Spreadsheet Reformatting Tool

Report #3

FHSU CSCI 441 Fall 2019

Group Members: John Mullane, Jeremy Pogue, Josh Lewis, Taylor Zwiebel

Project Website: **https://sites.google.com/view/csci441vaf19-hrisreportmanagem**
Github: **https://github.com/jepogue/HRIS-Report-Management**

Individual Contributions Breakdown

Report Section	John Mullane	Jeremy Pogue	Taylor Zweibel	Josh Lewis	Points
• Summary of Changes	100% (5)				5
1. Customer Statement of Requirements	100% (6)				6
2. Glossary of Terms	100% (4)				4
3. System Requirements		33% (2)	33% (2)	33% (2)	6
4. Functional Requirements Specification	23% (7)	37% (11)	37% (11)	3% (1)	30
5. Effort Estimation				100% (4)	4
6. Domain Analysis		40% (10)		60% (15)	25
7.a. Interaction Diagrams	33% (10)		33% (10)	33% (10)	30
7.b. Design Patterns		100% (10)			10
8. Class Diagram and Interface Specification		33% (3.33)	33% (3.33)	33% (3.33)	10
8.b. OCL Contract Specification			100% (10)		10
9. System Architecture and Design	71% (5)	29% (2)			7
10. Algorithms and Data Structures					n/a
11. User Interface Design and Implementation	45% (5)			55% (6)	11
12. Design of Tests		33% (4)	33% (4)	33% (4)	12
14. History, Current, and Future Work	100% (5)				5
15. References	100% (5)				5
PROJECT MANAGEMENT	50% (6.5)	17% (2.17)		33% (4.33)	13
TOTAL:	**30.3% (58.50)**	**23.1% (44.50)**	**20.9% (40.33)**	**25.7% (49.67)**	**100% (193)**

Contents

Summary of Changes

Due to time constraints, several key items have been slated for future development, should the customer choose to pursue. After speaking with the customer, we have determined that the core of their business problem is the time consuming process of repeatedly modifying files in the same manner. The manual process also opens the process up to error. By eliminating the root of the problem (manual file modification), this will satisfy the customer's needs at this time. The average user will still be able to perform a larger volume and cleaner work by using the program.

1. Project objectives
    a. Auto-Run Template - The additional functionality of scheduling an auto-run template has been removed as it cannot be completed based on the time frame allotted for the completion of this project.
    b. Email Modified file - this has also been removed due to time constraints
    c. File comparison tool - this was deemed unnecessary and was removed
2. Use case descriptions
    a. Use cases all remain intact, but those pertaining to the above requirements are now marked as future development.
3. User Interface
    a. Rather than a sequence of tabs, the core piece of the user interface for file modification (delete, rename, etc.) will be held in one screen, allowing the user to manipulate each column with the modification options applying to each column. Upon further work with the client, it was determined that it was very beneficial to allow the end user to see all the pending modifications to each column in the same view.
4. System design
    a. The design of the overall system remains unchanged with the exception of the removal of the “scheduler”, “emailer”, and “comparer” modules. These were all designed to be separate functionalities that were independent of each other, as well as of the core functionality. Therefore, the system design did not require any re-working.

Customer Problem Statement

a. Problem Statement

We have a Human Resources department with hundreds of employees using various 3rd party software to administer the different programs in the employee life-cycle. Each one of these software has some reporting functionality. Some programs have limited capability, and produce reports that must be modified to meet the needs of the business. Other programs have very robust reporting tools within the platform, and require a great deal of working knowledge in order to completely format the report to what is needed for the business purpose. Additionally, many employees are using multiple programs each day and do not know the intricacies of each program's reporting capabilities. Often, what the end-user needs is not exactly what the given software will produce (or it would take a great deal of time and knowledge to produce the desired result). As a result, our end-user employees manually manipulate reports in Microsoft Excel in order to get what they need on a daily, weekly, monthly, or quarterly basis. Some of our employees have to manipulate a spreadsheet each day for their business needs.

Some programs, due to the complexity of reporting and the sensitivity of the underlying data, our business only allows a small number of people to create reports. For example, our HRIS program contains sensitive employee information and so we limit the reporting capabilities to just four individuals. Furthermore, the HRIS program is very capable of producing reports in a manner that will output in exactly the format the end-user requires, but the reporting tool is quite complex and the average end-user cannot be expected to possess this knowledge just for a few reporting needs (nor could we afford to train them or use their time in this manner). Due to the limited number of people that are given access to the HRIS reporting tool, the number of customization requests by individuals is necessarily limited. If a customer needs a basic customization, such as the removal of a field, or the filtering of certain employees, they are denied a custom report, and must do the formatting themselves.

As discussed, for various reasons we have employees using Microsoft Excel for final formatting needs. This is not usually a matter of just putting in new data and having a chart or table automatically update. Most commonly, the end-user needs to delete some columns, filter some columns based on some criteria, sort the data on other criteria, and so forth. And, the employees are doing this on a recurring basis, sometimes daily. This is quite a was of time for the employees, even if it is only a matter of minutes. Their job is not to spend time formatting Excel spreadsheets. Some of the very tech-savvy employees have managed to write Microsoft Excel "macros" in order to complete this repeated formatting automatically. We asked the employees if they could apply this method for others, but it is too complicated and time consuming to write one single macro (there is programming involved in the background of Excel), and cannot be easily taught to others. Additionally, even those employees with the Excel "macros" still need to download the reports and run their program in order to get the final formatting complete.

We have already partially eliminated one step in the process, the downloading of the report for the end-user. Previously, users would need to go into their respective programs on a recurring basis and either download a report already produced, or need to wait for a new report to generate and then download (as there is no scheduler in some systems). Some of our software allows us to schedule reports to be emailed or sent via FTP or retrieved via API. This has helped eliminate some of the time spent with the daily retrieval of reports from the various systems, as we have scheduled the applicable reports to be moved automatically to their respective network locations for end-user retrieval.

In summary thus far, we have end-users of reports retrieving reports in spreadsheet format from a centralized repository and completing manual formatting of the files in Microsoft Excel. They are completing the same tasks on a recurring basis, as frequently as daily. Due to barriers of knowledge, and restrictions of the time of our subject matter experts, we do not have a way to eliminate these repetitive tasks. As such, we are coming to market for a software solution to our problem.

What this company needs is a software that will perform the repetitive formatting tasks that each end-user needs, without necessitating a high degree of time or training. Ideally, the software will perform these tasks automatically, on whatever schedule the end-user dictates. The software should allow the user to save multiple "templates" of the required formatting, as oftentimes employees have more than one special formatting need. This program must be as user-friendly as possible - we do not expect our employees to be able to do any sort of programming to setup these formatting tasks and schedules.

This is not a new problem for us. We have previously compiled a list of common tasks and ideal solutions from employees that are tasked with manually manipulating reports for formatting purposes.

For most applications, we foresee the reports in the centralized repository containing more fields/columns than are necessary. We will do this in order to minimize the maintenance and customization requirements for the applicable subject matter experts. Commonly, the employees that are experts in a specific reporting tool are asked to add or remove a field, which is not a good use of their time. However, we cannot, for various reasons discussed above, allow the end-user to modify or create reports themselves. We have had employees attempt to create basic Excel “macros” by using Excel’s built-in “macro recording” functionality. However, this fails if the report is ever modified. Adding or removing an additional column throws the whole thing off. We need the proposed software to dynamically delete a field/column based on the column name, and not simply on the location/ordering of the column within the spreadsheet. Furthermore, if this column name is changed or removed, we would expect the program to alert the end-user, and not simply skip the step or delete the wrong column (as was happening with our users home-grown “recorded macros”). We would like the user to be presented with a visual representation of the fields/columns that are in the report, and they should be able to deselect any fields they do not wish to have. A preview of the data within that field would also be nice to have. The program should delete those fields upon running.

Another common task employees perform is sorting and filtering. Different users care about different pieces of information, and different populations of people or things within that data that is contained in a large report. Our employees need to be able to filter the data by one or more fields/columns and subsequently sort that data by multiple columns. The output report should not contain the data that was “filtered out”.

