软件架构
Software Architecture

By, Fahad Sabah

TFSE_Fall2025_北京工业大学，中国

TFSE_Fall2025_Beijing University of Technology, China

软件架构的定义
Definition of Software Architecture

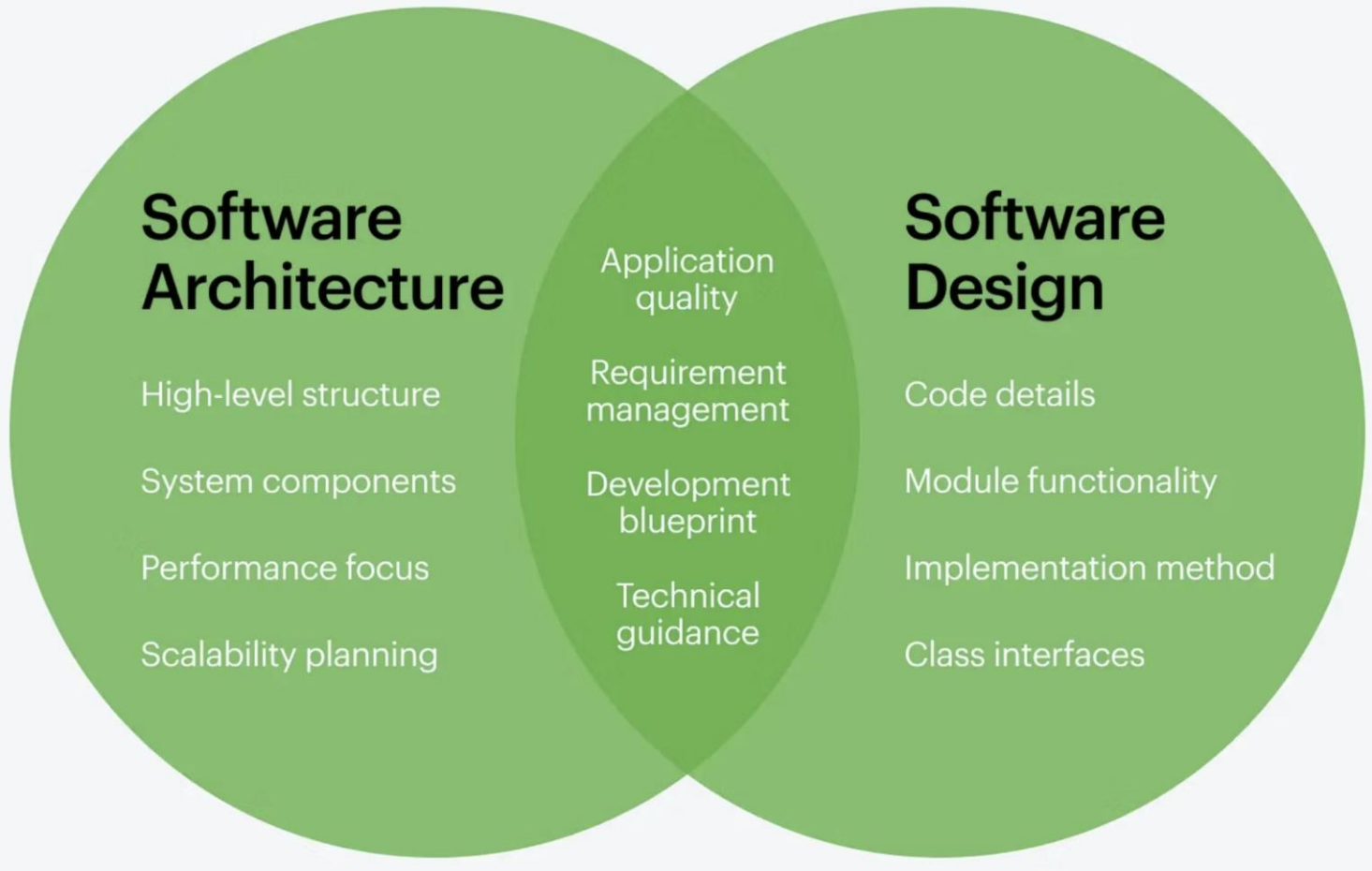
“The high-level structure of a software system, defining its components, their relationships, and how they interact”

Emphasize **high-level structure**, not low-level code

“软件系统的高层结构，定义其组件、关系及其相互作用”

强调高层结构，而非低层代码

软件架构与软件设计
Software Architecture vs. Software Design



架构 vs. 设计 vs. 实现
Architecture vs. Design vs. Implementation

Aspect	Definition	Scope / Level
Architecture	High-level structure, patterns, style choices	System-wide, decisions that affect quality attributes
Design	Component-level decisions, algorithms, data structures	Modules, classes
Implementation	Writing code to fulfill design	Lines of code, functions, methods

架构在编码开始之前就已经影响可维护性、可扩展性和性能。

Architecture impacts **maintainability, scalability, and performance** long before coding starts.

Common examples of architectural styles include:

常见的建筑风格例子包括:

- **Layered Architecture:** Organizes the system into horizontal layers, each providing services to the layer above it.
- **Microservices:** Structures an application as a collection of small, independent, and loosely coupled services that communicate over a network.
- **Event-Driven Architecture (EDA):** Focuses on the production, detection, consumption of, and reaction to events.
- **Model-View-Controller (MVC):** Separates the data (Model), the user interface (View), and the control logic (Controller) for handling input.
- **Client-Server:** Separates concerns between a provider of resources or services (server) and a requester of those resources (client).

Layered Architecture [分层架构]

- Logical layers (Presentation, Business Logic, Data)
- Closed vs. open layering
- Encapsulation and dependency rules

Usage [用法]

- Applications that are needed to be built quickly.
- Enterprise applications that require traditional IT departments and processes.
- Appropriate for teams with inexperienced developers and limited knowledge of architecture patterns.

Strengths & Weaknesses [优点与缺点]

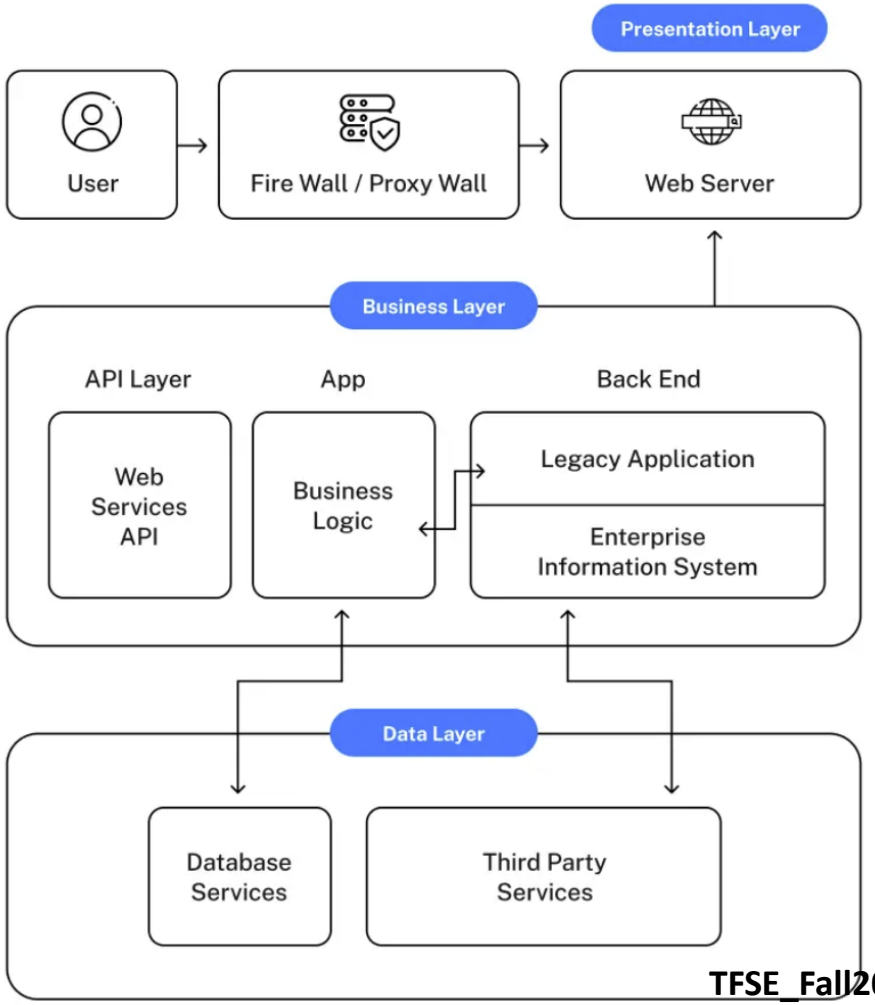
Strengths:

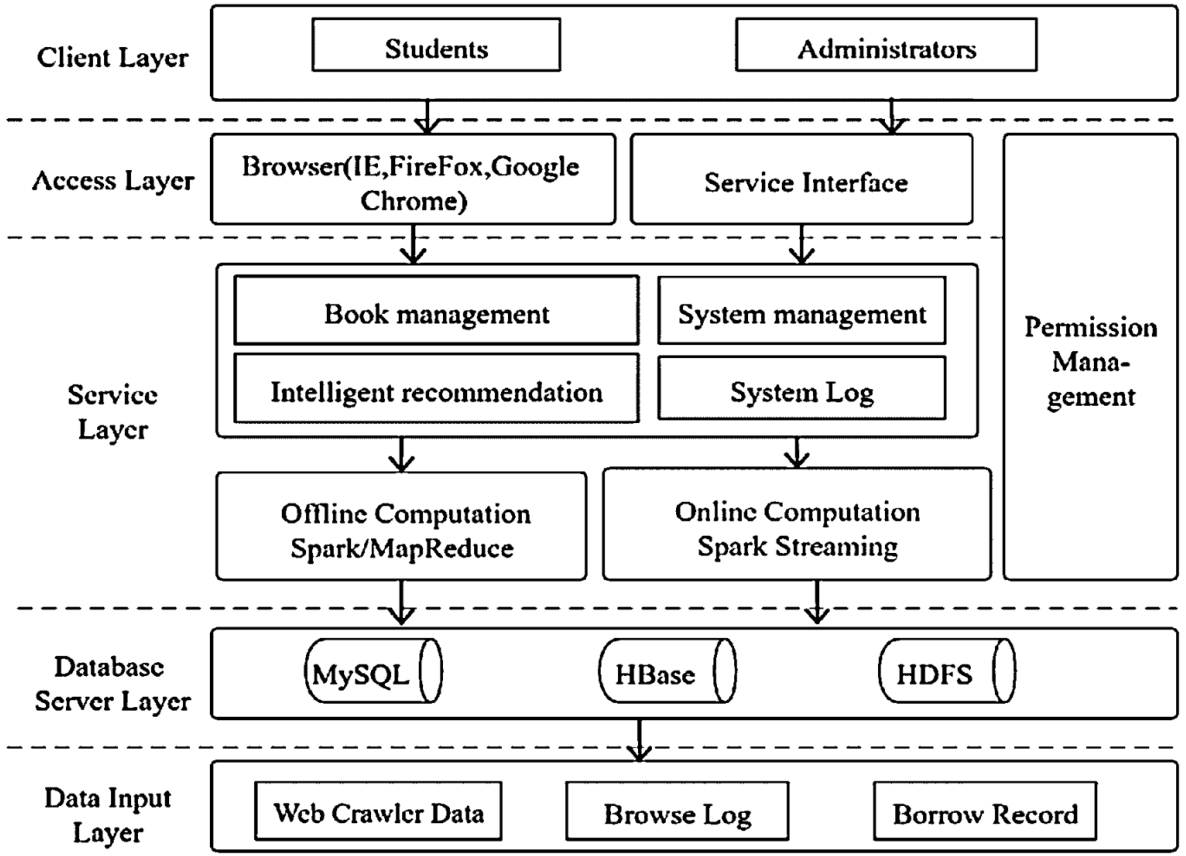
- ❑High modifiability
- ❑Clear separation of concerns
- ❑Well-suited for enterprise systems

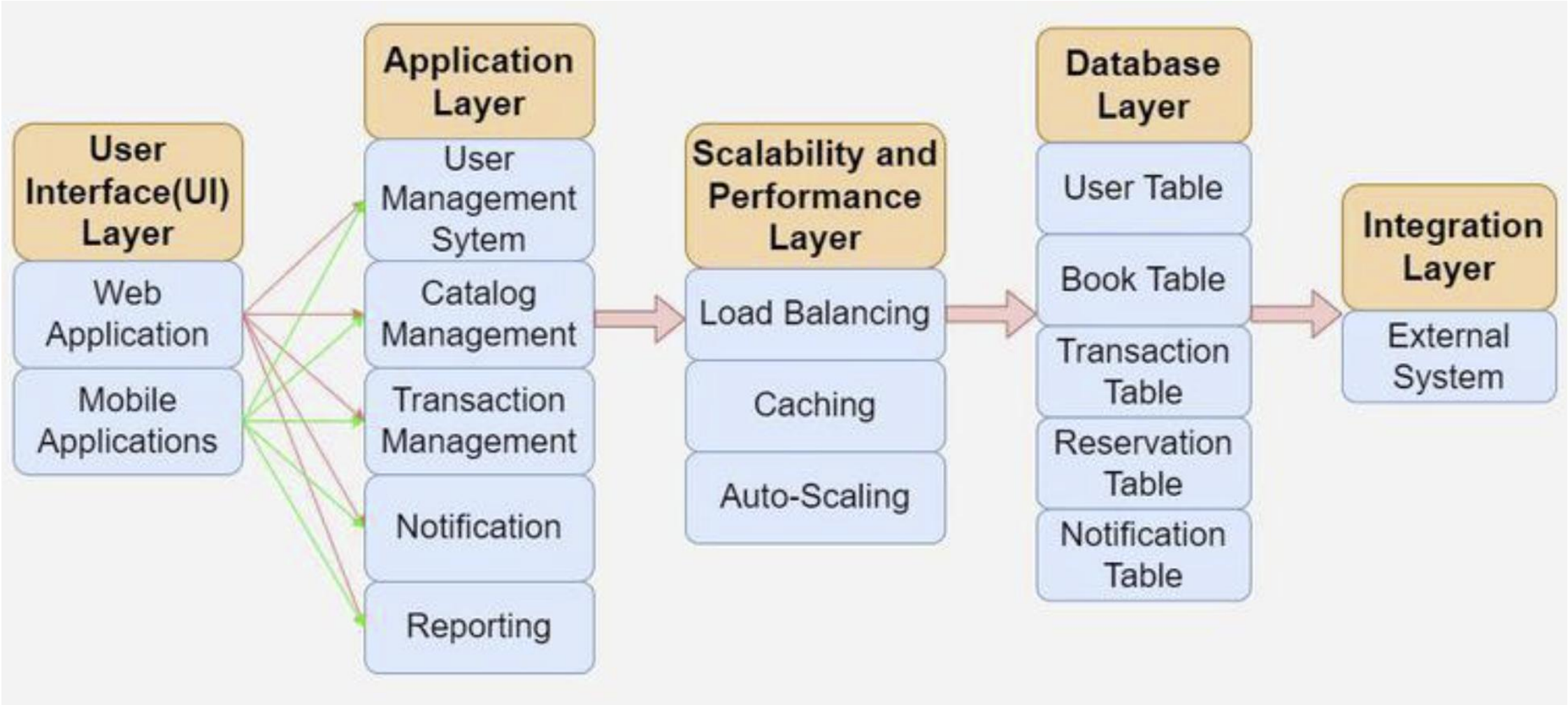
Weaknesses:

- ❑Unorganized source codes and modules with no definite roles can become a problem for the application.
- ❑Skipping previous layers to create tight coupling can lead to a logical mess full of complex interdependencies.
- ❑Basic modifications can require a complete redeployment of the application.

