SEEC-304

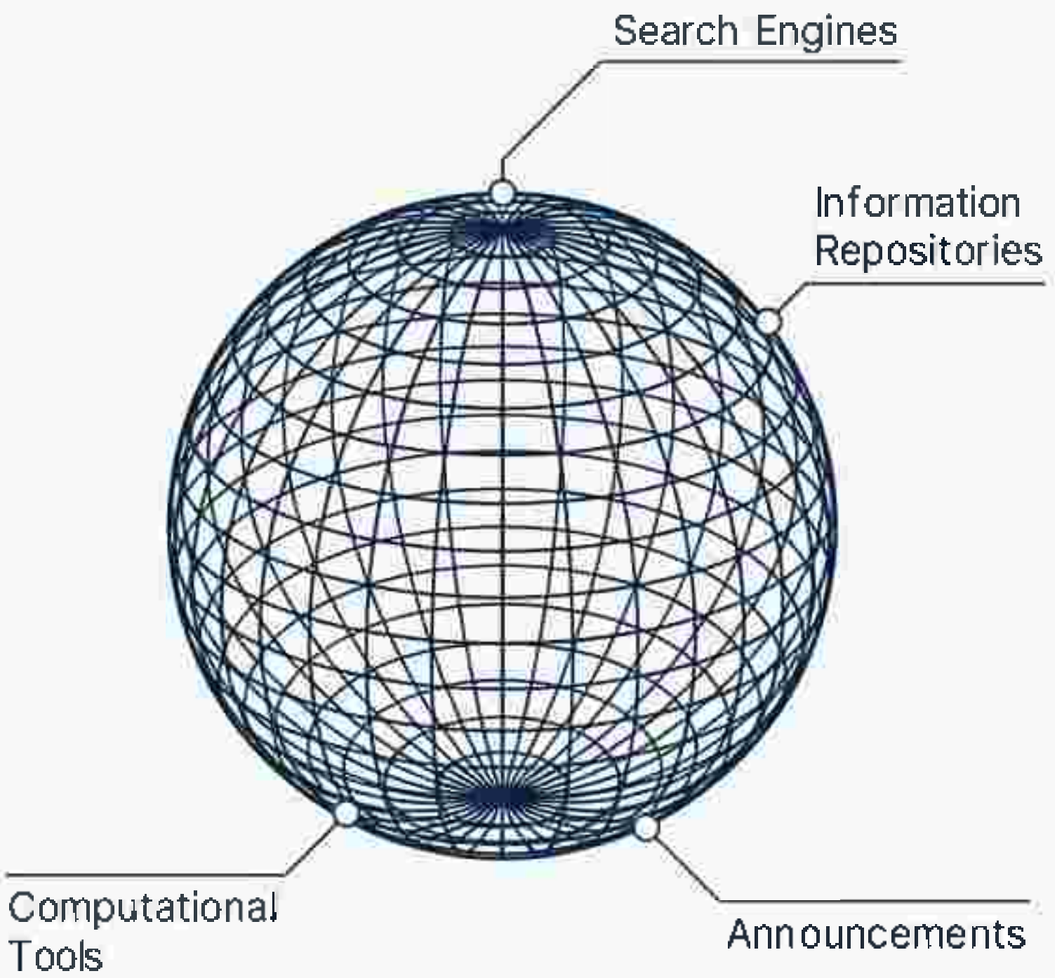
Web Engineering

Introduction to Web Engineering
Principles & Architectures

Dr. Ahmad Mustafa
Department of Software Engineering
University of Sargodha

The Omnipresent Web

- Global and permanent availability.
- Uniform access to information.
- Democratization of production: Shift from consumption to participation.
- Anyone can produce and publish contents.



Basic Paradigms & Architecture

Hypertext + Internet

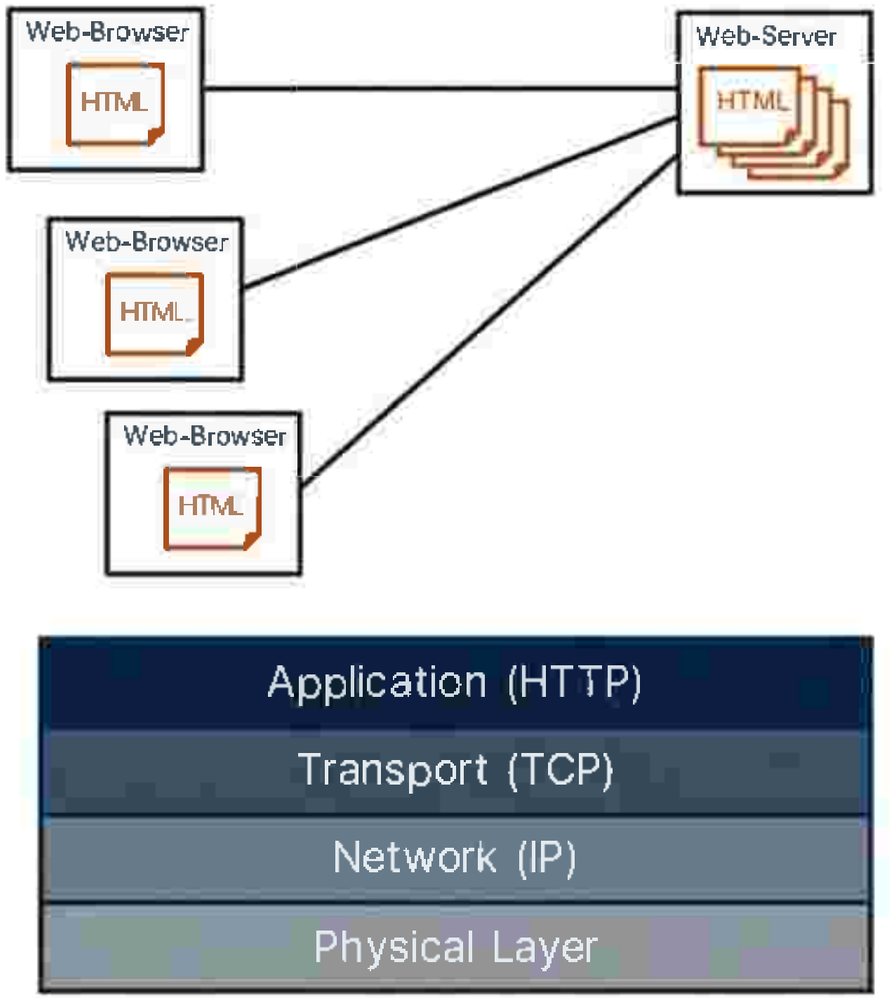
- Textual documents **interconnected by links**.

HTML

- HyperText Markup Language (Structure).

HTTP

- **HyperText Transfer Protocol** (Transport).



Evolution of the Network (1969–Present)

ARPA Network

First connection: Stanford,

Protocols

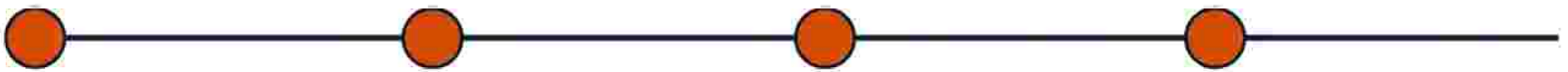
SMTP (Mail) & FTP (File Transfer).

The Web

Tim Berners-Lee introduces WWW.

W3CFormed

World Wide Web Consortium established.



1969

First connection: Stanford, UCLA, UCSB, Utah. Introduction of TCP/IP.

1973

SMTP (Mail) & FTP (File Transfer).

1989

Tim Berners-Lee introduces WWW.

1994

W3C Mission:

To lead the Web to its full potential by developing protocols and guidelines ensuring long-term growth.

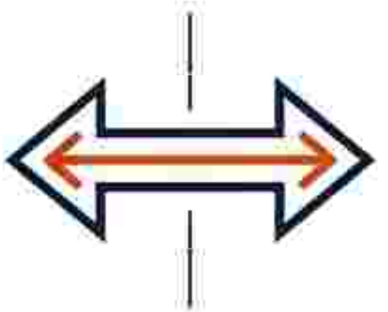
Defining the Web Application

"A software system based on W3C technologies and standards that provides Web-specific resources such as content and services through a user interface—the Web browser."

KAPPEL ET AL. 2004

Traditional Software

Installed locally on device.



- **Web Application**

  Logic resides on server.
  Browser acts as universal client.

The Spectrum of Applications: Static to Interactive

Informational

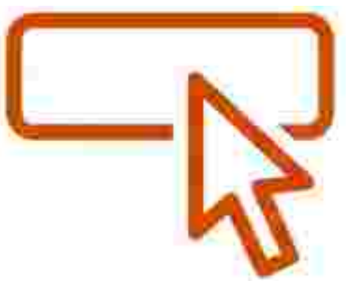
Read-only content.
Simple navigation.
(e.g., Online manuals).

Download

User downloads files from FTP servers.

Customizable

Content adapted to specific user needs (e.g., Personalized home pages).



Interactive

Dynamically generated content. Uses `Forms & CGI`. (e.g., Public transport schedules).

Transactional & Workflow Systems

Transactional

Read/write actions. Database transaction management. Focus on efficiency and consistency.
Examples: Online banking, E-shopping, Reservation systems.

Workflow-based

Complex services requiring a structured flow of activities.

Examples: B2B Integration, E-Government, Patient healthcare workflows.

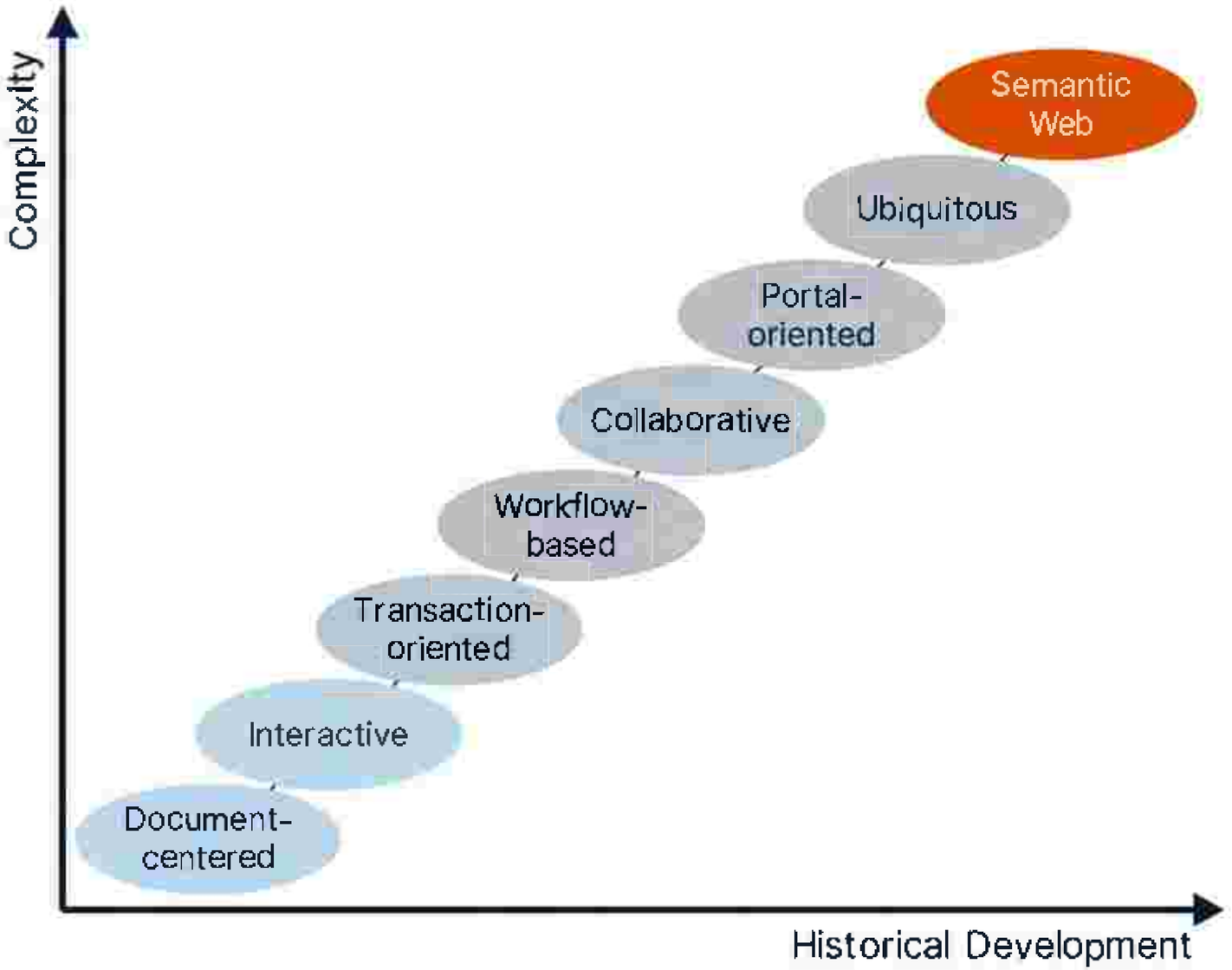
The Frontier: Collaborative, Ubiquitous, & Semantic

Collaborative: Unstructured flows, high communication (Wikis).

Portal-oriented: Single point of access (Enterprise portals).

Ubiquitous: Personalized, anytime/anywhere access (Context-aware).

Semantic: Knowledge management, derived knowledge (Web 2.0).



