**Software System Specification and Project Plan**

Comp 450, Software Engineering

**Invoice Recall System**

**Team Name**

Team leader: Stephen Tam

Ronald Jenkins, Kory Stansbury, Seth Loew

**Editor of this Document:** \_\_\_\_\_\_\_\_\_\_\_

Date of Delivery: mm/dd/yyyy

Corporate sponsor name

Corporate sponsor business address

Corporate sponsor web page address

Contact: Contact person’s name

Contact’s address

Contact’s email address

Contact’s phone number

**1 Narrative description of project**

Invoice Recall System will be an application that is set up to search for invoices and display them in various ways. It will work with PPG’s current system by searching in their archive folder, root folder, and caches. The program will search through those various folders and caches to find invoices. The search is dependent on the search parameters that the user inputs. Once found, those invoices will be displayed in either a .zip folder or a .pdf file. This software will be able to show if an invoice was not found. If an invoice was not found, it is either missing or corrupted. The user will be able to see which invoices were not found and be able to put them into their final destination. This saves PPG’s IT department time from searching through the folders to find the invoices and will allow more efficient work with searching for invoices.

**1.1 Context**

Description of the hardware/software/operations environment into which the software will be deployed. Indicate how the proposed software fits into the mission of the organization. Make explicit reference to at least one published work (manual, non-fiction literature, article, or web site) which contains more detailed information about the target organization.

**1.2 Overview of software product capabilities**

Informally describe the capabilities the software should have.

**1.3 Related systems.**

Include here at least two examples of software systems which are similar or related to the proposed system. For each example, make explicit reference to at least one published work which describes some aspect of the example system or the environment into which it is deployed.

**2 External view**

Invoice Recall System will run inside PPG’s current system. It will communicate with the servers inside of PPG’s network including the Archive folder, the Root folder, and various caches. There will be one interface that the user deals with. The user will be able to search for invoices using different parameters such as customer ID, date, range of dates, and invoice number. Once the user has entered in the specified information, the application will look through the current folders and caches. Once found, the interface will then display the invoices for the search in one of two ways: a .zip folder or a .pdf file of all the invoices. It will also list the invoices that were not found/corrupted.

**2.1 User scenarios**

**Scenario A.**

In order to properly configure our application to perform invoice searches, a PP IT employee would use edit the paths to the Root and Archive folders containing the invoices. Such a behavior is shown below in Use-Case i. This tool, to be implemented as a pop-out file search window, would store the file paths which the user wished to be searched. These file paths would be orderable by priority, and the use would be able to add additional paths to the base two if they so desired.

**Scenario B.**

Proper searches using our application would require the entering of several search parameters for the tool to use as shown in Use-Case ii. In particular, the PPG Help-desk employee using the software could narrow his search by entering a Customer ID number, an Invoice ID number, or date ranges for the search tool to use. In order to simplify the process of locating multiple date ranges worth of invoices, our application would allow the user to enter multiple date ranges at once.

**Scenario C.**

Our application is designed to output the located invoices in one of two ways. A PPG Help-desk employee may have the application output the located invoices in a .zip file, and/or concatenated in a .pdf file. Use-Case iii demonstrates this interaction. In addition to choose which of these outputs the user would like (either or both), the user may also choose whether or not to generate a failure report for the search. This report would indicate any invoices which the program would have expected but couldn’t find.

**Scenario D.**

In addition to setting the Archive and Root folders for the search component of the application, a PPG Help-desk employee would be required to set the output folder for any files creates by the search (Use-Case iv). This would simply take the form of Pop-out file explorer window for locating or creating a destination folder.

**Scenario E.**

Once configured, the program would be ready to search for invoices. A PP Help-desk employee, as shown in Use-Case v, would request a search from the system. Our system would then draw from the specified Archive and Root folders and perform a search based on the parameters that the user had provided. Finally, output files of the invoices would be generated and placed in the specified destination folder.

**2.2 Use-case diagrams**



**Use-Case i**



**Use-Case ii**

**Use-Case iii**

**Use-Case iv**



**Use-Case v**

**2.3 Integration issues**

There are several issues when integrating our application with PPG’s current system. Security issues limit the amount of testing that can be done at this time. The Invoice Recall System should look through different invoices and once those invoices are found, the application will display those invoices. Due to security issues, we are not able to search the invoices of PPG’s customers, we must make our own “customers” and make them as similar to PPG’s invoices as possible. This could potentially cause problems when integrating our application. We must be careful with how we program our application to make sure that it is easy to understand and interpret in order that PPG IT employees are able to change the necessary code. That way, PPG can use our application with their system. Failure to do so could result in the application not working with PPG’s current system even though the application works with our “customer” invoices.