Re-ordering the fields/columns in the report is something else our end-users find themselves doing on a regular basis. Frequently, there is a unique identifier located in one column that needs to be either the first column in the report or some other specific column depending on the final usage. Additionally, users have reported that they often need to rename the column headers in order to import into a system or otherwise make them more readable for their particular audience. Furthermore, with users importing into other systems, they are always having to save the files as a ".csv" before importing, and would like to have that step eliminated. Some users prefer to save a copy of both the "csv" and the Excel file, while other users have no need for a copy in an excel format.

Continuing along the lines of importing and dealing with other systems: users have reported that they are often merging reports from two different systems. For example, we have a user that takes data from the HRIS platform, and matches it with data from the Learning Management system in order to see which newly hired employees have completed their training. They are currently doing this by using a "vlookup" in Excel based on a unique identifier that is shared between the two systems. They would like to be able to have a program perform this operation automatically and just have to indicate which files to merge and which fields contain the unique identifier.

Once the formatting is complete, many employees are creating graphs and/or PDFs of the report output. They have asked for the ability to dictate what type of graph they would like and what data is applicable within the report, and have the graph be created automatically and either output as an image or into a PDF document. The report (without any graphs) should also be available to be exported into PDF.

Almost always, all of the work done thus far is not just for the employee's personal use. It is to be shared with other employees. Users are sometimes emailing their report or saving a copy to a shared location, or both.

To put this all into context, the following is a specific example we received of one user's daily process which contains most of the company's needs with regard to the proposed software solution:

Every morning I need to produce a report of the new hires and what training is required of them. I login and go to the report repository to retrieve two different reports. One report is from our HRIS and one is from our LMS. The report from the LMS contains the employees hired yesterday that need training. It has specific information about training classes that our HRIS does not keep on file. However, it doesn't have the employee's location and manager email, which I need in order to schedule training. That information is in a report from the HRIS,

and so I need to merge the two files together in order to get a listing that I can send to the managers at the various locations.
First, I open up the HRIS report and delete all the columns except the Employee ID, Employee Name, Manager Name, Manager Email, and employee location. Then, I put the LMS report into the same Excel file, and use "vlookup" to bring the HRIS information into the LMS report. After that, I take the merged report, and filter out any people in two specific locations, as they don't need to go through the training in those locations. Finally, I sort by location and then manager, so the managers can easily find their employees.

Before sending out, I quickly create a chart showing the amount of people needing training at each location, and take a screenshot so I can use it in a report I put together. Lastly, I make sure a copy is in a specific directory for later reference, and then I print it to PDF and email it to all the managers at a group email listing.

b. Glossary of Terms

.csv - (abbr) a comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text.

Employee life-cycle - an HR model that identifies the different stages a worker advances through in an organization and the role HR plays in optimizing that progress

HRIS program - a software or online solution that is used for data entry, data tracking and the data information requirements of an organization's human resources (HR) management, payroll and bookkeeping operations. A HRIS is usually offered as a database.

Learning Management system - a software application for the administration, documentation, tracking, reporting, and delivery of educational courses, training programs, or learning and development programs.

LMS - (abbr) Learning Management System

Macros - an automated input sequence that imitates keystrokes or mouse actions. A macro is typically used to replace a repetitive series of keyboard and mouse actions and are common in spreadsheet and word processing applications like MS Excel and MS Word.

Recorded macros - a macro created by a piece of software that records user actions for playback at a later time.

Subject Matter Experts - an individual with a deep understanding of a particular process, function, technology, machine, material or type of equipment.

Unique identifier - any identifier which is guaranteed to be unique among all identifiers used for those objects and for a specific purpose.

System Requirements

c. Enumerated Functional Requirements

Requirements	Priority	Description
REQ-1	5	User can create and save a template of formatting rules to be assigned to the chosen file
REQ-2	5	User can select a file for the system to import for reformatting.
REQ-3	5	User can delete columns
REQ-4	5	User can rename column headers
REQ-5	5	User can rearrange columns
REQ-6	5	User can choose to sort data by chosen criteria within a template before the report is reformatted.
REQ-7	5	User can filter the data based on one or more column values
REQ-8	5	User can merge two reports within the system based on a unique identifier
REQ-9	5	User can save formatted file to local drive
REQ-10	4	User can designate the output file after formatting has been run. This includes: name, save-location, and filetype.
REQ-12	3	User can group data by columns, similar to a “Pivot Table” in Microsoft Excel

REQ-15	3	User can choose to keep a copy of the original, or save only the newly formatted report.
REQ-18	2	Saved templates should be allowed to be edited by user for single-use runs without being altered (template will remain as is after single-use edit)
REQ-19	2	User will receive a prompt to pick a template when opening a file.

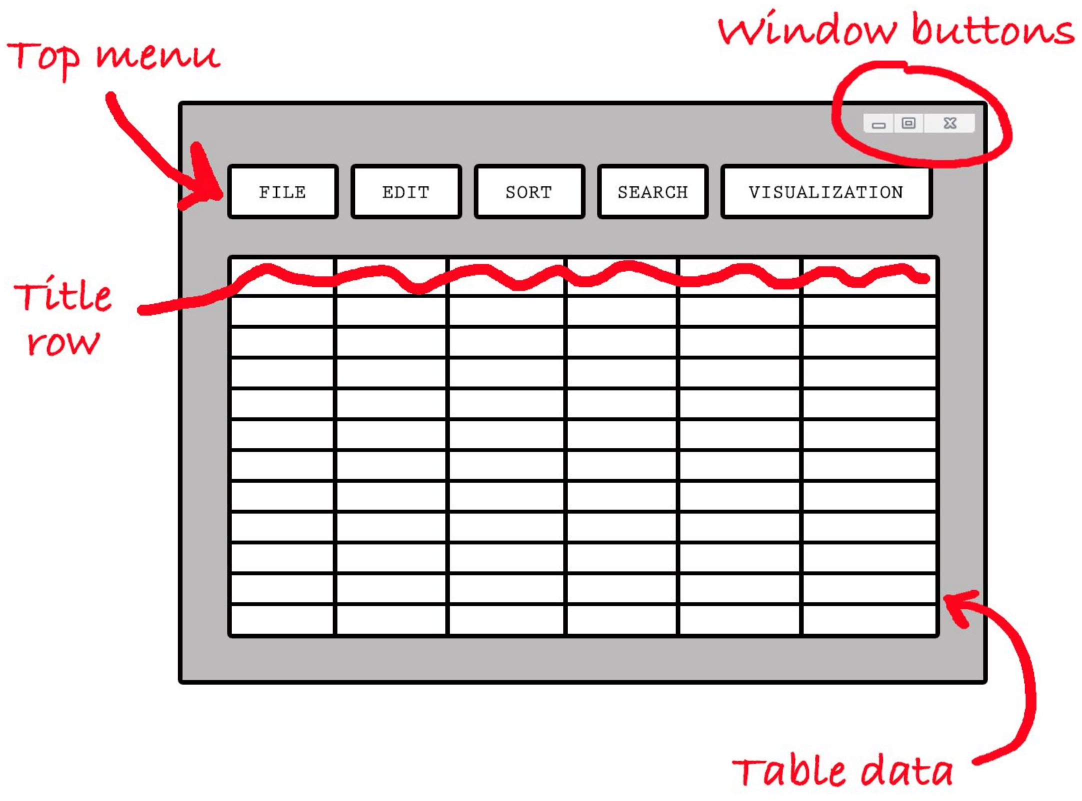
d. Enumerated Non-Functional Requirements

Requirements	Priority	Description
REQ-23	5	Program run time for templates should not exceed the amount of time it takes to manually perform the formatting
REQ-24	3	Program should be able to run while user is actively using other programs, and not substantially consume system resources
REQ-25	4	Program should be easy to install or access if a service
REQ-26	5	The average office employee should be able to operate the program without training
REQ-27	3	The program should be able to be serviced centrally or redeployed/updated as needed
REQ-28	5	Program should operate reliably, and without interruption

e. User Interface Requirements

Requirements	Priority	Description
REQ-29	3	Initial screen features simple step-by-step instructions for how to use the application.
REQ-32	3	Preview of the file/data will be available to the user when setting up or modifying the template

Interface Design Mock-Up



Notes regarding user interface requirements:

In speaking with the customer, it was clear they were concerned about usability from the standpoint of the average office employee. The customer expressed the desire to maintain similar interactivity with the program, similar to Microsoft Excel, as that is what the employees are using currently. In discussing further the range of capabilities with the customer, we noted that visual interactivity may not be essential as these tasks will no longer be repeated and thus the speed at which the user can setup the template is no longer such an important factor. The customer may be willing to sacrifice the similarity of interaction to Microsoft Excel in order to cut development time/costs. The overriding factor is that anyone that works in the company can easily use this program without extensive training.

