App Modernization Advanced Complexity Framework



App Modernization Advanced

Prepared for

Update [Customer Name] in Doc Properties

5-Oct-15

Version .1 Draft

Prepared by

**Jan Weisel (Bridge Partners)**

Add Author Position to Doc Properties

Add Author Email to Doc Properties

Contributors

**Add Contributors to Doc Properties**

Table of Contents

[Introduction 2](#_Toc414051578)

[1 Application Complexity Level/Indexes 2](#_Toc414051579)

[1.1 Objectives 2](#_Toc414051580)

[1.2 Level Definition 2](#_Toc414051581)

[2 Application Categories 4](#_Toc414051582)

[2.1 Objectives 4](#_Toc414051583)

[2.2 Code Syntax 4](#_Toc414051584)

[2.3 Code Complexity 5](#_Toc414051585)

[2.4 Security 6](#_Toc414051586)

[2.5 Server configuration 9](#_Toc414051587)

[2.6 Architecture 10](#_Toc414051588)

[2.7 Communication 14](#_Toc414051589)

[2.8 Monitoring &Logging 15](#_Toc414051590)

[2.9 Exception Management 16](#_Toc414051591)

[2.10 Data Access 18](#_Toc414051592)

[2.11 Non Functional Requirements 20](#_Toc414051593)

[2.12 Geolocation Requirements 22](#_Toc414051594)

[2.13 Compliance Requirements 23](#_Toc414051595)

[2.14 High Availability and Disaster Recovery 23](#_Toc414051596)

**[Remember to regenerate the TOC to capture any section changes, deletions, etc. Delete this note.]**

**\*\*\*\***

# Introduction

This guidance document is designed to help define and document the complexity framework taxonomy used for application categorization, including definition of technical Complexity Level/Indexes, application categories and mapping of Complexity Level/Indexes to specific application properties. The goal of this document is to establish overall foundation for the application diagnostics process and illustrate how a specific application will be matched with its target Complexity Level/Index.

1. Application Complexity Level/Indexes
   1. Objectives

The overall objectives of this section is to introduce the application Complexity Level/Index taxonomy that is later used for application migration estimates

* 1. Level Definition
     1. Very Simple

Application migration process can be considered as **very simple** if few changes need to be applied to migrate it to the cloud and the application architecture is straightforward enough to fit into the conventional multi-tier application design, with adjacent tiers (i.e. web, service and business) being combined into the single layer for simplicity purposes. SQL Server typically serves as backend database engine, the underlying relational database lacks complexity and data access management layer follows simple CRUD interaction pattern when communicating with the backend database. A **very-simple** application is further categorized by distinct characteristics listed further down in this document. Such an application is conducive to cloud migration with few major changes.

* + 1. Simple

Application migration process can be considered as **simple** if relatively few changes need to be applied to migrate it to the cloud and the application architecture is straightforward enough to fit into the conventional multi-tier application design. SQL Server typically serves as backend database engine, the underlying relational database lacks complexity and data access management layer follows simple CRUD interaction pattern when communicating with the backend database. A **simple** application is further categorized by distinct characteristics listed later in this document. Such an application is conducive to cloud migration with relatively few major changes.

* + 1. Medium

Application migration process can be considered of **medium** complexity if a rather substantial effort is involved in migrating it to the cloud. While the application architecture is still straightforward enough to fit into the conventional multi-tier application design, distinct tiers are clearly distinguished (i.e. web, service, business, etc.) and application logic is non-trivial enough to utilize multiple development technologies and frameworks. A **medium** application is further categorized by distinct characteristics listed later in this document. Such an application will require making significant changes to adopt it to the cloud stack.

* + 1. Complex

Application migration process can be considered **complex** if a major effort is involved in migrating it to the cloud. The application architecture may or may not fit into the conventional multi-tier application design, application logic may include complex asynchronous execution flows and the application is non-trivial enough to utilize multiple development technologies and frameworks. A **complex** application is further categorized by distinct characteristics listed later in this document. Such an application will require making significant changes to adopt it to the cloud stack.

