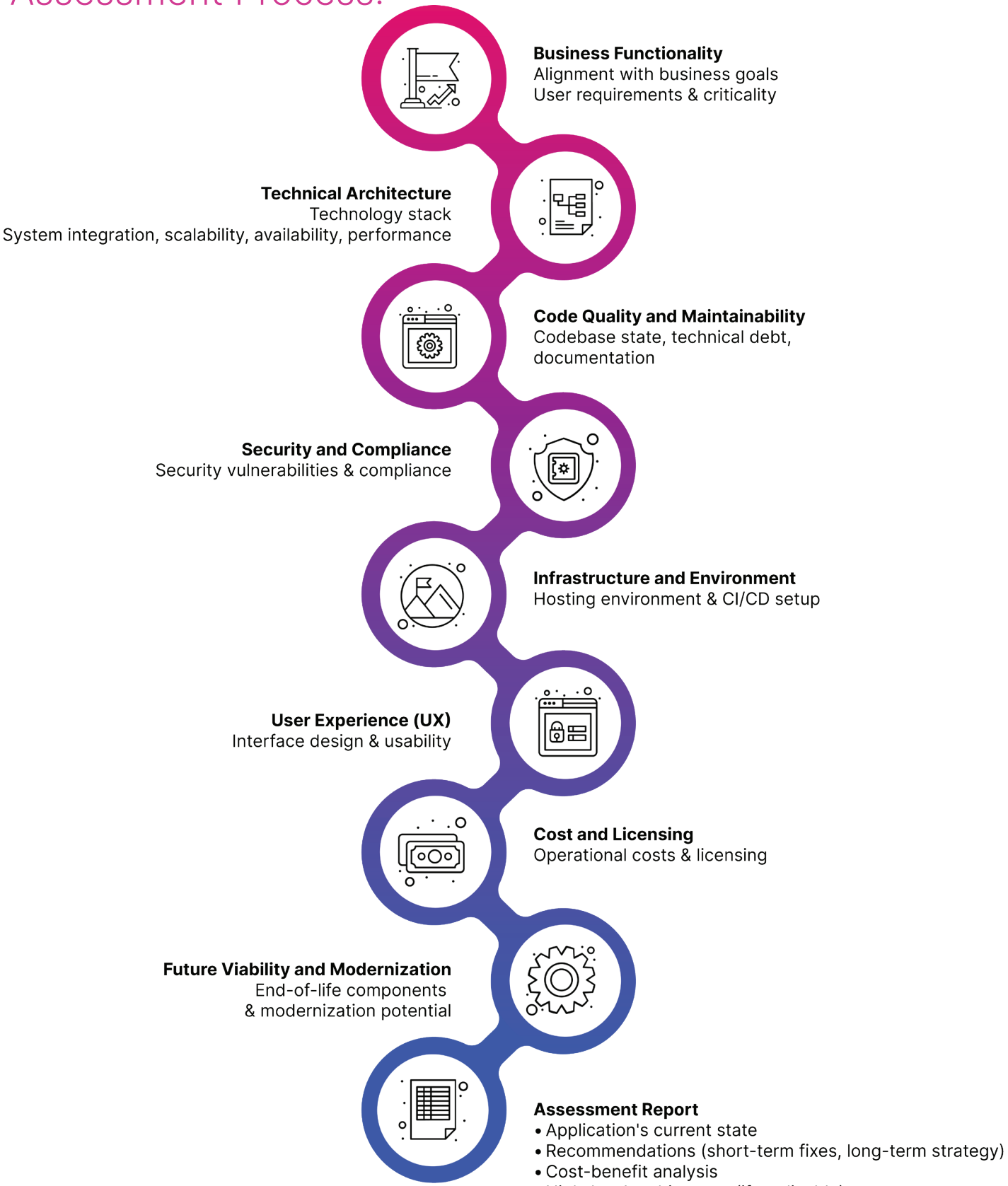
Legacy Application Assessment

Assessing a legacy application involves evaluating various aspects to determine its current state, pain points, and issues in the area of performance, security, scalability, maintainability, and future viability.