Engineering Challenge I: The Product

Content is King

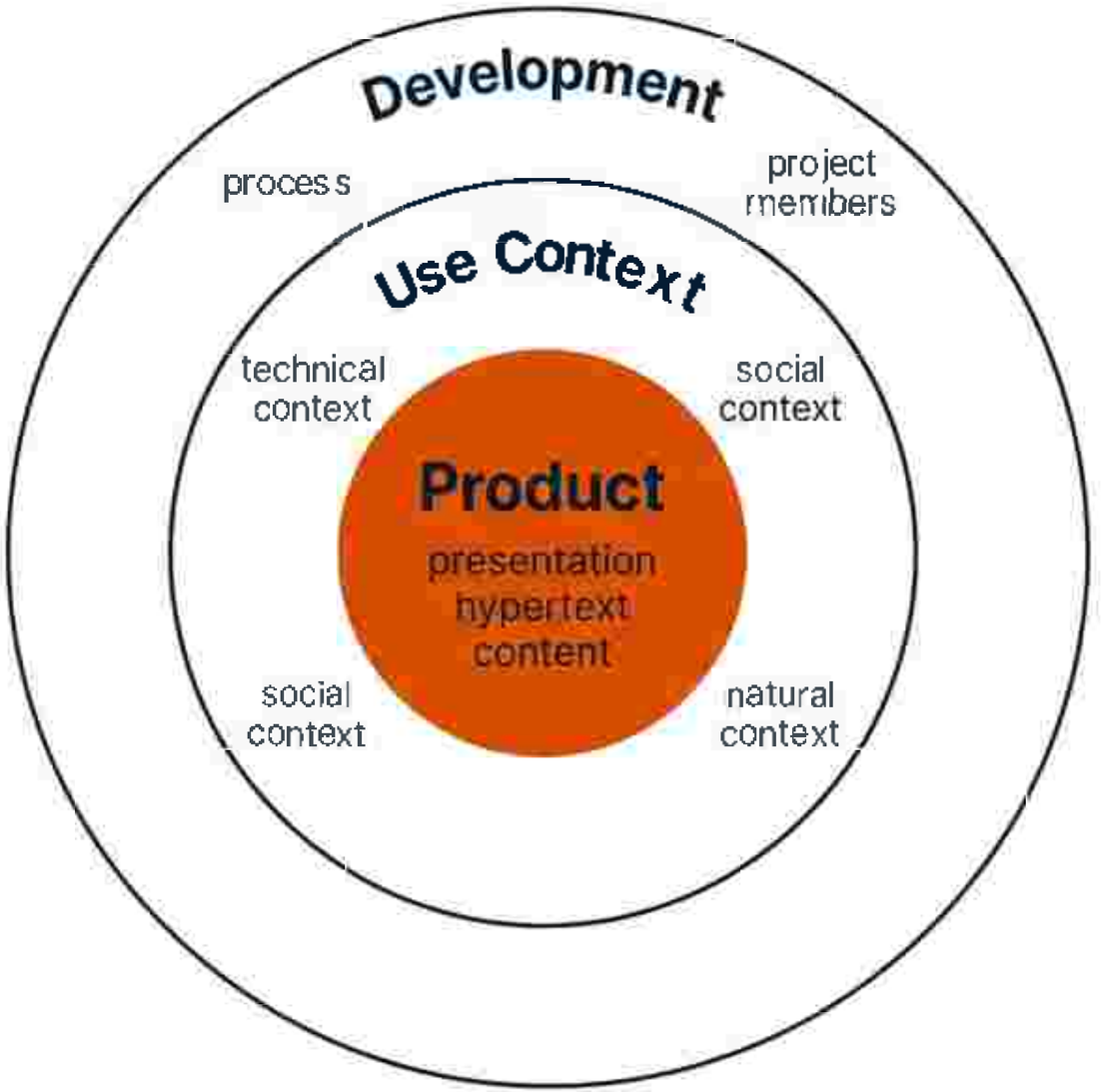
Objective: Communicating text, multimedia, audio, video. Quality and 'actuality' are critical.

Hypertext & Disorientation

Non-linearity leads to 'loss of sense of locality'. Risk of cognitive overload.

Aesthetics

High demands on look and feel. Must be intuitive without documentation.



Engineering Challenge II: Usage & Context

The Unknown User

Unknown numbers (scalability), anonymous users, limited knowledge of user preferences.

Multiculturality

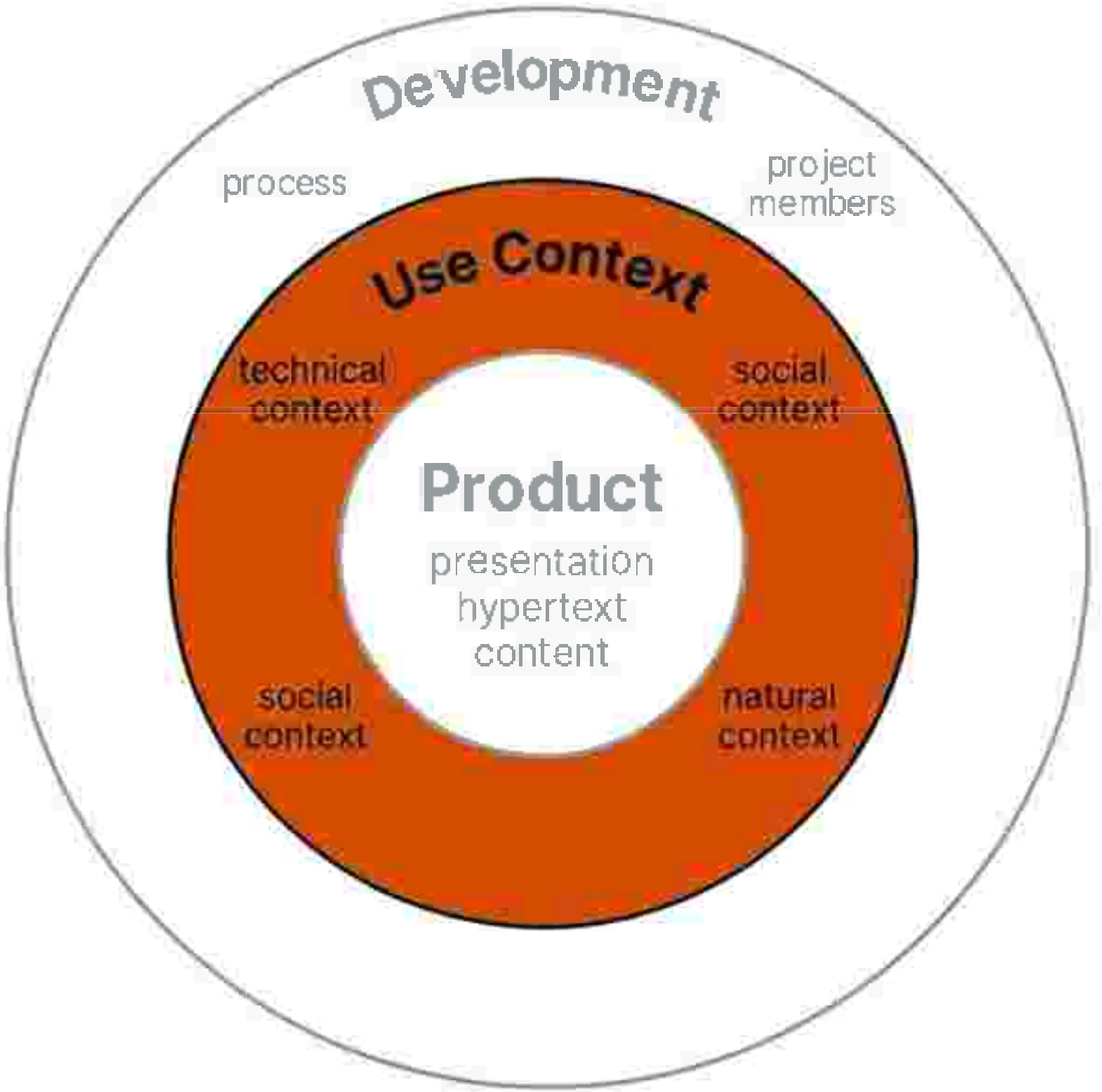
Internationalization requirements, regional and linguistic differences.

Technical Uncertainty

Unknown bandwidth, reliability issues.

Multi-platform Delivery

Must support diverse devices (PC, Tablet, Mobile) and browser versions.



Engineering Challenge III: The Development Process

Multidisciplinary Teams

A unique mix of Print Publishing (Arts/Marketing) and Software Development (CS/IT).

Inhomogeneity

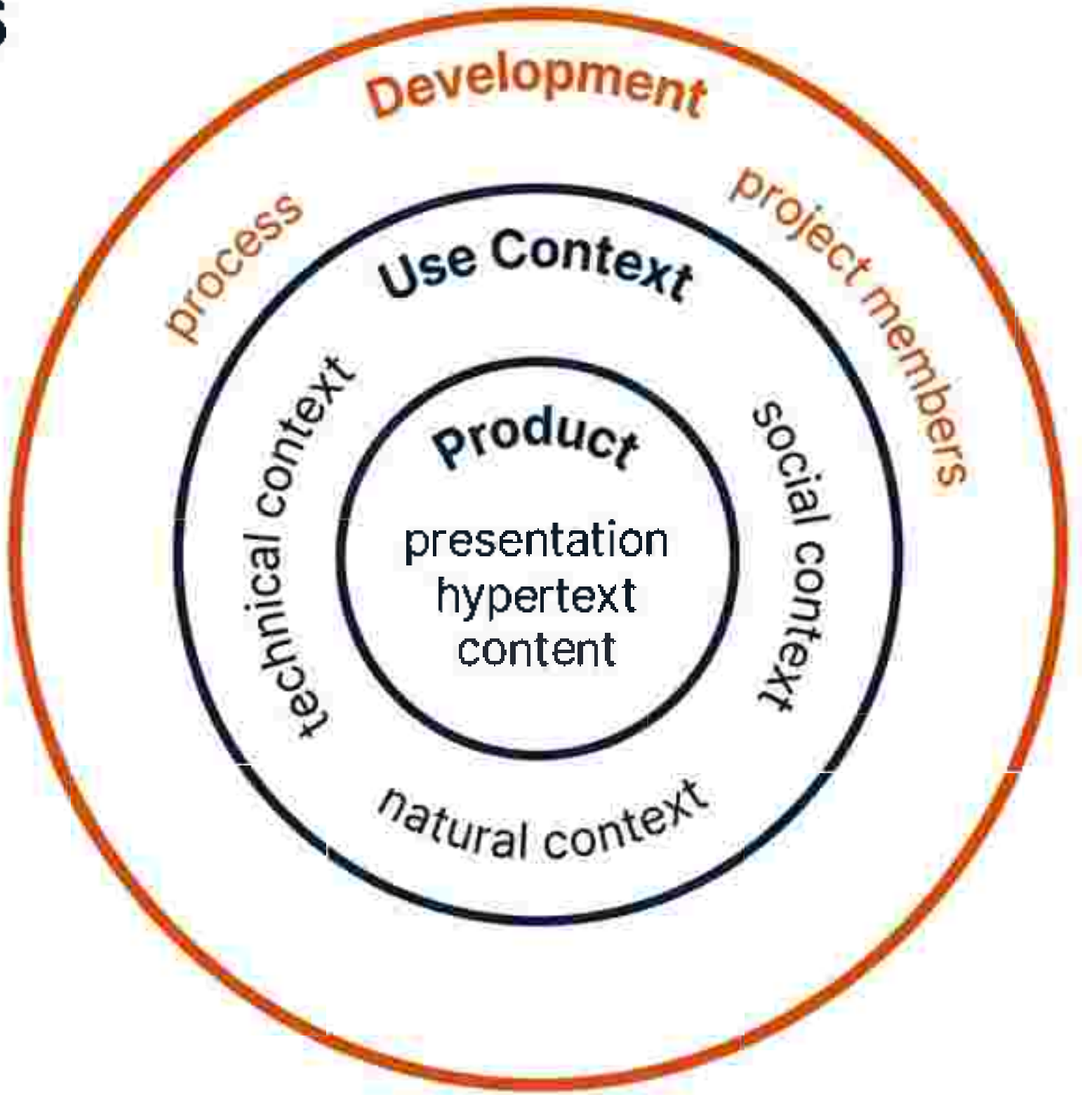
Split control: Server is controlled by developer; Browser is controlled by the user.

Immaturity

Buggy components due to time-to-market pressure.

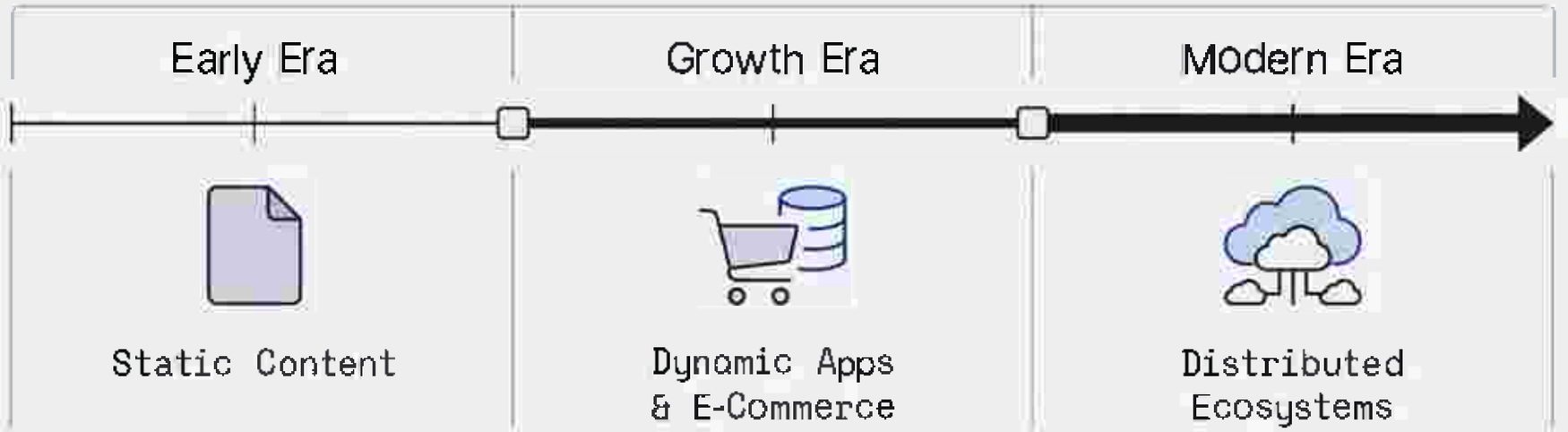
Process

Requires Agile, light-weight processes to handle changing requirements.