Functional Requirements Specification

f. Stakeholders

End-User employee (EU):

Employees that do not have extensive knowledge of the reporting tools of the 3rd party software they use will need the program to reformat spreadsheets as needed

Reporting Subject-Matter Expert employee (SME):

Employees with extensive knowledge of reporting tools of 3rd party software that are responsible for providing End-User employees with custom reports will need this program in order to eliminate some of their workload while satisfying their internal customers' requests

Managerial employees:

Managers have an interest in the amount of time and effort the program can save their employees (both EU and SME employees)

g. Actors and Goals

End-User Employee (EU) - an initiating actor that has the goal of reformatting a spreadsheet

Template Storage - a participating actor that stores the user's saved templates

Temp File Storage - a participating actor that stores a temporary copy of the original report to be reformatted

h. Use Cases

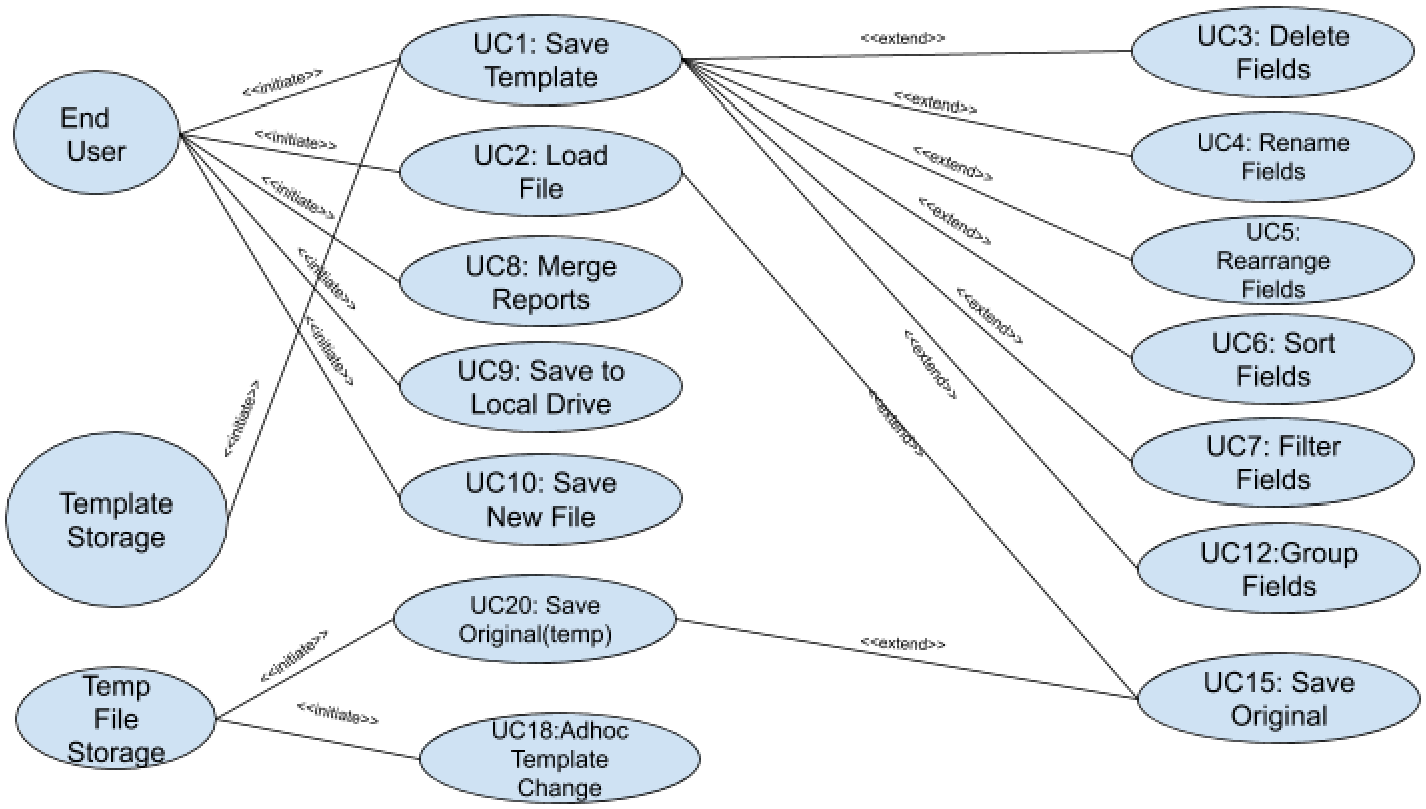
i. Casual Description

Use Case Identifier	Title	Casual Description
UC-1	Save Template	After setting up all reformatting, the user can save the settings as a template. The template can be assigned to a specific file name so it will always be applied to that file.
UC-2	Load File	User selects a file for reformatting from the file repository created by the company. The file is copied into the program and is ready for reformatting.
UC-3	Delete Fields	User can delete specific columns/fields.
UC-4	Rename Fields	User can rename column header(s).
UC-5	Rearrang e Fields	User can rearrange column order.
UC-6	Sort Fields	User can sort data from one or more columns.
UC-7	Filter Fields	User can filter data in one or more columns. The filtering will allow the user to select one or more of the values within that specific column, or allow the user to put in a partial search key. In example, a user wants to filter a field with job titles for anything that contains "sales". The user can enter in "sales" and filter for any records that contain that data, even if no job title is only marked as "sales"
UC-8	Merge Reports	User can merge two reports together by means of a unique key. They select the initial file and then another file with the same unique key. The user can merge the two datasets to either join them completely (includes all records regardless of a match), or only merge the matching records.
UC-9	Save to Local Drive	User can save their newly formatted file to their local user drive
UC-10	Save New File	After formatting of a file is complete, the user will choose where the newly formatted file should be saved, the filename and the file format
UC-11	Schedule Template	The saved templates can be scheduled to run automatically to reformat the reports to which they are assigned. The user will have no interaction after this is setup. The user would need to indicate