**3 Breakdown of software components to be produced**

The proposed system consists of 3 major components, each having an average of 3.7 subcomponents.

**3.1 Component 1 – User Interface**

Invoice Recall System includes a Graphical User Interface that the user will interact with. The main screen will give the user different parameters to search the invoice folders.

**3.1.1 Search Parameter subcomponent**

The user will be given parameters to search for invoices. Each invoice has a specific identification number including the date it was purchased, the customer number, and the invoice number. This data will then be used to query the database and process the invoices. The user may enter more than one parameter when searching for an invoice. The application would look at every parameter and make its search based off of those parameters.

**3.1.1.2 Customer ID subcomponent**

The user will be able to enter a customer ID when searching for different invoices. Each customer has their own customer ID and the user is able to search for invoices that only that specific customer has purchased.

**3.1.1.3 Invoice Number subcomponent**

The user will be able to enter an invoice number when searching for an invoice. Each transaction has a specific invoice number and will be used to search the folders for that invoice.

**3.1.1.4 Date subcomponent**

The user will be able to enter a specific date when searching for an invoice. The user could find what transactions were made by what date they were purchased on.

**3.1.1.5 Range of Dates subcomponent**

The user will also be able to enter a range of dates when searching for an invoice.

**3.2 Component 2 – Search/Process Invoices**

Once the user enters the search parameters, the application will process those parameters into a query. The query will include the different parameters that the user entered and search the folders and the caches to find the invoices. Then it will send those invoices back to the Invoice Recall System.

**3.2.1 Retrieve data from the search parameters subcomponent**

The data that the user entered will be received into a query. The query will consist of the parameters that the user has entered.

**3.2.2 Search for the invoice subcomponent**

The query will search into PPG’s system to find the specific invoices. The search will look into the archive and root folders. It will also look into the various caches where the invoices could be.

**3.2.1.1 Priority queue subcomponent**

There will be a priority queue on the order of where the query will look first. The query will first look into the caches, then it will look into the root folder, and the archive folder.

**3.3 Component 3 – Output File Manager**

After the search for the invoices is finished, the application will display all of the found invoices into one of two formats: a .zip folder and a .pdf file. It will also output a list of the invoices that were missing/corrupted.

**3.3.1 .zip subcomponent**

One of the ways the invoices will be shown is by a .zip folder. In the folder will be the list of invoices that were found with the invoice number showing.

**3.3.2 .pdf subcomponent**

The other way that the invoices will be displayed is by a .pdf file. This .pdf file will be a large file with all of the invoices in that specific .pdf file.

**3.3.3 Corrupted/Missing Invoices subcomponent**

While the query is searching for the invoices, if an invoice could not be found, the application will keep track of that. When the query is done searching for the invoices, it will have a list of invoices that were not found. These invoices will be produced in a list that will be displayed for the user. The user knows which invoices were not found and will be able to put the invoices that were not found to their final destination.

**4 Function Point Analysis**

Team \_\_\_\_\_ estimates the project encompasses \_\_\_ function points. We estimate the total effort at \_\_\_ person-hours, for an average of \_\_\_ person-hours per function point.

**4.1 External Inputs (EIs)**

The project incorporates \_\_\_ EIs, with a weighting factor of \_\_\_ (3 simple, 4 average, 6 complex), for a total count of \_\_\_.

**4.1.1 etc.**

Describe here each EI. An EI is an elementary processing function which resides inside the proposed software and uses data or control inputs coming from outside the system in order to maintain an ILF.

**4.2 External Outputs (EOs)**

The project incorporates \_\_\_ EOs, with a weighting factor of \_\_\_ (4 simple, 5 average, 7 complex), for a total count of \_\_\_.

**4.2.1 etc.**

Describe here each EO. An EO is an elementary processing function which resides inside the proposed software and performs at least one mathematical calculation to send processed information to the external environment.

**4.3 External Inquiries (EQs)**

The project incorporates \_\_\_ EQs, with a weighting factor of \_\_\_ (3 simple, 4 average, 6 complex), for a total count of \_\_\_.

**4.3.1 etc.**

Describe here each EQ. An EQ is an elementary processing function which resides inside the proposed software which makes inquiries to an external entity such as a user or an external database. An EQ makes no changes to the data it sends or the data it receives. It also does not change any ILFs.

**4.4 Internal logical files (ILFs)**

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

**4.4.1 etc.**

Describe here each ILF. An ILF is a user-visible logical grouping of data which resides within the boundary of the proposed software system. (Example: A set of branch office locations and phone numbers which is constructed and maintained by the proposed software.)