示例 [库系统的层架构]
Example [Layer architecture of a library system]

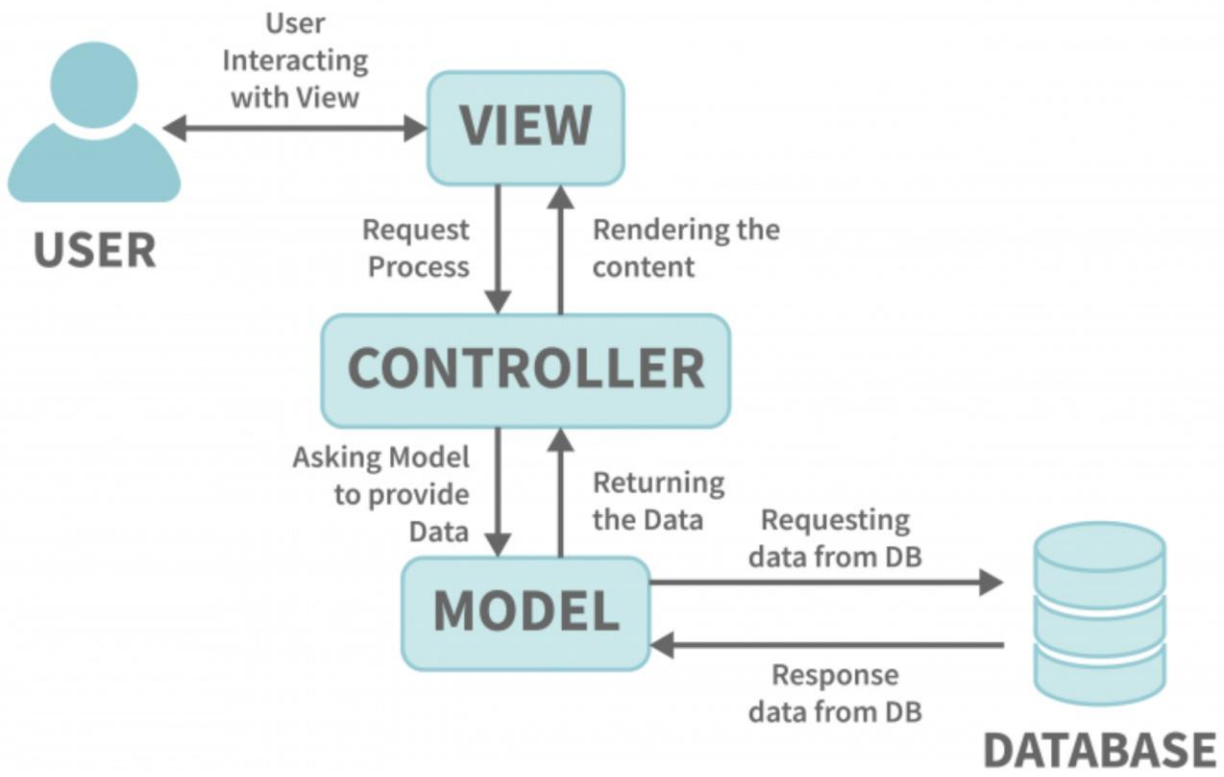





User Interface(UI) Layer
Web Application
Mobile Applications
Application Layer
User Management Sytem
Catalog Management
Transaction Management
Notification
Reporting
Scalability and Performance Layer
Load Balancing
Caching
Auto-Scaling
Database Layer
User Table
Book Table
Transaction Table
Reservation Table
Notification Table
Integration Layer
External System

模型-视图-控制器（MVC）架构
Model–View–Controller (MVC) Architecture

- Roles: Model, View, Controller
- Flow of events
- Where MVC fits today (web/mobile frameworks)



Strengths & Weaknesses [优点与缺点]

Strengths

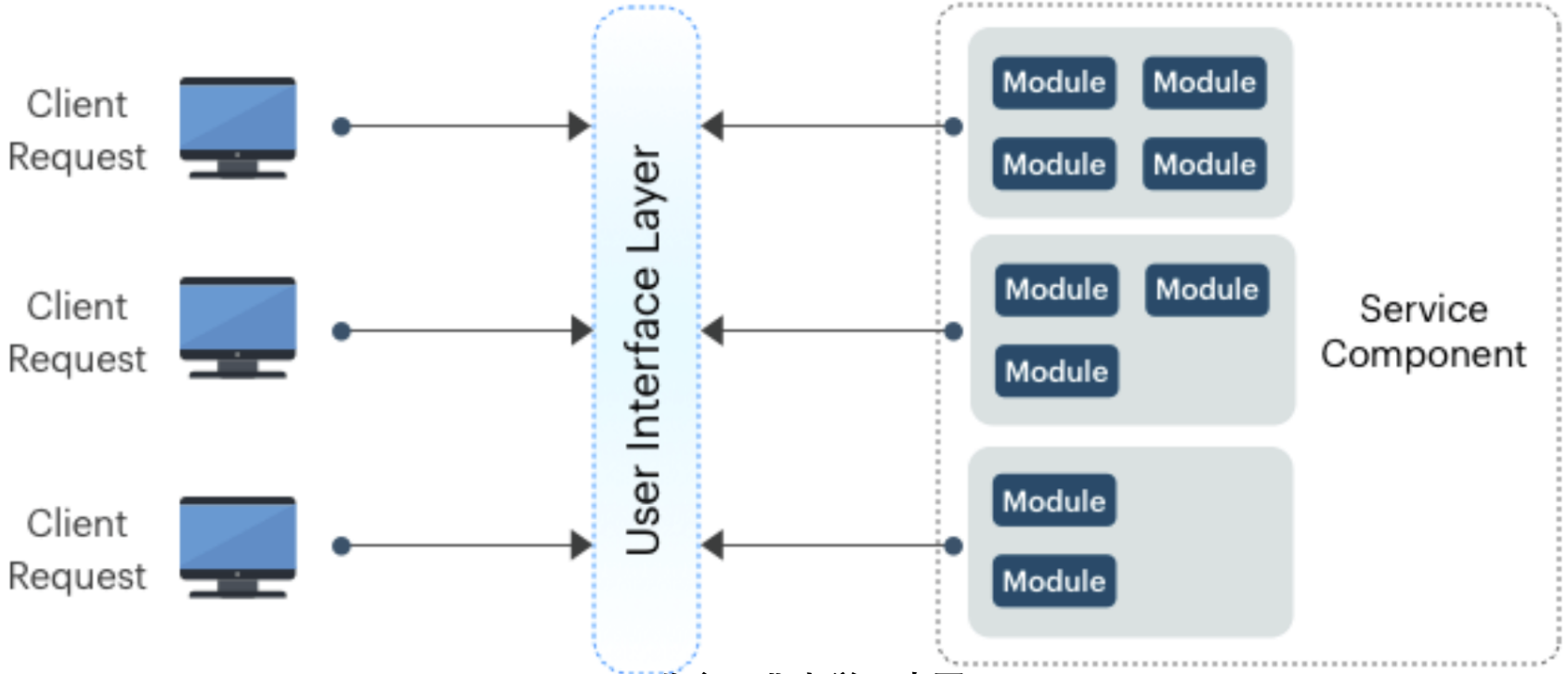
- ❑Clear separation of input, logic, and UI
- ❑Enables parallel development

Weaknesses

- ❑Can become tightly coupled if poorly implemented
- ❑Controller “bloat”

Microservices Architecture [微服务架构]

- Services as independently deployable units
- Bounded contexts (Domain-Driven Design influence)
- API gateways, service registries



Strengths & Weaknesses [优点与缺点]

Strengths

- ❑High scalability & fault isolation
- ❑Technology heterogeneity
- ❑Supports CI/CD and cloud-native architectures

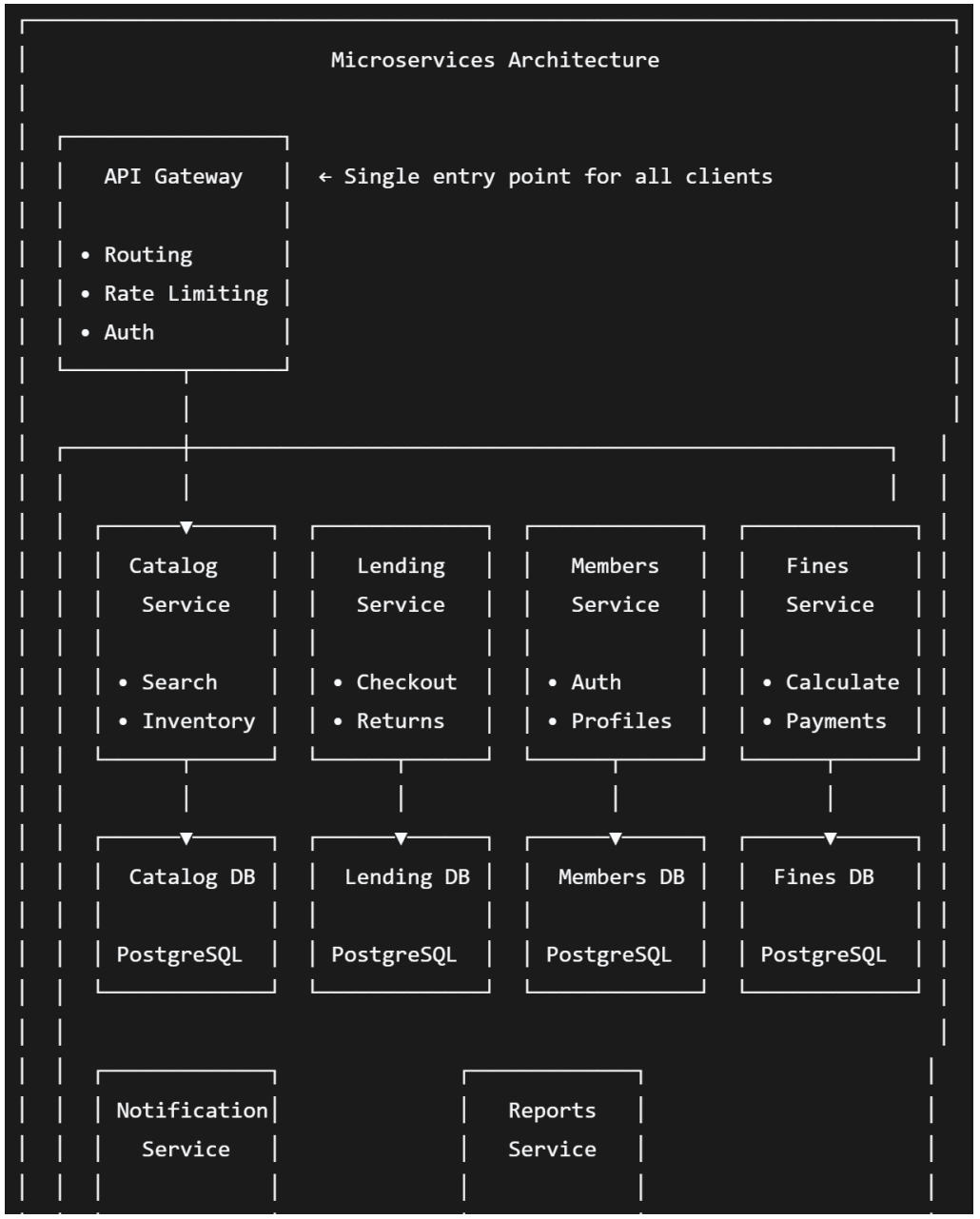
Continuous Integration/
Continuous Development/Delivery

Weaknesses

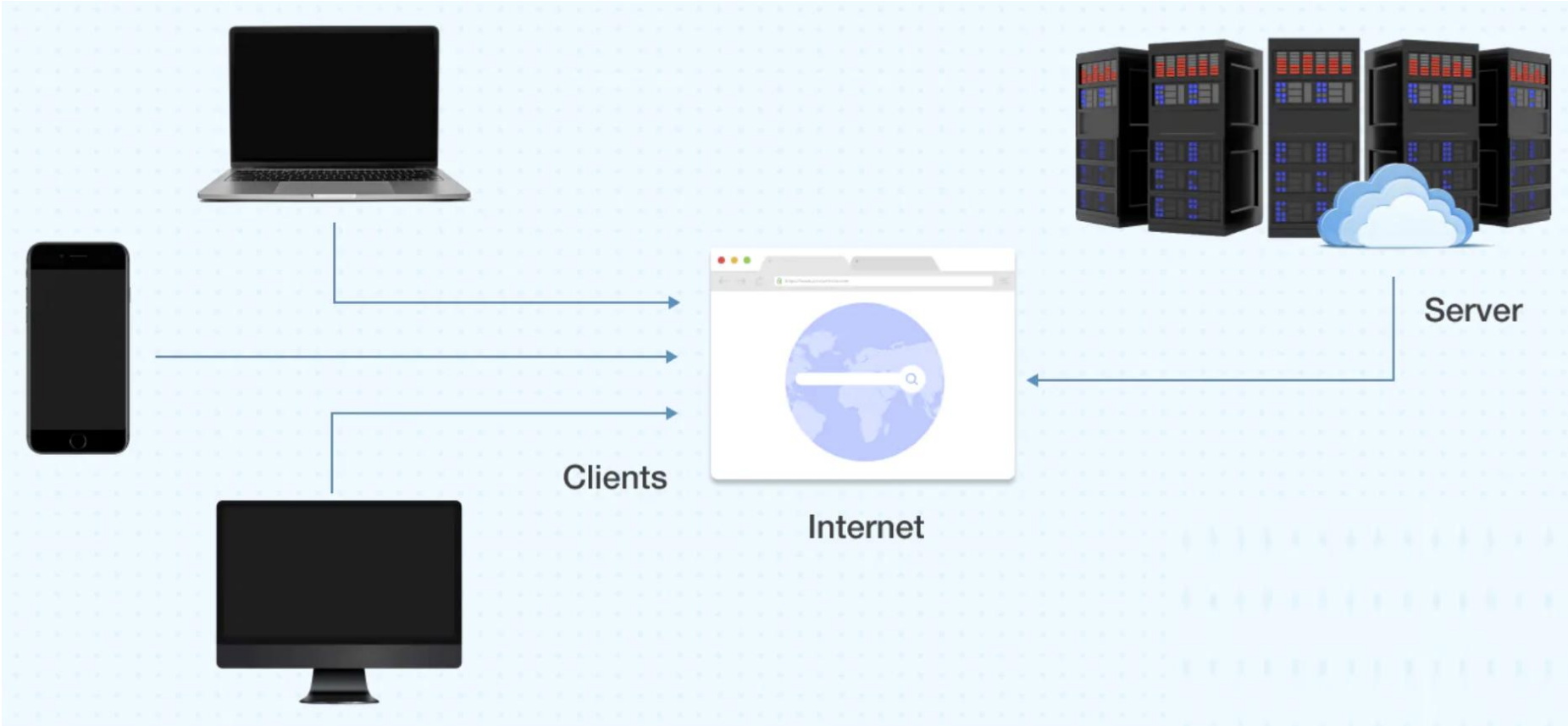
- ❑Operational complexity
- ❑Distributed system failure modes
- ❑DevOps/tooling overhead

Example [例子]