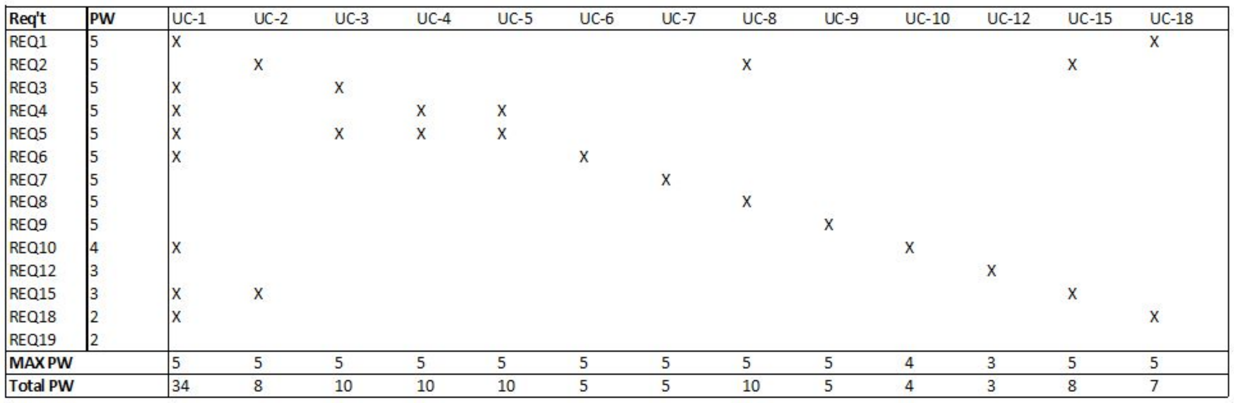
		the save location, filetype, and the original filename which is to be modified. This has not been implemented at the time of the final demo. This may be implemented at a later date.
UC-12	Group Fields	User can group data by columns. For example, there are 12 employees with 3 different locations among them. The user can choose to group the employees by state, and count the result. This is similar to a "pivot table" in excel.
UC-13	Compare Reports	User can load two files of the same format and the system will determine if the reports are the same. This has not been implemented at the time of the final demo. This may be implemented at a later date.
UC-14	Email File	User can choose to email the newly formatted file in addition to just saving the file to their local drive. The email address(es) would be part of the template and scheduling that is saved. This has not been implemented at the time of the final demo. This may be implemented at a later date.
UC-15	Save Original	User can choose to save a copy of the original file they copied for reformatting, in addition to the newly formatted file.
UC-16	Export to PDF	User can choose to export the reformatted report to PDF, and save to local drive and/or email. This has not been implemented at the time of the final demo. This may be implemented at a later date.
UC-17	Schedule Template Options	User can choose to schedule templates to run on various recurring bases. For example, weekly on Mondays, first of every month, last Friday of every month, etc. This has not been implemented at the time of the final demo. This may be implemented at a later date.
UC-18	Adhoc Template Change	User can edit a saved template without saving the new changes. For example, the want an additional column deleted, but they do not want to effect their saved template.
UC-19	Template Prompt	User will be prompted to open a file for formatting before beginning the formatting "wizard". This is optional, as users could be opening the program to modify a template. This has not been implemented at the time of the final demo. This may be implemented at a later date.
UC-20	Save Original (temp)	System will save the original file temporarily in the event that the user wants to revert to the original file. User will be given the option to revert back to the original file. This has not been implemented at the time of the final demo. This may be implemented at a later date.

UC-21	Download Link	User can choose to have an email sent containing a link to a downloadable file, rather than emailing the file directly. This has not been implemented at the time of the final demo. This may be implemented at a later date.
UC-22	Create Graph	User can create a graph or chart from the data and have this be included in their export or email of result. This has not been implemented at the time of the final demo. This may be implemented at a later date.

ii. Use Case Diagram



iii. Traceability Matrix



Req't	PW	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-12	UC-15	UC-18
REQ1	5	X												X
REQ2	5		X						X				X	
REQ3	5	X		X										
REQ4	5	X			X	X								
REQ5	5	X		X	X	X								
REQ6	5	X					X							
REQ7	5							X						
REQ8	5								X					
REQ9	5									X				
REQ10	4	X									X			
REQ12	3											X		
REQ15	3	X	X										X	
REQ18	2	X												X
REQ19	2													
MAX PW		5	5	5	5	5	5	5	5	5	4	3	5	5
Total PW		34	8	10	10	10	5	5	10	5	4	3	8	7

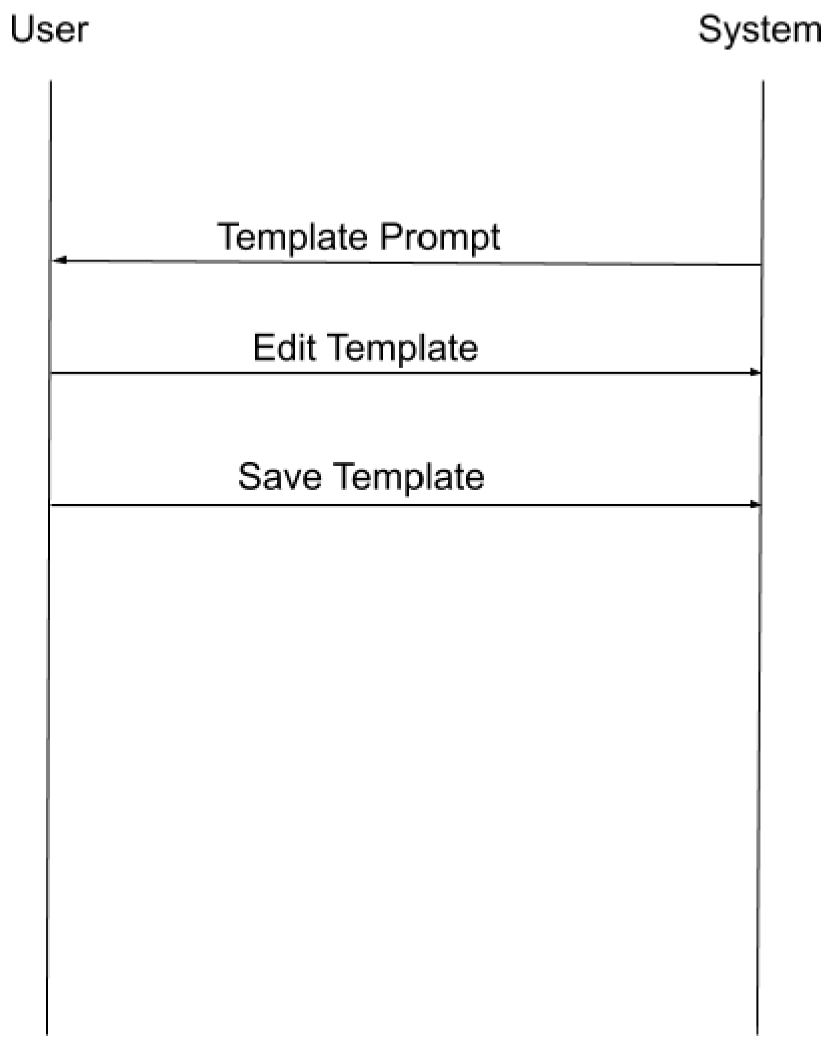
iv. Fully-Dressed Description

Use Case UC-1:	USE CASE Save Template
Related Requirements:	REQ-1, REQ-3, REQ-4, REQ-5, REQ-6, REQ-10, REQ-11, REQ-15, REQ-17, REQ-18, REQ-22
Initiating Actors:	End-User Employee
Actor’s Goals:	Create and Save a template for a spreadsheet to be reformatted
Participating Actor:	End-User Employee, Template Storage
Preconditions:	A spreadsheet has been loaded into the software
Postconditions:	The spreadsheet is formatted, and the template has been saved in the system.

