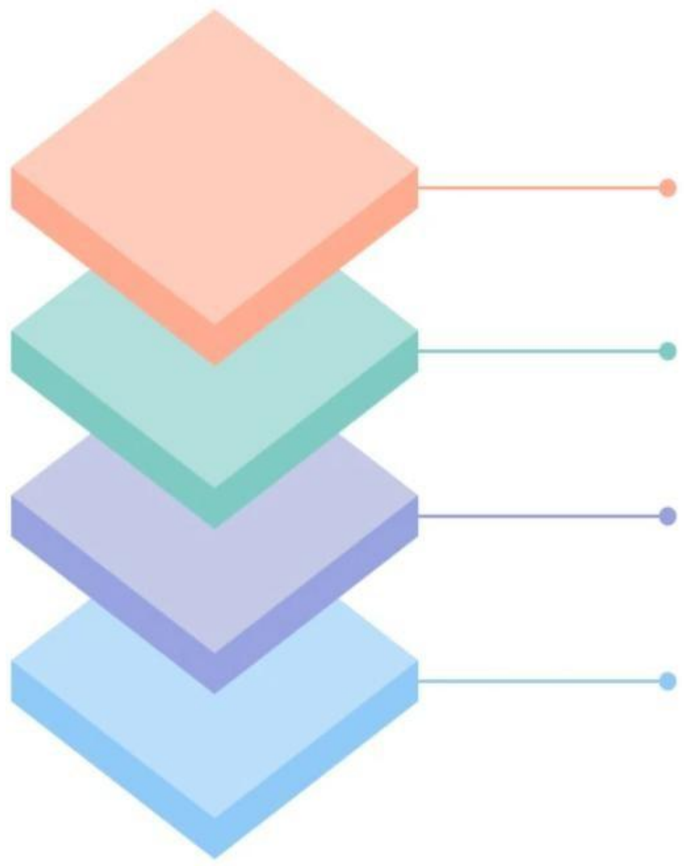
Modern Web Development

Nimesha Hewawasam
Lecturer
nimesha.h@nsbm.ac.lk

Layers Architecture in Software Architecture



Presentation Layer : responsible to display the user interface and manage user interaction.

Application Layer : (aka Business Layer) has all the business logic, rules and policies. It is the bridge between presentation layer and the data layer.

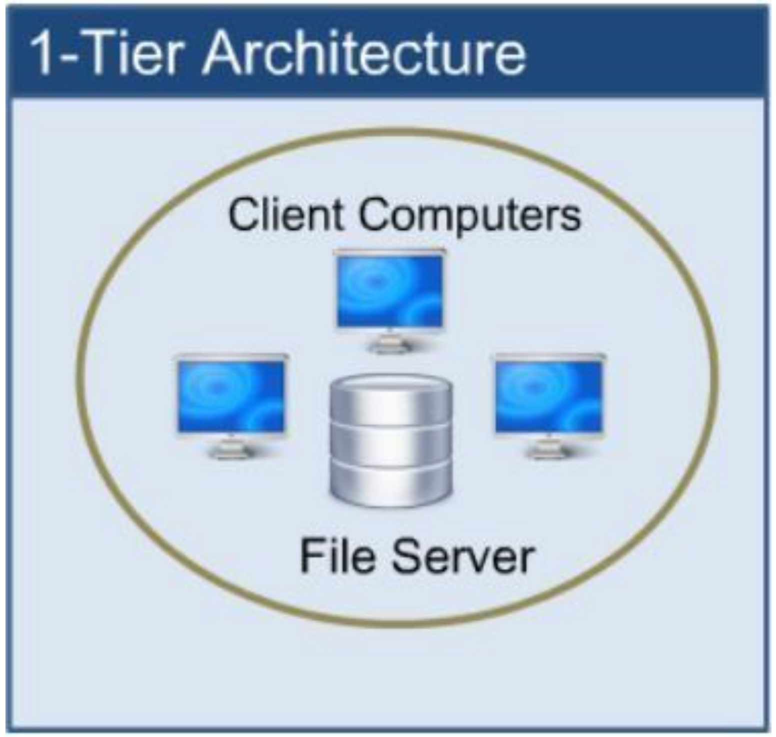
Service Layer : responsible to define and implement the service interface and the data contracts. The service layer communicates with the application layer

Data Layer : responsible for storing the data.