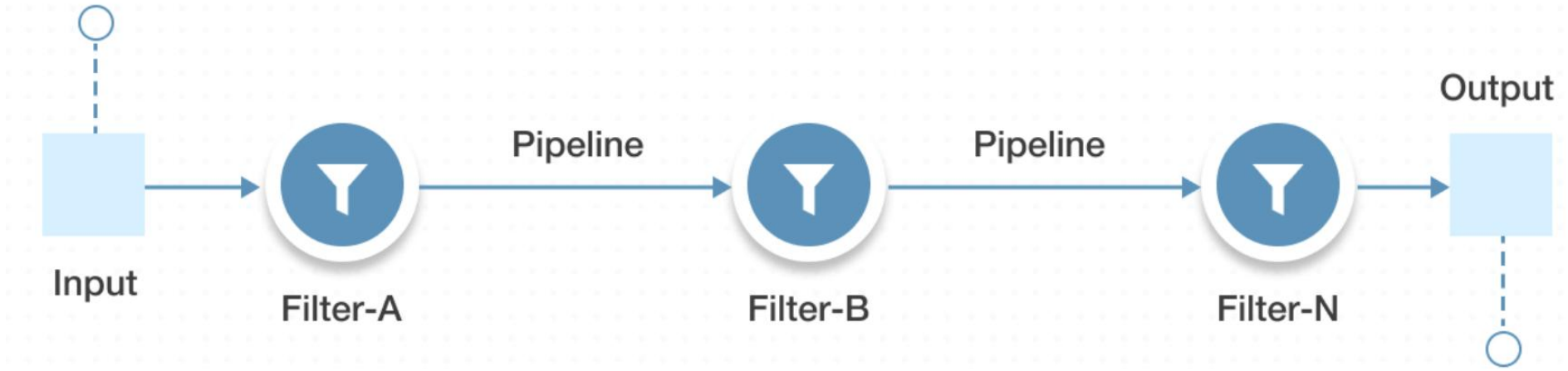
- Netflix, Amazon retail systems



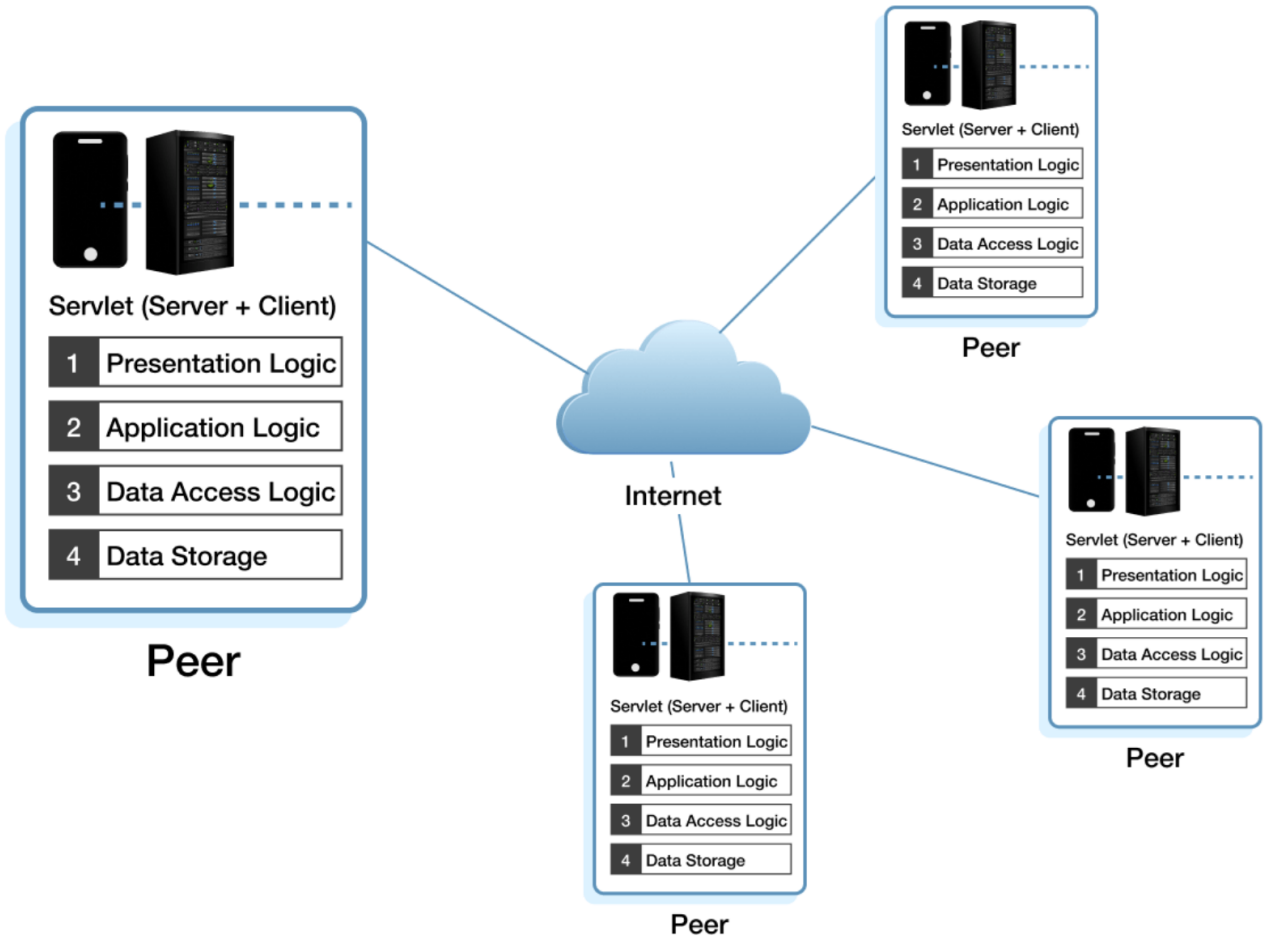
Client-Server Architecture



Pipe-Filter Architecture



Peer-to-Peer Architecture



Layered Pattern

Agility — Low

Difficult to make changes

Ease of Deployment — Low

Minor change requires redeployment of the entire app

Testability — High

Presentation component can be mocked to isolate testing

Performance — Low

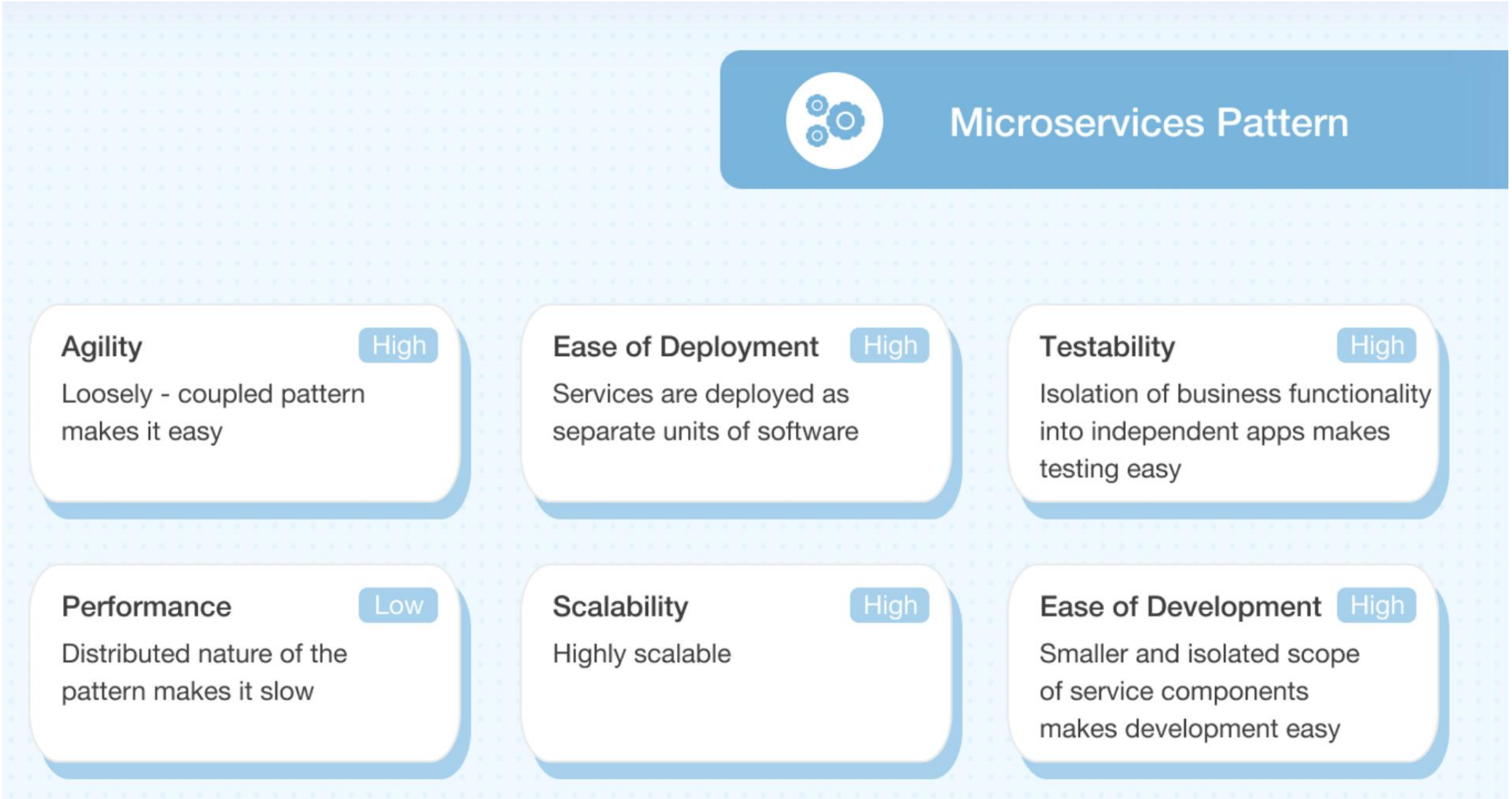
Pattern not friendly for high performant app