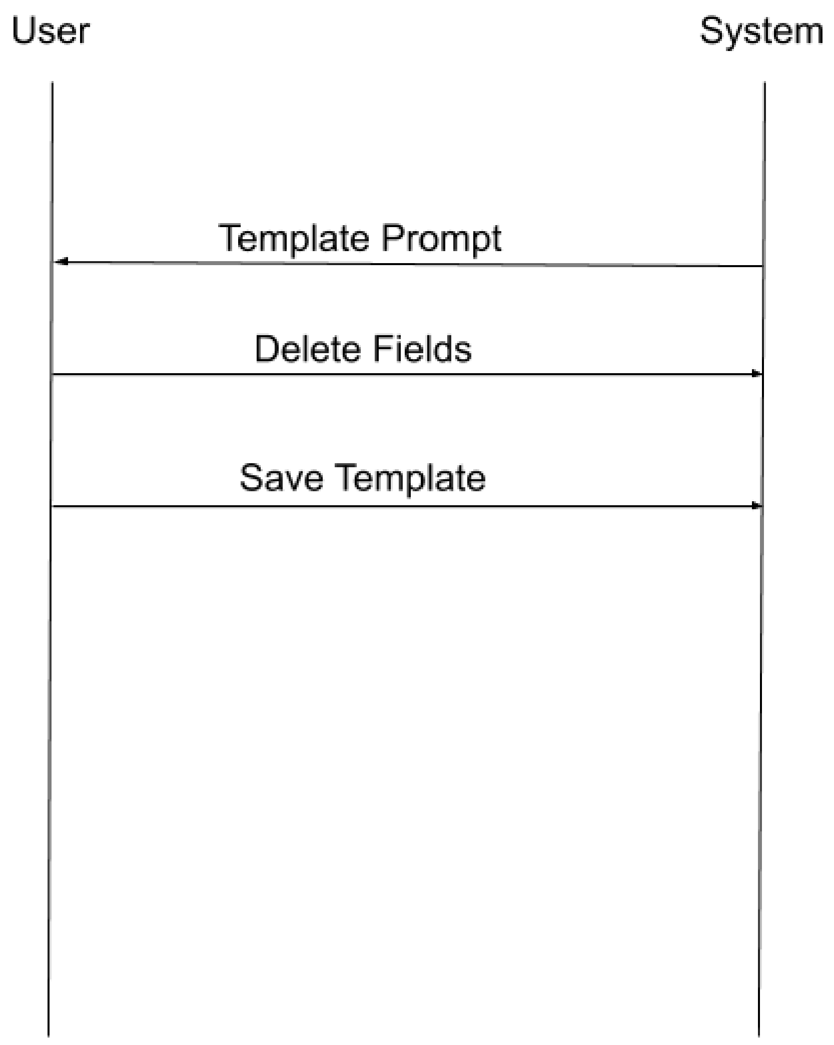
Flow of Events for Main Success Scenario: → 1. User makes changes to the report brought into the system. Including rearranging columns, sorting columns, deleting column, and filtering columns. Visuals may be added to the report as well. ← 2. System displays changes in real-time. → 3. When finished with editing the report, the user tells the system to save the report and template. ← 4. System prompts user for formatted report name, extension, save location. → 5. User selects save settings continues as prompted. ← 6. System prompts user to save the template for future use. → 7. User selects a name to save the template under and saves template, otherwise chooses not to save the template. ← 8. System displays successful save message, and allows user to bring in another report if the user chooses.
Flow of Events for Extensions (Initial File Not Found): → 1. User selects incorrect file or enters a corrupted file name. ← 2. System displays an error message and prompts the user to choose a file (*UC-2)*.

i. System Sequence Diagram

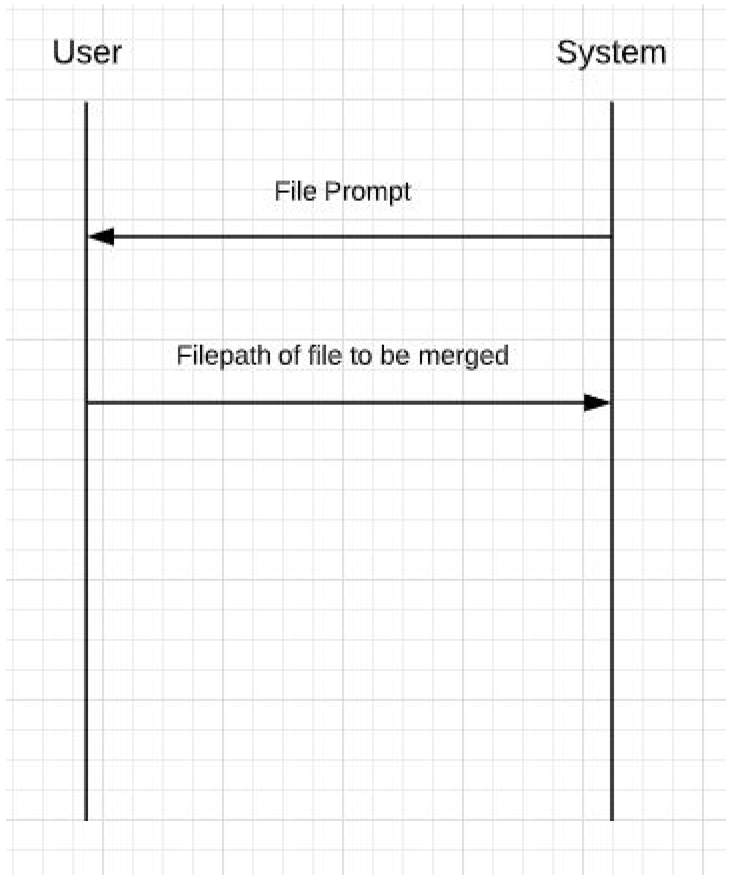
1. Save Template



2. Delete Fields



3. Merge Fields



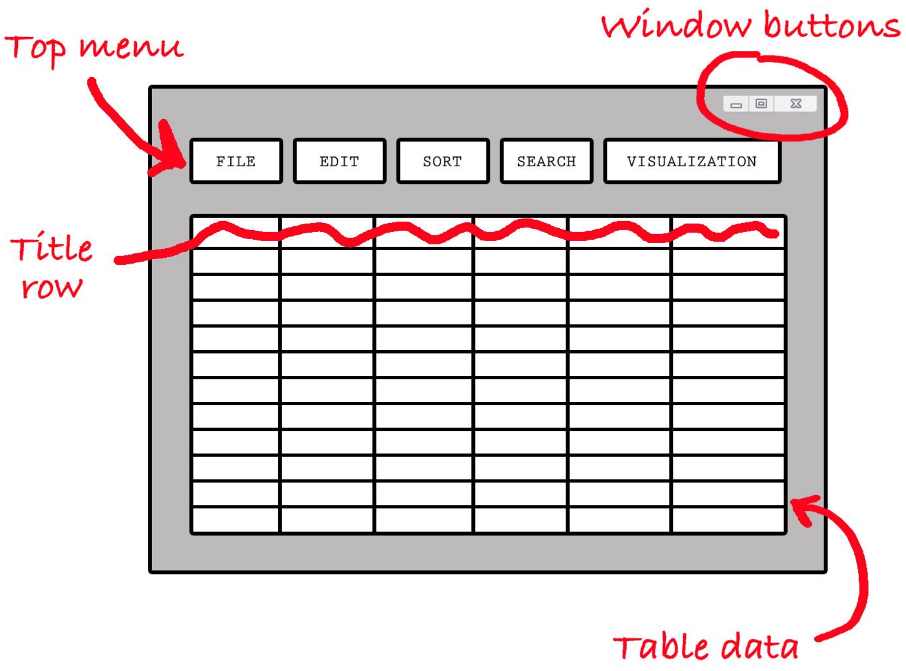
User Interface Specification

j. Preliminary Design

- The user's first action will be to select the "import" option.
- A file selection window will then open, allowing the user to browse for a file to import.
- File headers and top rows will be displayed to user for reference when making modifications
- Modification actions will be available to the user for each column/field
- More complex features like "group by", "pivot", "merge", and "save template" will have buttons to proceed with those actions.
- A "preview" function will be available to demonstrate the effect of the modification settings currently specified.
- The final product will then be exportable, with an option to either save the file as .csv or .xlsx

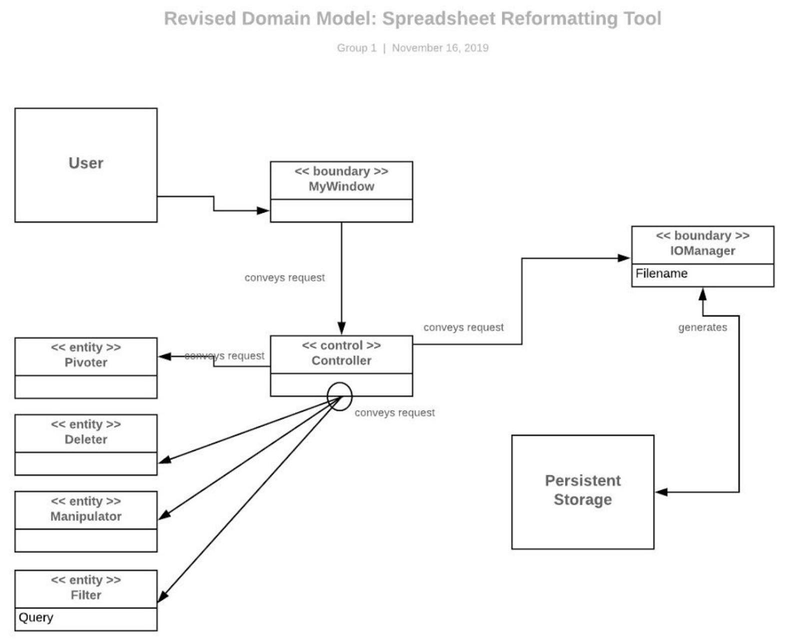
k. User Effort Estimation

- In top menu, click **File -> Import** OR Click the IMPORT button on the home screen, then browse local computer for csv or xls file to import
- Once a database cell has been selected, clicking **Edit** from the top menu to drop down the list of available actions (listed below)
- With a column selected, click the **Sort** button in the top menu to see a list of sorting options from the dropdown
- From the top menu, click the **Search** button to look for instances of a value in the database
- In the top menu, click **File -> Export -> Save As** to save a copy of the report to the local machine as a PDF
- In the top menu, click **File -> Close** OR click the "x" button in the top right corner to close the application