Tier Architecture

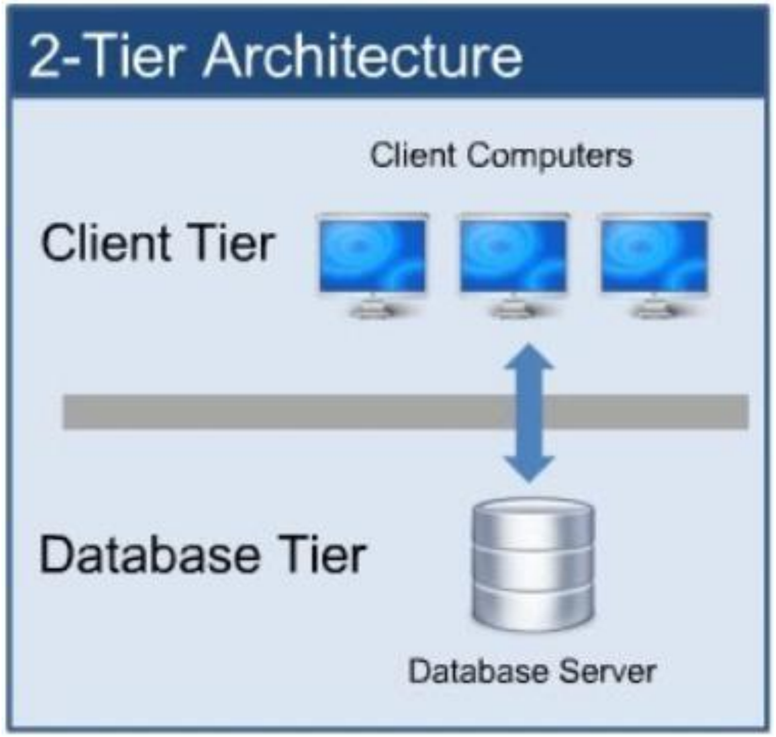
One-tier Architecture

- Has Presentation layer, Business layer and Data layers at the same tie.
- As the name suggested, all the layers and components are available on the same machine.
- *Browser (HTML/CSS/JS)*
- Ex: *MP3 players, Microsoft Office, Notepad, Paint, Calculator Because in these applications:*
  - *The **UI (presentation)**, the **logic (application)**, and the **data (files, settings, preferences)** are **all handled on the same system**.*
  - *There is **no external server**, **no database layer**, and **no network dependency***



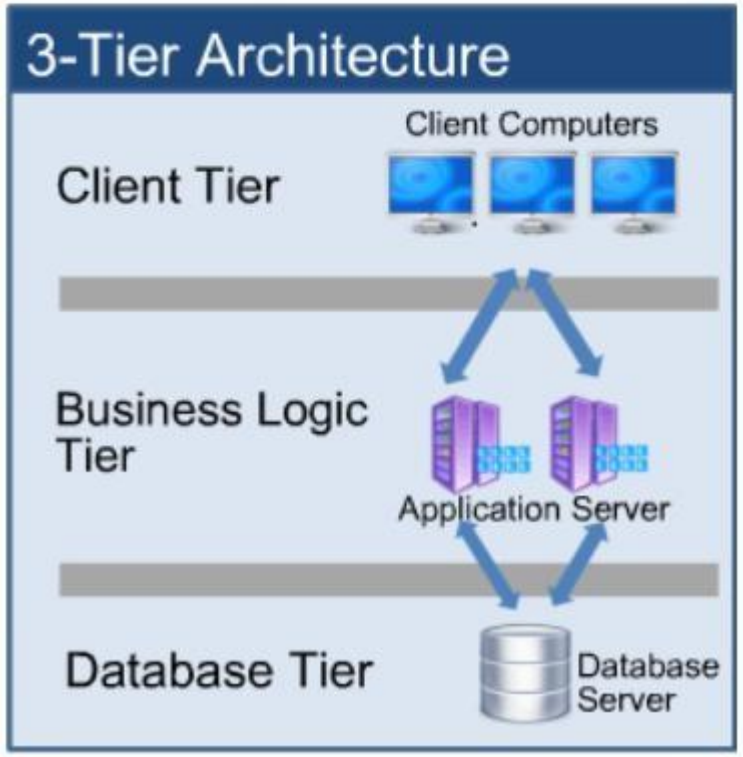
Two-tier Architecture

- The **client** tier handles both **presentation** and **application layers** and the **server** handles the **database layer**.
- AKA **Client-Server Application**.
- **Communication** takes place between the **Client and the Server**
- The **client** system **sends the request to the server** system and the **server** system **processes the request and sends** the **response** back to the **client** system.