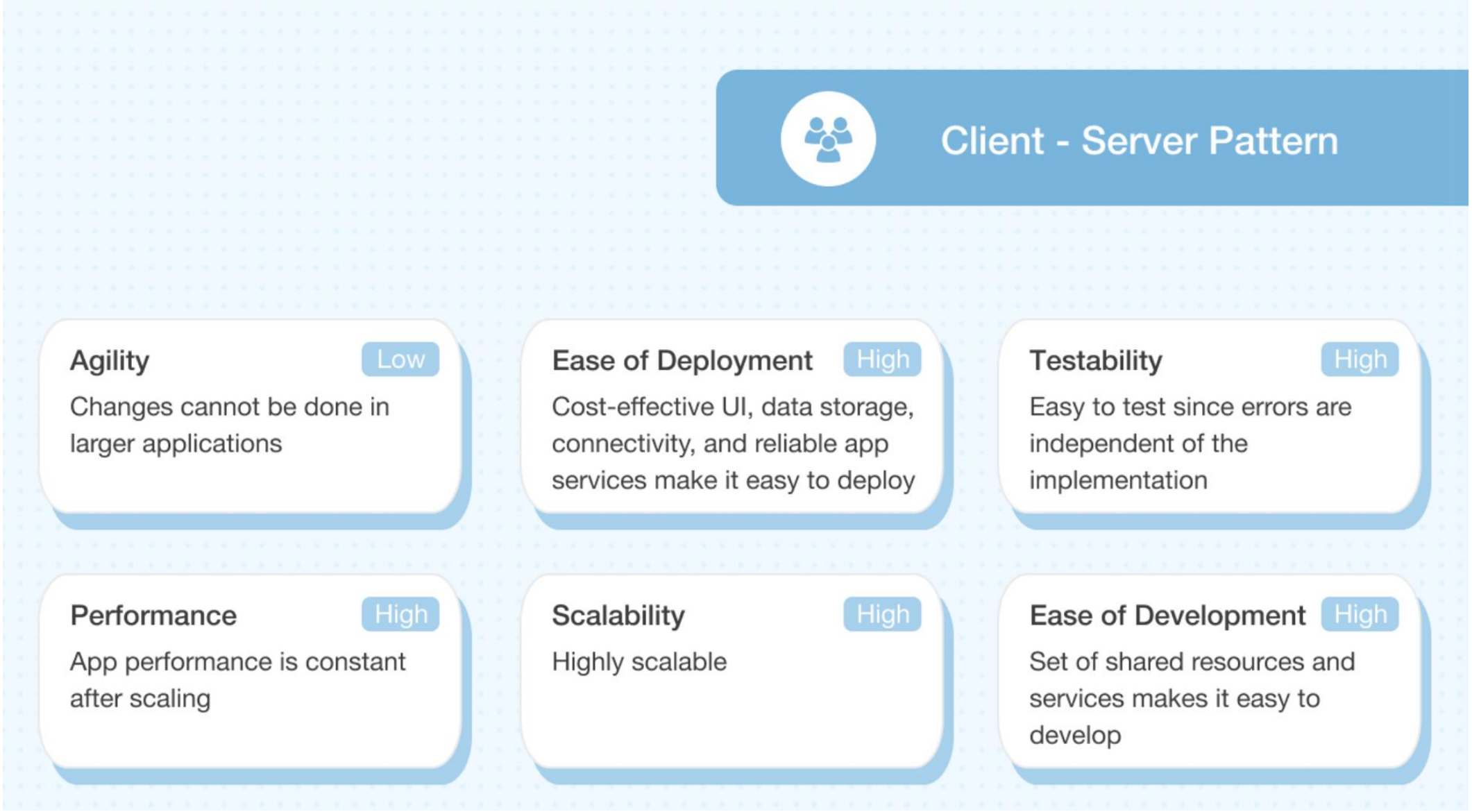
Scalability — Low

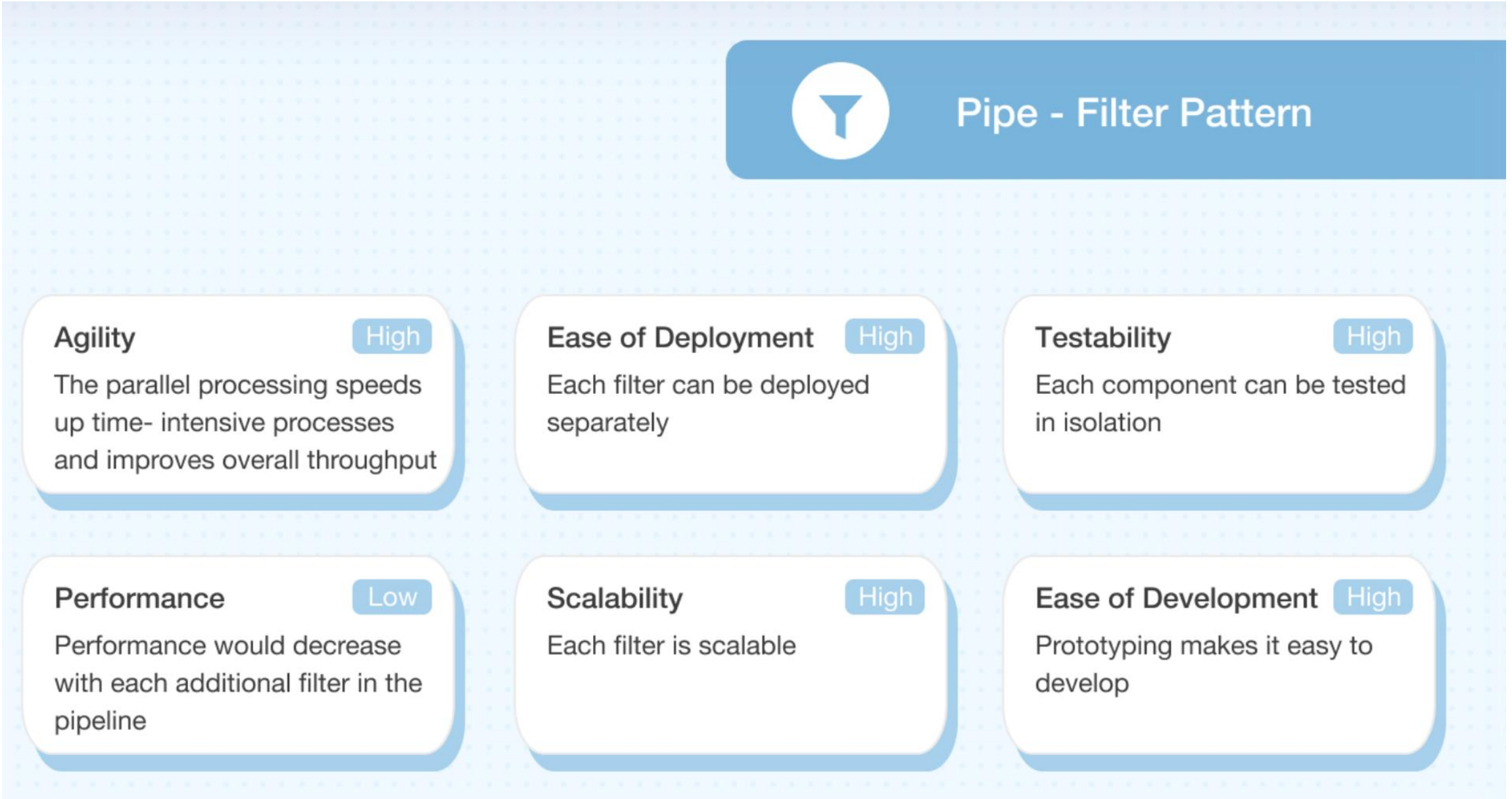
Difficult to scale

Ease of Development — High

Easy to develop


Microservices Pattern
Agility
High
Loosely - coupled pattern makes it easy
Ease of Deployment
High
Services are deployed as separate units of software
Testability
High
Isolation of business functionality into independent apps makes testing easy
Performance
Low
Distributed nature of the pattern makes it slow
Scalability
High
Highly scalable
Ease of Development
High
Smaller and isolated scope of service components makes development easy


Client - Server Pattern
Agility
Low
Changes cannot be done in larger applications
Ease of Deployment
High
Cost-effective UI, data storage, connectivity, and reliable app services make it easy to deploy
Testability
High
Easy to test since errors are independent of the implementation
Performance
High
App performance is constant after scaling
Scalability
High
Highly scalable
Ease of Development
High
Set of shared resources and services makes it easy to develop


Pipe - Filter Pattern
Agility
High
The parallel processing speeds up time- intensive processes and improves overall throughput
Ease of Deployment
High
Each filter can be deployed separately
Testability
High
Each component can be tested in isolation
Performance
Low
Performance would decrease with each additional filter in the pipeline
Scalability
High
Each filter is scalable
Ease of Development
High
Prototyping makes it easy to develop

LMS（图书馆管理系统）的预架构
Pre-Architecture for LMS [Library Management System]

- **Step 1:** Identify and Engage Stakeholders

- **How to Perform:** Brainstorming and using a Stakeholder Map. We can categorize stakeholders to ensure no one is missed.

- The project lead (or architect) would call a kick-off meeting with key business representatives. The goal is to brainstorm and create a comprehensive list of all individuals, groups, or organizations that will be affected by the new system.

Stakeholder Map for LMS (Business / Sponsors)

Category	Stakeholder Role	Description & Specific Examples
Business / Sponsors	Library Director	Ultimate decision-maker and budget holder. (e.g., Dr. Eleanor). Primary Concern: Return on Investment (ROI), increasing library membership, and improving community engagement metrics.
	Board of Trustees	Provides oversight and strategic direction for the library. Primary Concern: That the system aligns with the library's long-term vision and complies with public service mandates.

Stakeholder Map for LMS (End Users)

Category	Stakeholder Role	Description & Specific Examples
End Users	Librarians & Library Staff	The primary daily users of the system's administrative functions. (e.g., Sarah, the front-desk librarian; Tom, the inventory manager). Primary Concern: Efficiency, ease of use, and reducing time spent on repetitive tasks like check-out and fine management.
	Library Members (Patrons)	The public users of the system. They will use the online catalog and their personal accounts. (e.g., A student, a parent, a researcher). Primary Concern: Easy discovery of books, clarity of their account status (due dates, fines), and the ability to place holds from home.
	IT Administrator	The person responsible for maintaining the system, managing user accounts, and ensuring it runs smoothly. (e.g., Roy from the city's IT department). Primary Concern: System stability, security, ease of backup/restore, and minimal manual intervention required.

Stakeholder Map for LMS (Development Team)

Category	Stakeholder Role	Description & Specific Examples
Development Team	Project Manager	Manages the project timeline, budget, and resources. Primary Concern: Clear requirements, realistic deadlines, and managing stakeholder expectations.
	Software Developers	The team building the system. Primary Concern: Clear and non-contradictory requirements, a sensible technology stack, and adequate time for design and testing.
	QA / Test Engineers	Responsible for verifying the system works as intended. Primary Concern: Well-defined test cases, access to test environments, and clear criteria for what constitutes a "bug" vs. a "new feature."
	UI/UX Designer	Designs the user interface for both staff and members. Primary Concern: Creating an intuitive, accessible, and efficient user experience based on actual user workflows.

Stakeholder Map for LMS (External Systems)

Category	Stakeholder Role	Description & Specific Examples
External Systems	City IT Department	May impose constraints on hosting, security policies, or integration with other city-wide systems.
	Email Service Provider	The external system (e.g., SendGrid, the library's own SMTP server) used for sending notifications. Concern: Reliable delivery of emails.
	Payment Gateway Provider (Future)	For handling online fine payments. (e.g., Stripe, PayPal). Concern: Secure and compliant financial transactions.

Pre-Architecture for LMS (contd...)

- **Step 2:** Stakeholder Engagement Sessions
- **Goal:** Understand Everyone's Concerns and Needs
- **How to Perform:** Engagement Sessions (Interviews, Workshops etc.)

- After identifying the stakeholders, the architect conducts focused sessions (interviews, workshops) with each group to dig into their specific concerns.

利益相关者参与会议:
Stakeholder Engagement Sessions:

- 一对一访谈：图书馆馆长
- One-on-One Interview: Library Director
- 主持研讨会：图书馆员与图书馆工作人员
- Facilitated Workshop: Librarians & Library Staff
- 一对一面试：IT管理员
- One-on-One Interview: IT Administrator
- 用户反馈环节：图书馆会员（读者）
- User Feedback Session: Library Members (Patrons)
- 协作工作坊：开发团队
- Collaborative Workshop: Development Team

一对一访谈：图书馆馆长（埃莉诺·范斯博士）
One-on-One Interview: Library Director (Dr. Eleanor)

Format: One-on-One Interview (Best for C-level, confidential strategic concerns)

Attendees: Architect, Project Manager, Dr. Vance

Duration: 60 minutes

Architect's Opening: "Thank you for your time, Dr. Vance. Our goal today is to understand your vision for this new system. We want to make sure it delivers real value to the library and the community. There are no right or wrong answers; we're just focused on understanding your perspective."

Strategic & Business Questions:

- ❑ "Looking 3-5 years ahead, how should this new system change the library's relationship with the community?"
- ❑ "What are the top two or three key performance indicators (KPIs) you will use to personally judge this project a success? (e.g., a 15% increase in membership, a 20% reduction in staff time spent on manual tasks?)"
- ❑ "What is the budget range you have in mind for the initial build, and for ongoing annual maintenance?"
- ❑ "Are there any 'hard stops' or absolute constraints we must work within? For example, a mandatory go-live date or a specific technology we must use or avoid?"
- ❑ "Could you describe a frustrating limitation of the current system from a management or reporting perspective?"

图书馆馆长（主要事务）
Library Director (Primary Concerns)

- Budget & ROI: "This system must justify its cost by increasing efficiency and member engagement within 18 months."
- Metrics: "I need reports on popular genres, membership growth, and loan cycles to make purchasing decisions."
- Reputation: "The new system must be reliable and modern to enhance the library's image in the community."
- Compliance: "We must protect member data according to privacy laws."

Facilitated Workshop: Librarians & Library Staff (Sarah, Tom, etc.)

Format: Focus Group Workshop (Best for gathering diverse user experiences and building consensus)
Attendees: 4-6 library staff members, UX Designer, Architect (as facilitator), a Note-taker
Duration: 90 minutes
Materials: Whiteboard, sticky notes, markers.

Architect's Opening: "Thank you all for coming. You are the experts on the day-to-day operations. We need your help to build a system that actually makes your jobs easier and helps our members. Today, we're going to talk about what works, what doesn't."

Activity 1: The "Pain Point" Brainstorm (15 mins)
Instruction: "Thinking about your daily tasks—checking books in/out, managing holds, dealing with fines—write down the single most frustrating thing on a sticky note. One pain point per note. Be specific."
Examples from Staff: "It takes 5 clicks to check out a single book." "The system freezes if two people try to place a hold on the same item." "I have no way of knowing a book is overdue until the member brings it back.“

Activity 2: "A Day in the Life" Walkthrough (30 mins)
Instruction: "Let's walk through a busy Saturday. Sarah, you're at the front desk. A family of four comes in, each with a stack of books to check out. Then, a member wants to pay a fine. Then, another member is furious because a book they placed a hold on is missing. Walk us through what you do with the current system, step-by-step, and where the bottlenecks are."

Facilitated Workshop: Librarians & Library Staff (Sarah, Tom, etc.) contd...

Activity 3: "Magic Wand" Ideation (20 mins)
Instruction: "If you had a magic wand and could change anything, what would the perfect system do for you? Don't worry about how it's built. Think about features."
Probing Questions:
"What information should be on the very first screen you see when you log in?"
"How could the system help you proactively, like alerting you to potential problems?"
"Describe the fastest possible way to check out five books for one member.“

Activity 4: Prioritization (10 mins)
Instruction: "We've gathered all these pain points and ideas. Now, each of you gets three 'dots'. Place your dots on the three issues or ideas that are most important to you. This will help us know what to tackle first."

图书馆员与工作人员（Sarah，Tom）主要关注点：Librarians & Staff (Sarah, Tom) Primary Concerns:

- Efficiency: "I need to check out a stack of 5 books for a member in under 60 seconds. The process must be fast and have minimal clicks."
- Accuracy: "I cannot check out a book that is already on loan or marked as lost. The inventory count must be accurate."
- Usability: "The screen for managing overdue fines must be clear. I should not need a 100-page manual to do my job."
- Problem Handling: "The system must easily handle edge cases, like a member who has lost a book."

一对一访谈：IT管理员（罗伊）
One-on-One Interview: IT Administrator (Roy)

Format: Technical Deep-Dive Interview
Attendees: Architect, Lead Developer, Roy
Duration: 45 minutes

Architect's Opening: "Roy, we want to make sure the new system is a joy for you to maintain, not a burden. We need to understand your environment, your constraints, and your biggest fears so we can design for them."

Technical & Operational Questions:
- ❑ "What is the current standard for user authentication? Are we integrating with an existing Active Directory, or is local authentication acceptable?"
- ❑ "Walk us through your current backup and disaster recovery process. What would the new system need to do to fit seamlessly into that?"
- ❑ "What are the security and compliance policies we absolutely cannot violate? (e.g., password complexity, data encryption standards, audit logging)."
- ❑ "What are your expectations for monitoring? What metrics do you need to see on a dashboard? (e.g., database connections, application uptime, error rates)."
- ❑ "What is your preferred deployment process? Do you use any specific tools for server provisioning or configuration management?"

Scenario-Based Question:
- ❑ "It's 2 AM on a Sunday, and the library's website is down. From your perspective, what is the ideal process for diagnosing and resolving the problem? What tools and logs would you expect to have?"

IT管理员（罗伊）主要关注点：
IT Administrator (Roy) Primary Concerns:

- Maintainability: "I need clear documentation for backups, updates, and disaster recovery. I don't want to get paged at 3 AM."
- Security: "The system must enforce strong passwords and have a clear audit log for sensitive actions. It must be patched against known vulnerabilities."
- Integration: "It must work with our existing Windows Active Directory for staff logins (if required) and our email server."
- Performance: "It should run stably on our standard server hardware without constant performance tuning."

用户反馈环节：图书馆会员（读者）
User Feedback Session: Library Members (Patrons)

Format: Guerrilla Usability Testing & Short Survey (Best for end-users who are not invested in the project)
Attendees: UX Designer, a few voluntary library members (e.g., offered a free coffee for 10 minutes of their time)
Duration: 10-15 minutes per member

UX Designer's Opening: "Hi there! We're designing a new website and app for the library, and we'd love your input. It will only take about 10 minutes. We're testing ideas, not you, so please be brutally honest!"
Activity: Paper Prototype Test
Setup: The designer has paper printouts or simple digital mockups of key screens: the home page, search results, a book detail page, and the login/account page.
Instruction: "Okay, let's say you're at home and you remember a book you want to read, 'The Three-Body Problem'. Can you show me how you'd find it and see if it's available?"
Observer Notes: Where do they click first? The search bar? The navigation? Do they understand the "Place Hold" button?
Follow-up: "Great, you found it and it's available. Now, you want to make sure you can get it. What would you do next?" (This tests if they understand the hold process).

Probing Questions:
- ❑ "What on this screen is most important to you?"
- ❑ "Is there anything missing that you'd expect to see here?"
- ❑ "How would you check what books you currently have checked out?"

Quick Survey Questions (while they are there):
- ❑ "On a scale of 1-5, how frustrating is it when you can't tell if a book is on the shelf?"
- ❑ "What device do you use most to access the library's website: phone, computer, or tablet?"

图书馆会员（读者）主要关注点：
Library Members (Patrons) Primary Concerns:

- Convenience: "I want to search for a book from my phone and know if it's available before I come to the library."
- Transparency: "I need a clear list of the books I have borrowed and their due dates. I want to know exactly what my fines are and why."
- Simplicity: "Placing a hold on a book should be a one-click process. Renewing my books online should be easy."
- Reliability: "The system should not be 'down for maintenance' during evenings or weekends when I need it most."

协作工作坊：开发团队
Collaborative Workshop: Development Team

Format: Technical Planning Workshop
Attendees: Developers, QA Engineers, DevOps, Architect (as facilitator)
Duration: 60 minutes

Architect's Opening: "Team, we've gathered a lot of requirements from the business and users. Now, we need your expertise to pressure-test them and plan how we'll build this. Let's talk about risks, technology, and how we can set ourselves up for success."

Activity 1: "Requirements Clarification" (20 mins)
Instruction: "Here are the top 5 user stories we've heard. Let's read them one by one and ask the '5 Whys' to get to the root of what's needed. For example, 'The user needs to check out 5 books in under 60 seconds.' What does that really mean for the database, the UI, and the hardware?"

Activity 2: "Technology & Risk Storming" (30 mins)
Instruction: "Let's brainstorm on two whiteboards."
- ❑ Board 1 - Technology: "What technologies are we considering for the frontend, backend, and database? List pros and cons. Let's debate."
- ❑ Board 2 - Risks: "What are our biggest technical risks? (e.g., 'How will we handle peak load during summer reading program?')."