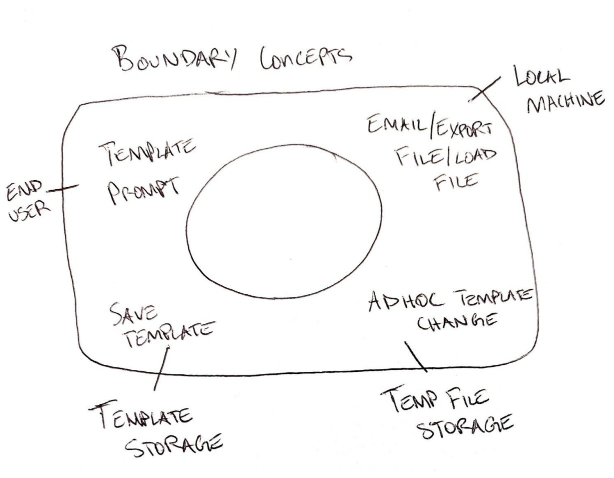

Top menu
Window buttons
FILE
EDIT
SORT
SEARCH
VISUALIZATION
Title row
Table data

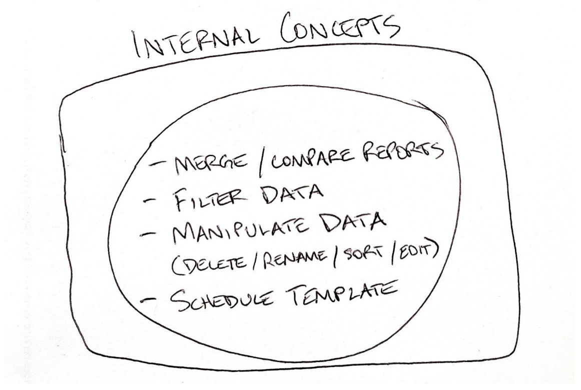
Domain Analysis

I. Domain Model



i. The domain model was first derived by drawing concept diagrams to determine boundary and internal concepts. From there, concept definitions were determined, followed by their respective associations. Below are the hand-drawn concept diagrams that preceded the following tables. These were merely the initial brainstorming terms, and are improved upon in the subsequent definitions.




INTERNAL CONCEPTS
– MERGE / COMPARE REPORTS
– FILTER DATA
– MANIPULATE DATA (DELETE / RENAME / SORT / EDIT)
– SCHEDULE TEMPLATE

1. Concept definitions

Responsibility Description	**Type**	**Concept Name**
Coordinate actions of all concepts associated with a use case, a logical grouping of use cases, or the entire system and delegate the work to other concepts.	D	Controller
Combine contents of selected reports and output the union of both data sets.	D	Combiner
Display relevant data based on user's filtering query (ie: cells containing specific string of characters, upper/lower boundary numeric values, or other conditional logic to be determined by user's needs).	D	Filter
Accommodate the manipulation of data contained in the imported table cells of reports. Actions include: modify, sort, delete, rename, cut, copy and paste.	D	Manipulator
Manages file input/output between the program and the local machine. This will facilitate the importing of reports, exporting of reports as files in various formats, and storing of temporary data.	D	IOManager

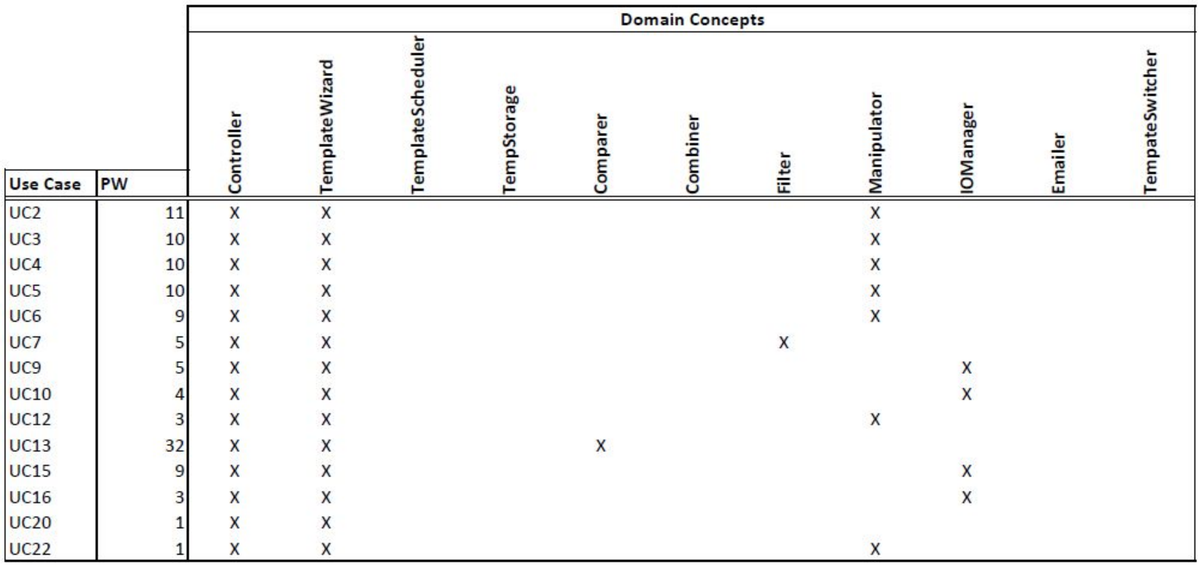
2. Association definitions

Concept Pair	**Association Description**	**Association Name**
Controller ←→ Combiner	Controller passes combine requests to Combiner	Conveys requests
Controller ←→ Filter	Controller passes filter requests to Filter	Conveys requests
Controller ←→ Manipulator	Controller passes modification requests to Manipulator	Conveys requests
Controller ←→ IOManager	Controller passes file I/O requests to IOManager, which imports data or generates a file export	Generates

3. Attribute definitions

Concept	**Attributes**	**Attribute Description**
Filter	Query	A given set of data for the Filter to drill down and focus on.
IOManager	Filename	The name and path of the file to both attach to an email or save into persistant storage.

4. Traceability matrix



		Domain Concepts										
Use Case	PW	Controller	TemplateWizard	TemplateScheduler	TempStorage	Comparer	Combiner	Filter	Manipulator	IOManager	Emailer	TempateSwitcher
UC2	11	X	X						X			
UC3	10	X	X						X			
UC4	10	X	X						X			
UC5	10	X	X						X			
UC6	9	X	X						X			
UC7	5	X	X					X				
UC9	5	X	X							X		
UC10	4	X	X							X		
UC12	3	X	X						X			
UC13	32	X	X			X						
UC15	9	X	X							X		
UC16	3	X	X							X		
UC20	1	X	X									
UC22	1	X	X						X			

m. System Operation Contracts

UC-1 Save Template -

Operation: Template Prompt
Preconditions: ▪ User has begun program
Postconditions: ▪ User has chosen a template to edit ▪ Use has chosen to create a template

Operation: Edit Template
Preconditions: ▪ User has selected a template to edit or is creating a new template ▪ User has a report or spreadsheet to import into the program
Postconditions: ▪ User has created a custom template using settings allowed by system

Operation: Save Template
Preconditions: · User has created a custom template using settings allowed by system · User has given this template a unique identifier (A name)
Postconditions: · Program stores template settings within persistent storage.

UC-2 Load File

Operation: Load File
Preconditions: · User has a spreadsheet or CSV file to load into the system · User knows location of stored file
Postconditions: · File is brought into the system.