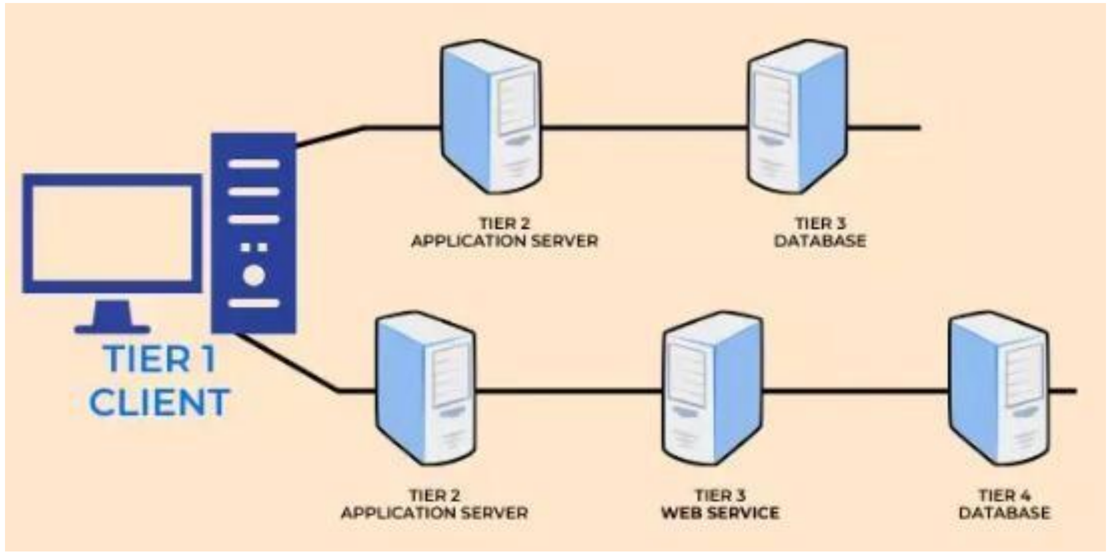
Three-tier Architecture

- All three **major layers** are **separated** from each other.
- **Presentation layer resides at client tier**.
- **Application layer** acts as **middleware** and **lies at business tier**.
- **Data layer** is **available at data tier**.
- This is a **very common architecture**.