Activity 3: "Definition of Done" (10 mins)
Instruction: "For a feature like 'Member can place a hold,' what does 'Done' mean? Is it just coded? Does it need unit tests, integration tests, UX review, and documentation? Let's agree on our standard."

开发团队主要关注点
Development Team Primary Concerns

- **Clarity = No Ambiguity**
  - Bad: "Fast", "Easy", "User-friendly"
  - Good: "Under 2 seconds", "3 clicks maximum", "95% user satisfaction"
- **Feasibility = Reality Check**
  - Ask: "Can we actually build this with our time, budget, and skills?"
  - Always have a Plan B for infeasible requests
- **Technology = Future-Proofing**
  - Choose technologies that won't be obsolete next year
  - Balance innovation with stability
  - Consider team skills and learning curve
- **Change Management = Project Protection**
  - Remember: It's easier to say "no" to a documented request than to an informal ask
  - Without process: Chaos, missed deadlines, blown budgets
  - With process: Controlled evolution, predictable outcomes

Summary Output of Phase 1, Step 1 and 2 Document: Stakeholder Analysis Register

Stakeholder	Role	Key Concerns & Needs	Influence (H/M/L)	Engagement Strategy
Dr. Eleanor	Library Director	ROI, Reporting, Community Reputation	High	Regular steering committee meetings, formal sign-offs.
Librarians (Sarah, Tom)	Primary Users	Efficiency, Accuracy, Ease of Use	High	Frequent demos and usability testing sessions; they are the "voice of the user."
Library Members	End Users	Convenience, Transparency, Reliability	Medium	Surveys, beta testing groups, feedback forms within the application.
Roy (IT Admin)	System Maintainer	Security, Maintainability, Stability	High	Involve in technical decisions, review deployment plans, and security design.
Development Team	Builders	Clear Requirements, Feasibility, Good Tech	Medium	Daily stand-ups, technical meetings, include in requirement analysis sessions.

利益相关者综合关注点
Consolidated Stakeholder Concerns

- **From Director:** "Must demonstrate ROI via increased membership and efficient resource allocation. Hard deadline for summer reading program launch."
- **From Librarians:** "Check-out process must be under 60 seconds. System must prevent double-loaning. Overdue fine screen must be crystal clear and easy to modify."
- **From Members:** "The mobile search experience is critical. Must see real-time availability. The process from search-to-hold must be seamless (max 3 clicks)."
- **From IT Admin:** "Must integrate with city-wide SMTP server. Must have comprehensive audit logs. Database backups must be automated and tested."
- **From Dev Team:** "Requirements for 'efficiency' need to be quantifiable. Need a clear CI/CD pipeline from day one."

These concrete outputs from the engagement sessions are the raw material that will now be formalized into Architectural Requirements in the next step.

A. Functional Requirements (What the system will DO)

- These are derived from the desired features.

ID	Requirement Description	Source (Stakeholder Concern)
FR1	The system shall provide a public search interface that allows users to search the catalog by Title, Author, ISBN, and Genre.	Member Convenience, Librarian Efficiency
FR2	The system shall display real-time availability status (Available, Checked-Out, On-Hold) for each item in the search results.	Member Transparency
FR3	An authenticated member shall be able to place a hold on any available or checked-out item.	Member Convenience
FR4	The system shall allow a librarian to check out multiple items to a member in a single transaction by scanning item barcodes and entering a member ID.	Librarian Efficiency (< 60 sec for 5 items)

ID	Requirement Description	Source (Stakeholder Concern)
FR5	The system shall automatically calculate and assign overdue fines based on configurable rules (e.g., $0.25/day per item, 14-day loan period).	Business Logic, Director ROI
FR6	The system shall provide a dedicated interface for librarians to view, waive, and process fine payments.	Librarian Usability (Clear fine screen)
FR7	The system shall send automated email notifications for: a) Available Holds, b) Overdue Items (3-day, 7-day, 14-day reminders).	Member Convenience, Director ROI
FR8	The system shall enforce Role-Based Access Control (RBAC): ❑ MEMBER: Can access FR1, FR2, FR3, and their own account. ❑ LIBRARIAN: Can access all Member functions + FR4, FR6, and inventory management. ❑ ADMIN: All access, including user role management.	IT Security, Business Rules
FR9	The system shall allow librarians to add, edit, and deactivate items in the catalog (Books, DVDs, etc.).	Librarian Efficiency

B. Non-Functional Requirements (How well the system will DO it)

- These are the quality attributes and constraints that shape the architecture. They are often measurable and testable.
- B.1. Performance & Scalability

ID	Requirement Description	Metric & Source
NFR-P1	Catalog Search Response Time: The public catalog search shall return results in under 2 seconds for 95% of queries under a normal load of 50 concurrent users.	Metric: ≤ 2 sec response time for 95th percentile. Source: Director (Modern Image), Member Convenience.
NFR-P2	Check-Out Transaction Time: A trained librarian shall be able to complete a check-out transaction for 5 items in under 60 seconds, from login to completion.	Metric: ≤ 60 sec end-to-end transaction time. Source: Librarian Efficiency (Workshop).
NFR-P3	Concurrent Users: The system shall support up to 100 concurrent active users without significant degradation of performance.	Metric: 100 concurrent users. Source: Projected growth, Summer Reading peak.

B.2. Availability & Reliability

ID	Requirement Description	Metric & Source
NFR-A1	System Uptime: The core system (catalog, check-in/out) shall be available 99.9% during library operating hours (6:00 AM - 10:00 PM local time, 7 days a week). Planned maintenance must be possible outside this window.	Metric: 99.9% uptime = ~ 44 minutes downtime/month. Source: Member Reliability, Director Reputation.
NFR-A2	Data Integrity: The system must prevent double-loaning of the same physical copy. The inventory count must be accurate at the time of transaction.	Metric: 0% occurrence of double-loans in acceptance testing. Source: Librarian Accuracy.

B.3. Security

ID	Requirement Description	Metric & Source
NFR-S1	Authentication: User passwords shall be stored using a strong, salted, and hashed algorithm (e.g., bcrypt).	Metric: Passwords verified as non-recoverable plaintext; use of industry-standard hashing in code review. Source: IT Security, Privacy Laws.
NFR-S2	Authorization: The system shall enforce RBAC as defined in FR8. A user with the 'MEMBER' role shall never be able to access librarian functions.	Metric: Penetration testing confirms access violations are impossible. Source: IT Security.
NFR-S3	Audit Logging: The system shall maintain an immutable log of all security-sensitive actions (logins, role changes, fine waivers over $10).	Metric: Logs exist and are searchable for key events. Source: IT Security (Audit Logs).

B.4. Usability

ID	Requirement Description	Metric & Source
NFR-U1	Mobile Responsiveness: The public catalog and member account pages shall be fully functional and usable on modern mobile devices (screen widths down to 360px).	Metric: UI passes responsive design tests on standard device emulators. Source: Member Convenience (Mobile Access).
NFR-U2	Accessibility: The system shall meet WCAG 2.1 Level AA guidelines to ensure accessibility for users with disabilities.	Metric: Audit report from an accessibility testing tool (e.g., axe). Source: Public Service Mandate.

B.5. Integrability & Operations

ID	Requirement Description	Metric & Source
NFR-I1	Email Integration: The system shall integrate with the library's existing SMTP server to send all notifications.	Metric: Test emails are successfully sent and received through the production SMTP server. Source: IT Integration.
NFR-O1	Backup & Recovery: The application and database must support a full backup and restore process. The system must be recoverable to a state no more than 24 hours old in the event of a critical failure.	Metric: A documented and tested RTO (Recovery Time Objective) of < 4 hours and RPO (Recovery Point Objective) of < 24 hours. Source: IT Maintainability.

C. Business & Technical Constraints

- These are non-negotiable decisions that frame the project.

ID	Constraint Description	Type	Source
C1	The system must be developed with a total budget not exceeding $150,000.	Business	Director (Budget)
C2	The system must be fully operational before the start of the Summer Reading Program on June 1st, 2025.	Business	Director (Hard Deadline)
C3	The system must be developed using the Java/Spring Boot technology stack to align with in-house developer expertise.	Technical	Development Team
C4	The system must be deployable on a standard Linux virtual machine hosted on the city's approved cloud provider (AWS).	Technical	IT Administrator

“建筑驱动力”
The "Architectural Drivers"

Now identify the Architectural Drivers—the handful of most critical requirements that will have the greatest influence on the architectural design.

For the LMS, the primary drivers are:

- ❑ NFR-P1 & NFR-P2 (Performance): The need for fast search and efficient check-out transactions directly points towards a well-structured database, effective caching strategies, and a responsive, simple UI.
- ❑ NFR-S1 & NFR-S2 (Security): The need to protect member data and enforce RBAC mandates a robust security framework integrated from the ground up.
- ❑ NFR-A1 (Availability): The requirement for high uptime during operating hours influences decisions around deployment, monitoring, and fault tolerance.
- ❑ C2 (Business Constraint - Deadline): The fixed deadline strongly favors a Modular Monolith over a more complex Microservices architecture, as it is faster to develop and deploy initially.

This complete, prioritized, and testable list of requirements is the final output of Phase 1.

It provides a crystal-clear, shared understanding for all stakeholders and gives the development team a solid foundation upon which to begin the architectural design.

建筑视角与模型
Architectural Views and Models

C4 Model Levels [C4模型等级]

The C4 model is a hierarchical approach to software architecture diagramming that provides different levels of abstraction, from high-level system context down to low-level code..

Level 1: System Context Diagram

Purpose: Shows the big picture of the system landscape

Answers: "What is the system and how does it fit into the world around it?"

Target audience: Non-technical stakeholders (business owners, end users, project managers)

Key Elements:

- ❑ Your system as a single box
- ❑ People (actors, users, roles) that interact with the system
- ❑ External systems that your system depends on or integrates with

Rules:

- ❑ No technical details
- ❑ No technology choices
- ❑ Show only the system boundary and external interactions
- ❑ Typically 1-2 pages maximum

Level 2: Container Diagram

Purpose: Shows the high-level technology shape of the system

Answers: "What are the main applications, data stores, and how do they interact?"

Target audience: Technical people both inside and outside the development team

Key Elements:

- ❑ Containers = separately runnable/deployable units
- ❑ Web applications/Mobile apps/Desktop applications
- ❑ Databases
- ❑ File systems
- ❑ Relationships between containers
- ❑ Technology choices for each container

Rules:

- ❑ Show all containers that make up the system
- ❑ Indicate the technology stack for each container
- ❑ Show how containers communicate
- ❑ Typically 3-5 containers for most systems

Level 3: Component Diagram

Purpose: Zooms into one container to show its internal structure

Answers: "What are the key logical components and how do they interact?"

Target audience: Software architects and developers

Key Elements:

- ❑Components = modular, replaceable parts with well-defined interfaces
- ❑Relationships between components
- ❑Technology-agnostic design (focus on responsibilities, not implementation)
- ❑Can show multiple diagrams for different containers

Rules:

- ❑Focus on one container at a time
- ❑Show major structural building blocks
- ❑Indicate responsibilities of each component
- ❑Can show both logical and physical components

Level 4: Code Diagram

Purpose: Shows how components are implemented as code

Answers: "What are the key classes, interfaces, and their relationships?"

Target audience: Developers (primarily those working on the system)

Key Elements:

- ❑Classes and interfaces
- ❑Methods and attributes
- ❑Relationships (inheritance, composition, dependency)
- ❑Design patterns being used

Rules:

- ❑Typically uses UML class diagrams
- ❑Should be generated from code where possible
- ❑Focus on important or complex areas only
- ❑Not every class needs to be shown

C4 Model Summary Table

Level	Scope	Audience	Key Question	Elements
1. Context	Entire system in environment	Everyone	"What does the system do and for whom?"	System, people, external systems
2. Containers	Applications and data stores	Technical stakeholders	"What's the high-level technology shape?"	Containers, technologies, interactions
3. Components	Single container	Developers, architects	"How is the container structured?"	Components, responsibilities, interfaces
4. Code	Single component	Developers	"How is the component implemented?"	Classes, interfaces, relationships

System Context Diagram (C4 Level 1)

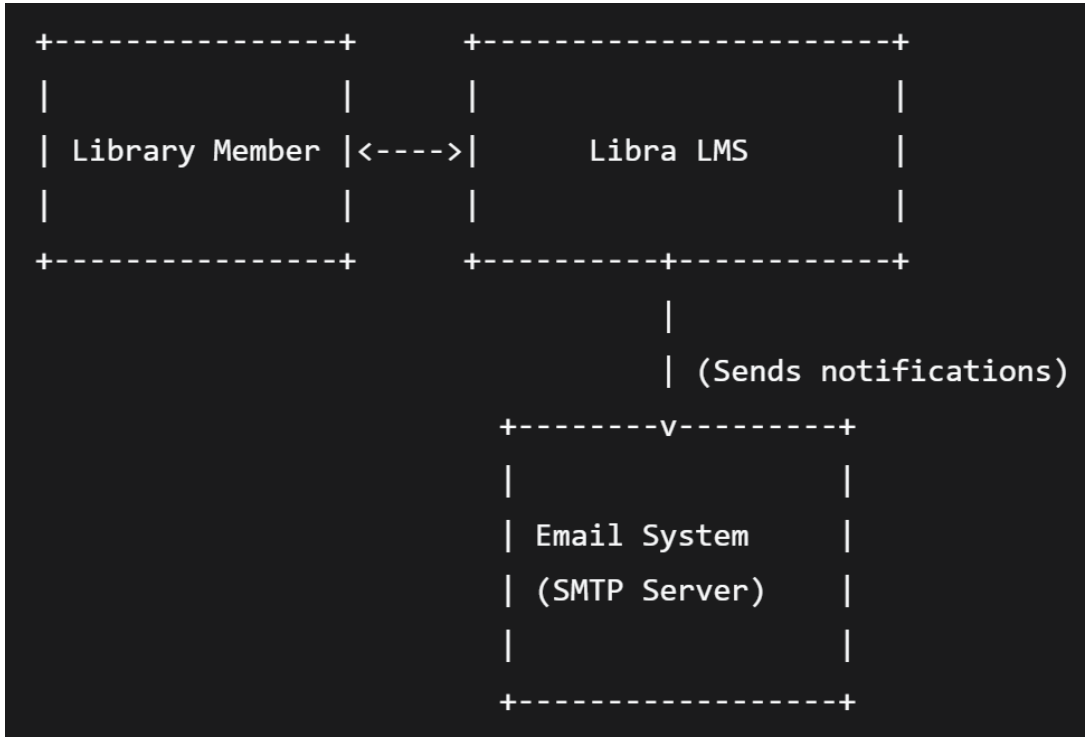
Purpose: Shows the system as a whole and its relationships with external entities and systems.

Key Elements:

- Library Member: Primary user who searches catalog, manages account, places holds
- Librarian/Staff: Administrative user who manages inventory, check-in/out, fines
- Email System: External SMTP server for sending notifications (holds available, overdue reminders)

Interactions:

- Members and Librarians interact with Libra LMS via web browsers
- LMS sends email notifications to members via external SMTP server
- No direct integration with payment systems in initial version (future consideration)



4.2. Container Diagram (C4 Level 2)

Purpose: Shows the high-level technology choices and how responsibilities are distributed across containers.