The Catalyst for Change

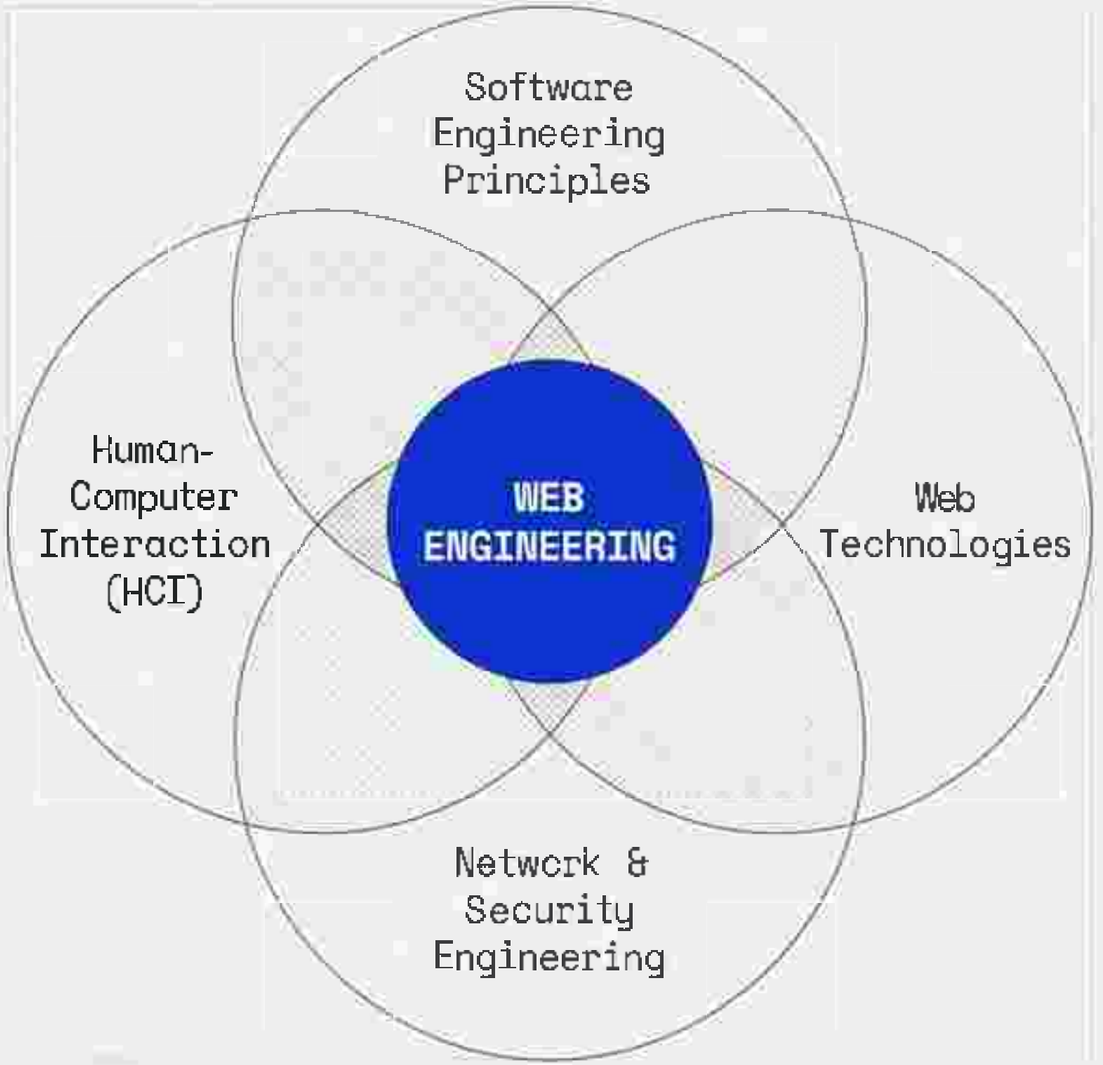
The rapid growth of the Internet transformed software systems from static, local installations into **distributed, user-centric ecosystems.** Traditional software development approaches proved insufficient when faced with the web's **unique pressures:** rapid changes, massive scalability requirements, and global accessibility.



Key Takeaway:
Increasing complexity demanded a shift from ad-hoc codingto engineering-level discipline.

Defining the Discipline

Web Engineering is not just coding for the browser. It is a systematic, disciplined, and measurable approach to the development, deployment, operation, and maintenance of web applications.



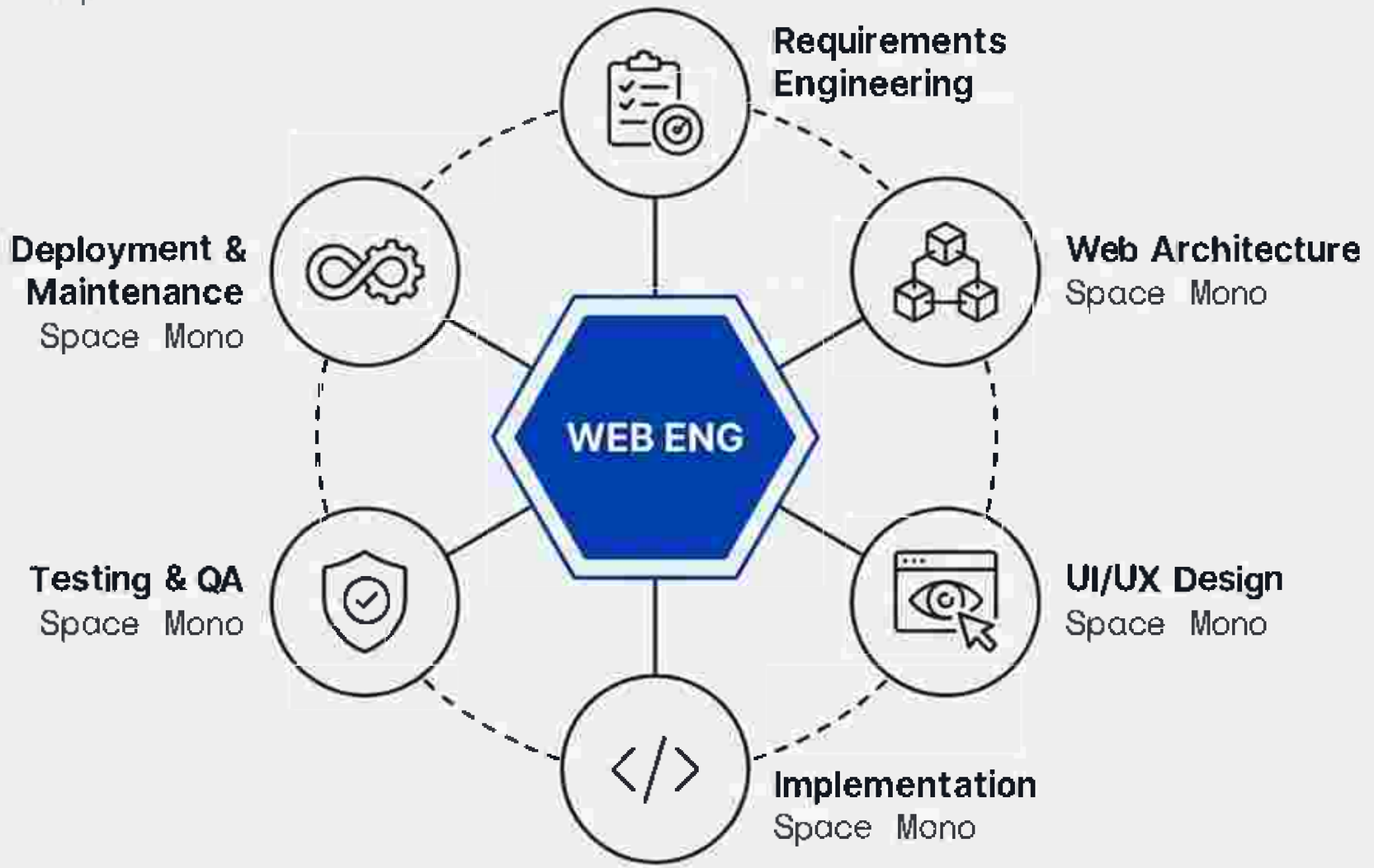
The Great Divergence: Web vs. Traditional Engineering

Web applications face unique constraints that traditional models cannot address, specifically the need to optimize for the unknown and the immediate.

Dimensions	Traditional SE	Web Engineering
Development Cycle	Long & structured	**Rapid & iterative**
User Base	Limited & known	Global & anonymous
Deployment	Periodic releases	Continuous deployment
Content Model	Code-centric	Code+ Content
Change Velocity	Slow	Very rapid

The Six Pillars of Web Engineering

To maintain quality, reliability, and maintainability in a high-velocity environment, Web Engineering integrates six core components.



Managing Uncertainty and Volatility

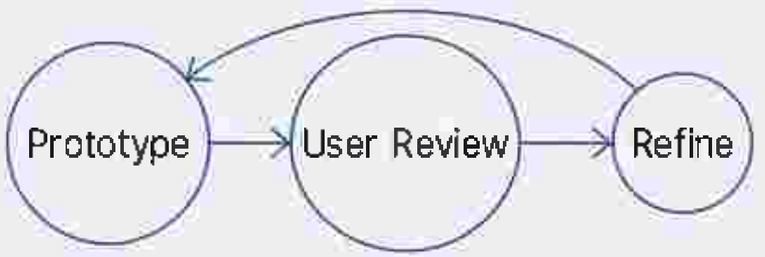
The Challenge

In web systems, requirements are volatile and often unclear at early stages. The

"waterfall" method of defining everything upfront fails here.

The Approach

- **Agile Alignment**: Embracing Scrum and Kanban to handle short cycles.
- **Feedback Loops**: Rapid prototyping and continuous user feedback replace static docs.
- **Focus Areas**: Heavy emphasis on non-functional requirements like security, scalability, and usability.

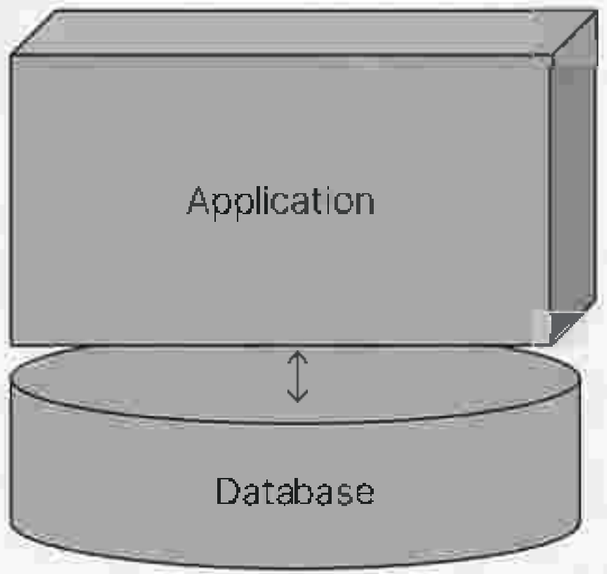


The Feedback Loop

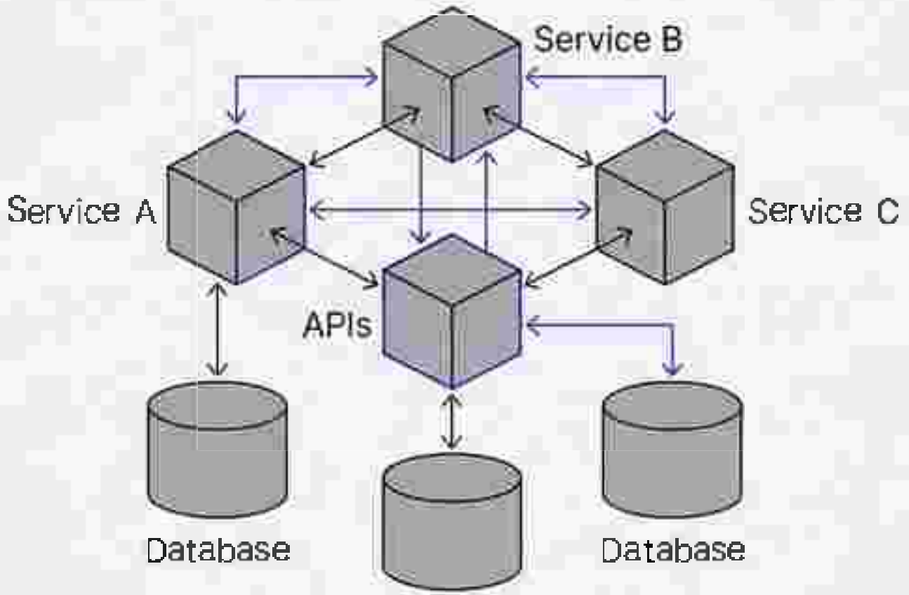
Architecting for Resilience

Architecture defines the structure of the application and determines its ability to survive growth. Modern web engineering prioritizes modularity and fault tolerance.

