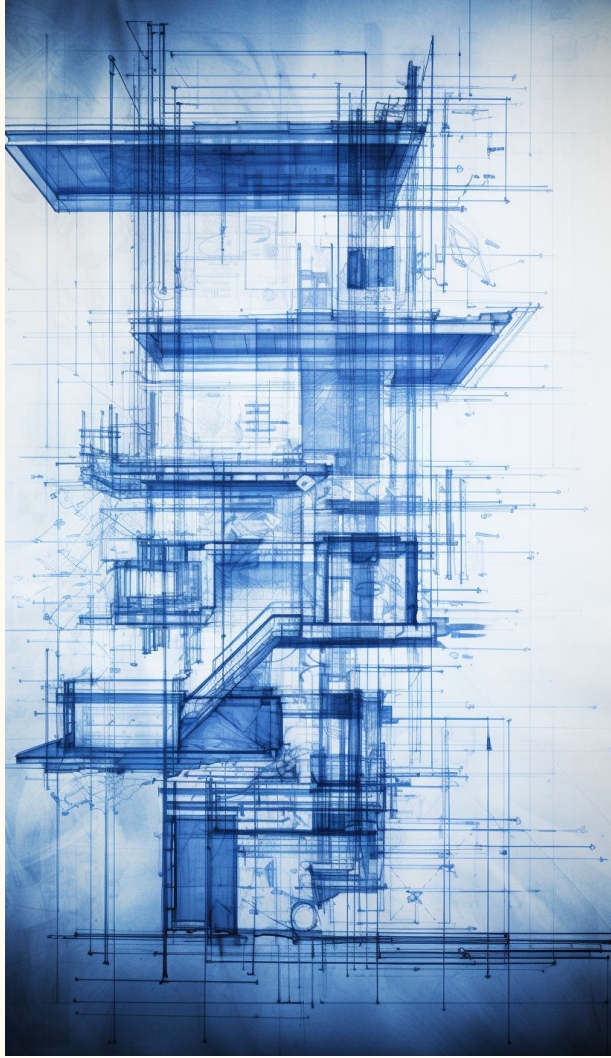
Lesson 3 - System Design

Navigating the World of System Design

High-Level and Detailed System Design

- High-level design outlines system components and interactions.
- Detailed design dives into specifics of each component, including algorithms and data structures.



Architecture Patterns

- Different architecture patterns (monolithic, microservices, client-server, etc.).
- Each pattern has its benefits and trade-offs.