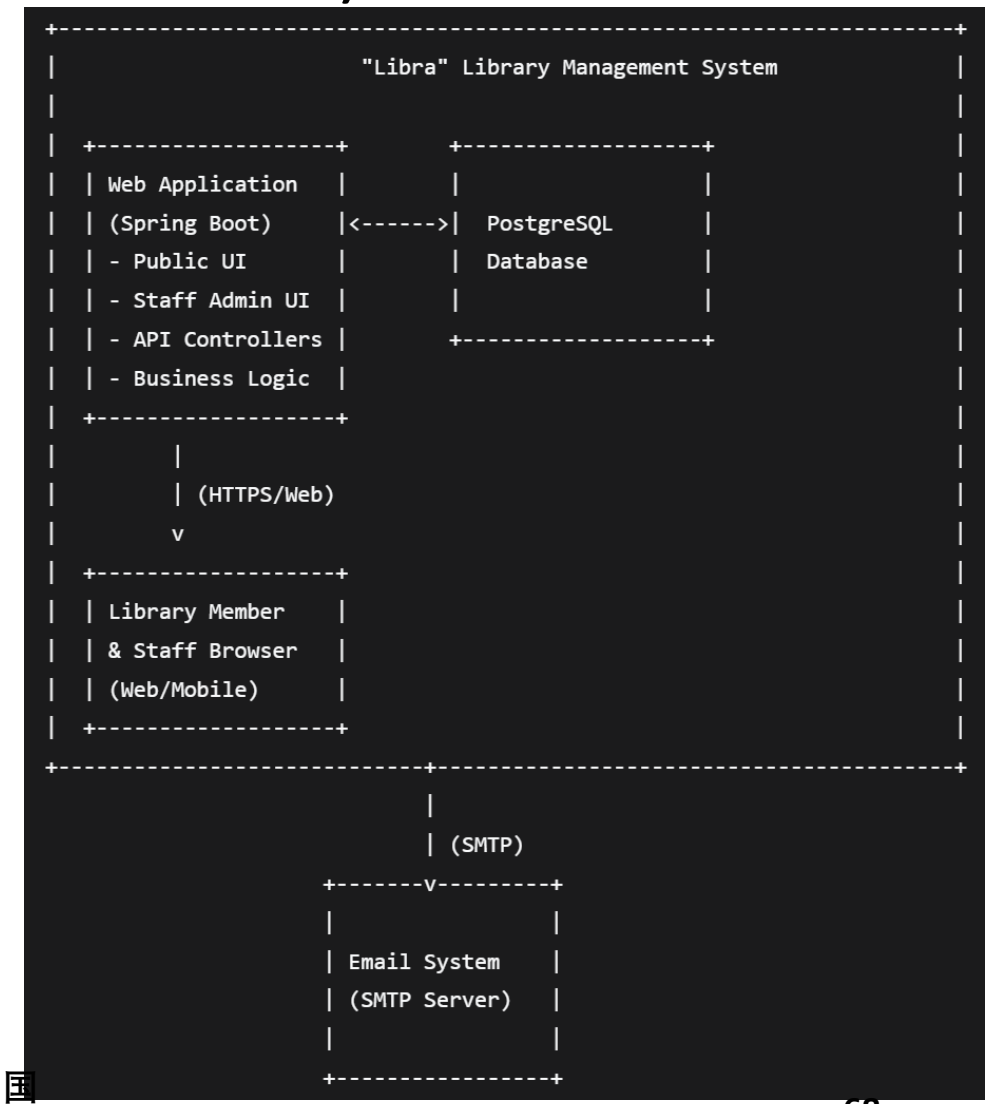
Web Application

Technology: Java 17, Spring Boot 3.x, Embedded Tomcat

Responsibilities:

- ❑Serve public-facing website for members
- ❑Serve staff administration portal
- ❑Host REST API endpoints
- ❑Execute business logic and validation
- ❑Handle authentication and authorization

Communication: HTTP/HTTPS with browsers, JDBC with database, SMTP with email server



4.2. Container Diagram (C4 Level 2) Contd…

PostgreSQL Database
Technology: PostgreSQL 15+
Responsibilities:
- ❑ Persistent storage of all application data
- ❑ Enforce data integrity through constraints
- ❑ Support complex queries for reporting and search

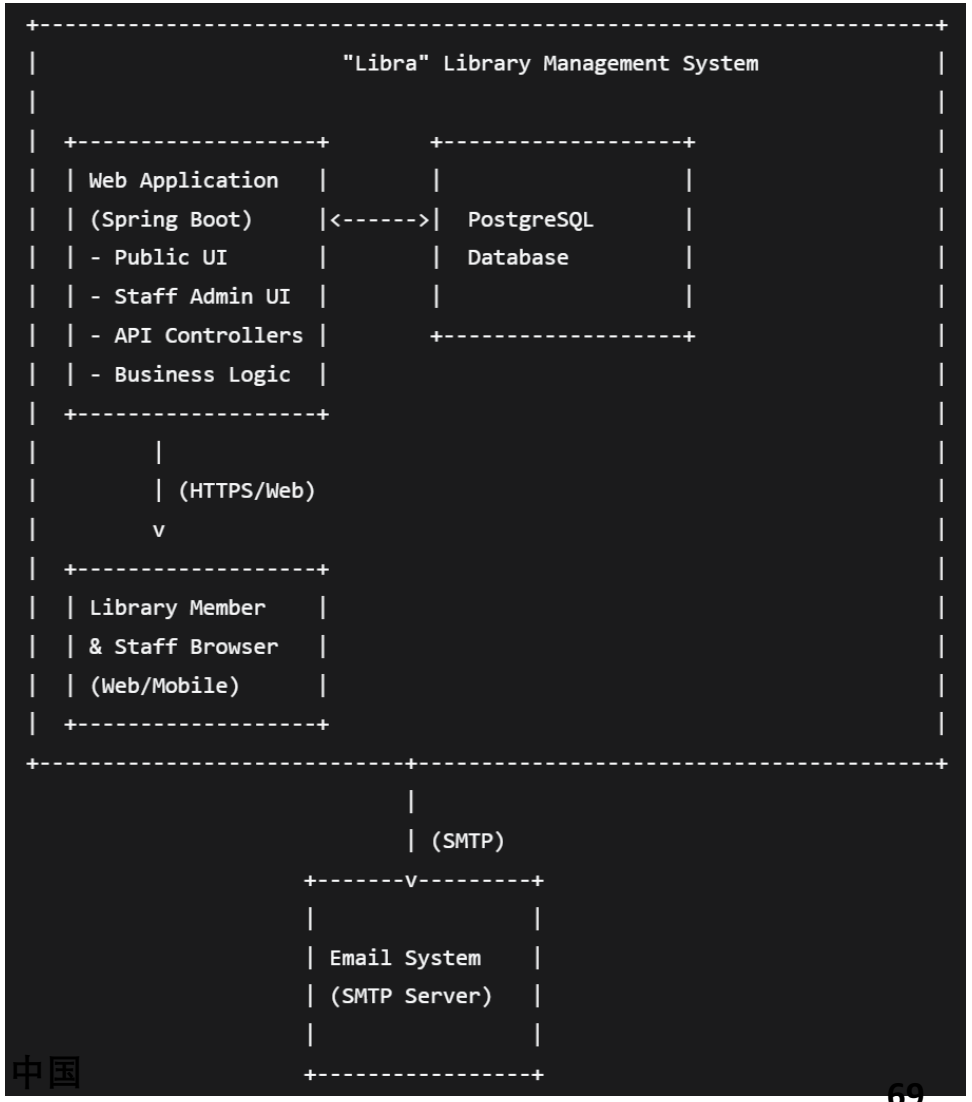
Data Includes: Users, books, loans, holds, fines, inventory

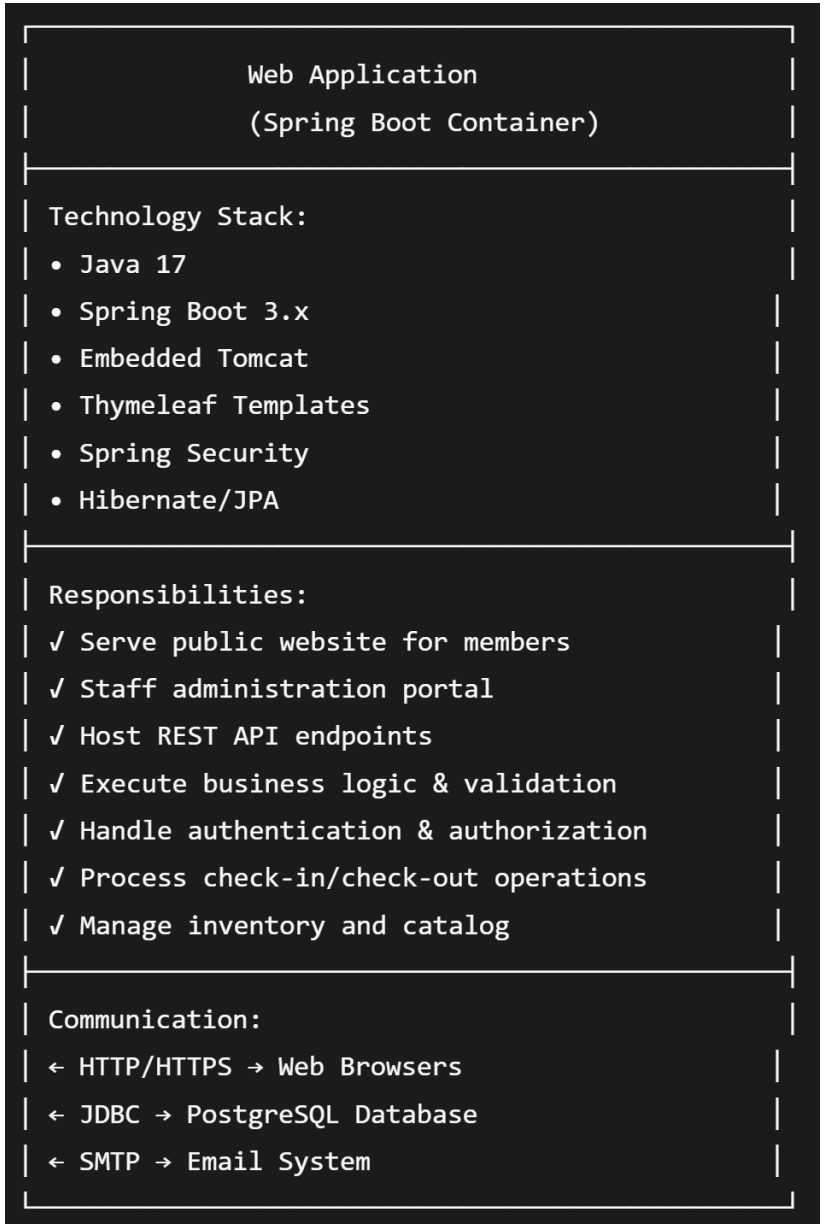
User Web Browser
Technology: Modern web browser (Chrome, Firefox, Safari, Edge)
Responsibilities:
- ❑ Render user interfaces
- ❑ Execute client-side validation
- ❑ Maintain user session

Access: Both desktop and mobile responsive

Email System
Technology: External SMTP server
Responsibilities:
- ❑ Deliver email notifications to members
- ❑ Handle email queuing and delivery

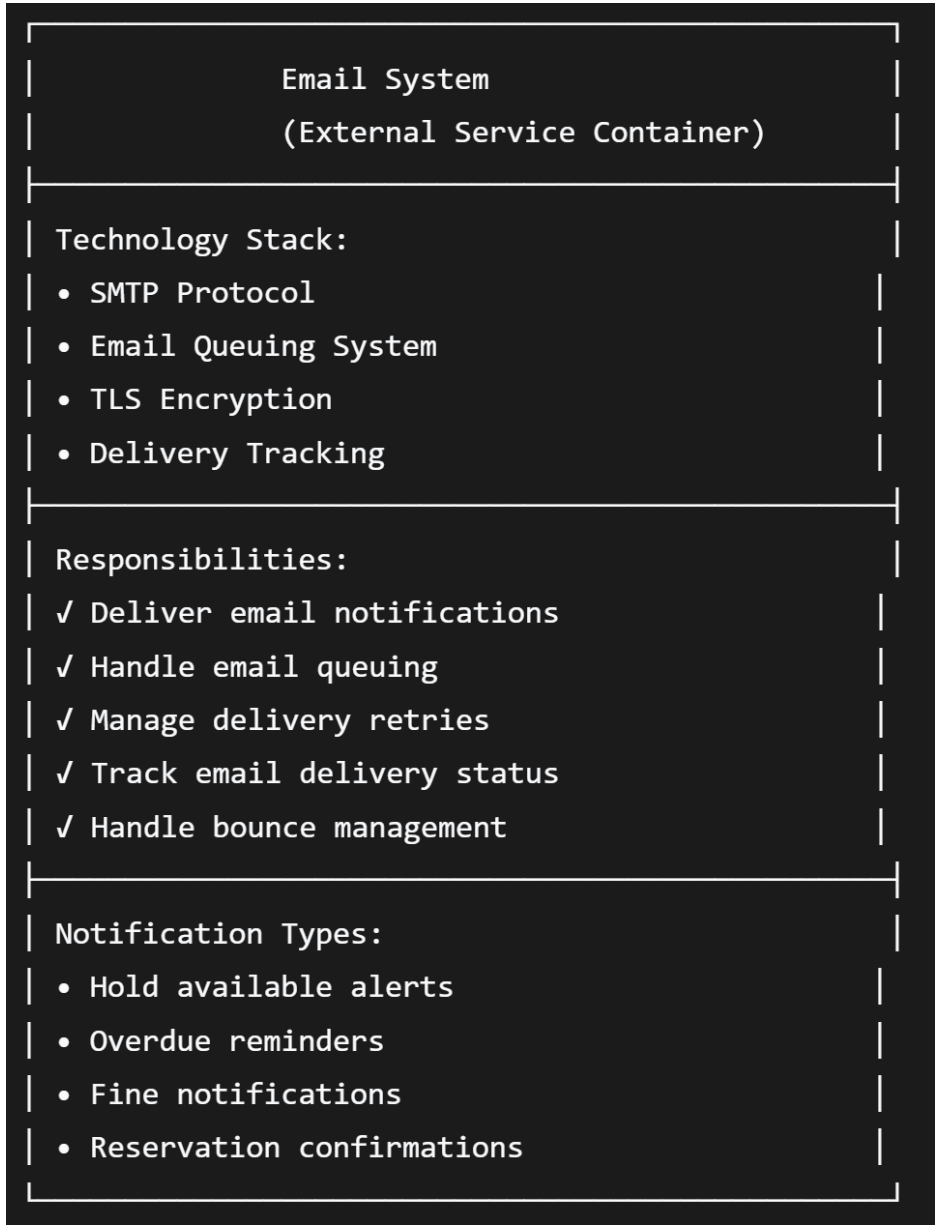
Protocol: SMTP




Web Application
(Spring Boot Container)
Technology Stack:
• Java 17
• Spring Boot 3.x
• Embedded Tomcat
• Thymeleaf Templates
• Spring Security
• Hibernate/JPA
Responsibilities:
✓ Serve public website for members
✓ Staff administration portal
✓ Host REST API endpoints
✓ Execute business logic & validation
✓ Handle authentication & authorization
✓ Process check-in/check-out operations
✓ Manage inventory and catalog
Communication:
← HTTP/HTTPS → Web Browsers
← JDBC → PostgreSQL Database
← SMTP → Email System


