**FeederApp Release Notes**

**Coding standards that has been followed: -**

* Pascal casing has been used for methods, class and properties.
* Underscore (\_) prefix for private fields.

**Naming Conventions: -**

* As per the functionality of the methods meaningful naming conventions have been followed.
* Control captions, Error messages are customized in resource file for easy addition and modifications. This will also support localization for various cultures.

**Architecture: -**

* The application has been developed following MVVM (Model-View-ViewModel) pattern.
* The presentation layer and the logic is loosely coupled.
* This enables a separation of responsibilities on teams.
* The architecture structure implemented is multi-tier architecture with grouped folders as per the actions such as Helpers, Models, Resources and Views. This will ensure Separation of Concerns and facilitate the reusability of any code. This adheres strongly to OOPS concepts.

**Design Patterns: -**

* Singleton design pattern is used in DAC layer DB instance creation, which will help in the reuse of the DB connection for all requests throughout the application life cycle. This will avoid multiple object instantiation and memory consumption.

**Coding Practises: -**

* Best coding practise with respect to .Net framework has been followed.
* Extensive use of Generics for Classes and return types. This reduces the number of lines of code, boxing and unboxing and also facilitates binary code reuse. This also improves the application response speed, code readability and Code Metrics and reduces the output DLL or package size.
* Centralized configuration file (App.config) is used for maintaining configuration related values and settings. As of now these are used for values like connections strings. Based on usage values like Application objects and other settings can be configured here.
* Concrete exception handling methodologies has been implemented using try catch blocks. Any exception will be routed back to the UI layer and appropriate error message can be shown. Specific exceptions have been implemented using specific exception catch blocks first followed by generic catch blocks, and the error will be logged which will help us analyse the cause easily.
* Errors are logged in a log file on each occurrence.
* Comments have been added to every method, Class and places wherever necessary.

**Unit Test: -**

* Unit test classes are written for every function as per Test Driven Development.
* This reduces the need to do complete testing on every bug fix or release. The problem in the modified code can be detected earlier by running the unit test while coding itself.