UC-3 Delete Fields

Operation: Delete Fields
Preconditions: · User has brought a file into the system. · User has selected a given column to delete.
Postconditions: · Column is removed from spreadsheet..

UC-6 Sort Fields

Operation: Sort Fields
Preconditions: · User has brought a spreadsheet into the system. · User has selected a field to sort by with sort radio button.
Postconditions: · Fields are sorted based on user input.

Project Size Estimation

Unadjusted Actor Weight breakdown:

Actor	Description	Complexity	Weight
User	Interacts with system via GUI	Complex	3
Template Storage	Gives and takes input from system	Average	2
Temporary File Storage	Holds file in case user needs it restored	Simple	1

Unadjusted Actor Weight (UAW)= 6 = 1*2 + 1*3 + 1*1

Unadjusted Use Case Weight breakdown:

Use Case Identifier	Title	User Interface Complexity	Steps for Success	Number of Participating actors	Category	Weight
UC-1	Save Template	Simple	1	3 - User, Template Storage, Local Machine	Average	10
UC-2	Load File	Simple	1	1 - User	Simple	5
UC-3	Delete Fields	Simple	1	1 - User	Simple	5
UC-4	Rename Fields	Simple	1	1 - User	Simple	5
UC-5	Rearrange Fields	Complex	1	1 - User	Complex	15
UC-6	Sort Fields	Average	1	1 - User	Average	10
UC-7	Filter Fields	Complex	1	1 - User	Average	10
UC-8	Merge Reports	Average	3	1 - User	Average	10
UC-9	Save to Local Drive	Simple	1	1 - User	Simple	5
UC-10	Save New File	Simple	1	1 - User	Simple	5
UC-12	Group Fields	Simple	1	1 - User	Simple	5
UC-15	Save Original	Simple	1	2 - User and Local Machine	Simple	5

UC-18	Adhoc Template Change	Simple	1	2 - User and Template Storage	Simple	5
UC-19	Template Prompt	Simple	1	1 - User	Simple	5
UC-20	Save Original (temp)	n/a	1	0	Simple	5

Unadjusted Use-case Weight (UUCW) = 105= 10*5 + 4*10 + 1*15

Unadjusted Use-Case Points (UUCP) = 6+105 = 111

Technical Factor	**Description**	**Weight**	**Perceived Complexity**	**Calculated Factor (Weight x Perceived Complexity)**
T1	Distributed system	2	3	2*3 = 6
T2	Users expect good performance but nothing exceptional	1	3	1*3 = 3
T3	End-user expects efficiency but there are no exceptional demands	1	3	1*3 = 3
T4	Internal processing is relatively simple	1	1	1*1 = 1
T5	Requirement for reusability	2	0	2*0 = 0
T6	Ease of install is moderately important	0.5	3	0.5*3 = 1.5
T7	Ease of use is very important	0.5	5	0.5*5 = 2.5

T8	No portability concerns	2	2	2*2 = 4
T9	Easy to change minimally required	1	1	1*1 = 1
T10	Concurrent use is not required	1	0	1*0 = 0
T11	Security is not a significant concern	1	1	1*1 = 1
T12	No direct access for third parties	1	0	1*0 = 0
T13	No unique training needs	1	0	1*0 = 0
Technical Factor Total:				23

TCF (Technical Complexity Factor) = C1 + C2 * SUM(Calculated Factors) =
0.6 + (0.01 * 23) = 0.83

Environmental factor	Description	Weight	Perceived Impact	Calculated Factor (Weight * Perceived Impact)
E1	Beginner familiarity with the Python-based development	1.5	1	1.5*1 = 1.5
E2	Some familiarity with application problem	0.5	2	0.5*2 = 1
E3	Some knowledge of object-oriented approach	1	2	1*2 = 2
E4	Beginner lead analyst	0.5	1	0.5*1 = 0.5
E5	Highly motivated, but some team members	1	4	1*4 = 4

E6	Stable requirements are expected	2	5	2*5 = 5
E7	No part-time staff will be needed to operate	-1	0	-1*0 = 0
E8	Programming language of average difficulty will be used, but not seen by the user	-1	3	-1*3 = -3
Environmental Factor Total:				11

ECF (Environmental Complexity Factor) = C1 + C2 * SUM(Calculated Factors) =
1.4 + (-0.03 * 11) = 0.938

Final Calculations:

UCP = UUCP * TCF * ECF = 111 * 0.83 * 0.938 = 86.41794 —> **86 Use Case Points**

Duration = UCP * PF = 86 * 28 hours = **2,408 hours**

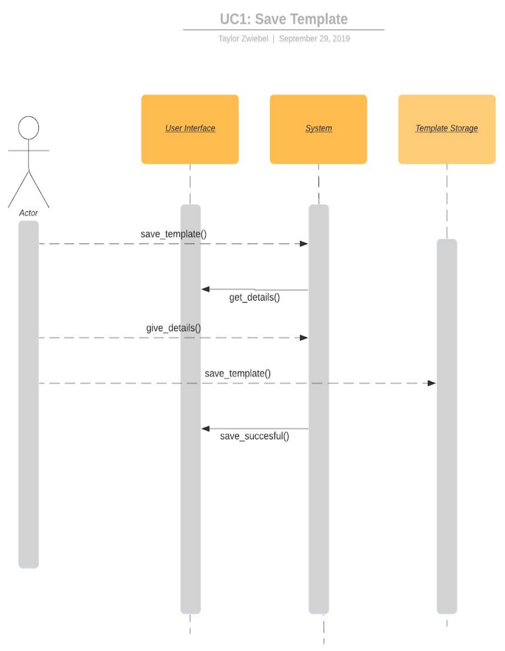
Product Ownership

The following list describes the product/feature ownership of this project. Each member will own that specific aspect of the project.

- **Select a file from the repository** - Jeremy Pogue
- **Save a template** - John Mullane
- **Merge with another file** - Taylor Zwiebel
- **Delete Fields** - Jeremy Pogue
- **Compare to prior report** - Taylor Zwiebel
- **Schedule Auto-Run** - Taylor Zwiebel
- **Email to myself or others -** Josh Lewis
- **Specify the output name of the file** - John Mullane
- **Save in a desired location** - Jeremy Pogue
- **Create a chart** - Josh Lewis
- **Export report/Chart, etc. to PDF** - Josh Lewis
- **Change file format to .csv or otherwise** - Jeremy Pogue
- **“Group By” / Pivot** - John Mullane