PostgreSQL Database
(Data Storage Container)
Technology Stack:
• PostgreSQL 15+
• Connection Pooling (HikariCP)
• ACID Compliance
• Advanced Indexing
Responsibilities:
✓ Persistent storage of all application data
✓ Enforce data integrity constraints
✓ Support complex search queries
✓ Transaction management
✓ Backup and recovery operations
Data Entities:
• Users (Members, Staff)
• Books (Titles, Copies)
• Loans (Checkouts, Returns)
• Holds (Reservations)
• Fines (Overdue charges)
• Inventory (Stock management)


Web Browser
(Client Container)
Technology Stack:
• Modern Browsers (Chrome, Firefox, Safari)
• HTML5/CSS3/JavaScript
• Responsive Design
• Session Storage
Responsibilities:
✓ Render user interfaces
✓ Execute client-side validation
✓ Maintain user sessions
✓ Handle form submissions
✓ Display real-time updates
Access Methods:
• Desktop computers
• Mobile devices (responsive)
• Tablets
• Public access terminals


Email System
(External Service Container)
Technology Stack:
• SMTP Protocol
• Email Queuing System
• TLS Encryption
• Delivery Tracking
Responsibilities:
✓ Deliver email notifications
✓ Handle email queuing
✓ Manage delivery retries
✓ Track email delivery status
✓ Handle bounce management
Notification Types:
• Hold available alerts
• Overdue reminders
• Fine notifications
• Reservation confirmations

容器交互序列
Container Interaction Sequence

1. Member Search Flow:

[Browser] → HTTP → [Web App] → JDBC → [Database] → Return Results

2. Book Checkout Flow:

[Browser] → HTTP → [Web App] → JDBC → [Database] → Update Status → SMTP → [Email] → Confirmation

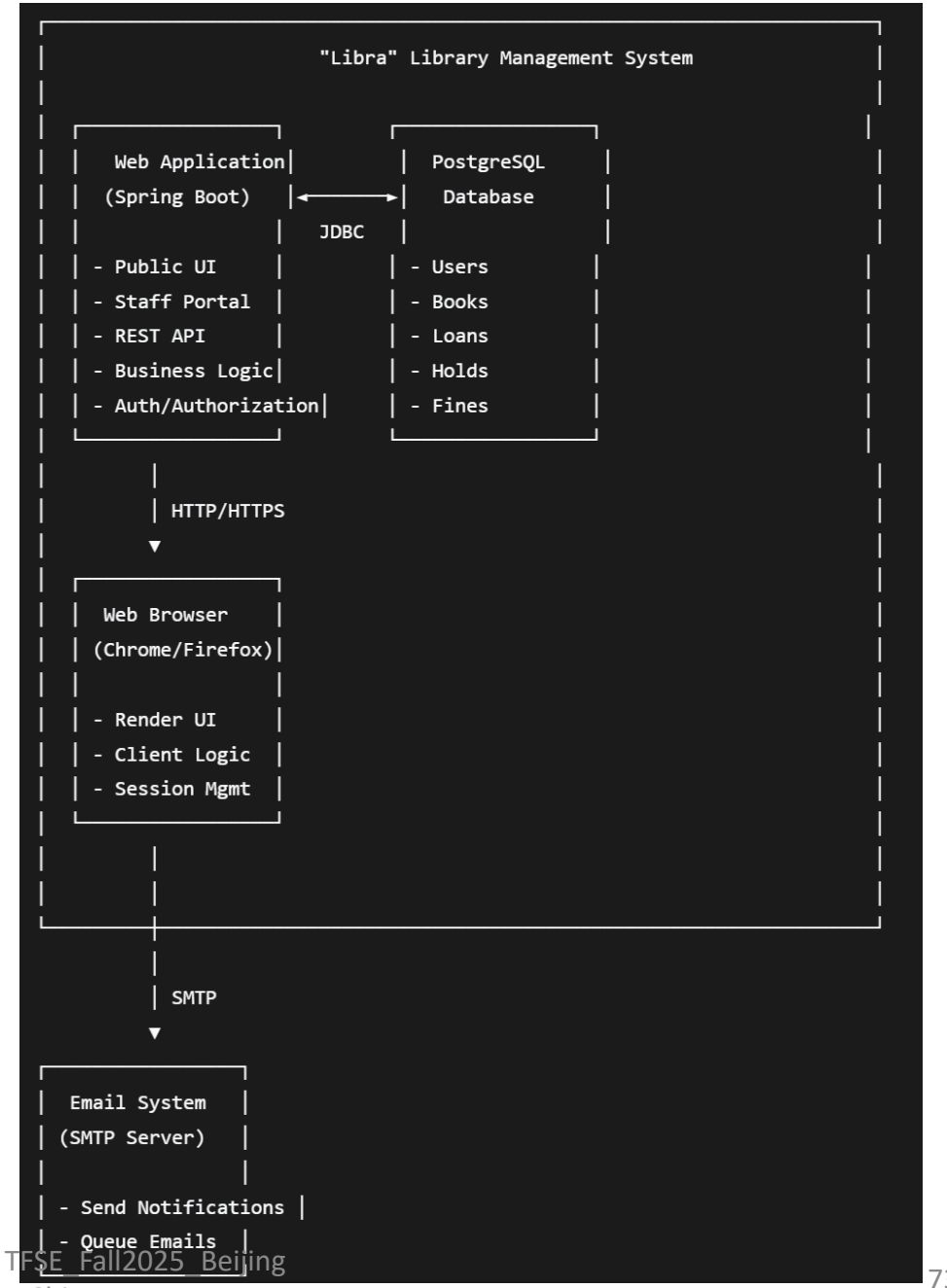
3. Staff Administration Flow:

[Browser] → HTTP → [Web App] → JDBC → [Database] → Return Reports → Render UI

4. Notification Flow:

[Web App] → Scheduled Job → JDBC → [Database] → Get Overdue → SMTP → [Email] → Send Reminders


"Libra" Library Management System
Web Application
(Spring Boot)
- Public UI
- Staff Admin UI
- API Controllers
- Business Logic
<------>
PostgreSQL
Database
(HTTPS/Web)
Library Member
& Staff Browser
(Web/Mobile)
(SMTP)
Email System
(SMTP Server)


"Libra" Library Management System
Web Application
(Spring Boot)
- Public UI
- Staff Portal
- REST API
- Business Logic
- Auth/Authorization
JDBC
PostgreSQL
Database
- Users
- Books
- Loans
- Holds
- Fines
HTTP/HTTPS
Web Browser
(Chrome/Firefox)
- Render UI
- Client Logic
- Session Mgmt
SMTP
Email System
(SMTP Server)
- Send Notifications
- Queue Emails

Component Diagram for the Web Application (C4 Level 3)

- Purpose: Shows the major structural building blocks and their interactions within the Spring Boot application.

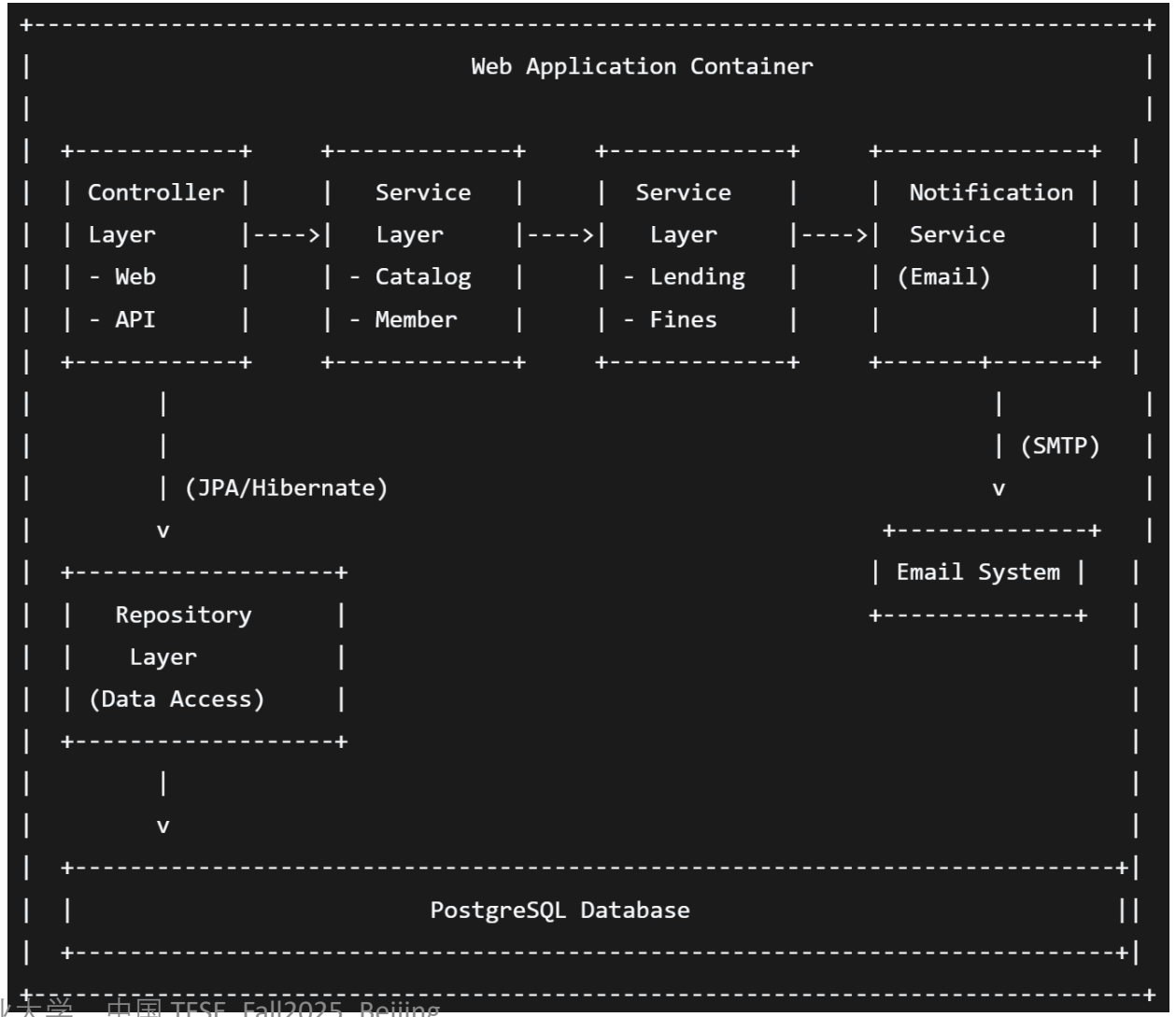
Component Diagram[Key Components Explained:]

- **Controller Layer**
  - ❑ CatalogController: Handles public catalog search and browsing
  - ❑ MemberController: Manages member accounts and self-service functions
  - ❑ LoanController: Processes check-in/check-out operations
  - ❑ AdminController: Provides staff administration functions
  - ❑ AuthenticationController: Handles login/logout and session management

- **Service Layer**
  - ❑ CatalogService: Implements search logic, inventory management
  - ❑ MemberService: Handles member registration, account management
  - ❑ LoanService: Core lending logic, due date calculation, renewal processing
  - ❑ FineService: Automatic fine calculation, payment processing, waiver logic
  - ❑ HoldService: Manage hold queues, notification triggers
  - ❑ SecurityService: Authentication, authorization, role management
  - ❑ ReportingService: Generate usage statistics and business reports

- **Repository Layer**
  - ❑ BookRepository: Data access for books, authors, inventory
  - ❑ MemberRepository: Member data and account management
  - ❑ LoanRepository: Loan history and current loans
  - ❑ FineRepository: Fine records and payment history
  - ❑ HoldRepository: Hold requests and queue management

- **Notification Service**
  - ❑ EmailService: Composes and sends email notifications
  - ❑ TemplateEngine: Generates formatted email content
  - ❑ NotificationScheduler: Triggers time-based notifications (overdue reminders)



Data Model (Entity-Relationship Diagram)

Purpose: Shows the major data entities and their relationships.

Key Relationships:

- Member 1---N Loan: A member can have multiple loans
- BookCopy 1---1 Loan: A physical copy can have one active loan
- Book 1---N BookCopy: A book title can have multiple physical copies
- Member 1---N Hold: A member can place multiple holds
- Book 1---N Hold: A book can have multiple holds in queue
- Loan 1---1 Fine: Each overdue loan generates a fine record

Critical Data Constraints:

- Status Enums:
- BookCopy.status: AVAILABLE, CHECKED_OUT, ON_HOLD, MAINTENANCE
- Loan.status: ACTIVE, RETURNED, OVERDUE
- Hold.status: ACTIVE, FULFILLED, EXPIRED, CANCELLED
- Fine.status: OUTSTANDING, PAID, WAIVED

Detailed Entity Specifications:

Member
(Primary Entity)

Primary Key:
- member_id (Auto-increment)

Attributes:
- email (Unique, Not Null)
- password_hash (Not Null)
- first_name (Not Null)
- last_name (Not Null)
- phone
- total_fines (Decimal, Default: 0.00)
- role (ENUM: MEMBER, LIBRARIAN, ADMIN)
- active (Boolean, Default: true)
- join_date (Date)
- last_login (DateTime)

Relationships:
- One-to-Many with Loan
- One-to-Many with Hold
- One-to-Many with Fine

Book
(Catalog Management)

Primary Key:
- book_id (Auto-increment)

Attributes:
- isbn (Unique, Not Null)
- title (Not Null)
- author (Not Null)
- genre
- description (Text)
- publication_year
- publisher
- total_copies (Integer, Default: 0)
- available_copies (Integer, Default: 0)
- created_date (DateTime)

Relationships:
- One-to-Many with BookCopy
- One-to-Many with Hold

BookCopy
(Physical Inventory)

Primary Key:
- copy_id (Auto-increment)

Foreign Keys:
- book_id (References Book.book_id)

Attributes:
- barcode (Unique, Not Null)
- status (ENUM: AVAILABLE, CHECKED_OUT, ON_HOLD, MAINTENANCE)
- purchase_date (Date)
- purchase_price (Decimal)
- location (Varchar)
- notes (Text)

Relationships:
- Many-to-One with Book
- One-to-Many with Loan

Detailed Entity Specifications:

Loan
(Circulation Transactions)

Primary Key:
- loan_id (Auto-increment)

Foreign Keys:
- member_id (References Member.member_id)
- copy_id (References BookCopy.copy_id)

Attributes:
- checkout_date (Date, Not Null)
- due_date (Date, Not Null)
- return_date (Date, Nullable)
- status (ENUM: ACTIVE, RETURNED, OVERDUE)
- renewal_count (Integer, Default: 0)
- last_reminder_sent (Date)

Relationships:
- Many-to-One with Member
- Many-to-One with BookCopy
- One-to-One with Fine

Hold
(Reservation System)

Primary Key:
- hold_id (Auto-increment)

Foreign Keys:
- member_id (References Member.member_id)
- book_id (References Book.book_id)

Attributes:
- placed_date (DateTime, Not Null)
- expiry_date (DateTime, Not Null)
- status (ENUM: ACTIVE, FULFILLED, EXPIRED, CANCELLED)
- position (Integer)
- notification_sent (Boolean, Default: false)

Relationships:
- Many-to-One with Member
- Many-to-One with Book

Fine
(Financial Transactions)

Primary Key:
- fine_id (Auto-increment)

Foreign Keys:
- member_id (References Member.member_id)
- loan_id (References Loan.loan_id)

Attributes:
- amount (Decimal, Not Null)
- status (ENUM: OUTSTANDING, PAID, WAIVED)
- reason (Varchar)
- created_date (DateTime)
- paid_date (DateTime, Nullable)
- waived_by (References Member.member_id)
- waived_reason (Text)

Relationships:
- Many-to-One with Member
- One-to-One with Loan

Key Architectural Decisions Model

- Module Structure within Monolith:

```
libra-lms/
├── lms-core/            # Domain models and business logic
├── lms-catalog/         # Search, browsing, inventory management
├── lms-lending/         # Check-in/out, holds, renewals
├── lms-accounts/        # Member management, authentication
├── lms-fines/           # Fine calculation and management
├── lms-notifications/   # Email and alert system
└── lms-web/            # Web controllers and UI templates
```

这种模块化结构允许:
This modular structure allows for:

- ❑Clear separation of concerns
- ❑Independent testing of business domains
- ❑Potential future extraction to microservices if needed
- ❑Team organization around functional areas
- ❑All views are consistent with our chosen Modular Monolith architecture and support the quality attributes defined in our requirements, particularly performance, security, and maintainability.