N-tier Architecture

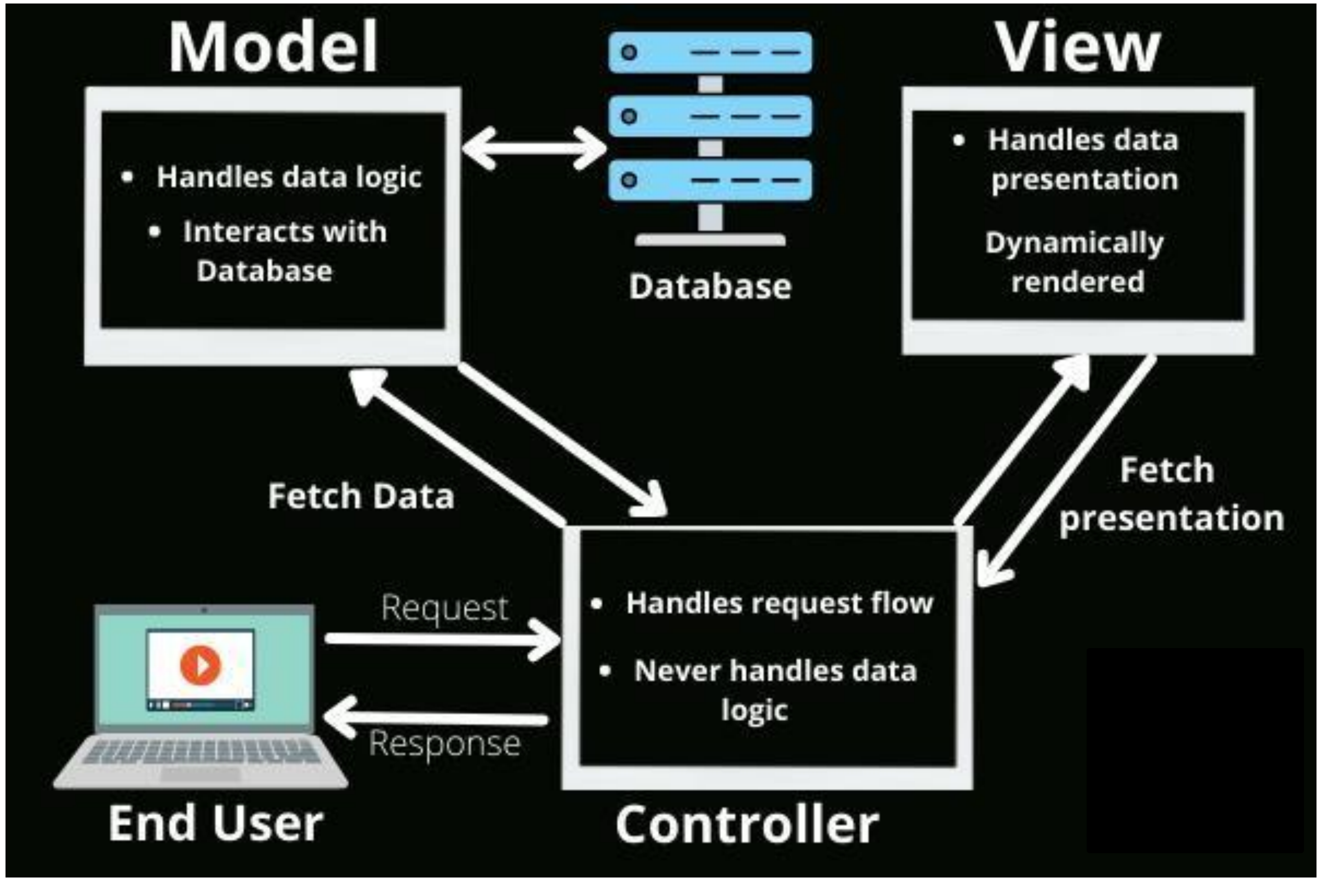
- Called **distributed architecture / multi-tier architecture**
- **Similar** to **three-tier architecture** but the **number** of the **application server is increased** and **represented in individual tiers** in order to distribute the business logic so that the **logic can be distributed.**



MVC Architecture

- What is MVC?
  - MVC stands for Model–View–Controller.
  - It is a design pattern used for developing software applications.
  - It separates the application logic into three interconnected components:
    - Model
    - View
    - Controller

Components of MVC

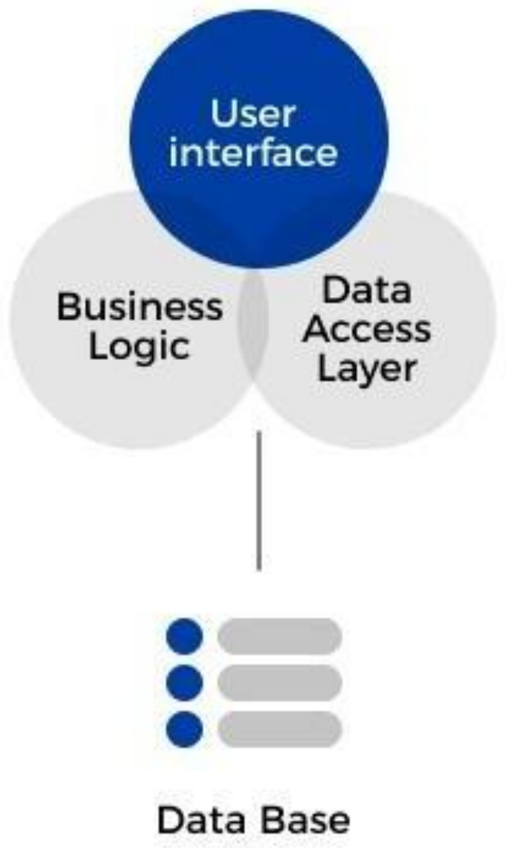


Monolithic Architecture

- Application is built as **one large unit**.
- All modules are **interconnected**.
- Deployed as a **single package**.
- Suitable for startups or simple applications.

