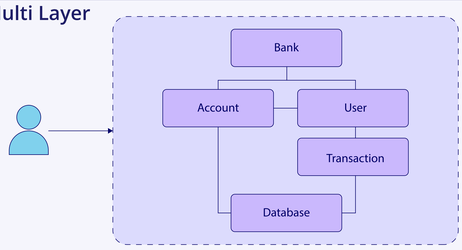
- **Single Layer Architecture**
  A simple and undivided structure where all components—including presentation, business logic, and data management—reside in a single layer or tier.
- **Multi Layer Architecture**
  A structure that divides a system into distinct layers—such as presentation, business logic, and data—promoting modular design, maintainability, and scalability.

Single Layer



Pros	Cons
Latency	Maintainability
Dev Overhead	Scalability
	Cohesion
	Coupling

Multi Layer



Pros	Cons
Maintainability	Coupling
Dev Overhead	Overhead
	Latency
	Scalability

1970s

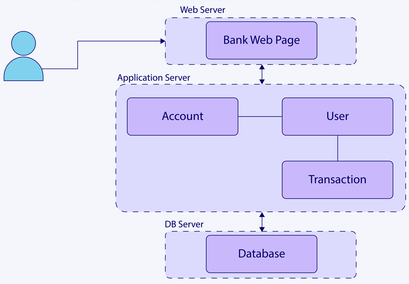
Early Networked Systems

A group of interconnected devices that can communicate with each other and share resources such as data, files, and applications.

TCP | UDP | HTTP(S) | OSI | Distributed Deployment | Asynchronous Communication | Network Security

This laid the foundation for ditributed computing, giving us architectural patterns like the Client-Server.

Network Enhanced Architecture



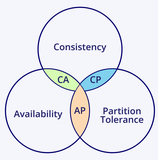
Pros	Cons
Maintainability	Dev Overhead
Coupling	Latency
Cohesion	

1980s

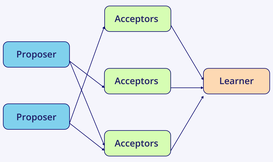
Distributed Systems

A network of independent computers working together to achieve a common goal.

CAP Theorem | PACELC Theorem | Consistency Model | Raft Algorithm | N-Phase Commits | Paxos Theorem | Geographically Distributed Comuting

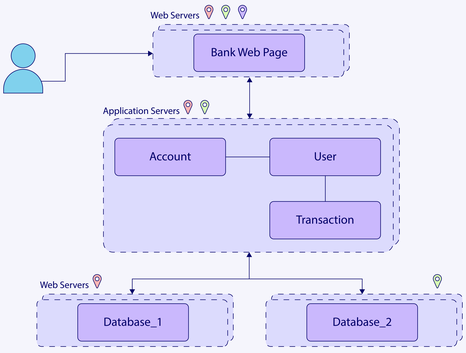


CAP Theorem



Paxos Theorem

Distributed Systems



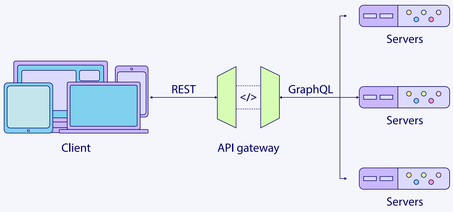
Pros	Cons
Maintainability	Consistency
Availability	Cost
Fault Tolerance	Dev Overhead
Performance	Complexity
Decentralization	

2000s

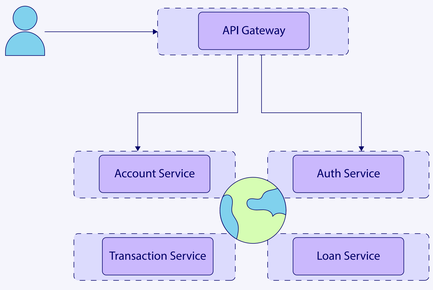
Microservices

A modular and decentralized approach to software development, where an application is composed of small, independent services.

REST | gRPC | GraphQL | API Gateway | Microservices Decomposition



Microservices

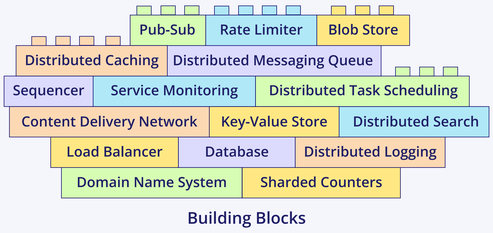


Pros	Cons
Scalability	Consistency
Flexibility	Complexity
Fault Isolation	Dev Overhead
Availability	
Latency	
Coupling	

2010s

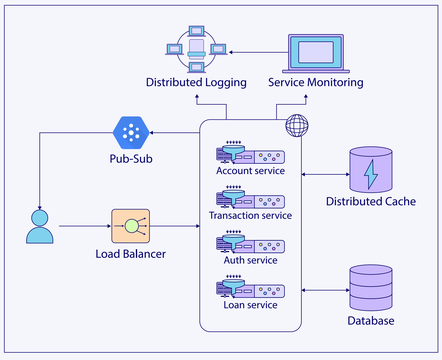
Modern System Design

Modern System Design leverages modular architectures, microservices, and DevOps practices to create scalable, flexible, and maintainable systems.



Building Blocks

Modern System Design



Pros	Cons
Scalability	Complexity
Flexibility	Dev Overhead
Fault Tolerance	
Reliability	
Availability	
Interoperability	

How to Design a System

A systematic approach to system design: RESHADED

Requirements	Estimation	Storage schema	High level design	API Design	Detailed design	Evaluation	Distinctive component
Undertsanding functional and non-functional needs of the system	Calculating resource requirements for the system	Creating a data model for the system	Identifying the main components and building blocks	Building appropriate interfaces for services	Finalizing a detailed design meets requirements	Justifying how our detailed design meets requirements	Identifying a unique aspect for each design problem and discuss it