Assessment is executed via interviews and surveys with appointed Stakeholders to understand their perspective on the application's performance, issues, and future needs, known pain points and desired improvements.

Assessment Process:



Aspects included and considered in the assessment

1. Business Functionality

- **Alignment with current Business Goals**: Determine if the application still supports current business processes and objectives.
- **User Requirements**: Assess whether the application meets the needs of its users and if there are any gaps in functionality.
- **Criticality**: Evaluate how critical the application is to daily operations and its impact on the business if it fails.

2. Technical Architecture

- **Technology Stack**: Identify the technologies used (programming languages, frameworks, databases, tools, etc.) and assess their relevance and supportability.
- **System Integration**: Determine how well the application integrates with other systems and whether these integrations are stable and secure.
- **Scalability**: Assess the ability of the application to scale to meet future demands.
- **Availability:** Assess if application meets current operational demands and application's uptime, reliability, and consistent accessibility to users when needed
- **Performance**: Collect currently available data and info about application's performance, including response times, throughput, and resource usage.

3. Code Quality and Maintainability

- **Codebase State**: Collect feedback on the current perceived level of codebase quality, including availability and adherence to coding standards, modularity, and readability.
- **Technical Debt**: Identify areas of the code that may have accumulated technical debt, making the system harder to maintain and extend.
- **Documentation**: Assess the availability of documentation, including architecture diagrams, code comments, and user manuals.
- **Dependency Management**: Check the status of external libraries and dependencies, including whether they are up-to-date and still supported. Is this already checked and considered?

4. Security and Compliance

- **Security Vulnerabilities**: Collect info of any past security issues caused by exploited vulnerabilities.
- **Backup Compliance**: Assess if application's data is protected, recoverable, and adheres to organizational and regulatory requirements.
- **Compliance**: Collect feedback on application's compliance with relevant industry standards and regulations, such as GDPR, HIPAA, or PCI-DSS.

5. Infrastructure and Environment

- **Hosting Environment**: Analyze the current hosting environment, including servers, operating systems, and network infrastructure.
- **CI/CD Environment Setup**: Evaluate how environments (development, staging, production) are configured and managed.

6. User Experience (UX)

- **Interface Design**: Gather feedback on pain points and areas where the application may be failing to meet user expectations. Assess the user interface for usability, accessibility, and adherence to modern design standards.

7. Cost and Licensing

- **Operational Costs**: collect data for the current costs associated with running the application, including hardware, software, licensing, and support.

8. Future Viability and Modernization

- **End-of-Life Components**: Identify any components or technologies that are approaching or have reached end-of-life status, requiring replacement or upgrade.
- **Modernization Potential**: Evaluate the feasibility of modernizing the application, including re-platforming, refactoring, extending, rearchitecting.

Assessment Report

A final assessment report is generated, providing clear insights into:

- **Application's current state,**
- **Recommendations and Action Plan**
    - **Short-Term Fixes:** Identified quick wins or urgent improvements that can stabilize or improve the application in the short term.
    - **Long-Term Strategy:** Recommendations for the long-term management of the application, whether that involves re-platforming, refactoring, extending, or rearchitecting.
    - **Cost-Benefit Analysis on the TCO for licensing and infrastructure level:** A cost-benefit analysis for any proposed actions, helping stakeholders make informed decisions.
- **If applicable, a proposed high-level architecture of the modernized application**
- **Recommended tools addressing identified pain points:**
    - **Code Analysis Tools:** To Review the codebase for quality, including adherence to coding standards, modularity, and readability
    - **Performance Testing Tools:** To Measure the application's performance, including response times, throughput, and resource usage
    - **Security Scanning Tools:** To identify potential security vulnerabilities during development and runtime
- **A planned roadmap for the implementations of recommended improvements**

Comtrade
Three-Sixty

info@comtrade360.com
comtrade360.com