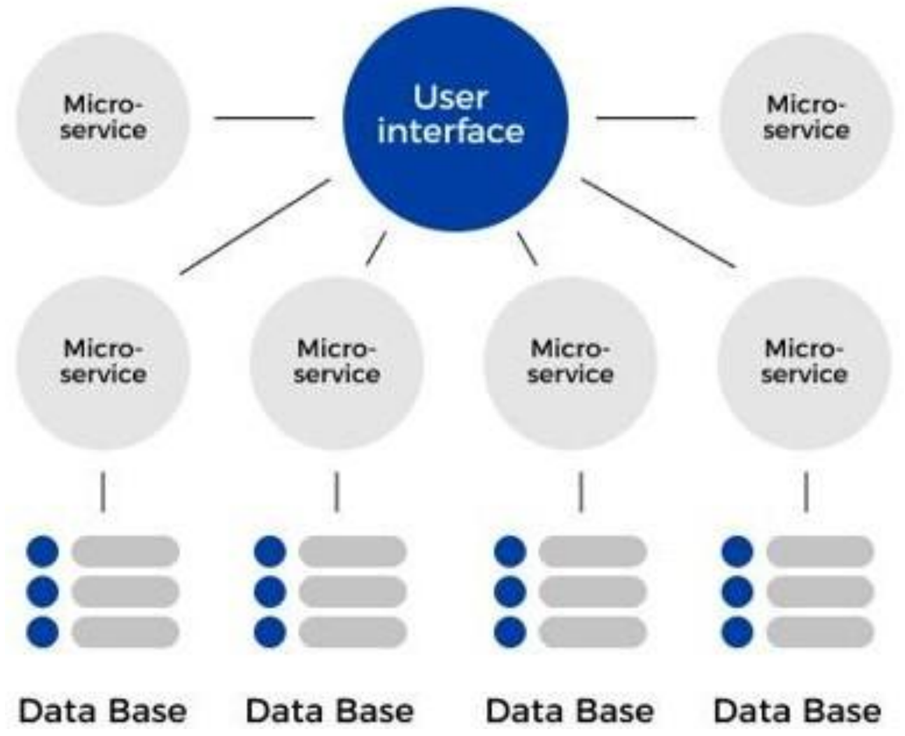
Challenges:

  – Scalability issues
  – Slow development and deployment
  – Limited technology flexibility
  – Difficult to upgrade
  – A failure in one function dose crash the entire application.