Architecture Pattern: Client-Server

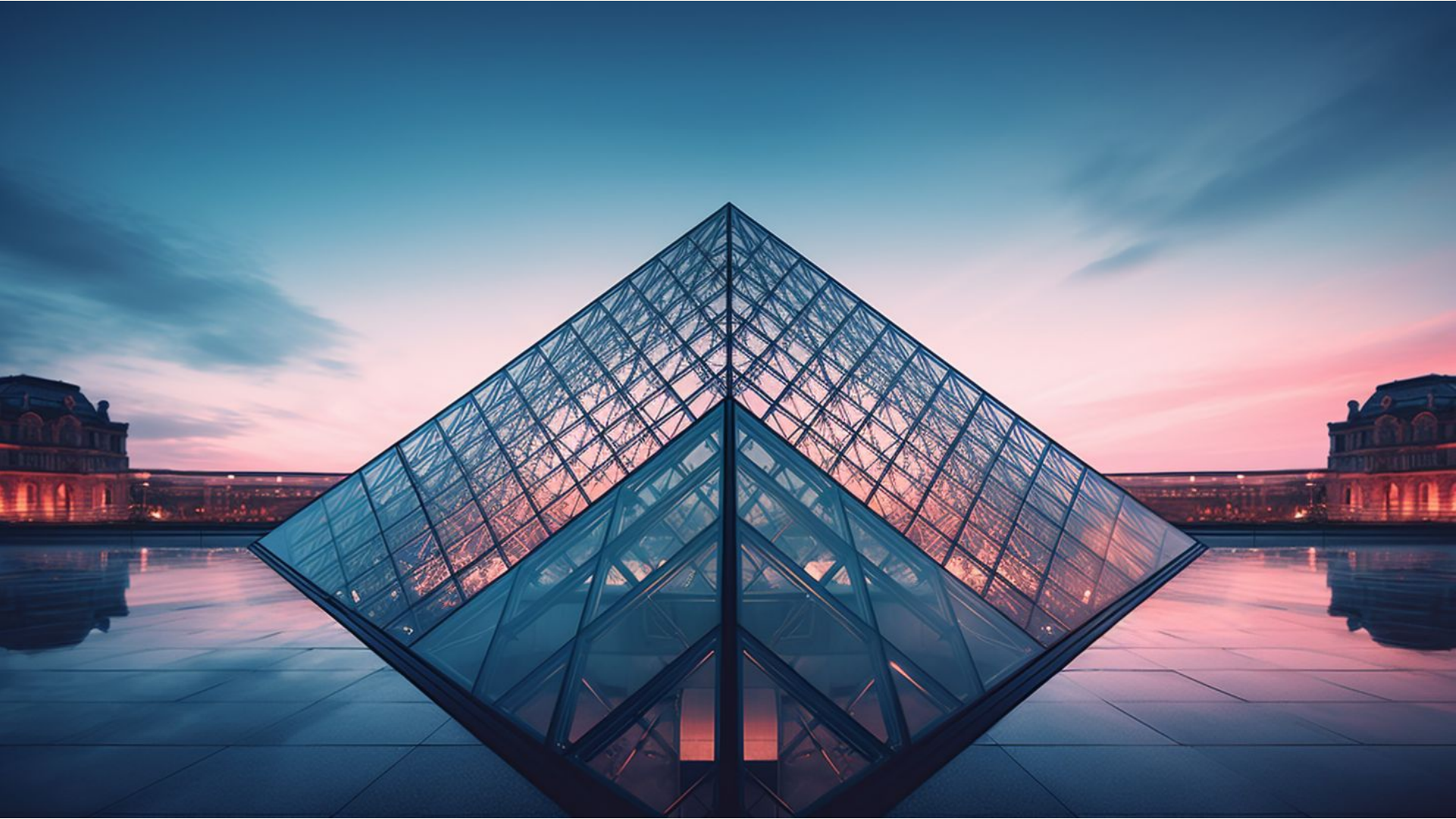
- A way to structure computer systems where two types of devices, clients and servers, interact.
- **Clients**: Devices used by users to access services or resources.
- **Servers**: Powerful computers that provide services, resources, or data to clients.
- **Communication**: Clients and servers communicate over a network (e.g., the internet).
- **Advantages**: Scalability, centralized data management, efficient resource utilization.

Architecture Pattern: Monolithic

- A software structure where all components of an application are tightly integrated into a single codebase.
- **Single Unit**: The entire application, including user interface, logic, and database access, is combined.
- **Advantages**: Simple to develop and deploy, easy to maintain for small projects.
- **Challenges**: Can become complex and hard to manage as the project grows.
- **Scaling**: Requires scaling the entire application, which might be inefficient for specific components.
- **Changes**: Small changes can impact the entire application, testing and deployment can be challenging.