**4.5 External Interface Files (EIFs)**

The project incorporates \_\_\_ EIFs, with a weighting factor of \_\_\_ (5 simple, 7 average, 10 complex), for a total count of \_\_\_.

**4.5.1 etc.**

Describe here each EIF. An EIF is a user-visible logical grouping of data which resides outside the boundary of the proposed software system. (Example: A data table in an already-existing database not constructed nor maintained by the proposed software.)

**4.6 Value Adjustment Factors (VAFs)**

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

|  |  |
| --- | --- |
| Factor | Weight |
| 1. Backup and recovery |  |
| 1. Data communications |  |
| 1. Distributed processing |  |
| 1. Performance |  |
| 1. Heavy use |  |
| 1. On-line data entry |  |
| 1. Multiple screens |  |
| 1. On-line update |  |
| 1. Data complexity |  |
| 1. Processing complexity |  |
| 1. Reusability |  |
| 1. Conversion/installation |  |
| 1. Multiple installation |  |
| 1. Facilitate change |  |
| WEIGHTED SUM | *(VAF total)* |

**4.7 Function point computation**

The weighted sum of all information domain counts (from sections 4.1 – 4.5) is (count total), and the weighted sum of all VAFs is (VAF total), giving a final function point count of

FP = count total × [0.65 + (0.01 × VAF total)] = \_\_\_\_\_

**5 Task breakdown for the conduct of the project**

Using a task breakdown analysis, team \_\_\_\_\_ has identified \_\_\_ individually scheduled subtasks, requiring a total of \_\_\_ person-hours. (Include here all work products, including documentation items such as this plan and the three mini-plans)

**5.1 Document preparation**

\_\_\_ separate documents must be prepared, requiring a total effort of \_\_\_ person-hours.

**5.1.1 Subtask – Project description and elevator talk**

This task involved \_\_\_ individuals and required \_\_\_ person-hours.

**5.1.2 Subtask – System specification / Project plan and classroom presentation**

This task involved \_\_\_ individuals and required \_\_\_ person-hours.

**Etc.**

…

**5.2 Task**

**5.2.1 Subtask**

**5.2.2 Subtask**

…

…

**6 Schedule**

The schedule covers an elapsed time of \_\_\_ days, and includes \_\_\_ individually scheduled subtasks.

**6.1 PERT graph**

Use Visio to draw a PERT graph for major tasks, showing task names, task dependencies, and elapsed time allocated to each task. This must be pasted into the MS Word document, so do not make it so detailed as to be unreadable.

**6.2 GANTT chart**

Use Microsoft Project to construct a GANTT chart for tasks, showing task names, task dependencies, and elapsed time allocated to each task. Same comments apply here as for the PERT diagram.

**6.3 Project Deadlines**

Tabular schedule of completion dates. Include all the tasks and subtasks from section 5, with a completion date for each.

**7. Risk table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Probability** | **Impact** | **RMMM** |
| Time loss to other Classes | 0.85 | 2 | Flexibility with work distribution among team members and purposeful time management |
| Team member not contributing | 0.02 | 2 | Scheduled group meetings and work checkups |
| Gaps in critical knowledge of C# | 0.1 | 3 | Member to member honesty about C# hang-ups and frequent use of online tutorials |
| Software incompatibility issues with PPG’s system | 0.35 | 1 | Communication with PPG Sponsor and early backend testing with actual system |
| Losing a team member | 0.01 | 1 | Redistribute work and reevaluate goals with Sponsor is necessary |
| Difficulty meeting or communicating with Sponsor | 0.5 | 2 | Ensure members have open schedules for Sponsor meetings and communicates |
| Change in delivery date | 0.1 | 2 | Plan sprints to complete project before the Sponsor’s deadline |
| Poor documentation on the part of the team | 0.4 | 3 | Schedule mutual documentation evaluations throughout each sprint |
| Application will not meet expectations of the Sponsor | 0.2 | 1 | Acquire strong mutual understanding of final application prior to beginning construction |
| Sponsor augments final application expectations | 0.15 | 2 | Retain record of Sponsors expectations for the final product |
| Project is too small to meet class standards | 0.3 | 1 | Correspond with Dr. Yeager before beginning construction to ensure the success of the project |

**8 Bibliographical references**

Insert here a list of references, including both software engineering references and application-area references. This should include your textbook, at least two additional non-web sources, and at least two web sources. Use the MLA style guide (<http://www.lib.washington.edu/help/guides/44mla.pdf>) to determine proper form, for all categories of sources including books, articles, and web sites.