Introduction to Microservices

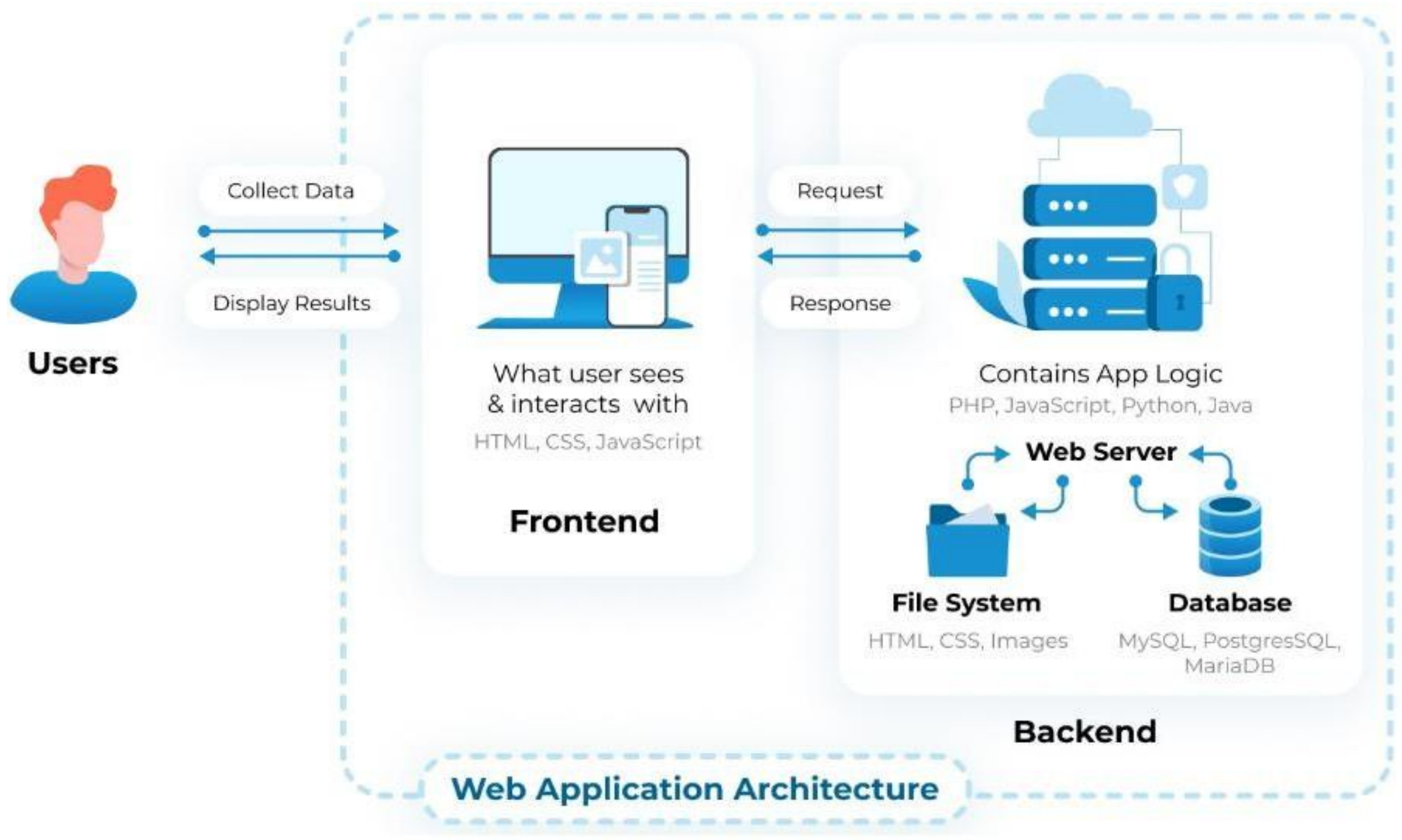
- Application functions broken into **multiple independent services.**
- Services **handle specific tasks.**
- Individual **services can scale** based on demand.
- Services **communicate via APIs** but **function independently.**
- A **failure** in **one service doesn't crash** the entire **system.**