LEGACY: Monolithic



MODERN: Microservices



Key Objectives

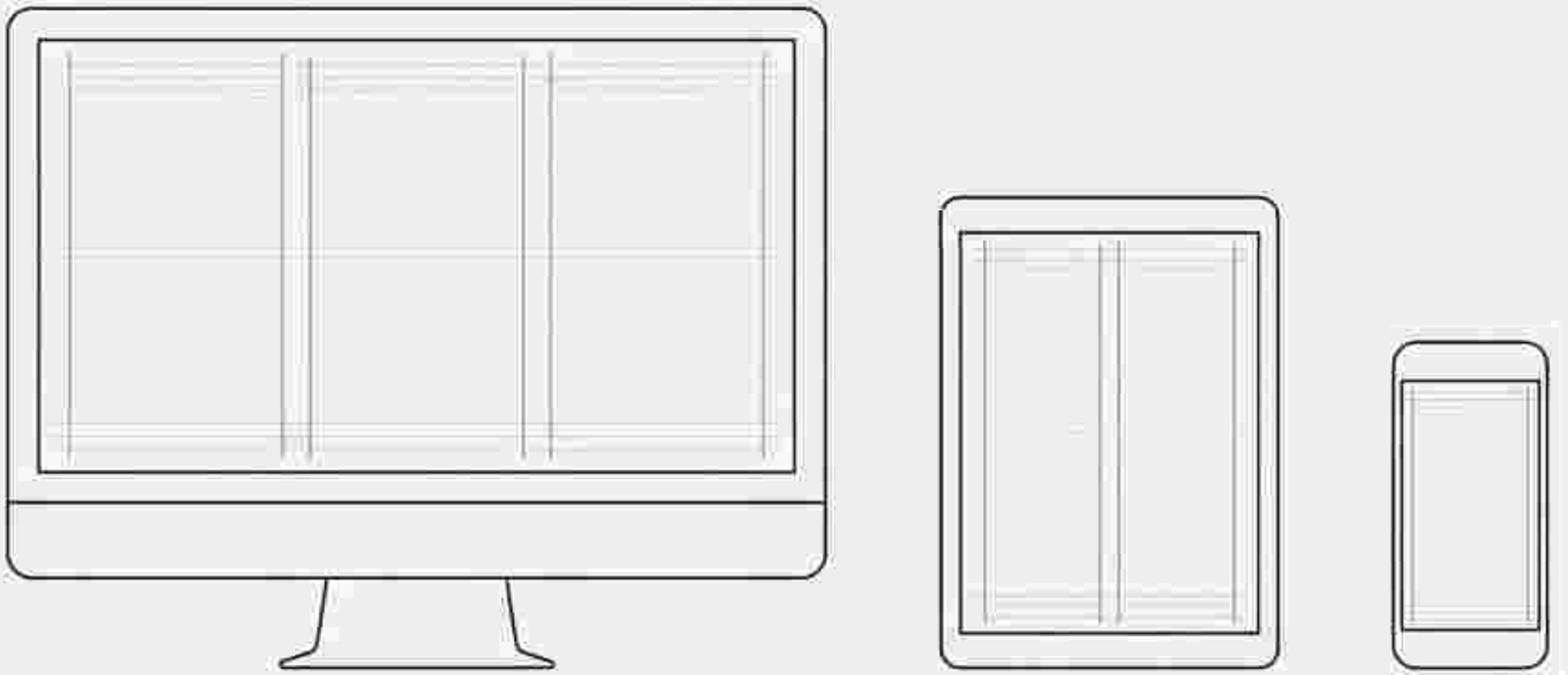
Modularity
Independent updates

Scalability
Handling growth

Fault Tolerance
System survival during failure

The Human Element

Web applications are highly user-facing. Unlike backend systems, the user interface IS the application utility. Poor UX equals system failure.



Responsiveness

Layouts adapt to multi-platform access.

Accessibility

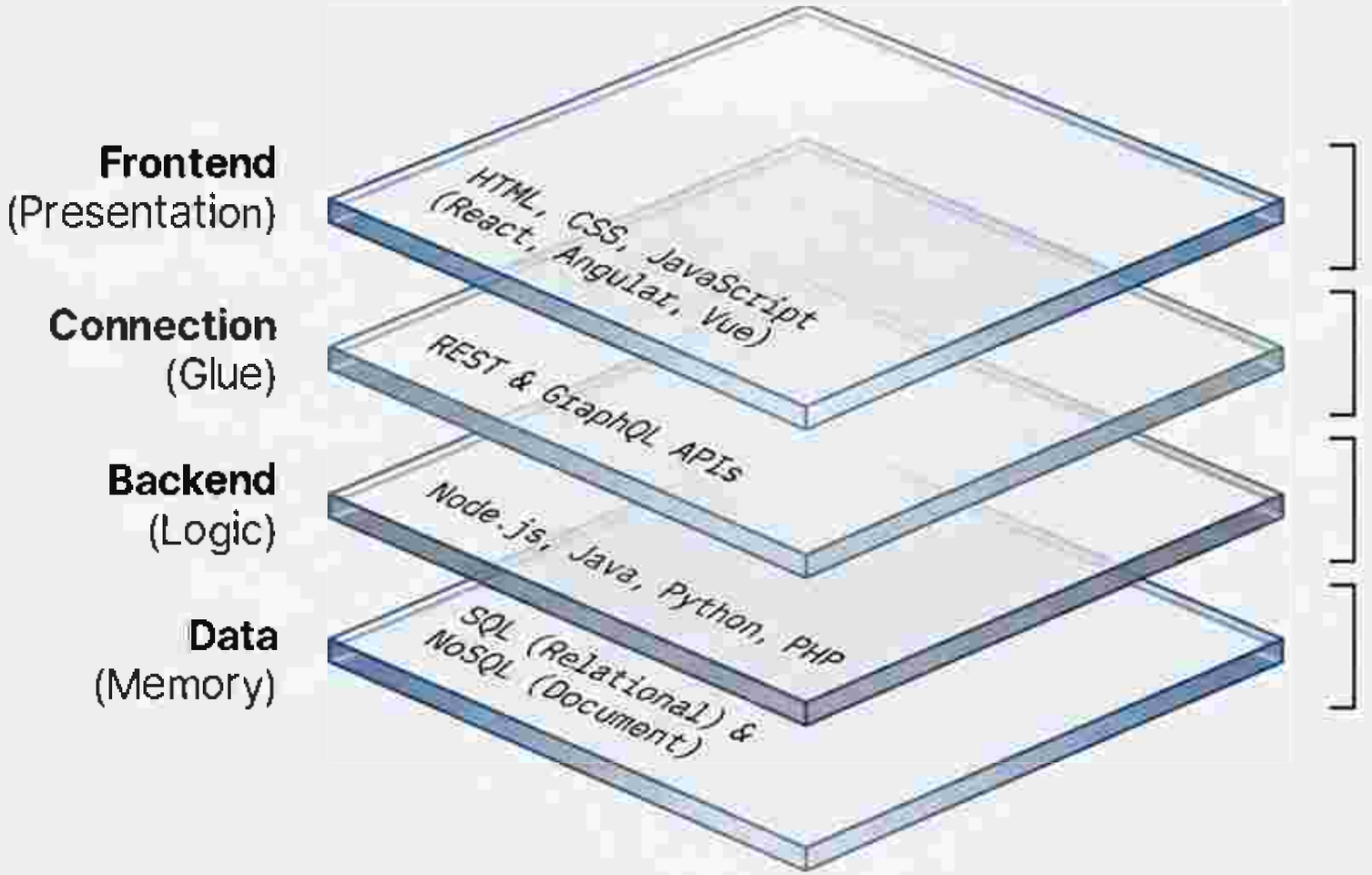
Ensuring access for diverse user bases.

Usability

A core engineering activity, not just design.

The Modern Implementation Stack

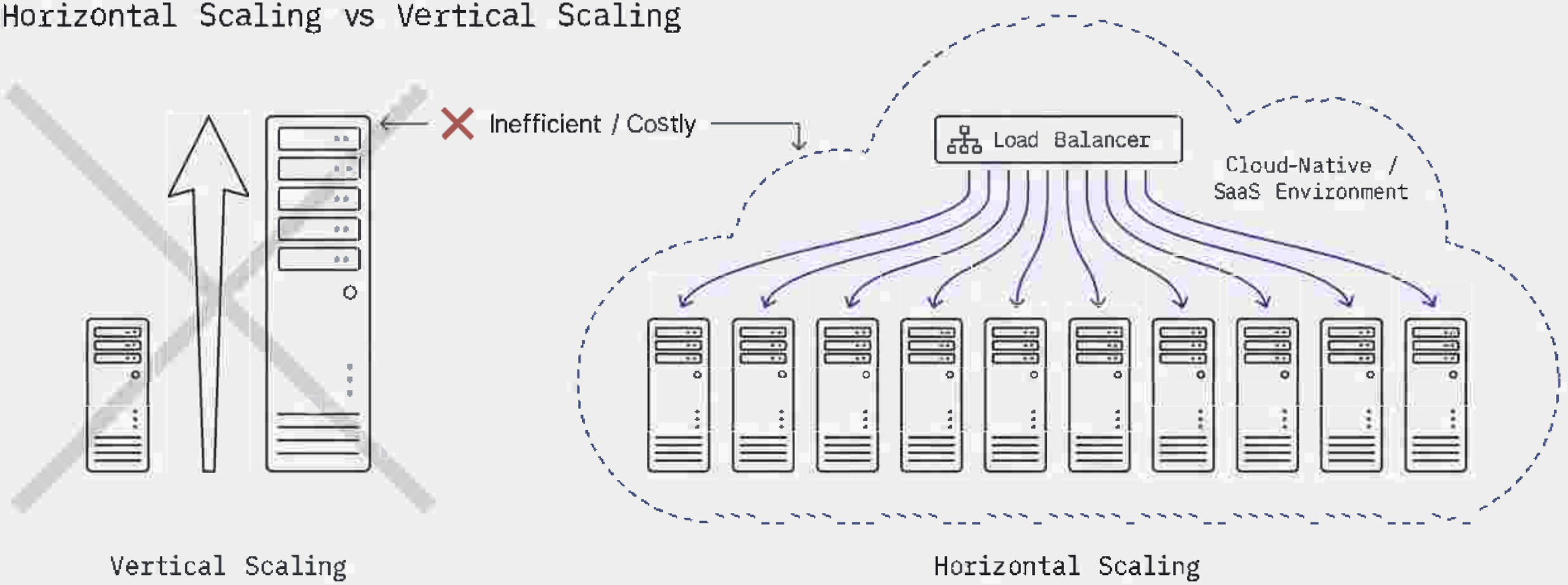
Implementing a web system requires orchestrating a diverse set of technologies across four distinct layers.



Engineering for Scale

Modern systems must withstand high traffic peaks and serve millions of concurrent users without degradation. High availability is a non-negotiable feature.

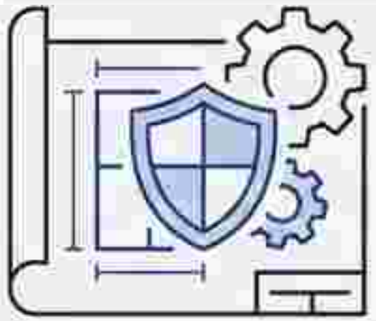
Horizontal Scaling vs Vertical Scaling



Security in an Open World

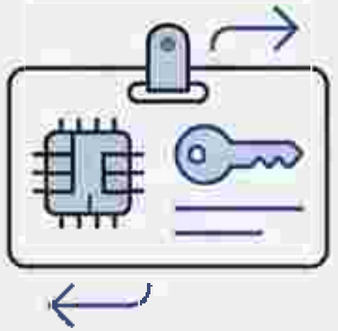
Global accessibility invites global threats. Security must be integrated throughout the lifecycle.

Design Phase



Secure Design Principles: Shifting security "left".

Access Layer



Access Control: Authentication & Authorization.

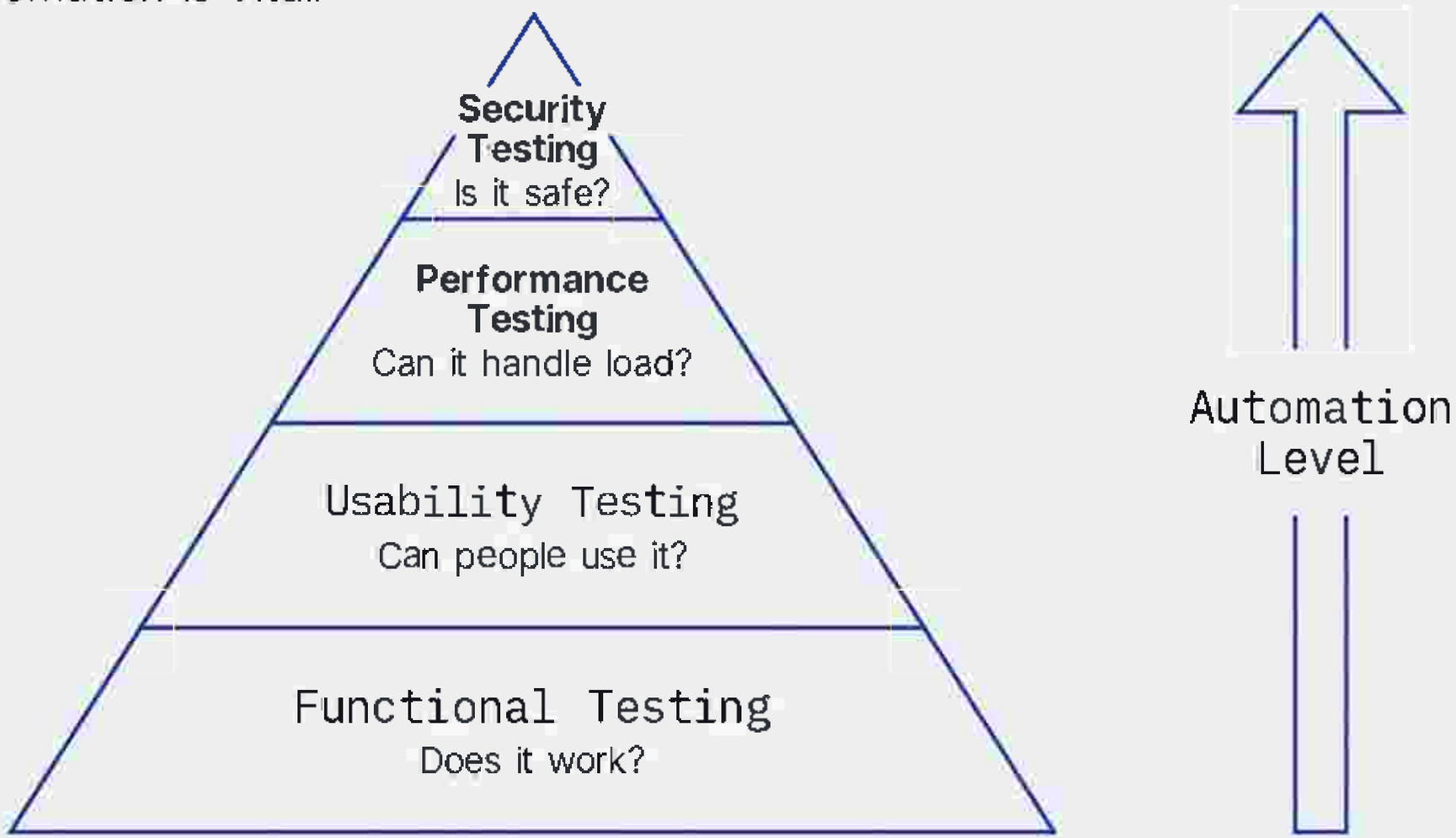
Data Layer

Data Protection: Encryption & Privacy.