* + 1. Very Complex

Application migration process can be considered **very complex** if a major, resource-intensive effort is involved in migrating it to the cloud. The application architecture may or may not fit into the conventional multi-tier application design, application logic may include complex asynchronous execution flows, include multiple integration points with external systems, third-party products and the application can be non-trivial enough to utilize wide spectrum of Microsoft and open-source development technologies and frameworks. A **very**-**complex** application is further categorized by distinct characteristics listed later in this document. Such an application will require a major engagement to adopt it to the cloud stack.

1. Application Categories
   1. Objectives

The overall objectives of this section is to introduce wide spectrum of application categories that can be used to map application architecture properties to migration Complexity Level/Index

* 1. Code Syntax

The following application properties can be distinguished to map an application to Complexity Level/Indexes based on Code Syntax categorization type:

Table 1: Code Syntax Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very Simple  Index: 1 | Source code components consist of:   * Basic language statements * Standard operators * Few local variables * Primitive control structures |
| Simple  Index: 2 | Source code components consist of:   * Basic language statements * Standard operators * Local and static variables * Basic control structures (conditional, iteration, jump statement) |
| Medium  Index: 3 | Source code components use:   * A variety of language statements * Standard operators * Local and static variables * Constants and read-only fields * Numerous control structures (complex conditional, iteration, jump statement), * Exception handling statements |
| Complex  Index: 5 | Source code components use:   * A variety of language statements * Interface and class definitions, * Iterator execution control structures, * Hierarchical standard operators * Local and static variables * Constants and read-only fields * Numerous control structures (complex conditional, iteration, jump statement), exception handling statements |
| Very Complex  Index: 10 | Source code components use:   * A variety of language statements * Asynchronous method modifiers * Interface and class definitions * Iterator execution control structures * Hierarchical standard operators * Local and static variables * Constants and read-only fields * Numerous control structures (complex conditional, iteration, jump statement) * Exception handling statements |

* 1. Code Complexity

The following application properties can be distinguished to map an application to Complexity Level/Indexes based on Code Complexity categorization type:

Table 2: Code Syntax Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very simple  Index: 1 | Source codebase consists of:   * Simple class definitions with basic methods * Few local variables * Primitive control structures * Few dependencies on other modules   Cyclomatic Complexity of the source code base does not exceed a limit of 2-3 |
| Simple  Index: 2 | Source codebase:   * Employs simple synchronous execution flow patterns * Includes references between various modules * Incorporates simple class hierarchy * Implements exception handling management routines.   Cyclomatic Complexity of the source code base does not exceed a limit of 3-4 |
| Medium  Index: 3 | Source codebase employs:   * Mostly synchronous execution flow patterns * Substantial number of references between various modules * Interface and class hierarchy * Exception handling management routines.   Cyclomatic Complexity of the source code base does not exceed a limit of 4-6 |
| Complex  Index: 5 | Source codebase employs:   * Complex execution flow patterns * Substantial number of references between various modules * Complex interface and class hierarchies * Polymorphic class hierarchies, * Deep exception handling stack unwinding routines.   Cyclomatic Complexity of the source code base does not exceed a limit of 10 |
| Very complex  Index: 10 | Source codebase employs:   * Asynchronous execution flow patterns * Recursive computational routines * Complex interface and class hierarchy * Polymorphic class hierarchies * Deep exception handling stack unwinding routines.   Code components implement:   * A wide spectrum of common design patterns * State machines * Graph-based algorithms, * Numeric computations * Statistical analysis routines * Complex procedures implementing industry-specific process etc. * Implementation of other non-trivial computer algorithms   Cyclomatic Complexity of the source code reaches/exceeds a limit of 15 |

* 1. Security

The following application properties can be distinguished to map an application to Complexity Level/Index based on Security categorization type:

Table 3: Security Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very Simple  Index: 1 | Application:   * Has no or few security-related requirements, * Is exposed to external users without authentication or authorization * Data is not sensitive to possible leaks, * No data security requirements are identified * No regulatory/Industry requirement such as HIPAA/EU Data protection act * There are no firewall restrictions on outgoing and incoming HTTP requests |
| Simple  Index: 2 | Application:   * Has trivial security-related requirements, * Uses standard HTTP Basic or form-based authentication, * Features simple role-based authorization capabilities uses standard SSL protocol for data security in transit * Application data at rest is secured using out-of-box SQL encryption features. * No regulatory/Industry requirement such as HIPAA/EU Data protection act * There are no firewall restrictions on outgoing and incoming HTTP requests |
| Medium  Index: 3 | Application has:   * Clearly-identified security requirements * There are firewall restrictions on outgoing and incoming HTTP requests * Authentication requirements: * Uses HTTP Basic, form-based or Windows Integrated authentication against corporate Active Directory * Azure Active Directory Integration * Authorization requirements: * Incorporates role-based authorization capabilities using AzMan, Active Directory groups or custom implementation * Data security: * Uses SSL protocol for data security in transit, * Application data at rest is secured using out-of-box SQL encryption features.   --TCP 1433 outbound/Inbound port open |
| Complex  Index: 5 | Application has:   * Complex security requirements * Specific Security constraint for connecting to third party application * There are firewall restrictions on outgoing and incoming HTTP requests * The application store secret data that should be updated independently from application code * Require VPN access to resources on Premises or Vice Versa * HBI including PII data handled by application * Authentication requirements: * Uses advanced authentication patterns (i.e. Claims-based authentication, Kerberos multi-hop delegation, federated authentication, integrating with third-party Identity providers, etc.) * Authorization requirements: * Implements claims-based authorization or role-based authorization (using AzMan, Active Directory groups, SQL-based store or custom implementation). * Data security:   The data is secured:   * In transit via transport-based security (TLS/SSL, WCF secure bindings, etc.) * At rest using one of many encryption capabilities (SQL encryption features, DPAPI for configuration data, custom encryption before storing data in centralized repository) |
| Very complex  Index: 10 | Application has:   * Very complex security requirements * There are firewall restrictions on outgoing and incoming HTTP requests * HBI including PII data handled by application * Authentication requirements: * Uses advanced authentication patterns (i.e. Claims-based authentication, Kerberos multi-hop delegation, federated authentication, integrating with third-party identity providers, mutual authentication between its major tiers, etc.) * Integrates with Microsoft or 3rd-party identity products * Authorization requirements: * Implements claims- (or role-) based authorization (using AzMan, Microsoft identity manager, active directory groups or custom implementation). * Data security:  The data is secured: * In transit via transport-based security (TLS/SSL, WCF secure bindings, etc.) * At rest using one of many encryption capabilities (SQL encryption features, DPAPI for configuration data, custom encryption before storing data in centralized repository). * Regulatory/Industry requirement such as HIPAA/EU Data protection act * Data Integrity security requirements like digital signatures/MAC   Application:   * Is enabled with implements extensive audit capabilities to implement data non-repudiation principle; * Offers delegated administration capabilities |

* 1. Server configuration

The following application properties can be distinguished to map an application to Complexity Level/Index based on server configuration categorization type:

Table 3: server configuration Complexity Level/Index mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category definition |
| Very simple  Index: 1 | An application:   * Has no or few server configuration requirements * Runs on out-of-box server OS and web server configurations * Maintains minimum (a few or none) configuration settings, no additional server configuration requirements are identified |
| Simple  Index: 2 | An application:   * Has minimum server configuration requirements, such as basic IIS configuration settings * Has no dependencies on additional OS features beyond standard installation * Requires trivial web server configuration changes (re: authentication, security, etc.), no additional server configuration requirements are identified |
| Medium  Index: 3 | An application:   * Has clearly identified server configuration requirements, such as IIS configuration, application pool custom settings, custom machine, configuration settings, etc. * Requires installation of additional OS features (ex: MSMQ, application server features, etc.), * Has a set of requirements to apply to web server configuration (re: authentication, certificates, security, etc.), no additional server configuration requirements are identified |
| Complex  Index: 4 | An application:   * Has substantial server configuration requirements, such as custom setup of server roles and features, extensive IIS configuration, application pool custom settings, major machine, configuration customizations, including custom sections, etc. * Relies on installation of additional OS features (ex: MSMQ, application server features, etc.) * Relies on networking appliance for SSL Encryption/Decryption, XML/Json Formatting and /or transformations * Dependent on 3rd-party products installed within the execution environment   Requires legacy code, Com+, 32 bit or native code.   * Requires custom domain with SSL support and multiple subdomains are required. * Has a set of complex requirements to apply to web server configuration (to enable authentication/Kerberos delegation, certificates, security, etc.). |
| Very complex  Index: 5 | An application:   * Has very complex server configuration requirements, such as extensive IIS configuration, application pool custom settings, custom machine, configuration sections, COM+/DCOM machine-wide configurations, custom boot settings, specialized custom drivers. * Relies on installation of additional OS features (ex: MSMQ, application server capabilities, etc.), * Relies on networking appliance for SSL Encryption/Decryption, XML/Json Formatting and /or transformations * Is dependent on 3rd-party products installed within the execution environment, * Has a set of complex requirements to apply to web server configuration (to enable authentication/Kerberos delegation, certificates, security, etc.).   Server configuration can be centrally managed and there is an automated change control infrastructure verifying and updating server configuration. |

