A. Core Architecture & Best Practices

1. **MVVM Implementation:**
    - ***Question:*** **In a large WPF application, what are the key principles and benefits of the MVVM pattern? Can you explain pitfalls you've seen in real projects and how to avoid them?**
2. **Data Binding Performance:**
    - ***Question:*** **If a large number of bindings are causing slow UI responsiveness, what are the possible bottlenecks and remedies?**
3. **Dependency Injection:**
    - ***Question:*** **How would you integrate DI into a WPF application for better maintainability? What challenges have you faced with lifetime management in WPF?**
4. **Resource Dictionaries:**
    - ***Question:*** **How do you structure Resource Dictionaries for a large WPF application to avoid bloat and duplication while ensuring styles and templates are easily maintainable?**

B. Performance Optimization

5. **Virtualization:**
    - ***Question:*** **How does UI virtualization work in WPF, and what are common mistakes developers make that break virtualization (e.g., in ListView or DataGrid)?**
6. **Freezable Objects:**
    - ***Question:*** **What is a Freezable in WPF? How does freezing them improve performance, and when should you use them?**

C. Advanced Binding & Tricks

9. **Binding Priority & Precedence:**
    - ***Question:*** **In WPF, multiple sources (local value, style, binding, etc.) can set a property. Can you explain dependency property value**

precedence and a tricky case where this caused unexpected behavior?

Dependency Property Value Precedence in WPF

WPF determines a property's final value using a **well-defined precedence order** — from highest to lowest priority:

10. **Local value**
    - Set directly in XAML (`<Button Width="100"/>`) or in code (`btn.Width = 100;`).
11. **Animations with a HoldEnd behavior** (high priority while active).
12. **TemplatedParent template values** (values coming from a control template).
13. **Style triggers** (like `DataTrigger`, `EventTrigger` inside styles/templates).
14. **Template triggers** (inside ControlTemplates).
15. **Style setters** (values set in a style, without triggers).
16. **Theme styles** (from OS theme or WPF default theme resources).
17. **Inherited property values** (e.g., `FontSize` inherited from parent).
18. **Default value** from the dependency property metadata.

Example of Unexpected Behavior

Scenario:

Suppose you bind a property in XAML:

```xml
<TextBox Text="{Binding UserName}" Width="Auto" />
```

Later, a style is applied with a `Setter`:

```xml
<Style TargetType="TextBox">
    <Setter Property="Width" Value="200" />
</Style>
```

✅ **Expected by junior dev:** The bound `Width="Auto"` wins because it was set in XAML.

❌ **Reality:** The **local value** (`Width="Auto"`) *does* win — but if you programmatically set `Width` to something else (local value in code-behind), the **binding is replaced** because a local value has *higher precedence* than style setters.

19. IMultiValueConverter Usage:

- ***Question:*** **When would you use an IMultiValueConverter? Can you give an example where chaining converters or using PriorityBinding was more effective?**

20. Binding to Non-Dependency Objects:

- ***Question:*** **How can you bind to a property in a non-dependency object and still receive UI updates?**

21. DataTemplateSelector:

- ***Question:*** **Can you explain a real-world scenario where a DataTemplateSelector drastically simplified your UI logic?**

D. Styling, Theming & Reusability

13. Dynamic vs. Static Resources:

- ***Question:*** **Explain the difference between StaticResource and DynamicResource. How do you decide which to use for performance and flexibility?**

14. ControlTemplates vs. DataTemplates:

- ***Question:*** **How do you decide when to use a ControlTemplate vs. a DataTemplate?**

15. Theming:

- ***Question:*** **How would you implement a runtime theme switcher in WPF for a large application without introducing performance or flicker issues?**

16. Attached Behaviors:

- ***Question:*** **Can you describe a situation where using an attached behavior was a better solution than using code-behind or extending a control?**

E. Debugging, Maintenance & Edge Cases

17. Binding Debugging:

- ***Question:*** **If a binding fails silently, how would you troubleshoot it? What tools and techniques do you use?**

18. Memory Leaks in WPF:

- ***Question:*** **What are common causes of memory leaks in WPF applications, and how would you detect and fix them?**

19. **Dispatcher Priority:**
    - ***Question:*** **Can you explain how DispatcherPriority works, and give an example where adjusting it solved a UI lag issue?**
20. **Interoperability:**
    - ***Question:*** **Have you integrated WinForms or DirectX inside WPF? What are the pitfalls, especially in terms of performance and threading?**

1. MVVM Implementation & Best Practices *Explain the MVVM pattern in WPF and describe how you would implement INotifyPropertyChanged efficiently. What are the potential memory leak issues with event handlers, and how do you prevent them?*

Expected Answer Points:

- Understanding of Model-View-ViewModel separation
- WeakEventPattern or proper event unsubscription
- Use of CallerMemberName attribute
- Base class implementations or Fody.PropertyChanged

2. Advanced Data Binding Scenarios *Explain MultiValueConverter and a scenario where it can be used*.

Expected Answer Points:

- PriorityBinding, MultiBinding with converters
- Null propagation in binding paths
- FallbackValue and TargetNullValue
- Custom converter implementations
- Binding.DoNothing for conditional binding