Critical Challenge: Balancing open user access with protection against malicious actors.

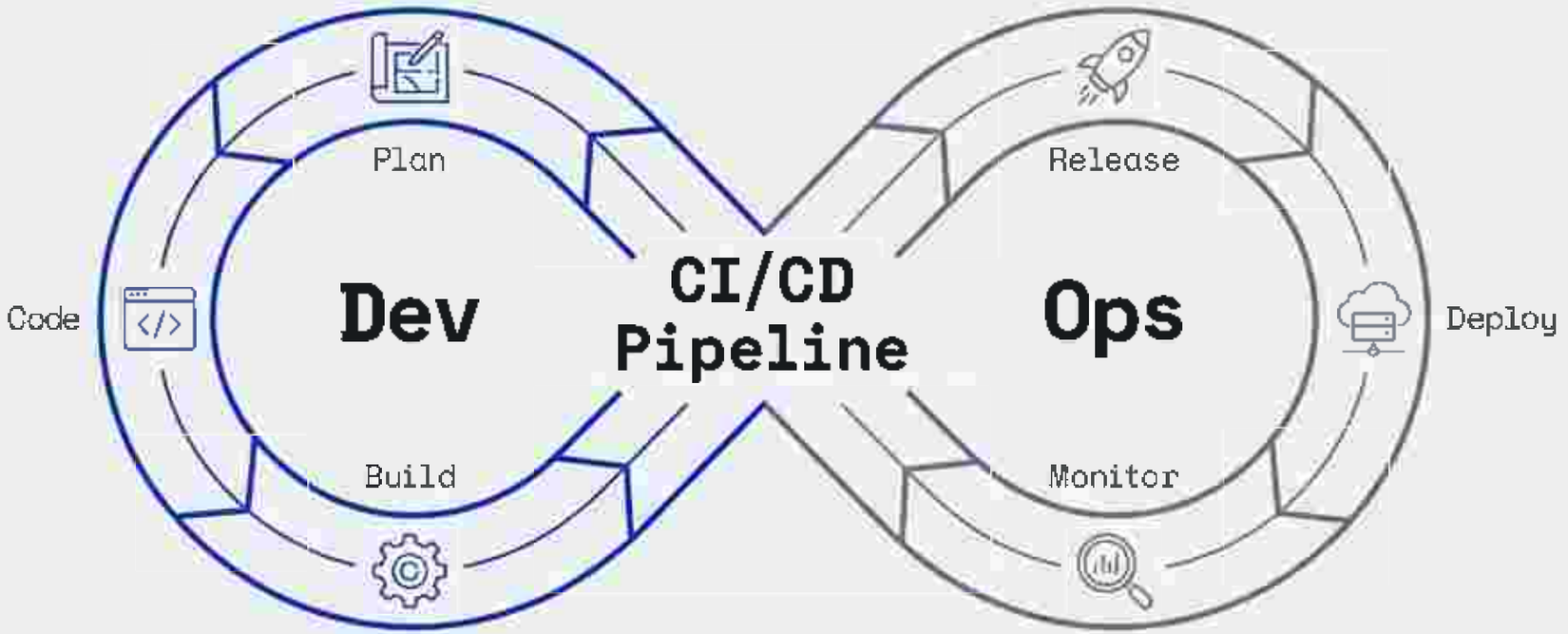
Continuous Quality Assurance

In a continuous deployment environment, testing must be continuous and multi-dimensional. Automation is vital.



The DevOps Engine

Deployment and Maintenance are not the end of the project; they are a continuous loop of operation.



CI/CD Pipelines

Automated Integration/Deployment

Active Maintenance

Bug fixing, enhancements

Integration

Bridging Dev and Ops

Future Vectors

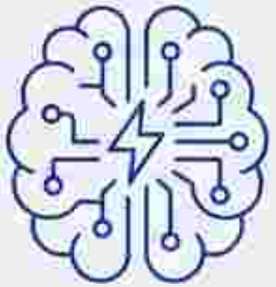
The discipline continues to evolve as the web platform expands its capabilities.

PWAs

Bridging mobile and web.

AI-Driven Systems

Integrating intelligence into flows.

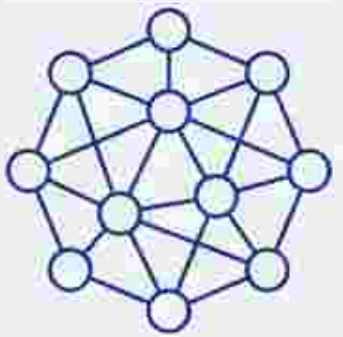


Serverless

Abstracting infrastructure.

Web3.0

Decentralization.



The Essential Discipline

Web Engineering provides the necessary structure to extend traditional software engineering concepts into the modern era. It is the vital framework for building systems that are reliable, scalable, and secure.

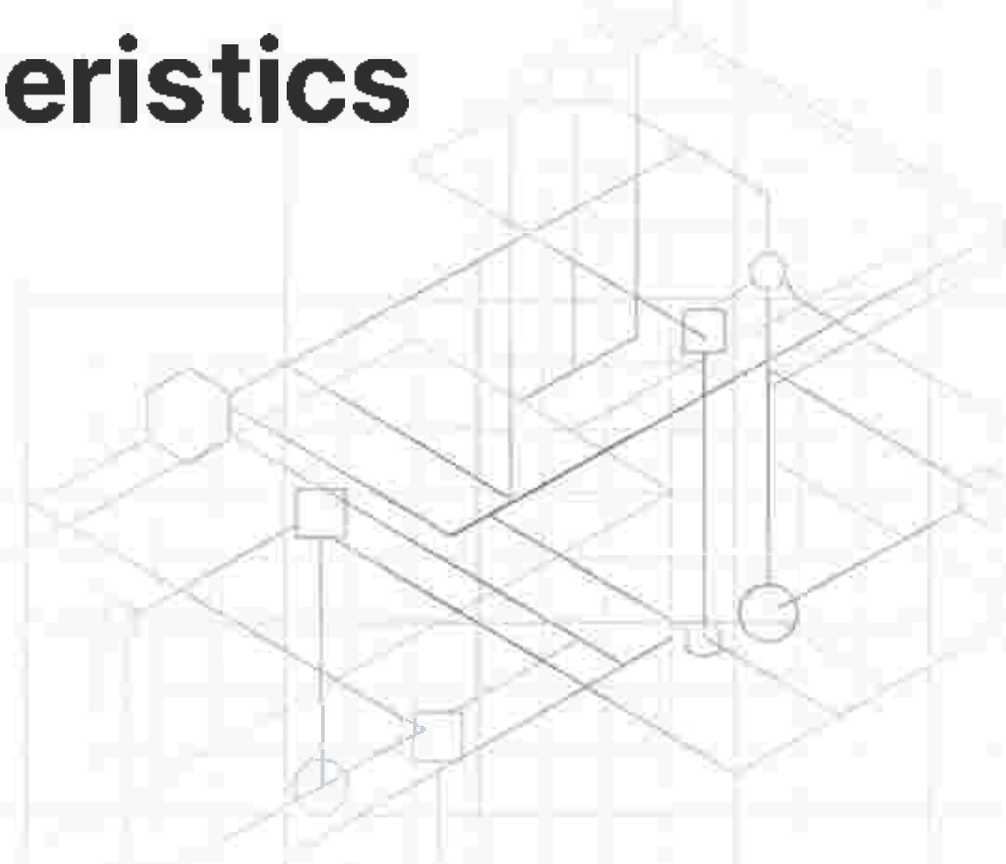
Industry Relevance

Banking Systems	Government Portals	E-commerce Platforms	Learning Management

Final Reflection: The web engineer must possess a unique blend of technical mastery, user empathy, and architectural foresight to navigate the future of software.

Categories and Characteristics of Web Applications

A Structural Taxonomy for Web Engineering

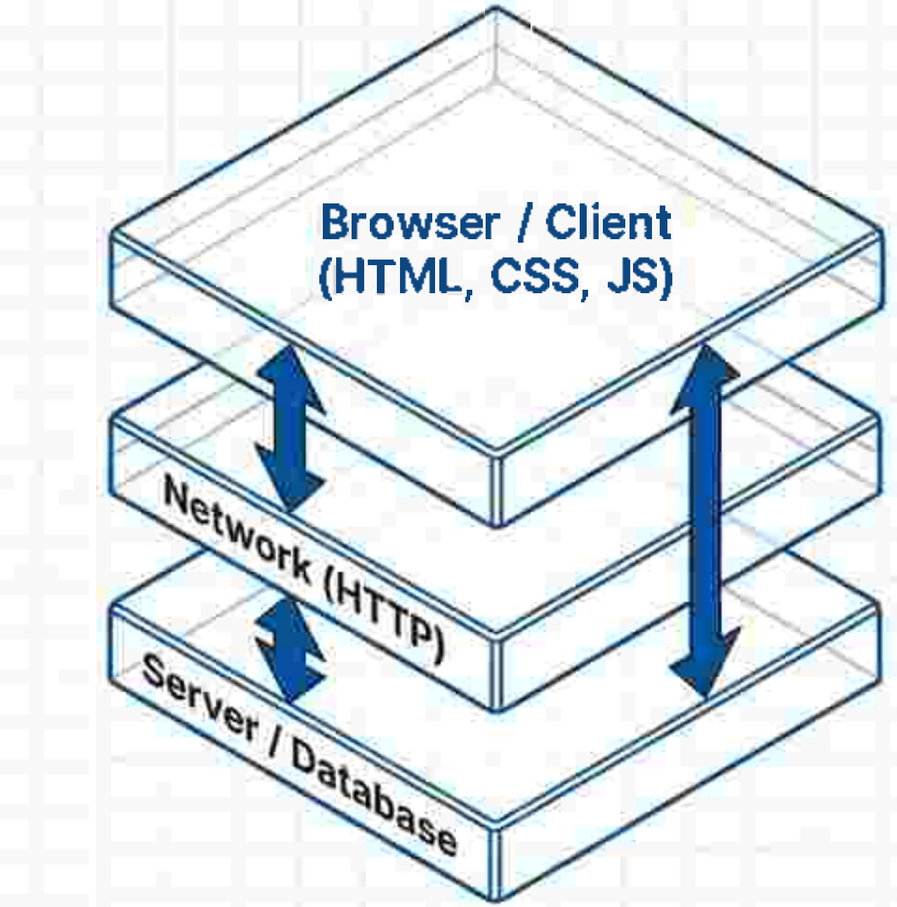


The Backbone of Modern Software Systems

> **Definition:** A software system that runs on a **web server**, is accessed via a **browser**, and relies on **HTTP**, **HTML**, **CSS**, and **JavaScript**.

Why Understanding Matters

- Prerequisite for Proper Design
- Essential for Technology Selection
- Critical for Performance and Security Planning



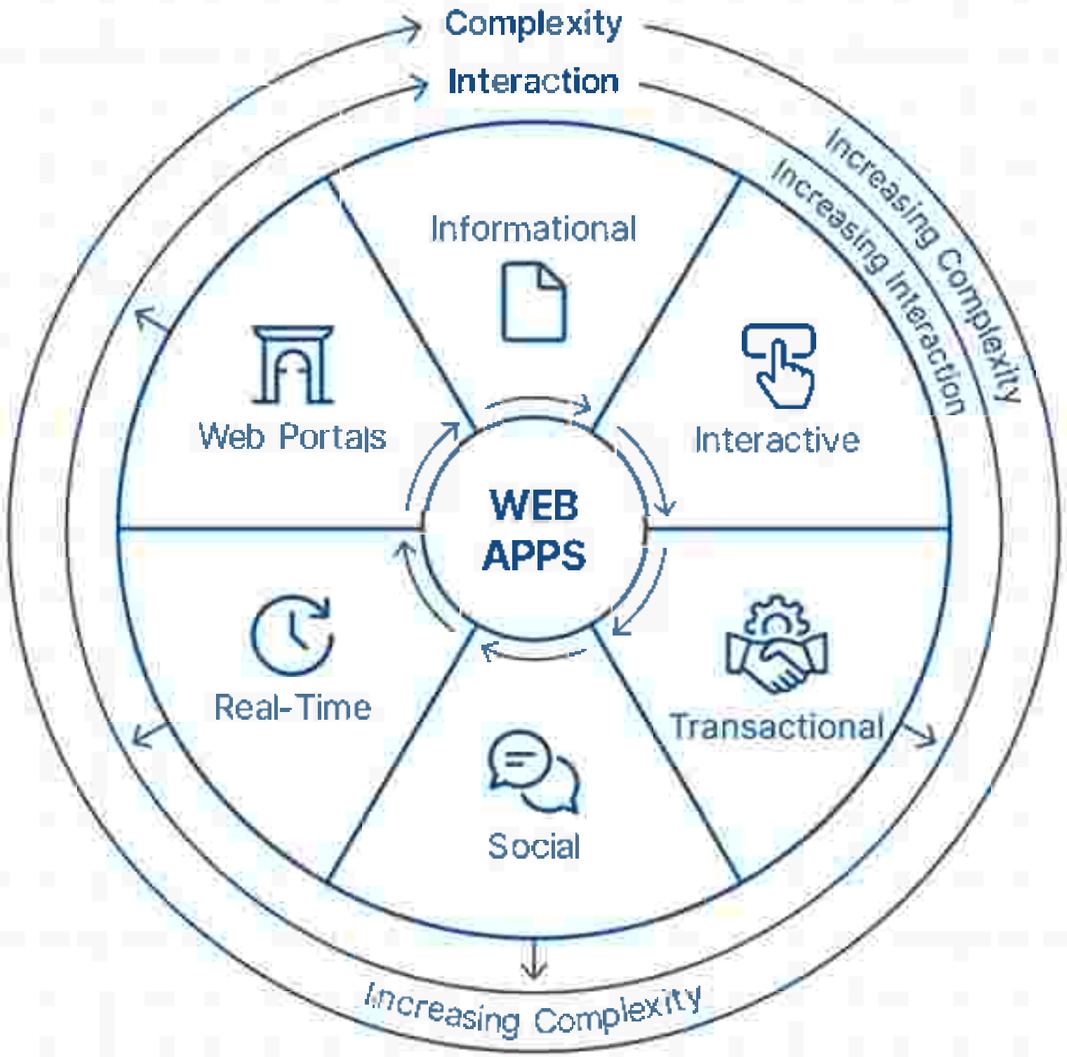
E-commerce

Online Learning

Banking Portals

Mapping the Application Landscape

We classify applications based on three distinct vectors: Functionality, Level of user interaction, and System complexity.



