1. **3-Tier System Architecture**

A **3-tier architecture** separates applications into independent layers for better organization and scalability:

* **Presentation Tier (UI):**
  + Handles user interactions.
  + Technologies: HTML/CSS (web), WPF or WinForms (desktop), or Xamarin/MAUI (mobile).
  + Sends requests to the Application Tier.
* **Application Tier (Logic):**
  + Processes requests from the UI and applies business rules.
  + Communicates with the Data Tier for database operations.
  + Often deployed on application servers like IIS or cloud platforms.
* **Data Tier:**
  + Stores and retrieves data using a database system (e.g., SQL Server, MongoDB).
  + Ensures data integrity, scalability, and backup.

**Benefits:**

* **Modularity:** Easier to manage and modify individual layers.
* **Scalability:** Layers can be scaled independently (e.g., multiple application servers).
* **Security:** Sensitive data is processed and stored in separate layers.

**2. Application Server (Tier) Layered Architecture**

The application layer can be divided into smaller, specialized layers:

* **Presentation Layer:**
  + Responsible for rendering data for users.
  + Example: Return a JSON response for RESTful APIs or an HTML view for web applications.
  + Technologies: Razor Views (MVC), Blazor, or React (front-end integrated).
* **Business Logic Layer (BLL):**
  + Implements core business rules and workflows.
  + Example: A rule that ensures discounts are only applied to eligible products.
  + Typically includes services or managers (OrderService, UserManager).
* **Data Access Layer (DAL):**
  + Contains methods to perform CRUD operations on the database.
  + Example: Using Entity Framework to define DbContext for database interactions.
  + Abstracted using repositories for better maintainability.
* **Infrastructure Layer:**
  + Provides shared services like logging, caching, or email notifications.
  + Example: Use of Serilog for logging or Redis for caching.



**3. Data Transfer Objects (DTO)**

* **Purpose:**
  + Prevent exposing internal models (e.g., database entities) directly to clients.
  + Reduce over-fetching of data by only including required fields.

**Example:** Entity (Database Model):



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

