UiPath The Foundation of Innovation™

Solution Design & Work Breakdown Principles:

Solution Design Principles:

Follow principles such as avoiding unnecessary complexity, ensuring maintainability, readability, flexibility, reliability, extensibility, scalability, modularity, and building an efficient design.

- **Avoid Unnecessary Complexity:**
  - Keep the technical solution simple and easy to understand. Complexity is measured by how difficult it is to analyze, explain, and develop. Aim for a solution that meets the requirements without being overly complicated. Find a balance between tightly connected and independent code.
- **Maintainability:**
  - Ensure that the design of the code follows good standards and has a clear structure. Too many configuration files or dependencies can make it hard to maintain.
- **Lead Readability Through Design:**
  - Standardize the process structure, encouraging clear development practices. Use meaningful names for workflow files, activities, arguments, and variables. Consider using a well-known framework.
- **Flexibility:**
  - Keep environment settings in external configuration files or Orchestrator to make it easy to run automation in different environments. Centralize changes to parts like the user interface, environment settings, and output naming conventions for consistent updates throughout the code.
- **Reliability:**
  - Design the process to handle exceptions, report errors, and consolidate reporting. Ensure that re-attempts are easy and performed on the latest data, avoiding unnecessary reruns by the controllers.
- **Design and Code Should be Extensible:**
  - Make the design and code ready for new use cases, changes in technology or approach, and the addition of functionalities.
- **Scalability:**
  - Plan for automations to run in multiple bots, reducing time consumption and speeding up the process output with a multi-bot architecture.
- **Ask Targeted and Specific Questions:**
  - When designing, ask specific questions about the influx of data and transactions. Design the bot architecture to balance the load through queuing and functionality distribution between automation bots to meet SLAs and deadlines for time-sensitive automations.
- **Modularity:**
  - Separate functionality with dedicated workflows for fine-grained development and testing. Extract and share reusable components and workflows between projects, avoiding too many functionalities or activities in a single workflow or module.
- **Build an Efficient Design:**
  - Measure process capability based on how well the process output complies with given specifications. Consider capability indices, reflecting the set design parameters or customer demands.

Work Breakdown Principles:

- **Don't Reinvent the Wheel:** use what's already available. Try to use existing components, activities, and integrations from UiPath rather than creating something new. This saves time and ensures reliability.
- **Exhibit Uniformity:** keep things consistent. Follow design conventions and use a similar approach to achieve functionalities throughout your project. Consistency makes it easier for everyone to understand and work together.
- **Exceptions Allowed:** There can be exceptions. For example, use an Object repository for some UI Automation modules and only selectors for others. However, remember this is a guideline, not a strict rule.
- **Avoid Repetition (DRY Principle):** don't repeat yourself. If there's overlapping functionality in your code, try to avoid duplicating it. Instead, make it modular so it can be reused elsewhere in your project.
- **Cohesive Modules:** Each module should do one thing well. Focus on implementing one functionality per XAML/module. This makes it easier to manage and less likely to need frequent changes.
- **Identify Reusable Components:** look for common functionalities that can be used in different processes or projects. Reusable components make development easier, reduce complexity, and lead to shorter implementation times.
- **Efficiency Through Reusable Components:** reusable components reduce development efforts, simplify use cases, provide better change management, and speed up implementation timelines.
- **Design Testable Modules:** make sure your modules are easy to test independently. This helps with debugging and ensures the reliability of your code.
- **Avoid Unnecessary Features:** focus on implementing necessary features. Don't add features that are not required. This keeps your project focused and efficient.
- **Write Readable Wrappers:** Use readable wrappers. When developing with the help of work breakdown, create parent modules that bring together smaller functional modules. This improves readability and maintainability.
- **Leverage Existing Components:** Use components or libraries that already exist, such as those in Orchestrator feed or common project repositories. Mention them in your work breakdown during whiteboarding. Note the versions to be used for developers to leverage the same.