Index:

1. Revision History
2. Project Overview
3. Information Description
   1. User Interface
   2. Data Elements Dictionary
4. Functional Description
   1. Functions
   2. Processing Narrative
   3. Design Constraints
5. Performance Requirements
6. Implementation Priorities
7. Foreseeable Modifications and Enhancements
8. Acceptance Criteria
   1. Functional Tests
   2. Documentation Standards
9. Revision History

|  |  |  |
| --- | --- | --- |
| Author/Editor | Date | Summary |
| Donnie Gray | 12 June 2024 | Created initial document version with an Index and eight sections, Project Overview, Informational Description, Functional Description, Performance Requirements, Implementation Priorities, Foreseeable Modifications and Enhancements, Acceptance Criteria, and Revision History respectively. |
| Donnie Gray | 24 June 2024 | Moved revisions section to the beginning of document and re-implemented as a table per Todd McAllister’s recommendation. |

1. Project Overview:This project is meant to generate a custom report which users can utilize to submit prescription numbers (Rxno’s) in order to see related claims and invoice information. Currently, users utilizing 835 Electronic Remittance files often have a list of “unmatched” Rxno’s. This implies that these prescriptions have claims for which a payment has not been found, or “matched.” Users then must investigate these prescriptions, find the relevant customer ID and relevant claim dates, then input that customer ID into a separate custom query along with an appropriate date range, to receive a list of potential transactions, which must then be parsed to find the appropriately relevant Invoice Number (Invoice), all in order to verify appropriate payment and enter relevant information into Sage 300 to confirm as much. The intent of this query is to submit an Rxno and receive the relevant information which can be manually entered into Sage without manual investigation.
2. Information Description
   1. User Interface

The UI will consist of a file which can by opened by Microsoft Excel on machines connected to the appropriate FrameworkLTC and Sage 300 databases. It will will have a field to enter a prescription number, a button with which to run the query, and will generate a sheet of appropriate query results consisting of a column for prescription number (labeled “RxNo”), a column indicating which customer ID is related to the transaction (labeled “CustID”), a column indicating whether it is an insurance claim or other method of payment (labeled “MOP”), a column indicating whether the claim is the primary or secondary payment option if it is a claim (labeled “Priority”), a column showing date for the claim (labeled “Claim Date”), a column showing the related invoice number (labeled “Invoice”), and a column showing the invoice date (labeled “Invoice Date”).

* 1. Data elements

Prescription Number (Rxno)

Customer ID (CustID)

Method of Payment (MOP)

Primary vs Secondary vs Nonclaim (Priority)

Date which claim was made (Claim Date)

Invoice Number (Invoice)

Date which invoice was made (Invoice Date)

1. Functional Description:
   1. Processing Narrative
      1. Verification
         1. A user wants to see the billing transactions related to a specific prescription number. They open the customer query, enter a prescription number, and click “Query.” A new sheet is generated with the claims and related Invoices for the prescription.
   2. Design Constraints
      1. Big Data
         1. Due to the possibility of claim rejection and claim reversal, some prescriptions will have multiple claims that have been sent, accepted, rejected, and/or reversed. It may not be possible to match every reversal and rejection to the claim which initiated it in order to scrub irrelevant results. This may result in several records per Rxno. To keep these results easy to parse, it may be in the best interest to limit queries to one prescription at a time. If this is found to not be an issue, results querying multiple Rxno’s will be ordered by Rxno and then Invoice Date for easier parsing.
      2. Matching Invoices in Sage
         1. This report will not be able to input the appropriate data directly into Sage 300, and so when the query results are provided, a user will still have to parse them and enter the correct information manually.
      3. Database Structure Updates
         1. Both Sage and SoftWriters are the creators, designers, and primary distributors of the products Sage 300 and FrameworkLTC respectively. Because of this, changes to their database structure may be made at any time without warning when providing software updates to their products. Any such change may result in this report becoming defunct, either due to no longer being necessary, or because the information it’s querying is no longer present or labeled the same way on the tables from which it is searching.
2. Performance Requirements

A machine meeting the requirements for running both FrameworkLTC and Sage 300, per the versions of these products available to the user, that is also connected to the server database for these products.

1. Implementation Priorities
2. Getting Customer ID from RxNo
3. Getting transaction dates from RxNo and CustID
4. Getting Invoice Numbers from RxNo, CustID, and Claim Date
5. Foreseeable Modifications and Enhancements
6. Multiple RxNos per query
7. Scrubbing rejected and reversed claims from query results
8. Database Structural changes from SoftWriters or Sage
9. Acceptance Criteria
   1. Functional Tests
      1. Query Prescription Number
      2. Enter:
         1. Prescription Number
      3. Generate query results
         1. Compare query results to manual data investigation to confirm accuracy and relevancy, especially including claims and invoices as matched to Rxno’s.

7.2 Documentation Standards

A. Documents to be delivered:

* + 1. Report Requirements Specification