Technology Stack [技术栈]

Layer	Technology Choice	Justification
Backend	Java 17, Spring Boot	Robust ecosystem, strong ORM support, high performance.
Web Layer	Spring MVC, Thymeleaf	Server-side rendering simplifies initial development and is SEO-friendly for the public catalog.
Database	PostgreSQL	Reliable, open-source RDBMS. Excellent for structured data like members, books, and transactions.
ORM	Hibernate (JPA)	Simplifies data access and reduces boilerplate code.
Security	Spring Security	Industry standard for handling authentication and authorization (RBAC).
Testing	JUnit, Mockito, TestContainers	Comprehensive testing framework.
Deployment	Docker, AWS EC2	Containerization for consistent environments; EC2 for simplicity and cost-effectiveness.

交叉切割关注设计
Cross-Cutting Concerns Design

Security Architecture

- **Authentication:** Members log in with email and password. Staff use a username and password.
- **Password Storage:** Passwords are hashed using bcrypt.
- **Authorization (RBAC):**
  - MEMBER: Can view own account, search catalog, place holds.
  - LIBRARIAN: Can perform all member actions + check-in/out, manage inventory.
  - ADMIN: All librarian actions + system management (user roles, etc.).
- **Web Security:** All communication over HTTPS.

交叉切割关注设计
Cross-Cutting Concerns Design

Data Design

- **Database Schema:** A normalized schema to reduce data redundancy.
- **Key Entities:** Member, Book, BookCopy, Loan, Hold, Fine.
- **Example:** A Book (title, author, ISBN) has one-or-more BookCopy (barcode, purchase_date). Each Loan is associated with a Member and a BookCopy.

Architecture Decision Records (ADRs) - Making and Documenting Critical Choices

"ADRs are documents that capture important architectural decisions made along with their context and consequences. They're the 'why' behind the 'what' in your architecture."

Think of ADRs as:

- ❑Architectural meeting minutes that don't fade from memory
- ❑A decision log that new team members can read to understand past choices
- ❑A way to avoid re-discussing the same decisions repeatedly

ADR #1: 2024-10-27 - Selection of Modular Monolith over Microservices

- **Status:** Accepted
- **Context:** The Library Management System has well-defined, cohesive functionality. The development team is small, and time-to-market is a priority. The initial user load is predictable and moderate.
- **Decision:** We will build a Modular Monolith architecture, separating concerns into clear modules (catalog, lending, members) within a single deployable unit.
- **Consequences:**
  - ❑ ☑ Positive: Simplified development, testing, and deployment. Easier database transactions and data consistency.
  - ❑ ☑ Positive: Lower operational complexity (no orchestration needed).
  - ❑ ⚠ Negative: Components are coupled and must be deployed together. A bug in one module can bring down the entire system.
  - ❑ ⚠ Negative: Limits the ability to scale individual components independently initially.

```
Two Paths We Could Take:


OPTION A: Microservices Architecture
[User] → [API Gateway] → [Catalog Service] → [Catalog DB]
                         [Lending Service] → [Lending DB]
                         [Member Service] → [Member DB]


OPTION B: Modular Monolith Architecture
[User] → [Single Application] → [Shared Database]
         ├── Catalog Module
         ├── Lending Module
         └── Member Module
```

Breaking Down the Context[解析背景]

- **Why This Decision Mattered:**
  - ❑ This is the most fundamental architectural choice
  - ❑ It affects everything: development, testing, deployment, and scaling
  - ❑ Hard to change once implemented

- **Our Specific Context (The Reality Check):**
  - ❑ "Well-defined, cohesive functionality"
  - ❑ Library systems have clear, related functions: catalog, lending, members
  - ❑ "Small development team“
  - ❑ Reality: We have 3-5 developers
  - ❑ Microservices require more operational overhead per service
  - ❑ "Time-to-market is a priority“ The library needs this system running before summer reading program, Modular monoliths are faster to develop initially
  - ❑ “Predictable and moderate load” We're not building Amazon or Netflix, 100 concurrent users is manageable for a well-designed monolith

Understanding the Decision
What is a Modular Monolith?

Key Point: We get separation of concerns WITHOUT distributed system complexity.

```
// This is how we structure our code internally
libra-library/
├── catalog-module/          // Handles books, search, inventory
│   ├── src/
│   ├── tests/
│   └── module-config.java
├── lending-module/          // Handles checkouts, returns, holds
│   ├── src/
│   ├── tests/
│   └── module-config.java
├── member-module/           // Handles members, accounts, authentication
│   ├── src/
│   ├── tests/
│   └── module-config.java
└── application/             // Glues everything together
```

ADR #2 - Server-Side Rendering vs SPA

- Two Approaches to Web Applications:

```
OPTION A: Single Page Application (SPA)
Browser → [HTML Shell] → [JavaScript Framework] → [REST API]
          Fast subsequent navigation
          Rich, app-like experience
          SEO challenges

OPTION B: Server-Side Rendering (SSR)
Browser → [Server] → [Fully Rendered HTML]
          Faster initial load
          Better SEO
          Full page reloads
```

ADR #2: 2024-10-27 - Use of Server-Side Rendering over SPA

- **Status:** Accepted
- **Context:** The application requires a high degree of accessibility and SEO for the public catalog. The team has stronger backend Java skills than modern JavaScript framework expertise.
- **Decision:** Use Spring MVC with Thymeleaf templates for server-side rendering instead of a Single Page Application (SPA) framework like React or Angular.
- **Consequences:**
  - ❑ ☑ Positive: Faster initial page loads, simpler architecture, built-in SEO.
  - ❑ ☑ Positive: Easier to build accessible HTML forms and leverage Spring Security's built-in CSRF protection.
  - ❑ ⚠ Negative: Less dynamic and interactive user experience compared to a SPA.
  - ❑ ⚠ Negative: More server load as each page request requires a full round-trip.

Breaking Down the Context

- **Why This Decision Mattered:**
  - ❑ Affects user experience, development workflow, and SEO
  - ❑ Different skill requirements for the team
  - ❑ Different performance characteristics

- **Our Specific Context:**
  - ❑ "High degree of accessibility and SEO“ Public catalog must be searchable by Google
  - ❑ Library serves diverse users including those with disabilities
  - ❑ "Team has stronger backend Java skills“
  - ❑ Reality:
    - ❑ Our team knows Spring MVC well
    - ❑ Learning React/Angular would slow us down significantly
    - ❑ We can deliver faster with technologies we know
  - ❑ Library application characteristics
    - ❑ Mostly form-based interactions (search, checkouts, account management)
    - ❑ Not highly interactive like a real-time dashboard or game

Glossary [词汇表]

- This glossary ensures all team members and stakeholders have a common understanding of the terms used throughout the architecture documentation and project discussions.

- **A. Technical Terms**
- **Architecture & Design**
  - ADR (Architecture Decision Record): A document that captures an important architectural decision along with the context and consequences.
  - C4 Model: A hierarchical set of software architecture diagrams (Context, Containers, Components, Code).
  - Container: A separately runnable/deployable unit (e.g., web application, database, mobile app).
  - Component: A modular, replaceable part of a system with well-defined interfaces.
  - Monolith: A single deployable unit containing all application functionality.
  - Microservices: An architectural style where an application is composed of loosely coupled, independently deployable services.
  - ORM (Object-Relational Mapping): A technique that lets you query and manipulate data from a database using an object-oriented paradigm.

A. Technical Terms (contd…)

- **Web Development**
  - ❑SPA (Single Page Application): A web application that loads a single HTML page and dynamically updates content without page reloads.
  - ❑SSR (Server-Side Rendering): Rendering web pages on the server before sending them to the client.
  - ❑API (Application Programming Interface): A set of rules that allows programs to talk to each other.
  - ❑REST (Representational State Transfer): An architectural style for designing networked applications.
  - ❑CRUD (Create, Read, Update, Delete): The four basic operations of persistent storage.

A. Technical Terms (contd…)

- **Security**
  - ❑RBAC (Role-Based Access Control): A security approach that restricts system access to users based on their roles.
  - ❑Authentication: The process of verifying who a user is.
  - ❑Authorization: The process of verifying what a user has access to.
  - ❑CSRF (Cross-Site Request Forgery): An attack that tricks a user into executing unwanted actions.
  - ❑BCrypt: A password hashing function designed to be slow and computationally expensive to prevent brute-force attacks.

A. Technical Terms (contd…)

- **Quality Attributes**
  - ❑Scalability: The ability of a system to handle increased load.
  - ❑Performance: How quickly the system responds to user actions.
  - ❑Availability: The proportion of time a system is in a functioning condition.
  - ❑Maintainability: The ease with which a system can be modified.
  - ❑Reliability: The ability of a system to perform required functions under stated conditions.

B. Library Domain Terms

- **Core Entities**
  - ❑Member/Patron: A registered user of the library system who can borrow materials.
  - ❑Librarian/Staff: Library employees with administrative privileges in the system.
  - ❑Book: A title in the library's collection (e.g., "The Great Gatsby").
  - ❑Book Copy: A physical instance of a book title that can be borrowed.
  - ❑ISBN (International Standard Book Number): A unique numeric commercial book identifier.
  - ❑Barcode: A unique identifier for a physical book copy.
- **Library Operations**
  - ❑Loan/Circulation: The act of borrowing a book copy from the library.
  - ❑Check-out: The process of issuing a book to a member.
  - ❑Check-in: The process of returning a book to the library.
  - ❑Due Date: The date by which a borrowed item must be returned.
  - ❑Renewal: Extending the loan period for a borrowed item.
  - ❑Hold/Reservation: A request to be next in line for a currently checked-out item.

B. Library Domain Terms (contd..)

- **Fines & Fees**
  - ❑Fine: A monetary penalty for returning items after the due date.
  - ❑Overdue: The status of an item that has not been returned by its due date.
  - ❑Waiver: The act of canceling a fine, typically by library staff.
  - ❑Payment Processing: Handling financial transactions for fines.
- **Inventory Management**
  - ❑Catalog: The complete collection of books and materials in the library.
  - ❑Inventory: The process of tracking and managing physical book copies.
  - ❑Acquisition: The process of adding new materials to the library collection.
  - ❑Weeding: The process of removing old or unused materials from the collection.

C. Business & Project Terms

- **Project Management**
  - ❑Stakeholder: Any person or organization affected by the system.
  - ❑KPI (Key Performance Indicator): A measurable value that demonstrates how effectively objectives are being achieved.
  - ❑ROI (Return on Investment): A measure of the profitability of an investment.
  - ❑Constraint: A limitation or restriction on the project (time, budget, technology).

- **Quality Metrics**
  - ❑Response Time: The time taken for the system to respond to a user action.
  - ❑Throughput: The number of transactions processed in a given time period.
  - ❑Uptime: The time when the system is operational and available.
  - ❑Concurrent Users: The number of users actively using the system simultaneously.

D. Technology Stack Terms

- **Database**
  - ❑ PostgreSQL: A powerful, open-source object-relational database system.
  - ❑ ACID (Atomicity, Consistency, Isolation, Durability): A set of properties that guarantee database transactions are processed reliably.
  - ❑ JDBC (Java Database Connectivity): A Java API for connecting and executing queries on databases.
  - ❑ Connection Pooling: A cache of database connections maintained for reuse.
- **Testing**
  - ❑ JUnit: A unit testing framework for the Java programming language.
  - ❑ Mockito: A mocking framework for unit tests in Java.
  - ❑ TestContainers: A Java library that supports JUnit tests, providing lightweight, throwaway instances of common databases.
- **Deployment & Infrastructure**
  - ❑ Docker: A platform for developing, shipping, and running applications in containers.
  - ❑ AWS EC2 (Elastic Compute Cloud): A web service that provides resizable compute capacity in the cloud.
  - ❑ CI/CD (Continuous Integration/Continuous Deployment): A method to frequently deliver apps to customers by introducing automation.
  - ❑ Virtual Machine: An emulation of a computer system.

E. System-Specific Terms

- **LMS Components**
  - ❑Catalog Module: Handles book search, browsing, and inventory management.
  - ❑Lending Module: Manages check-outs, check-ins, renewals, and holds.
  - ❑Member Module: Handles user accounts, authentication, and personal information.
  - ❑Fines Module: Manages overdue fines, payments, and waivers.
  - ❑Notification Service: Handles email alerts and reminders.

- **System States**
  - ❑Available: A book copy that can be checked out.
  - ❑Checked Out: A book copy that is currently borrowed.
  - ❑On Hold: A book copy that is reserved for the next member in queue.
  - ❑Maintenance: A book copy that is temporarily unavailable due to repairs.
  - ❑Active Loan: A currently borrowed item.
  - ❑Overdue Loan: A borrowed item that has passed its due date.

F. Acronyms Quick Reference

Acronym	Full Form	Description
ADR	Architecture Decision Record	Documents why an architectural decision was made
API	Application Programming Interface	Allows software components to communicate
AWS	Amazon Web Services	Cloud computing platform
BCrypt	Blowfish Crypt	Password hashing function
CI/CD	Continuous Integration/Continuous Deployment	Automated software delivery process
CRUD	Create, Read, Update, Delete	Basic data operations
CSRF	Cross-Site Request Forgery	Web security vulnerability
EC2	Elastic Compute Cloud	AWS virtual servers
HTML	HyperText Markup Language	Standard markup for web pages
ISBN	International Standard Book Number	Unique book identifier
JDBC	Java Database Connectivity	Java database connectivity API
JPA	Java Persistence API	Java specification for object-relational mapping
JVM	Java Virtual Machine	Runtime environment for Java applications
KPI	Key Performance Indicator	Measurable value showing effectiveness
LMS	Library Management System	Software for managing library operations
MVC	Model-View-Controller	Software design pattern
ORM	Object-Relational Mapping	Programming technique for data conversion
RBAC	Role-Based Access Control	Security based on user roles
REST	Representational State Transfer	Architectural style for web services
ROI	Return on Investment	Profitability measure
SMTP	Simple Mail Transfer Protocol	Email transmission protocol
SPA	Single Page Application	Web app that loads a single page
SQL	Structured Query Language	Language for managing databases
SSR	Server-Side Rendering	Rendering web pages on the server
UI	User Interface	Space where interactions occur
UX	User Experience	Overall experience of using a product

Key Takeaways for Students [学生要点]

About ADRs:

- ❑ADRs make thinking explicit - They force you to express your reasoning
- ❑Context is king - The same decision might be wrong in a different context
- ❑Trade-offs are inevitable - Every architecture choice has pros and cons
- ❑ADRs are living documents - They can be superseded when context changes

About Specific Decisions:

- ❑Start simple - Modular monoliths are often the right starting point
- ❑Productivity matters - Choose technologies that match your team's skills
- ❑Consider your users' needs - SEO and accessibility are real requirements
- ❑Plan for evolution - Good architectures can grow with your needs