Foundation and Interaction

Informational Web Applications

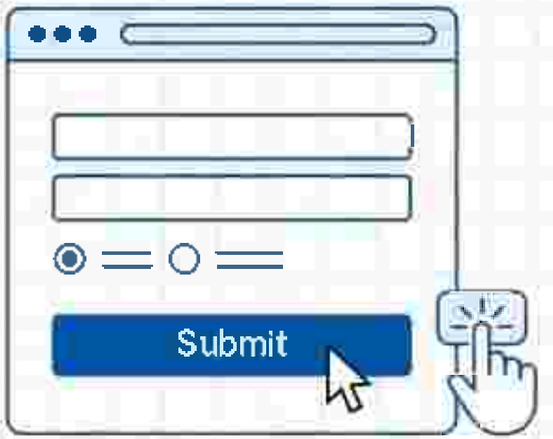
Focus: Providing information; content-driven.

Tech State: Mostly static or semi-dynamic.

Pros: Easy to maintain, low development cost.

Examples: University websites, News portals, Corporate websites.

Interactive Web Applications



Focus: Allowing user interaction; accepting input and providing responses.

Tech State: Dynamic content using scripting technologies; relies on Client-Server interaction.

Examples: Online forms, Search engines, Learning Management Systems (LMS).

Commerce and Centralized Access

Transactional Web Applications

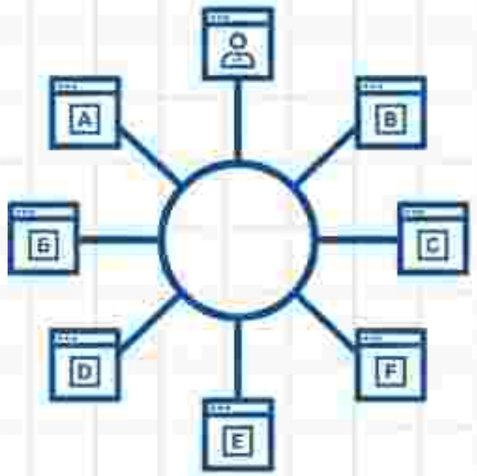
Focus: Supporting business processes and online transactions.

Critical Data: Handles sensitive user and financial data.

Requirements: High security, data integrity, reliability.

Examples: E-commerce systems, Online banking, Airline reservation systems.

Web Portals



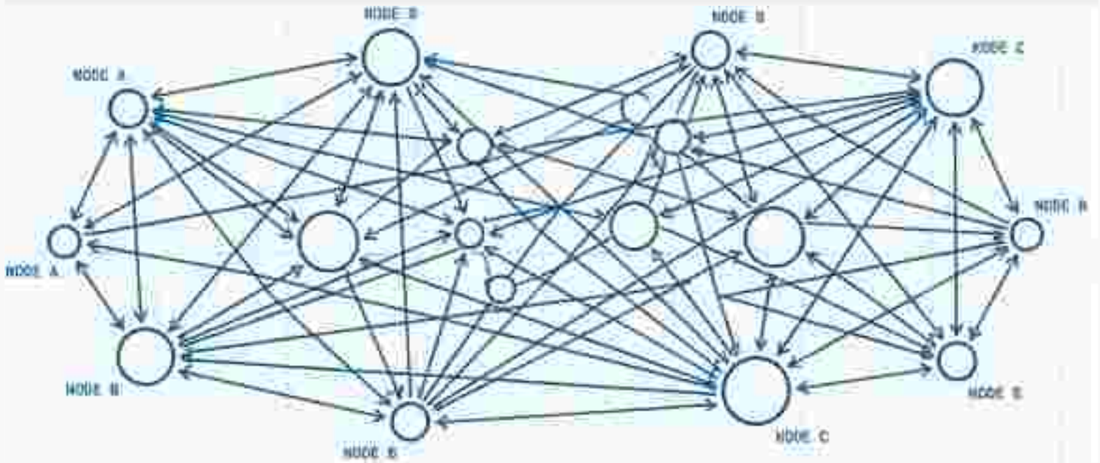
Focus: Providing centralized access to multiple services, often role-based.

Features: User authentication, integration of disparate systems.

Examples: University student portals, Government service portals, Enterprise dashboards.

Connectivity and Instantaneity

Social Web Applications

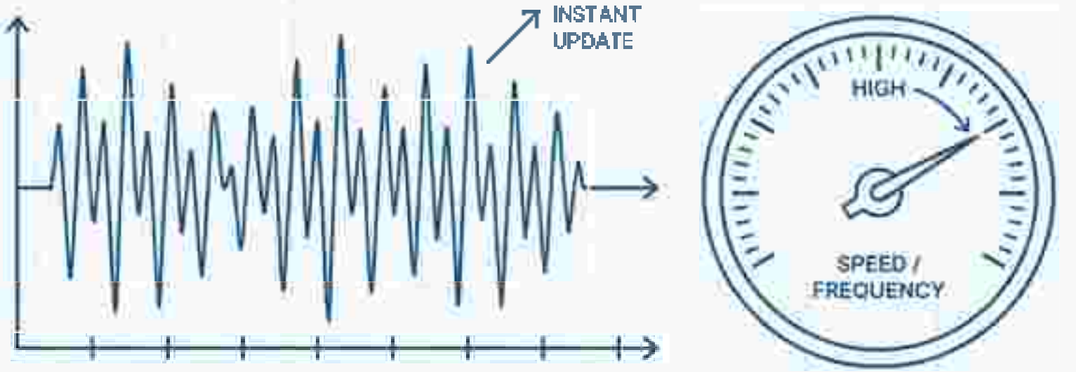


Focus: Communication, collaboration, and user-generated content sharing.

Challenges: Scalability, privacy/data protection, content moderation.

Examples: Facebook, Instagram, LinkedIn.

Real-Time Web Applications



Focus: Instant data updates requiring continuous server-client communication.

Tech Stack: WebSockets, Server-Sent Events, Real-time databases.

Examples: Chat applications, Online gaming platforms, Stock trading systems.

The Interaction Spectrum

Summary Matrix of Web Application Categories

Category	Interaction Level	Prime Example
Informational	**Low**	News website
Interactive	**Medium**	LMS
Transactional	**High**	E-commerce
Social	**Very High**	Facebook
Real-Time	**Instant**	Chat apps
Web Portal	**Mixed**	University portal

Engineering Characteristics of the Modern Web

Unlike traditional desktop software, web applications face unique constraints:

Global Accessibility

Multi-platform Usage

Continuous Evolution

SCALABILITY **SECURITY** **PERFORMANCE** **USABILITY** **RESPONSIVENESS**

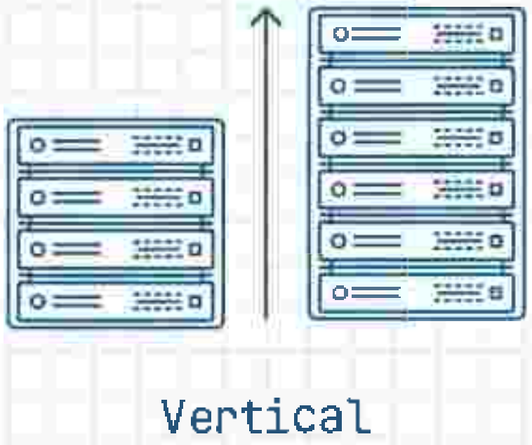
Non-Functional Requirements

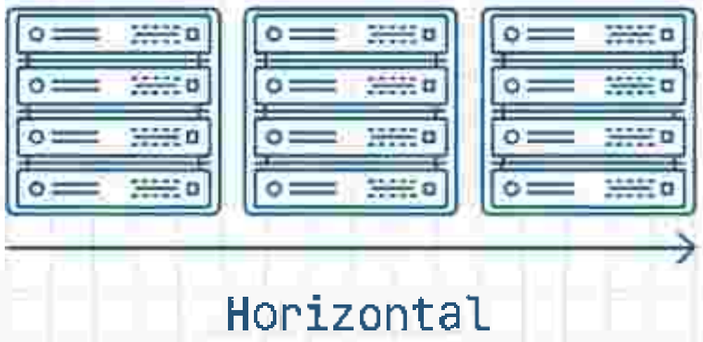
System Health and Growth

Scalability

Definition: Handling increases in users, data volume, and requests.

Critical to support growth and prevent failure.





Performance

Definition: Speed and efficiency of response.

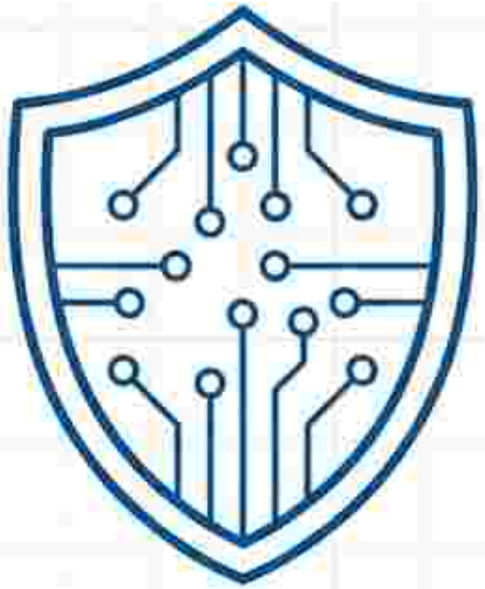
Goal is fast response times.

Factors:

- Server capacity
- Network latency
- Database efficiency

Protection and Uptime

Security



Goal: Protection from unauthorized access.

Threat Landscape: SQL Injection, Cross-Site Scripting (XSS), CSRF.

Defensive Measures: Authentication/Authorization, HTTPS, Secure coding practices.

Availability & Reliability

Availability: System is accessible when needed.

Reliability: System works correctly over time.

Techniques: Redundant servers, Fault tolerance, Backup and recovery.

The User Experience Layer

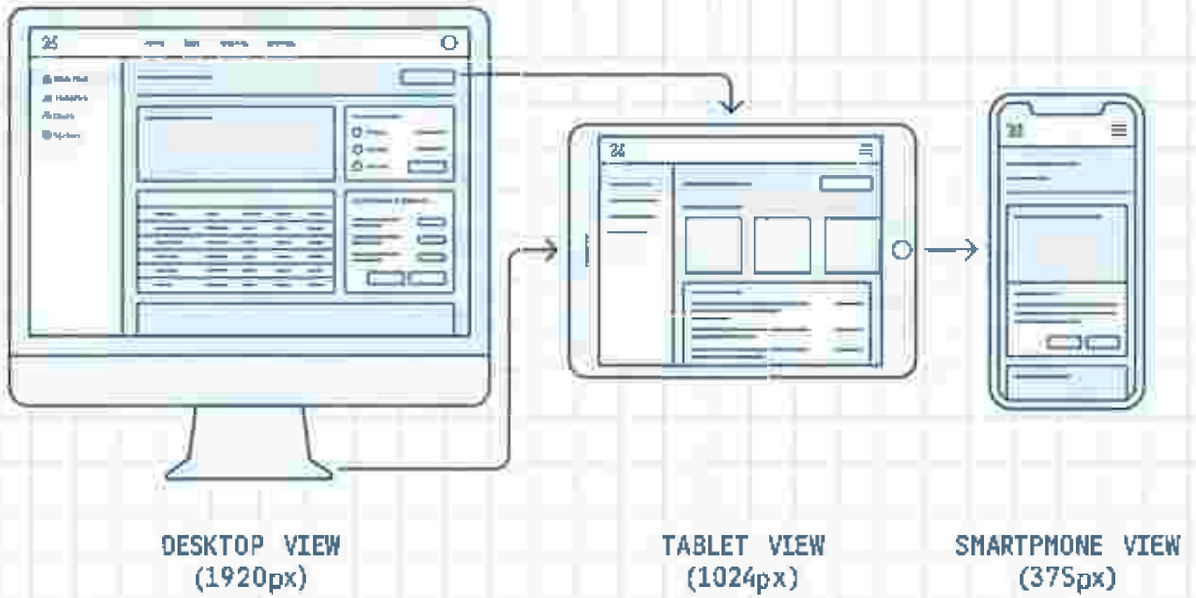
Usability

Definition: Ease of use and learnability; the driver of user satisfaction.

| Simple navigation | Clear interface | Accessibility

Responsiveness

Definition: Adapting to different devices and screen sizes. Critical for mobile-first users.

