**4 Function Point Analysis**

Team KRSS estimates the project encompasses 60. We estimate the total effort at 120 person-person hours, for an average of 2 person-hours per function point.

**4.1 External Inputs (EI)**

The project incorporates 3 EIs, with a weighting factor of 5 (5, simple, 7 average, 10 complex), for a total count of 15.

**4.1.1 Searching for Invoices**

This function uses control inputs from the user to determine which invoice(s) to return to the user in the Invoice Search Results ILF. This function will use textboxes and pick lists to record inputs and then format the inputs so that the program can search the folders for matches. This has a simple weighting factor because it only draws input from the user.

**4.1.2 Change Results Filter**

The results returned to the user after the initial search can be sorted and filtered and this function will utilize textboxes and pick lists to manage control inputs. The Filtered Search Results ILF will be adjusted according to the values returned by this function. This task has a simple weighting factor.

**4.1.3 Change Settings**

The PPG Invoice Finder will allow the user to change the folders the application searches through for invoices, so this function will modify the Search Locations ILF. The user will use textboxes and windows browser to modify the folders. This EI has a simple weighting factor.

**4.2 External Outputs (EOs)**

The project incorporates 2 EOs, with a weighting factor of 7 (5 simple, 7 average, 10 complex) for a total count of 21.

**4.2.1 Sort Results**

The search results will be capable of being sorted. The sorts include rearranging by customer ID, store ID, region ID, or date. This will be accomplished by a series of value comparisons. This function will be automatically called once when the invoice search results are returned, sorting the data by date (newest to oldest). This task has an average weight factor.

**4.2.2 Find Invoices**

When searching for invoices matching user criteria the software will perform a series of comparison statements to determine matching invoices. Because the user can give varying specificity, the number of comparisons resulting as matches (true) could be very large, resulting in a lot of processing. This function will be used every time the user does an invoice search. This task has a complex weighting factor due to the response time hurdle that will result from the thousands of invoices that could end being compared against.

**4.2.3 Filter Results**

This function will use a simple comparison statement to filter out invoices in the search results. This function will be very similar to the Find Invoices function. This will be utilized every time the user enters filter criteria and hits the filter button. This task has a simple weighting factor.

**4.3 External Inquiries (EQs)**

The project incorporates 0 EQs.

**4.4 Internal logical files (ILFs)**

The project incorporates 3 ILFs, with a weighting factor of 10 (7 simple, 10 average, 15 complex), for a total count of 30.

**4.4.1 Invoice Search Results**

A table consisting of the invoice file names that matched based on the search criteria, their location, the filename broken into transaction data, a email request box, and an export select box. The data displayed in this table can be adjusted in the settings. This task has an average weighting factor due to the learning curve involved with abstracting the GUI from the C# code.

**4.4.2 Filtered Search Results**

The set of invoices returned to the user based on the search criteria can be further narrowed down and sorted with filter settings. The filter options will be the same options available on the search criteria page. The data can be sorted by value based on the various data associated with the invoice file name. This task has an average weighting factor due to the learning curve involved with abstracting the GUI from the C# code.

**4.4.3** **Search Locations**

The folders that the application searches in for invoices are displayed in a basic table. The table displays the folders path, its internal nickname, and its status upon application startup. This task has an average weighting factor due to the learning curve involved with abstracting the GUI from the C# code.

**4.5 External Interface Files**

The project incorporates 0 EIFs.

**4.6 Value Adjustment Factors**

The sum of all VAFs is 34, based on the following weights (0 to 5, where 0 indicates the factor has no relevance and 5 indicates that the factor is absolutely essential).

|  |  |
| --- | --- |
| Factor | Weight |
| 1. Backup and recovery | 0 |
| 1. Data communications | 2 |
| 1. Distributed processing | 0 |
| 1. Performance | 5 |
| 1. Heavy use | 5 |
| 1. On-line data entry | 0 |
| 1. Multiple screens | 2 |
| 1. On-line update | 0 |
| 1. Data complexity | 2 |
| 1. Processing complexity | 3 |
| 1. Reusability | 5 |
| 1. Conversion/installation | 5 |
| 1. Multiple installation | 5 |
| 1. Facilitate change | 0 |
| WEIGHTED SUM | *34* |

**4.7 Function point computation**



The weighted sum of all information domain counts (from sections 4.1-4.5) is 66, and the weighted sum of all VAFs is 34, giving a function point count of

FP = 66 \* [0.65 + (0.01 \* 34)] = 65.34.

This rounds down to a final function point count of 65.