* 1. Architecture

The following application properties can be distinguished to map an application to Complexity Level/Index based on Architecture categorization type:

Table 3: Architecture Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very Simple  Index: 1 | The application has a very simple conventional web application multi-tier architecture:   * Presentation/business layer connecting to the backend SQL Database. * No additional architectural components are identified. |
| Simple  Index: 2 | The application has a conventional web application multi-tier architecture:   * Presentation, business and data access layer connecting to the backend relational Database. * Presentation tier may incorporate session state management functionality * Frequently used static data can be kept in a cache. |
| Medium  Index: 3 | The application has a conventional multi-tier architecture:   * Presentation, service façade, business and data access layer connecting to the backend relational database. * Presentation tier may incorporate session state management functionality * Frequently used static data can be kept in a cache. Application middle-tier may possess simple OLTP capabilities. * Utilizes Enterprise Library 4.x-6.0 components |
| Complex  Index: 6 | The application has a complex multi-tier architecture:  General aspects:   * Physically separate presentation, service façade, business and data access layer which can be connecting to multiple backend data repositories, including: * Relational databases * NoSQL databases * Other types of persistent stores. * Application depends on windows service/Batch jobs etc. * Workflow defined for application process. * Application may call Win32 API/COM/DCOM objects. * Application can utilize multiple third party components which require complex startup tasks. Application has the functionality to send mail. * Application at any layer accesses registry or depend on registry entries. * Application communicates with On-premises database/SSIS/SSRS, * or application currently utilizes hybrid cloud connectivity * Application utilizes Reporting controls like Microsoft Report Viewer/Crystal Reports with data originating from on premises and Azure SQL Database * Impact on upstream and downstream critical systems   Presentation tier:   * May incorporate session state management functionality * May utilize 3rd-party frameworks, such as Spring MVC, nHibernate, etc. * May keep frequently used static data can be kept in a dedicated cache. * Application may incorporate update notification subsystem based on SQL Server Notification Services, SignalR, etc   Middle Tier:   * The application may use asynchronous data communication patterns to communicate between its major components, including: * Loose-coupled enterprise integration pattern implementations * Service bus communication patterns * MSMQ. queued COM components, asynchronous COM+/Serviced Components, * Application has Web service endpoints like FTP, Email, and Spreadsheet Input. * Application may incorporate scheduled tasks support subsystem. * Application middle-tier may possess OLTP capabilities. * Application may be integrated with other systems, such as: * SharePoint 2007/2010/2013 * O365 SharePoint Online * BizTalk Server 2006/2010/2013 * Application may incorporate data synchronization pattern with end user connected via unstable links * The primary backend relational database may be accompanied by Data Warehousing solution   On-Premises Dependencies:   * The application has components that run on legacy hardware (mainframe, midrange, other systems) and can't be rearchitected during the cloud migration cycle due to time/resource constraints * The application has components that require designated authenticated hardware to run * The application has components that require vendor-supplied hardware * The application has components that run on high-end hardware not conducive to rearchitecture for cloud-based commodity hardware * The application has components that must run on premises due to compliance requirements * The application has components that must run on premises due to proprietary security requirements * The application has components that have real-time processing requirements with strict latency requirements * The application has components that have to run in a designated hoster due to non-technical considerations (legal, contractual, etc.)   DevOps:   * Application may use custom performance counters for production monitoring |
| Very complex  Index: 10 | The application has a very complex multi-tier architecture:  General aspects:   * Physically separate presentation, service façade, business and data access layer which can be connecting to multiple backend data repositories, including: * relational databases * NoSQL databases and other types of persistent stores. * Application depends on windows service/Batch jobs etc. * Workflow defined for application process. * Application may call Win32 API/COM/DCOM objects. * Application can utilize multiple third party components which require complex startup tasks. * Application communicates with On-premises database/SSIS/SSRS * or application currently utilizes hybrid cloud connectivity * Application utilizes Reporting controls like Microsoft Report Viewer/Crystal Reports with data originating from on premises and Azure SQL Database * Impact on upstream and downstream critical systems   Presentation tier:   * May incorporate session state management functionality implemented in a dedicated state provider * May utilize 3rd-party frameworks, such as Spring MVC, nHibernate, etc. * May keep frequently used static data can be kept in a dedicated cache. * Application includes custom WCF component infrastructure comprised of custom channels, encoders, bindings...   Middle tier:   * The application may use asynchronous data communication patterns to communicate between its major components and with: * external systems * Line-of-business applications, including complicated distributed computing communication patterns. * Application may be implementing various enterprise-integration patterns * Application may use: * conventional service bus communication patterns * MSMQ. queued COM components, asynchronous COM+/Serviced Components, * Application uses third-party Enterprise service bus products * Application may be integrated with 3rd-party messaging products such as TIBCO, MQ Series, etc. * Application has Web service endpoints like FTP, Email, and Spreadsheet Input. * Application may incorporate data synchronization pattern with end user connected via unstable links * Application may be integrated with BizTalk Orchestrations, Kafka, Windows Workflow, etc. * Application may incorporate update notification subsystem based on SQL Server Notification Services, SignalR, etc. * Application middle-tier may incorporate long-running persistent workflows and/or possess OLTP capabilities and/or integrate with large number of external systems, including: * Customer line-of-business application * Third-party supplemental services. * Application may be integrated with other systems, such as: * 3rd-party application server such as IBM WebSphere, Oracle WebLogic, etc. * SharePoint 2007/2010/2013 * O365 SharePoint Online * BizTalk Server 2006/2010/2013 * Value Added Network/Clearinghouse   Data-related aspects:   * Application may implement large data transfer management facilities * Application may incorporate scheduled tasks support subsystem. * The primary backend relational database may be accompanied by data warehousing solution * The application may be using a non-SQL data store, such as ISAM. NoSQL, etc. * Application follow OLAP semantics or it may be incorporating transactional processing * The Application database stores may be managed in a distributed transaction using transaction processing monitors.   On-Premises Dependencies:   * The application has components that run on legacy hardware (mainframe, midrange, other systems) and can't be rearchitected during the cloud migration cycle due to time/resource constraints * The application has components that require designated authenticated hardware to run * The application has components that require vendor-supplied hardware * The application has components that run on high-end hardware not conducive to rearchitecture for cloud-based commodity hardware * The application has components that must run on premises due to compliance requirements * The application has components that must run on premises due to proprietary security requirements * The application has components that have real-time processing requirements with strict latency requirements * The application has components that have to run in a designated hoster due to non-technical considerations (legal, contractual, etc.) |