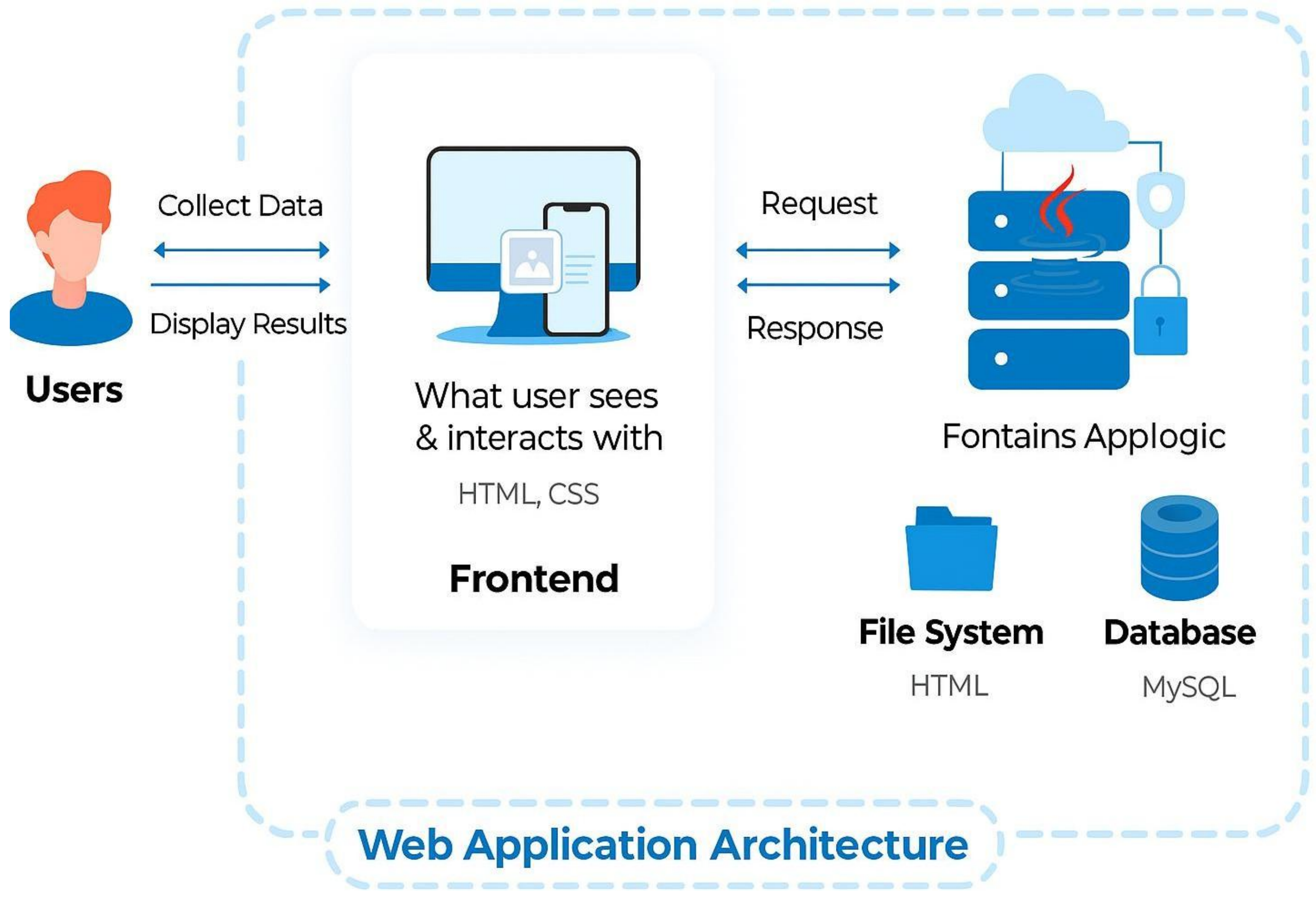
Introduction to Microservices

- When to Uses:
  - Large, complex applications
  - High scalability needs
  - Continuous deployment and updates
  - Multiple development teams
  - Cloud-native applications
  - Need for high availability
- Challenges:
  - Increased complexity
  - Network latency
  - Security challengers
  - Higher initial development effort

Web Application Architecture



With Java


Users
Collect Data
Display Results
What user sees & interacts with
HTML, CSS
Frontend
Request
Response
Fontains Applogic
File System
HTML
Database
MySQL
Web Application Architecture

What is a Servlet?

- A **Servlet** is a **Java program** that runs on a **web server** and handles **HTTP requests** and **responses**.
- It is used to:
  - Process user input from web forms (like login pages)
  - Interact with databases
  - Dynamically generate web content (HTML, JSON, etc.)
  - Manage sessions, cookies, and redirects

Why Servlets?

- Servlets are the foundation of Java web applications.
- Modern frameworks like Spring Boot are built on Servlet APIs.
- Understanding Servlets helps you grasp how requests, responses, and web containers work.

Modern Development Setup

- IDEs: **IntelliJ IDEA, Eclipse**, VS Code
- Servers: **Apache Tomcat 10+,** embedded Jetty
- Java: **JDK 17+** recommended
- Build Tools: **Maven** / Gradle
- Database: **MySQL** / PostgreSQL
- **Spring Boot** (for abstraction and simplification)

Where Are Servlets Used?

- - Login and authentication forms
- - Dashboard data rendering
- - PDF or Excel generation
- - Dynamic form processing
- - Legacy systems still in production in finance, telecom, etc.

Beyond Servlets: Transition to Frameworks

- - Spring Boot internally uses Servlets (via DispatcherServlet)
- - Simplifies setup: embedded server, auto-config
- - REST APIs are often built using @RestController on top of Servlet layers
- - Understanding Servlets makes learning Spring easier

Why Spring?

“We already learned how to build web apps using Java Servlets and JSP. But as applications grow, Servlets become hard to manage. That's where Spring comes in — it simplifies everything.”

Core Concepts of Spring Framework

Core Concept	What to Cover
Inversion of Control (IoC)	Explain with a real-life analogy (e.g., restaurant waiter calling kitchen vs kitchen auto-notifying waiter)
Dependency Injection (DI)	Show how Spring “injects” objects instead of you creating them manually
Beans & Configuration	What is a Spring Bean, and how Spring manages it
ApplicationContext	Basic intro to Spring container

What is Spring Boot?

- Built on top of Spring Framework
- Auto-configured
- Embedded Tomcat
- No web.xml or XML config needed

Install Spring Boot

JDK 24

IJ

Maven™

spring boot

spring initializr