3. Dependency Injection in WPF – *Tell me about approaches to make your WPF app more testable.*

Expected Answer Points:

- Container integration (Autofac, Unity, built-in DI)
- ViewModel locator patterns
- View-first vs ViewModel-first approaches
- Handling design-time data

4. Command Pattern Implementation *Describe different ways to implement the Command pattern in WPF. When would you use DelegateCommand vs RelayCommand vs custom implementations? How do you handle async operations?*

Expected Answer Points:

- ICommand interface implementation
- Async command handling patterns
- Command parameter passing
- CanExecute logic and performance considerations

Performance & Memory Management (Questions 5-8)

5. WPF Performance Optimization *A WPF application with large datasets is experiencing slow scrolling and high memory usage. Walk me through your systematic approach to diagnose and optimize performance.*

Expected Answer Points:

- Virtualization (VirtualizingStackPanel, VirtualizingPanel)
- Data template optimization
- Profiling tools (PerfView, Visual Studio Diagnostic Tools)
- Lazy loading and data paging
- UI virtualization vs data virtualization

6. Memory Leaks in WPF *What are the most common sources of memory leaks in WPF applications? Provide specific examples and mitigation strategies.*

Expected Answer Points:

- Event handler subscriptions
- Static event handlers
- Dispatcher timers
- Binding to static properties

- Routed events and tunneling/bubbling
- WeakReference patterns

7. Rendering Performance *Explain the WPF rendering pipeline and how you would optimize rendering performance for complex visual scenarios with many UI elements.*

Expected Answer Points:

- Measure, Arrange, Render phases
- Visual tree vs logical tree optimization
- Hardware acceleration and RenderOptions
- Freezable objects and their benefits
- Custom drawing with OnRender vs DrawingVisual

Custom Controls & Advanced UI (Questions 9-12)

9. Custom Control vs UserControl *When would you create a custom control versus a user control? Walk me through the process of creating a custom control with custom dependency properties and explain the control template approach.*

Expected Answer Points:

- Composition vs inheritance decision factors
- Dependency property implementation
- Control template and theme integration
- Visual state management
- Lookless control concepts

10. Advanced Styling and Templating *Describe how you would create a reusable, themeable button control that supports different visual states and can be easily customized. Include discussion of triggers, animations, and resource organization.*

Expected Answer Points:

- Style inheritance and BasedOn
- Template triggers and property triggers
- Storyboard animations
- Resource dictionary organization

- Theme switching mechanisms
-

Data Management & Binding (Questions 13-16)

13. Complex Data Validation *Implement a robust validation system that supports both synchronous and asynchronous validation, displays errors at both field and entity level, and integrates well with MVVM.*

Expected Answer Points:

- IDataErrorInfo vs INotifyDataErrorInfo
- ValidationRule implementations
- Async validation patterns
- Cross-field validation
- Validation error templates and styling

14. Data Templating Strategies *You have a heterogeneous collection of different object types that need different visual representations. How would you implement this efficiently?*

Expected Answer Points:

- DataTemplateSelector usage
- Resource key-based template selection
- Implicit data templates
- Performance implications
- Dynamic template switching

15. Collection Synchronization *How do you handle updating UI collections from background threads safely? Discuss different approaches and their trade-offs.*

Expected Answer Points:

- Dispatcher.Invoke vs BeginInvoke
- BindingOperations.EnableCollectionSynchronization
- ObservableCollection thread safety
- Producer-consumer patterns
- Performance impact of cross-thread operations

16. Data Transformation and Aggregation *Implement a solution for displaying aggregated data (sums, averages, grouping) from a large dataset that updates in real-time without blocking the UI.*

Expected Answer Points:

- CollectionViewSource and grouping
- Background data processing
- Incremental collection updates
- LINQ performance considerations
- Reactive extensions (Rx.NET) usage

Architecture & Best Practices (Questions 17-20)

17. Application Architecture *Design the architecture for a large WPF enterprise application with multiple modules, shared components, and plugin capabilities. Discuss your choices and reasoning.*

Expected Answer Points:

- Modular architecture patterns (Prism, MEF)
- Shared library strategies
- Plugin loading and isolation
- Configuration management
- Inter-module communication

18. Testing Strategies *How do you approach testing WPF applications? Include unit testing, integration testing, and UI automation testing strategies.*

Expected Answer Points:

- MVVM testability benefits
- UI automation frameworks (Coded UI, White, FlaUI)
- Mocking dependency services
- Design-time data strategies
- Continuous integration considerations

19. Deployment and Distribution *Discuss modern deployment strategies for WPF applications, including ClickOnce alternatives, package management, and update mechanisms.*

Expected Answer Points:

- MSIX packaging and deployment
- ClickOnce limitations and alternatives
- Auto-update implementations
- Prerequisites and runtime dependencies
- Microsoft Store distribution

20. Migration and Modernization *You're tasked with modernizing a legacy WinForms application to WPF. Outline your migration strategy, including gradual migration approaches and maintaining business continuity.*

Expected Answer Points:

- Incremental migration strategies
- WinForms-WPF interop (ElementHost, WindowsFormsHost)
- Business logic extraction
- Data access layer modernization
- User training and change management