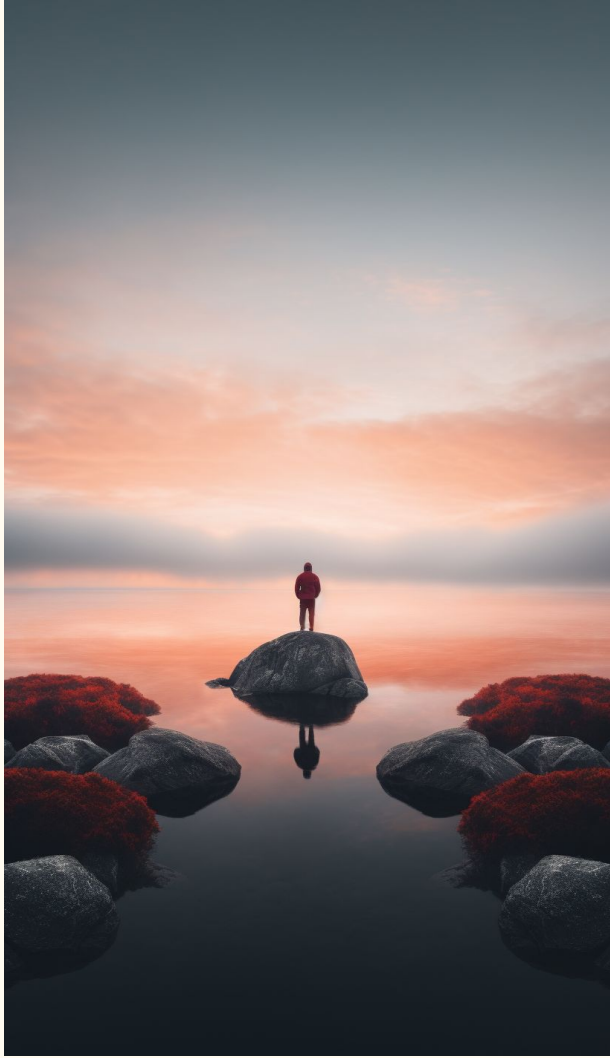
Architecture Pattern: Microservices

- A way to design software as a collection of small, independent services that communicate.
- **Services**: Each service handles a specific function or task within the application.
- **Decoupled**: Services are separate and can be developed, deployed, and scaled independently.
- **Advantages**: Scalability, flexibility, easier maintenance, technology diversity.
- **Challenges**: Complexity in managing distributed systems, communication overhead.
- **Communication**: Services communicate through APIs (Application Programming Interfaces).
- **Consideration**: Microservices suit complex applications with different components that need to scale and evolve independently.



Design Principles

- Design principles like SOLID (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion).
- Principles aid in creating maintainable and extensible systems.



User Interface (UI) and User Experience (UX)

- UI design focuses on visual elements and layout.
- UX design considers overall user experience, including usability and flow.

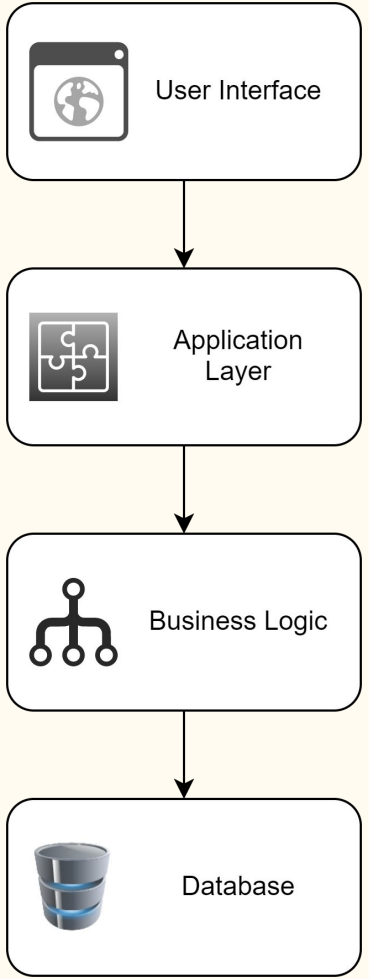
Data Modeling and Database Design

- Data modeling defines the structure and relationships of data.
- Database design involves creating tables, indexes, and optimizing queries.

Case Study: E-Commerce Platform (1)