* 1. Communication

The following application properties can be distinguished to map an application to Complexity Level/Index based on communication categorization type:

Table 3: Communication Complexity Level/Index mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category definition |
| Very simple  Index: 1 | The application:   * Exposes conventional HTTP external endpoints * Communicates directly to SQL database using its native protocol. * No additional network-related components are identified. |
| Simple  Index: 2 | The application:   * Exposes conventional HTTP endpoints * Employs TLS/SSL network protocol to secure data in transit. * No additional network-related components are identified |
| Medium  Index: 3 | The application has a medium complexity multi-tier architecture:   * The application presentation tier exposes conventional HTTP endpoints * The application Employs TLS/SSL network protocol to secure data in transit. * Application middle-tier may expose request/response type binary protocol-based endpoint for efficiency purposes (ex: WCF TCP binary binding-based endpoints) |
| Complex  Index: 5 | The application has a complex multi-tier architecture:   * The application presentation, service façade, middle tiers expose conventional http endpoints * The application employs TLS/SSL network protocol to secure data in transit. The application can also implement health status management check networking functionality. * The application can send/receive very large messages or there is transformation required on incoming & outgoing message. * Besides request/response type binary protocol-based endpoint, the application middle-tier may employ lower-level networking communication pattern with other components and external systems, not only including conventional http synchronous calls but also asynchronous socket communication, AMQP messaging with external systems. |
| Very complex  Index: 8 | The application has a complex multi-tier architecture:   * The application presentation, service façade, middle tiers: * Expose conventional http endpoints * Employ TLS/SSL network protocol to secure data in transit * Can also implement health status management check networking functionality. * Besides request/response type binary protocol-based endpoint, the application middle-tier may employ lower-level networking communication pattern with other components and external systems, not only including conventional http synchronous calls but also asynchronous socket communication, AMQP messaging with external systems. * The application can send/receive very large messages or there is transformation required on incoming & outgoing message. * The application can send/receive EDI Message. * The application also can support custom adaptors to implement proprietary networking protocol for communication with backend systems and/or devices. |