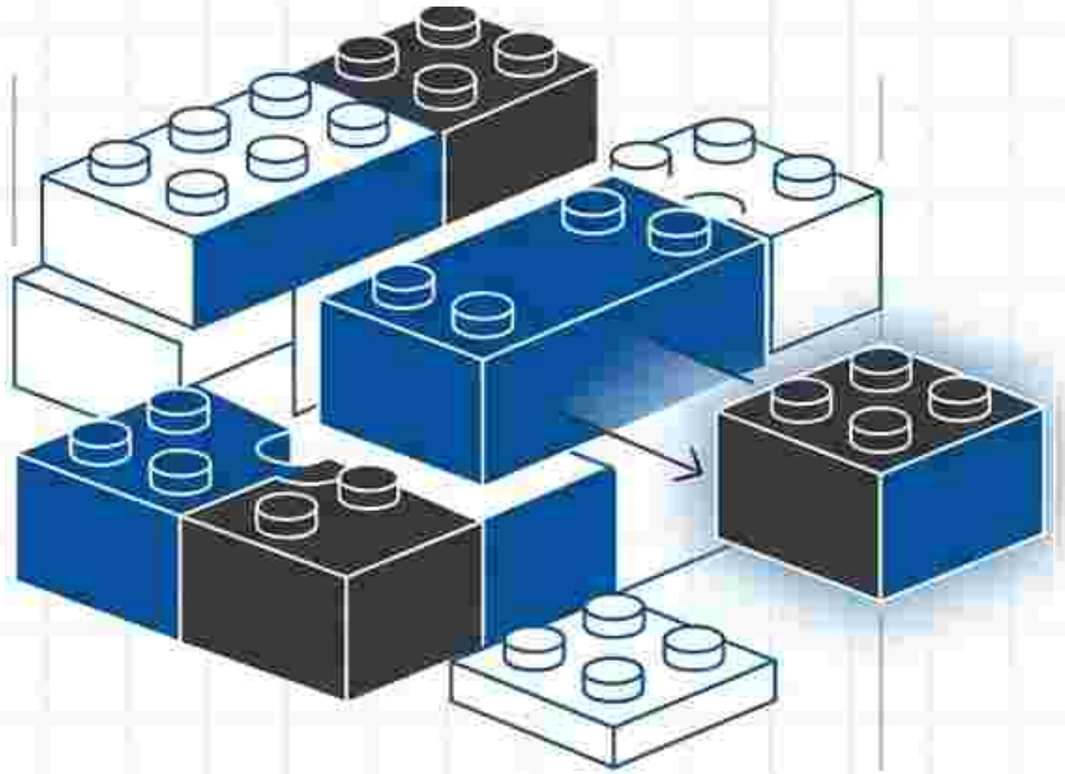


Techniqu: Responsive web design, CSS media queries, Front-end frameworks.

Long-Term Viability

Characteristic: Maintainability

Definition: The ease of updating and enhancing the system.

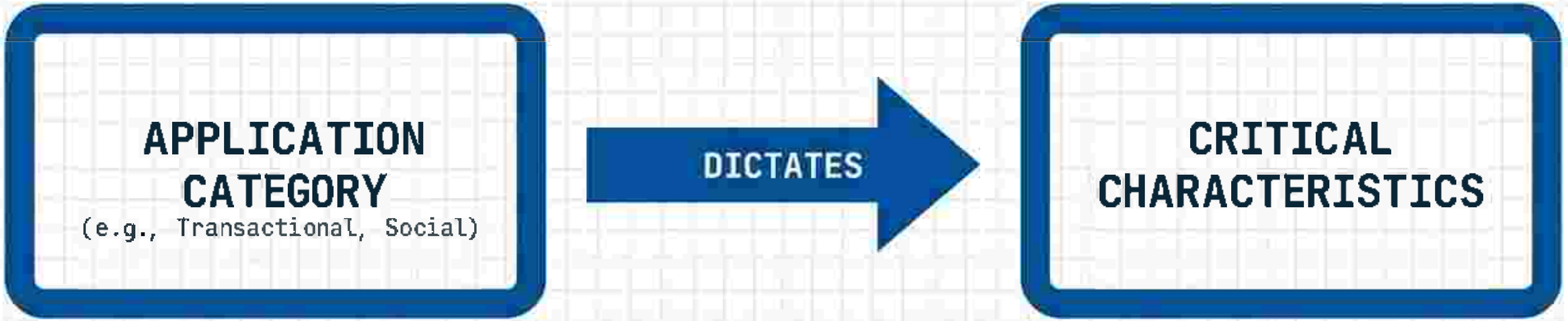


Modular Design Architecture

The Reality: Web applications evolve continuously. High maintainability reduces long-term costs.

The Approach: Modular design and clean code practices.

Architectural Decision Making



Transactional App → Requires High Security

Social App → Requires High Scalability

Real-Time App → Requires High Performance

Outcome: Proper understanding leads to better architecture and higher quality software.

Conclusion and Critical Analysis

Summary

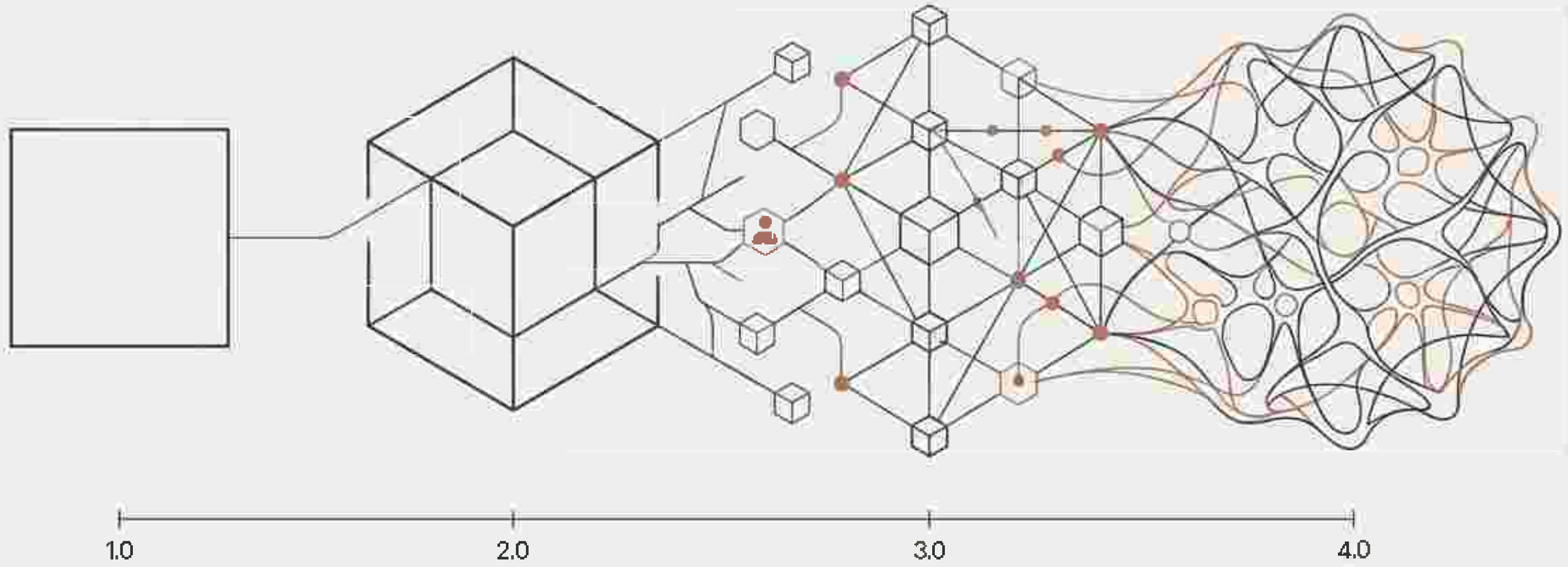
- Apps are classified by functionality and interaction.
- Each category imposes unique engineering challenges.
- Success relies on balancing **Scalability**, **Security**, and **Responsiveness**.

Discussion Points

1. Which category requires the highest security and why? (Hint: `Transactional`).
2. How does **scalability** differ between `Social` and `Transactional` apps?
3. Why is **responsiveness** non-negotiable in modern web systems?

The Living Web

From Static Pages to Symbiotic Intelligence: A Chronicle of Web Evolution (1.0 – 4.0)

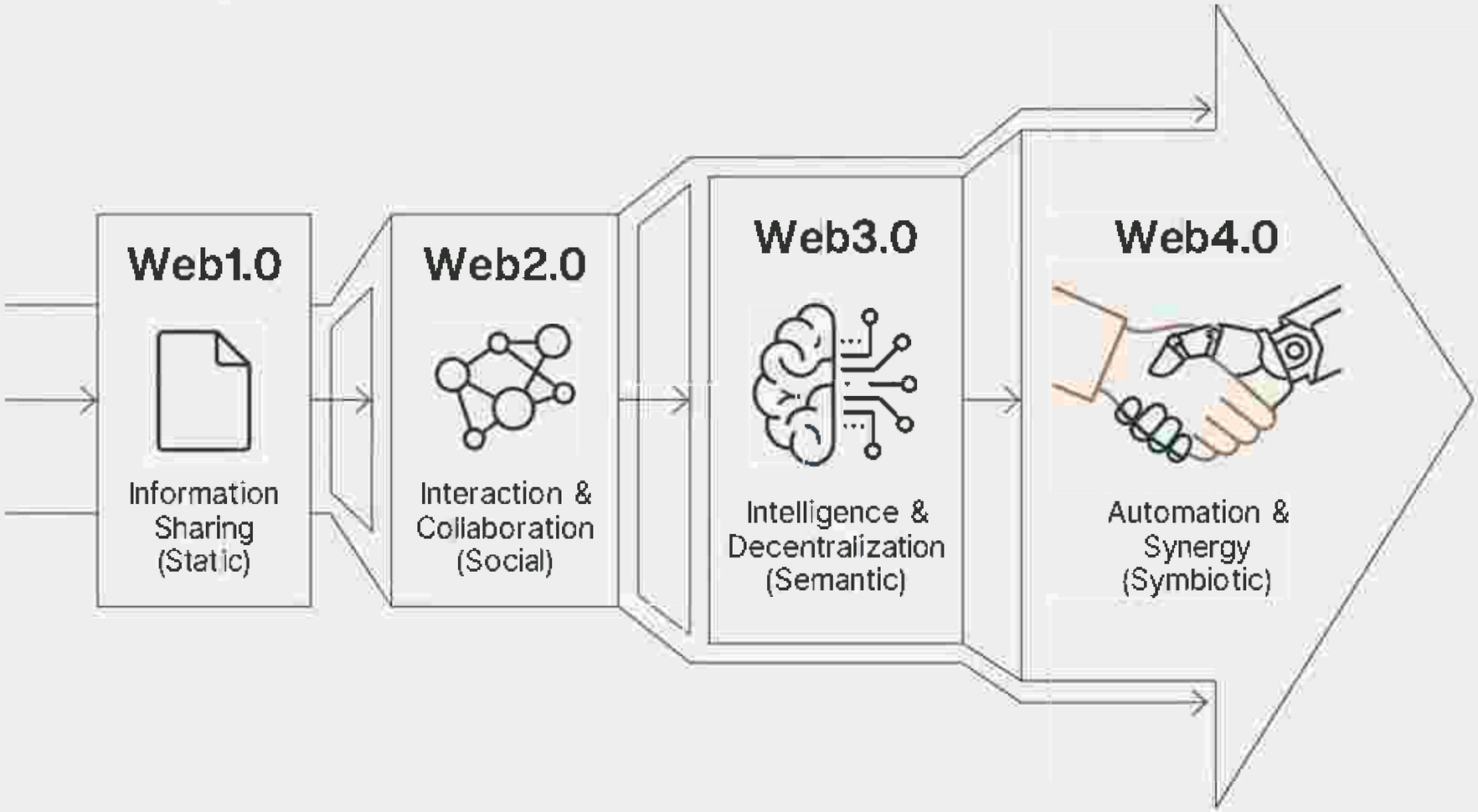


A SLIDEDOC FOR SOFTWARE ENGINEERS & STRATEGISTS

The web has transformed from a document repository into an autoinomous environment.

The World Wide Web has moved far beyond its 1990s inception as a system for sharing static documents. It has evolved into a highly interactive, intelligent, and user-centric platform. This evolution is defined by a shift in agency: from the user simply watching the screen, to the machine anticipating the user's needs.

Key Takeaway:
Understanding this trajectory is critical for designing systems that are scalable, secure, and future-ready.



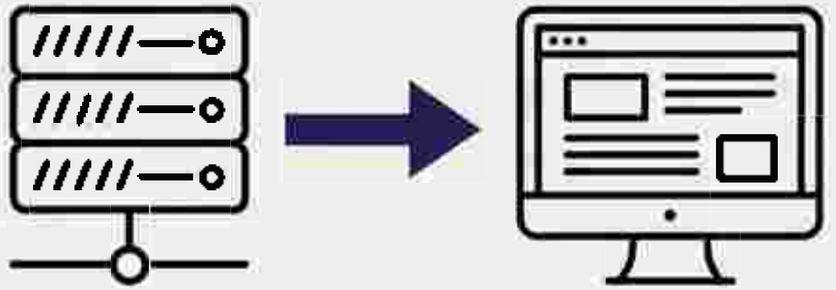
The Digital Library (Web 1.0)

Known as the "Static Web" or "Read-Only Web." In this era, the internet functioned like a digital brochure. Communication was strictly one-way (Server→ User). Content was published by a limited number of authors/companies for consumption by the many.

Evidence & Data:

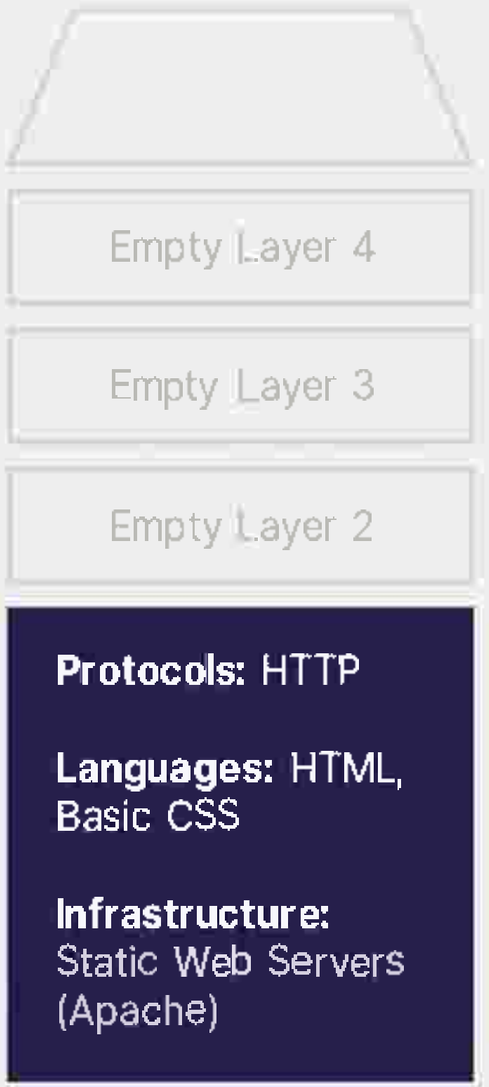
- **Examples:** Personal homepages, early company websites, online brochures.
- **Limitations:** No user participation; updates required specific technical coding expertise; zero personalization.