Architecture Design

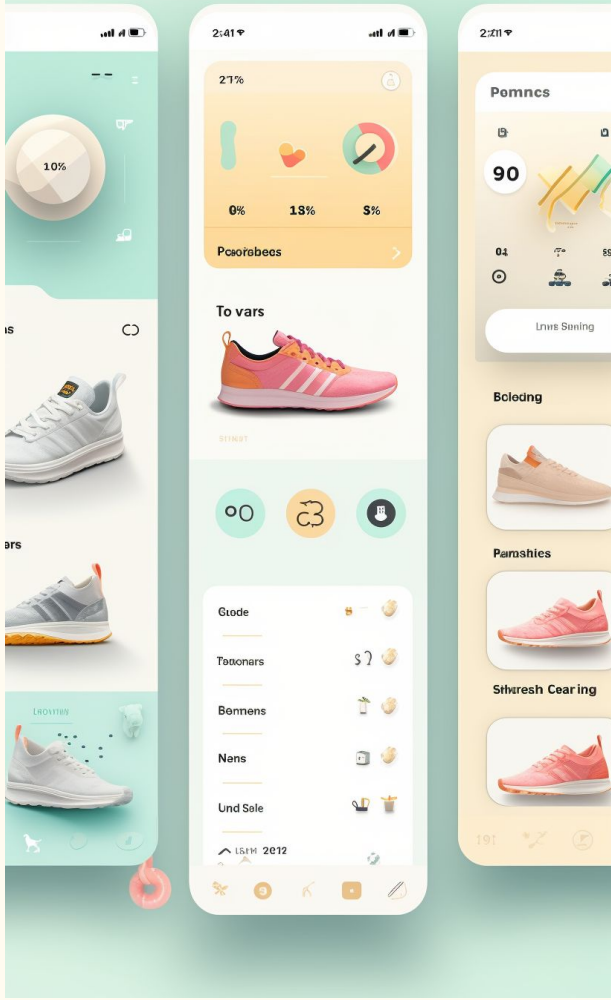
- The E-Commerce Platform follows a traditional client-server architecture.
- The User Interface interacts with users and displays product listings, categories, and user accounts.
- The Application Layer handles user requests, authentication, and communication with the Business Logic layer.
- The Business Logic layer manages core e-commerce functionality such as shopping cart, checkout, order processing, and recommendation algorithms.
- The Database stores user data, product information, and order details.



Case Study: E-Commerce Platform (2)

UI/UX Design

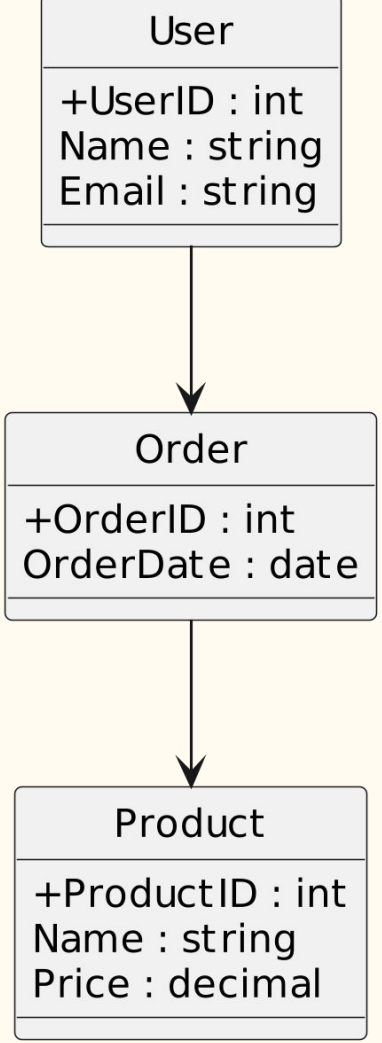
- The UI features a responsive design accessible through web browsers and mobile devices.
- The homepage displays featured products and categories, enabling users to quickly find products of interest.
- Users can browse products, view detailed information, and read reviews.
- The shopping cart allows users to add and remove items, with a visible summary of the selected items.
- The checkout process includes user authentication, address selection, and secure payment methods.
- User accounts offer personalized order history, tracking, and recommendations based on purchase history.



Case Study: E-Commerce Platform (3)

Data Modeling

- The User entity stores user information, including a unique UserID, user's Name, and Email.
- The Product entity holds product data, including a unique ProductID, product Name, and Price.
- The Order entity represents an order placed by a user. It includes a unique OrderID and an OrderDate.



Iterative Design Process

- Design is an iterative process with constant feedback and refinement.
- Continuous improvement ensures alignment with user needs and project goals.

Conclusion

- System design transforms concepts into functional solutions.
- Combining architectural understanding, design principles, and user-centricity.

Question?