* 1. Monitoring &Logging

The following application properties can be distinguished to map an application to Complexity Level/Index based on Logging categorization type:

Table 3: Logging Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very Simple  Index: 1 | The application:   * Has little (or none) features to support logging functionality. * Only low-level tracing information maintained by underlying OS about the running application comprise the overall logging dataset. |
| Simple  Index: 2 | The application:   * Has basic logging functionality including error tracing. * Uses file system or OS-level event log as a logging store * Lacks logging level configuration capabilities. |
| Medium  Index: 3 | The application:   * Has well architected logging facilities, including error tracing, major event notification messaging as well as configurable logging level – from none to verbose. * Uses file system or OS-level event log as a logging store and can switch between log providers using standard configuration capabilities. |
| Complex  Index: 4 | The application:   * Has extensive logging capabilities, including error tracing, major event notification messaging as well as configurable logging level (from none to verbose). * Integrated with logging development framework (Ex: Enterprise Library logging block, log4net, etc.) * Has sophisticated configuration capabilities to plug-in a variety of log providers. |
| Very Complex  Index: 5 | The application has:   * Rich and extensible logging capabilities, including error tracing, major event notification messaging as well as configurable logging level (from none to verbose). * Employs logging development framework (Ex: Enterprise Library logging block, log4net, etc.). * Has sophisticated configuration capabilities to plug-in a variety of log providers. * Integrated with centralized log management system, provides end-to-end tracing, logging correlation capabilities Supports industry standard tools for log data consumption and analysis. * Application may use custom performance counters for production monitoring. * The application may be integrated with monitoring products, such as System Center, OpsNet, Microsoft Operation Manager, etc. |

* 1. Exception Management

The following application properties can be distinguished to map an application to Complexity Level/Index based on Exception Management categorization type:

Table 3: Exception Management Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very Simple  Index: 1 | The application:   * Has little (or none) features to support exception management functionality. * Uses standard unhandled exceptions handler provided by underlying platform. |
| Simple  Index: 2 | The application:   * Has basic features to support exception management functionality. * Provides application-wide unhandled exceptions handler to trace captured errors via logging subsystem. |
| Medium  Index: 3 | The application:   * Implements standard exception management functionality. * Includes custom-defined exception classes/messages. * Has a solid exception stack unwinding strategy to properly handle abnormal use cases. * Collects exception information to trace captured errors via logging subsystem. |
| Complex  Index: 4 | The application:   * Implements standard exception management functionality. * Includes custom-defined exception classes/messages. * Has a solid exception stack unwinding strategy to properly handle abnormal use cases. * Collects exception information to trace captured errors via logging subsystem. * Integrated with exception handling development framework (Ex: Enterprise Library exception handling block, etc.). * Has sophisticated configuration capabilities to persist resulting errors using a variety of preconfigured log providers. |
| Very Complex  Index: 5 | The application:   * Implements standard exception management functionality. * Includes custom-defined exception classes/messages. * Has a solid exception stack unwinding strategy to properly handle abnormal use cases. * Collects exception information to trace captured errors via logging subsystem. * Employs exception handling development framework (Ex: Enterprise Library exception handling block, etc.). * Sophisticated configuration capabilities to persist resulting errors using a variety of preconfigured log providers is integrated with centralized management system. * Supports industry standard tools for system monitoring, data consumption and analysis. |

* 1. Data Access

The following application properties can be distinguished to map an application to Complexity Level/Index based on Data Access categorization type:

Table 3: Data Access Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very Simple  Index: 1 | The application:   * Does not write data to SQL server/any other persistent store. * Uses simple CRUD data access patterns to access backend SQL database, employs standard out-of-box development library for database calls. * Does not require any data migration. * Does not plan to use Azure Storage Table/Blobs/Queues. * Does not require database Backup and Restore service. |
| Simple  Index: 2 | The application:   * Uses simple CRUD data access patterns to access backend SQL database employing standard out-of-box development library for database calls. * May incorporate different data access techniques based on the operation (read-only vs. Updates). * Does not require any data migration. * Does not plan to use Azure Storage Table/Blobs/Queues. * Does not require database Backup and Restore service. |
| Medium  Index: 3 | The application:   * Uses local storage to store the persistent data. * Uses standard data access patterns to access backend relational database. * May incorporate different data access techniques based on the operation (read-only vs. Updates). * The underlying SQL database may be using a variety of simple types and database objects, including stored procedures and parameterized queries. * Perform batch updates in database. * Requires data migration for small volume of data. * Does plan to use Azure Storage Table/Blobs with small volume of data. * Does not plan to use Azure Storage Queues. * Does not frequently read and update data in Azure Storage Table/Blobs. * Requires Database regular backup and Restore. * Uses retry/failure pattern to connect to SQL |
| Complex  Index: 4 | The application:   * Uses local storage to store the persistent data. * Uses a variety of data access patterns to access backend database repositories, including relational databases, NoSQL persistent stores. * Employs a sophisticated data access development framework, may include object-relational mapping library. * Has sophisticated configuration capabilities to tune data access performance. * Optimize database to improve performance and user experience of the application. * The backend database stores may be managed in a distributed transaction using transaction processing monitors. * The backend SQL Database uses SQL CLR components. * Perform batch updates in database. * Uses SQL server analytics. * Uses SQL server services like Reporting Services, Notification Services or Service Broker. * Uses Cross database/Cross server queries. * Uses retry/failure pattern to connect to SQL. * Requires data migration for very large volume of data. * Does plan to use Azure Storage Table/Blobs with large volume of data * Does plan to use Azure storage queues with frequently operations in the queue. * Frequently read and update data in Azure Storage Table/Blobs. * Requires Database regular backup and Restore and have defined retention policies. * Requires SQL Full Text search feature. * Data tier code accesses the registry. |
| Very Complex  Index: 8 | The application:   * Uses local storage to store the persistent data. * Uses a variety of data access patterns to access backend database repositories, including relational databases, NoSQL persistent stores. * Employs a sophisticated data access development framework. * Optimize database to improve performance and user experience of the application. * May include object-relational mapping library and has sophisticated configuration capabilities to tune data access performance. * The backend database stores may be managed in a distributed transaction using transaction processing monitors. * The backend SQL Database uses SQL CLR components. * The data layer implements data synchronization and replication pattern (P2P, transactional/merge replication) between application deployments spread across multiple geographical locations. * Perform batch updates in database. * Uses SQL server analytics. * Uses SQL server services like Reporting Services, Notification Services or Service Broker. * Uses Cross database/Cross server queries. * Uses retry/failure pattern to connect to SQL. * Uses federated database to fetch application’s data. * Requires data migration for very large volume of data. * Does plan to use Azure Storage Table/Blobs with large volume of data. * Does plan to use Azure Storage queues with frequently operations in the queue. * Frequently reads and updates data in Azure Storage Table/Blobs. * Requires Database regular backup and Restore and have defined retention policies. * Requires SQL Full Text search feature. * Uses third party products for backup, compression etc. * Data tier code accesses the registry. |

* 1. Non Functional Requirements

The following application properties can be distinguished to map an application to Complexity Level/Index based on Non Functional Requirements categorization type:

