General Requirements

# 1.0 Database Design Requirements

## 1.1 Naming Convention

1. Names should clearly represent an attribute or property of the Entity(ies) they represent.
2. Use an underscore to separate the words within a name / identifier, for example “Case\_Type”.
3. Use longer names instead of abbreviations. Although abbreviations are discouraged if those are adopted make sure that the larger community clearly understands those.
4. It is recommended to prioritize the reference Entity first and the adjective or other later. This is targeted to remove any possible ambiguity, therefore a “First\_Name” is written as “Name\_First”, “Name” the entity and “First” the position of the name part. Eventually a parser, data analysts, and AI operations can be more accurate recognizing Entities being represented.
5. Complex names that reference more than one entity such as those for associations will have the Entity references first and the attribute that relates both at the end.
6. Names that express actions (verbs) like “Delete”, “Update” or other will be placed. For example, instead of “Deleted\_Activity” it will be “Activity\_Deleted”.

## 1.2 Database Schemas

1. Each Database Schema should manage a single area of concern and attend the smallest possible number of independent business entities for example the “Entity” schema manages the following business data objects:
   1. Organization – Agencies, Corporations, Institutions, or any other unit which conducts sort of business or operations.
   2. Officer – an Agent that works and/or provide services for an Organization.
   3. Person – details about a Human being.
   4. Contact – a data concept for a person or organization to contact.
2. Each Database Schema should be small, self-sustained and be isolated from other schemas.
3. A Schema could be removed from a Database and be placed in another Database without any modifications to any table, constraint or resource and be fully functional and accessible by just repointing the SQL instance.
4. Other data objects not in the Schema should be accessible only through a sort of a “Reference ID” and no FOREIGN KEY or any other dependency should be placed upon those. Referential integrity should be managed by an externa resource that may keep track of all data objects.
5. Duplicate Code Sets or some other duplication may be required to achieve schema isolation.
6. Data Objects within a Database Schema can fully reference and be dependent on each other as needed.

## 1.3 Unique Identifiers

1. IDs should not be built to be interpreted in any way, if they are created with some recognizable structure, such details should be ignored and not be dependent on.
2. All IDs should be generated by a reliable redundant singleton service upon request and will guaranty that no ID is repeated.
3. Numeric Database generated IDs cannot be used since a schema could be moved and/or merged with other identical schemas at any time.
4. While merging with other identical schemas there will be no conflict with any identifier since all should had been generated with the same singleton.
5. The singleton ID generation service could be substituted with any method that will guaranty the required uniqueness.
6. IDs should be registered with additional metadata identifying its target reference object and identity.
7. Generated IDs should be able to be used to search for related business objects.
8. The ID generator service should allow the request of many instances upon a single call to be used in mass insertions.
9. An ID that had been used to identify an object or data entity should not be ever reused.

## 1.4 Reference IDs

1. As suggested, Reference IDs should be used to point to artifacts outside a Schema.
2. Any ID used should exist and be supported as consistent as possible.
3. Removal of an ID should be fully logged and documented in detail.
4. If for some reason orphan references are found those should be registered as so and be set to “null”.

## 1.5 Reference Data

1. Code Set tables should be easily identified and their names will always end with “\_Code” or “\_Type”.
2. The purpose of the code sets is to identify a code or type and its description and therefore those will be the expected names.
3. Code Sets may contain a “Category” column that should be used to further classify the code as needed.
4. Code Sets may contain a “Tags” column that will be used to “Tag” the column with Privacy tags (see “Privacy and Security” section) or other as needed.
5. Code Sets should contain a “CodeSet\_Name” column that should be used to provide the standard code set name that it relates too.
6. Code Set exceptions may include a few more columns in addition to the expected code – description. Just a few of these, if any, are expected.
7. Each Code or Type columns will include a “Code\_ID” (or “Type\_ID”), a description and a Category\_ID column.

## 1.6 Privacy and Security

1. Privacy initiatives that should be supported include:
   1. HIPAA – Health Insurance Portability and Accountability Act
   2. GDPR - General Data Protection Regulation
   3. PII - Personal Identifiable Information
   4. NIST\_PII – Person Identifiable Information
2. Data elements related to any person identifiable information should be tagged with a label corresponding to the related initiative.
3. Data elements related to any person identifiable information in conjunction with healthcare data should be tagged with a label corresponding to the related privacy healthcare initiative.

# 2.0 Functional Areas Requirements

## 2.1 User Session Management

1. Upon login, into the system and internally a corresponding time-limited "Session" is created.
2. Associated with the Session a "Session\_ID" is created along with other relevant session information (see Application.Session).
3. The session life is bounded by time or usage, a valid session is requested upon login, and it is closed and no longer valid after logout or the max session lifetime expires as configured.
4. The session ID should be recorded during all database transactions done to any database object during the lifetime of the session (see the “Session\_Updated\_ID” column).
5. It is possible that a (period-limited) Access-Token associated to the Officer is provided in the request and used to validate that the user is authorized to login (see Application.AccessToken).

## 2.2 Follow-Up List

1. Follow-Up or Checklists should be available and presented to Managers, Officers, or Users to work through pending requests, follow-up tasks or other related actions (see sample list in 4.0 (2)). These lists help the Users to keep track of pending tasks, priorities, or as a reminder of things to do.
2. Lists are configurable therefore, Managers or assigned Officer should be able to create, update or retire a list or items within a list.
3. Within a list, Managers should be able to assign tasks to Officers.
4. An Officer should be able to add or move items from a list to another as progress is made or as needed.