User Role: Observer (Read Only)



Server **User Screen**

The Stack



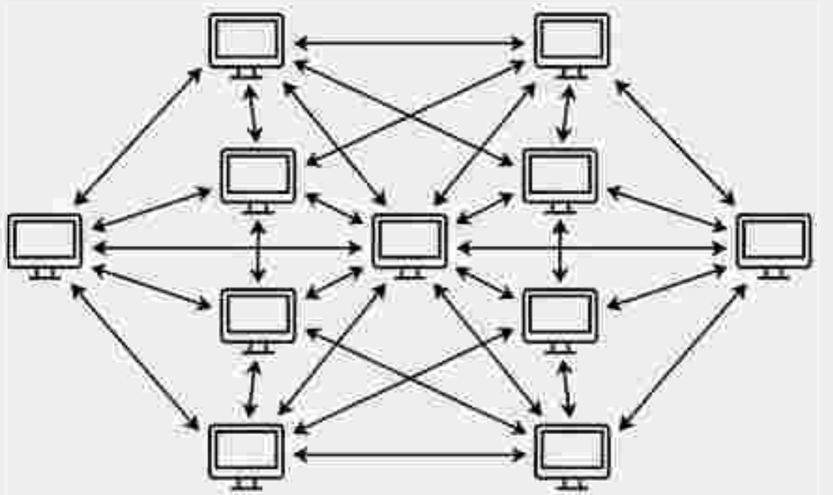
The Town Square (Web 2.0)

The "Social & Interactive Web" (Read-Write). The web became a two-way conversation. The distinguishing factor was the rise of User-Generated Content (UGC), enabling anyone to become a creator. This era birthed the digital economy and massive collaboration platforms.

User Role: Contributor (Read / Write)

Evidence & Data:

- **Examples:** Facebook, YouTube, Twitter, Wikis, E-commerce platforms.
- **Impact:** A shift from consumers to contributors; improved user engagement.
- **Limitations:** Centralized data ownership (platform dependency); privacy and security concerns.



The Stack

Empty Layer 4

Empty Layer 3

Languages: HTML5, CSS3, JS, PHP

Methods: AJAX (Dynamic Updates)

Data: Databases (MySQL)

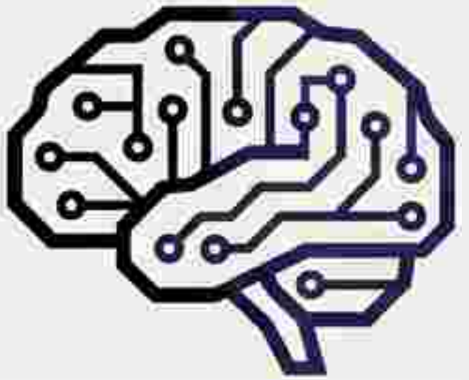
Protocols: HTTP

Languages: HTML, Basic CSS

Infrastructure: Static Web Servers (Apache)

The Semantic Brain (Web 3.0)

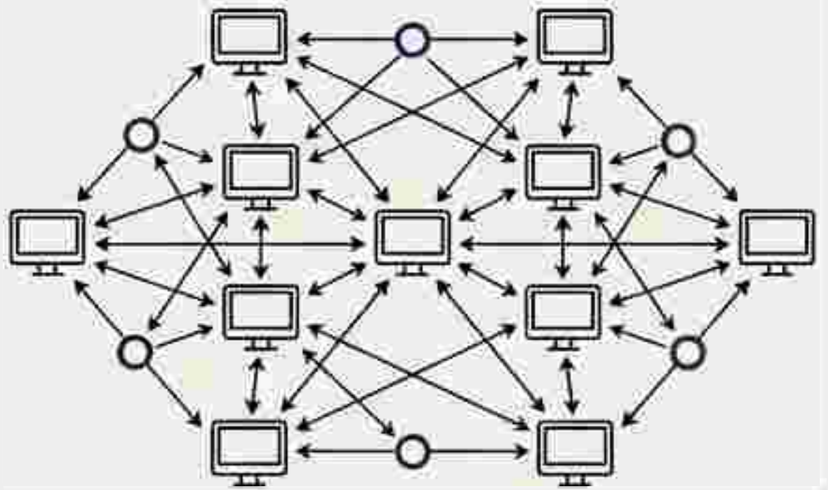
The "Intelligent Web" (Read-Write-Execute). Here, the focus shifts to machine-understandable data. The web is no longer just connecting people; it is processing information to provide personalized, context-aware services. It introduces concepts of decentralization to counter platform dependency.



User Role: Owner (Execute/ Understand)

Evidence & Data:

- **Examples:** Knowledge Graphs, Decentralized Apps (DApps), Personalized recommendations.
- **Impact:** User-centric data control; improved data interoperability.
- **Limitations:** High technical complexity; scalability challenges; lack of mature standards.



Tech Stack

Empty Layer 4

Semantic Data: RDF, OWL, SPARQL
Intelligence: Machine Learning Models
Infrastructure: Blockchain, Smart Contracts

Languages: HTML5, CSS3, JS, PHP
Methods: AJAX (Dynamic Updates)
Data: Databases (MySQL)

Protocols: HTTP
Languages: HTML, Basic CSS
Infrastructure: Static Web Servers (Apache)

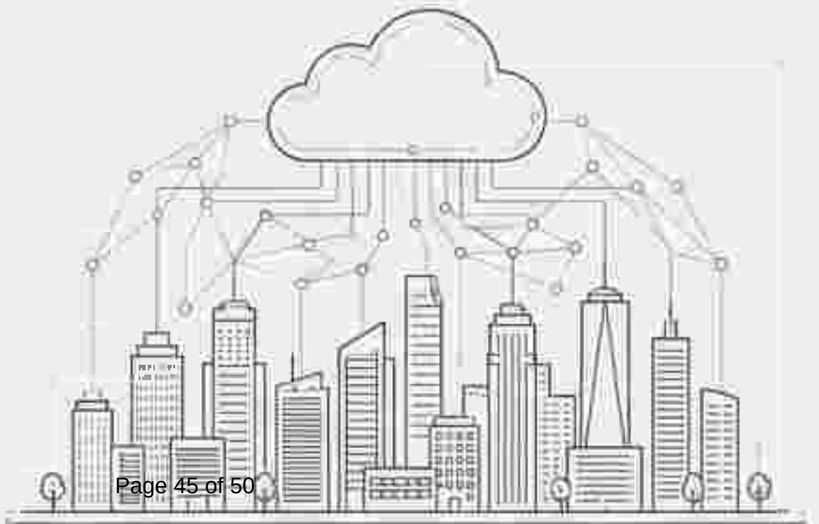
The Symbiotic Nervous System (Web 4.0)

The "Symbiotic & Intelligent Web." The boundary between the digital and physical worlds dissolves. Systems possess autonomous decision-making capabilities, driven by real-time intelligence and ubiquitous connectivity. The web is no longer a place you go; it is an environment you live within.

Evidence & Data:

- **Examples:** Smart Cities, Autonomous Systems, Smart Assistants.
- **Impact:** Seamless automation of complex tasks; integration of physical/digital realities.
- **Limitations:** High dependency on infrastructure; ethical privacy concerns; security risks.

User Role: Partner (Human-Machine Synergy)



The Stack

Intelligence: Advanced General AI
Connectivity: Internet of Things (IoT)
Processing: Big Data, Cloud & Edge

Semantic Data: RDF, OWL, SPARQL
Intelligence: Machine Learning Models
Infrastructure: Blockchain, Smart Contracts

Languages: HTML5, CSS3, JS, PHP
Methods: AJAX (Dynamic Updates)
Data: Databases (MySQL)

Protocols: HTTP
Languages: HTML, Basic CSS
Infrastructure: Static Web Servers (Apache)

Architecture Evolution: Under the Hood

Metric	Web 1.0	Web 2.0	Web 3.0	Web 4.0
Primary Interaction	One-way (Server-to-User)	Two-way (Social/UGC)	Personalized / Decentralized	Autonomous / Symbiotic
Data Handling	Static File Systems	Relational Databases (SQL)	Knowledge Graphs / Blockchain	Big Data & Real-time Streams
Key Tech	HTML/ HTTP	AJAX/ JS / PHP	RDF / SPARQL / AI	IoT / Edge Computing

The Shift in Agency: From Observer to Partner

Web 1.0: The Consumer

Passive consumption.
No influence on content.
The user watches.

Web 2.0: The Contributor

Active participation. Creates value but does not own the platform.
The user speaks.

Web 3.0: The Owner

Control over data.
Context-aware interactions.
The user decides.

Web 4.0: The Partner

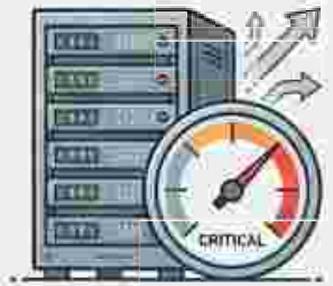
Synergy. The user sets the goal; the system handles execution.
The user lives within it.

The Cost of Complexity

Each generation builds upon the previous one, increasing capability but also introducing new engineering and ethical challenges.

Data Sovereignty

Moving from centralized ownership (2.0) to the complexity of user-centric control (3.0).



Scalability & Performance

The jump from serving static text (1.0) to processing real-time autonomous decisions (4.0) requires massive infrastructure.

Security & Ethics

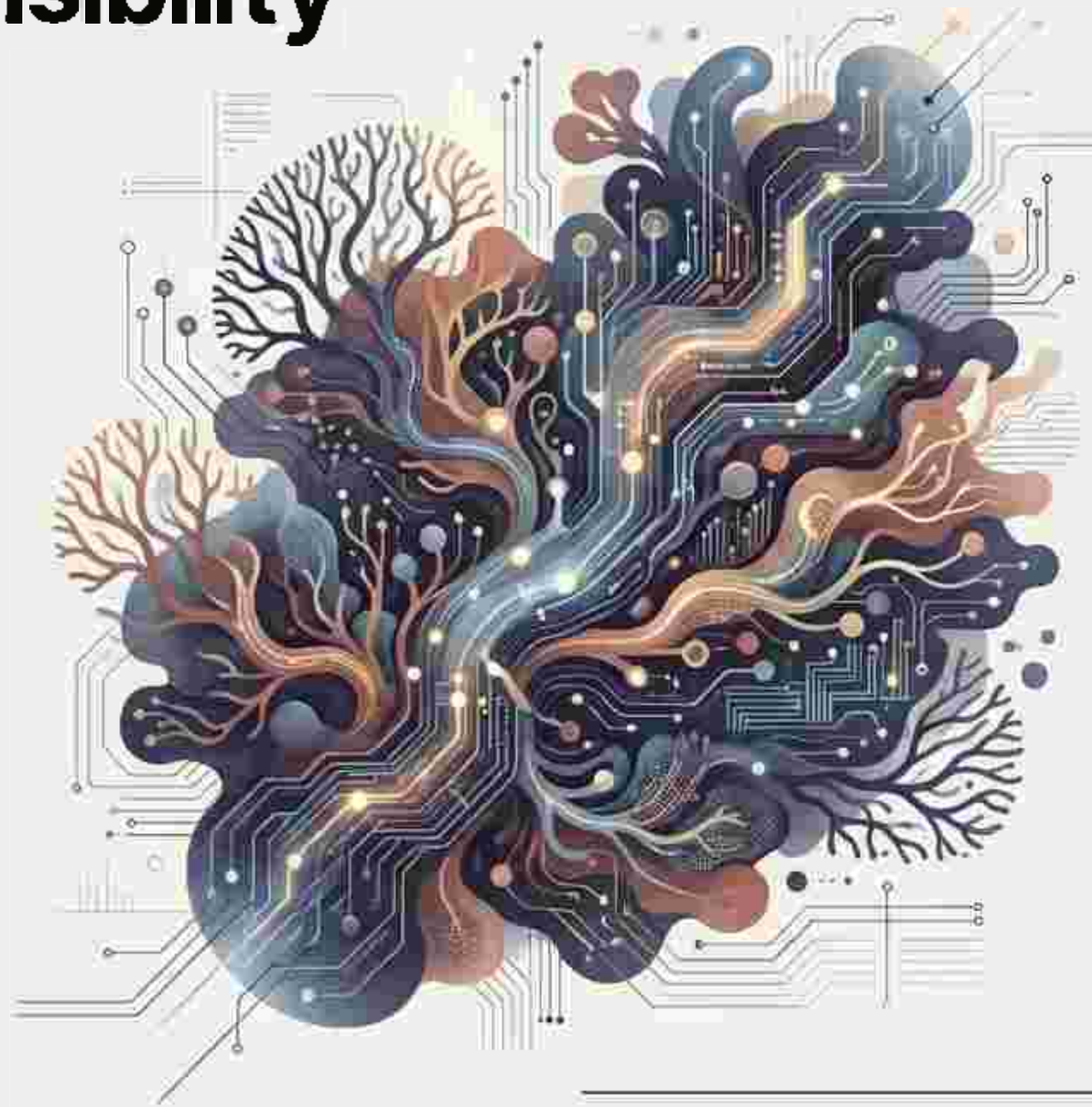
As we move to **Web 4.0**, risks shift from website defacement to critical infrastructure failure and ethical autonomous decision-making.

Era Summary Matrix

Feature	Web1.0	Web2.0	Web3.0	Web4.0
Focus	Info Sharing	Interaction	Intelligence	Automation
Read/Write	Read-Only	Read-Write	R-W-Execute	Symbiotic
Key Tech	HTML	Social / Mobile	Semantic / AI	IoT / Edge
Example	Brochure Site	Facebook / Wiki	DApps	Smart City
Impact	Global Info	Community	Personalization	Integration

The Engineer's Responsibility

As the web evolves from simple files to autonomous environments, the burden of design increases. We are no longer just building pages; we are architecting the nervous system of the modern world.



The transition to Web 4.0 requires a focus not just on code, but on ethics, security, and the human impact of symbiotic systems.

Design for the symbiotic future.