Table 3: Non Functional Requirements Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very Simple  Index: 1 | * The application has few (or none) Non Functional Requirements identified. |
| Simple  Index: 2 | * The application has basic performance requirements measured as user-perceived latency of loading web pages. * The application does not have any load spikes in general. * No code change required to achieve scalability. |
| Medium  Index: 3 | * The application has a set of requirements identified around performance, availability and scalability with corresponding metrics measured via standard performance monitoring tools. * The application does not have any load spikes in general. * The application uses caching mechanism to cache some of the application data. |
| Complex  Index: 4 | * The application has a set of formal requirements identified around performance, availability and scalability. * The application has very short SLAs. * The application has very high throughput / high user load requirement. * The application has frequent load spikes like monthly/Quarterly. * The application has large memory footprint or consuming large non-persistent storage. * The application uses caching extensively and different type of cache handling mechanism is used across the application. * Application has very short Recovery Point Objective and Recovery time Objective defined. * The application is built for DevOps, heavily instrumented to allow easy production monitoring with corresponding metrics measured via standard performance monitoring tools. |
| Very Complex  Index: 5 | * The application has a set of formal requirements identified around performance, availability and scalability. * The application has very short SLAs. * The application has very high throughput/high user load requirement. * The application has frequent load spikes like daily/weekly. * The application has large memory footprint or consuming large non-persistent Storage. * The application uses caching extensively and different type of cache handling mechanism is used across the application. * Application has very short Recovery Point Objective and Recovery time Objective defined. * The application is built for DevOps, heavily instrumented to allow easy production monitoring with corresponding metrics measured via standard performance monitoring tools. * It also supports industry standard system monitoring suites and well integrates into overall single-pane of glass view of enterprise infrastructure. |

* 1. Geolocation Requirements

The following application properties can be distinguished to map an application to Complexity Level/Index based on Geolocation categorization type:

Table 3: Non Functional Requirements Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very Simple  Index: 1 | The application has few (or none) Geolocation Requirements identified. |
| Simple  Index: 2 | The application has basic Geolocation requirements with minor UI differences based on the location of the end users. |
| Medium  Index: 3 | The application has a set of Geolocation requirements including;   * Application logic including varied business rules based on end user location. |
| Complex  Index: 4 | The application has a set of Geolocation requirements including;   * Application logic including business rules based on end user location. * The application implements ring-fencing for specific locales restricting the user end population to a geographical area. |
| Very Complex  Index: 6 | The application has a set of Geolocation requirements including;   * Application logic including business rules based on end user location * The application topology consists of numerous deployments across the major geographies with underlying data store being synchronized in near- or real-time. * The application implements ring-fencing for specific locales restricting the user end population to a geographical area. |

* 1. Compliance Requirements

The following application properties can be distinguished to map an application to Complexity Level/Index based on Compliance categorization type:

Table 3: Non Functional Requirements Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very Simple  Index: 1 | The application has few (or none) Compliance requirements identified. |
| Simple  Index: 2 | The application has basic Compliance requirements mandating data security in transit. |
| Medium  Index: 3 | The application has various Compliance requirements mandating data security in transit, at rest, and requirement to follow commonly used industry practices. |
| Complex  Index: 5 | The application has complex Compliance requirements including industry certification, such as HIPAA, SOC, etc. |
| Very Complex  Index: 8 | The application has complex Compliance requirements including industry certification, such as HIPAA, SOC, as well as extensive requirements around data privacy and credit card processing (PCI, etc.). |

* 1. High Availability and Disaster Recovery

The following application properties can be distinguished to map an application to Complexity Level/Index based on High Availability and Disaster Recovery Requirements categorization type:

Table 3: Non Functional Requirements Complexity Level/Index Mapping

|  |  |
| --- | --- |
| Complexity Level/Index | Category Definition |
| Very Simple  Index: 1 | The application has few (or none) High Availability and Disaster Recovery identified. |
| Simple  Index: 2 | The application has basic High Availability and Disaster Recovery requirements expressed as conventional load balancing of the front-end tier. |
| Medium  Index: 3 | The application has a set of requirements identified around High Availability of each tier:   * Load-balanced web front end * Load-balanced middle-tier * Clustered backend database   The application has a set of requirements identified around Disaster Recovery:   * Implement cold stand-by disaster recovery configuration with a separate deployment being stood up on demand during the downtime caused by disaster event. * The database tier implements capabilities for disaster recovery scenarios. |
| Complex  Index: 4 | The application has a set of formal requirements identified around High Availability and Disaster Recovery:   * Load-balanced web front end * Load-balanced middle-tier * Clustered backend database   The application has a requirement to implement warm stand-by disaster recovery configuration with a separate dedicated deployment and data replication. |
| Very Complex  Index: 7 | The application has a set of formal requirements identified around High Availability and Disaster Recovery:   * Load-balanced web front end * Load-balanced middle-tier * Clustered backend database   The application has a requirement to implement hot stand-by disaster recovery configuration with a separate always online dedicated deployment and real-time cross deployment data replication. |