8. Interaction Diagrams

UC-1 Interaction Diagram



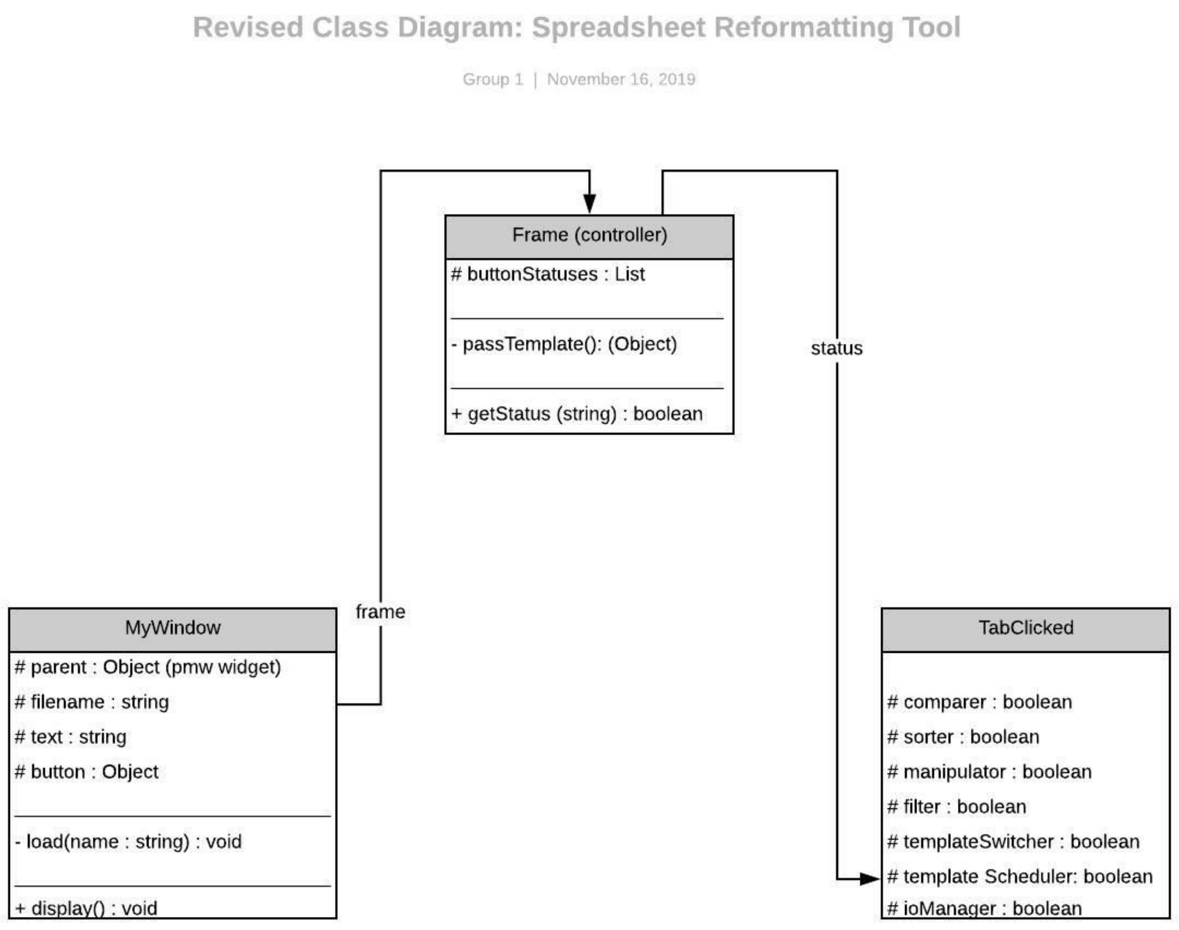
Traceability and Interaction Diagram #1

Our first interaction diagram details the process of UC-1, which entails a user saving a template they have created. It evolved from the first sequence diagram in the report. In this sequence the system prompts the user to create or edit a template. The user manipulates the template as needed and changes are saved within the system. From our domain model, this interaction requires the concepts of IOManager, TemplateWizard, and Controller. Furthermore, the classes involved with this interaction will be MyWindow, Frame, TabClicked, and Main. As extensions from these domain concepts.

The template class is an extension of the user interface's interaction with the "controller". The controller can take input from either a template, or the user interface. This "facade" design makes the program more robust and able to have the user interface modified for ease of use. A large part of the motivation for this program's creation was a more powerful user-interface, focused on the file modification ease. The implementation of the "facade" of the controller allows for this flexibility when needed.

9. Class Diagrams, System Architecture and Design

a. Class Diagram


Revised Class Diagram: Spreadsheet Reformatting Tool
Group 1 | November 16, 2019
Frame (controller)
# buttonStatuses : List
- passTemplate(): (Object)
+ getStatus (string) : boolean
status
frame
MyWindow
# parent : Object (pmw widget)
# filename : string
# text : string
# button : Object
- load(name : string) : void
+ display() : void
TabClicked
# comparer : boolean
# sorter : boolean
# manipulator : boolean
# filter : boolean
# templateSwitcher : boolean
# template Scheduler: boolean
# ioManager : boolean

b. Data Types and Operation Signatures

i. Window Class

The window class accesses the buttons, file, and text views and is responsible for window views. Essentially the window is the view to the user.

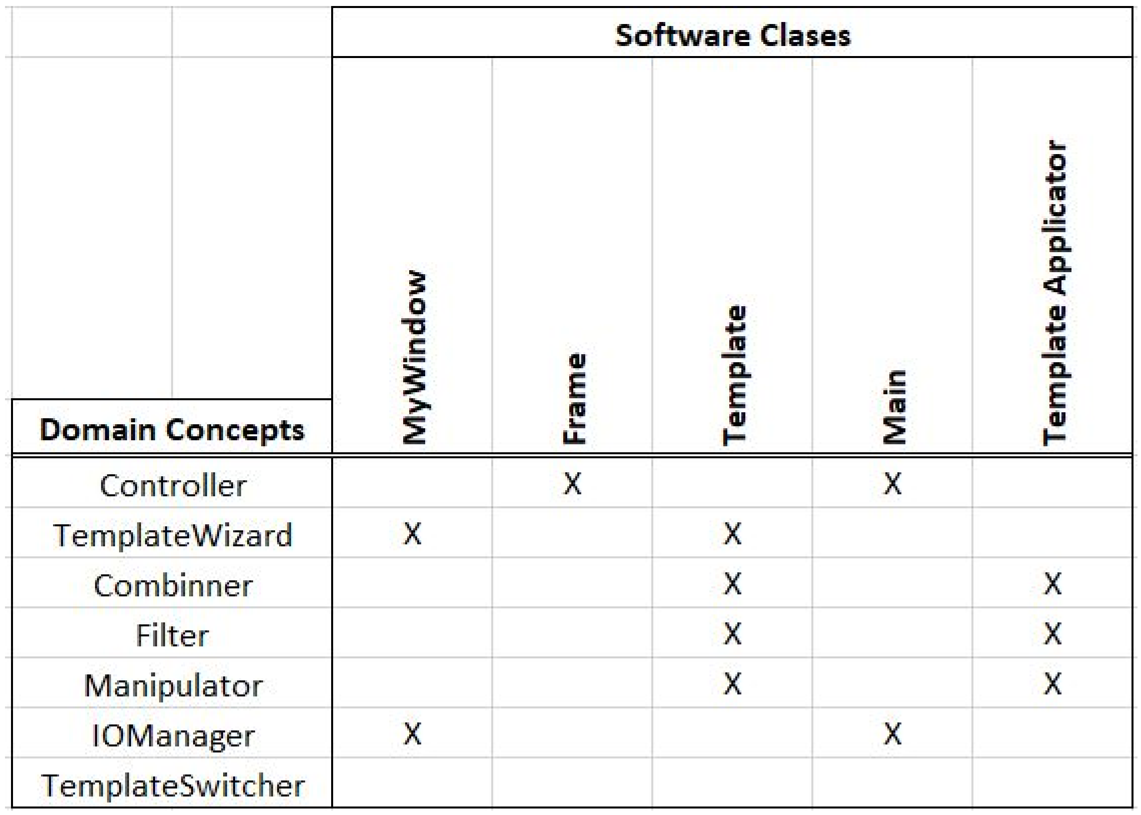
- parent: Object
  -The parent object such as a previous window.
- filename: string
  -The file names used for pulling files.
- text: string
  -Text that is viewable in the window.
- button: Object
  -Buttons used for different actions in the window
- load(name : string): void
  -Loads a file using the name of the desired template.
- display(): void
  -The window the user sees.

ii. Frame Class

The frame class acts as the controller to the system. It accesses the status of buttons and pulls the status to determine what to do next.

- buttonStatuses: List
  -Tells the statuses of buttons so the program knows what to do next.
- getStatus(string): boolean
  -Retrieves the current status of a button.

c. Traceability Matrix



	Software Clases				
Domain Concepts	MyWindow	Frame	Template	Main	Template Applicator
Controller		X		X	
TemplateWizard	X		X		
Combinner			X		X
Filter			X		X
Manipulator			X		X
IOManager	X			X	
TemplateSwitcher					

d. Design Patterns

i. The “facade” design pattern is used in this program. The individual components are passed into a “controller” or “facade” module. This “controller” module takes in all the individual inputs from the components and makes changes to the files. Rather than the individual modification components of the user interface making changes to the file directly, they interact with the “controller” which then applies the core modification functionalities based on the input within the user interface. This allows the user interface to be malleable. As long as the user-interface modules pass the same arguments into the controller function, the controller function will not need modification.

ii. Furthermore the controller uses a list of commands to run on the file being edited. This is a slightly different formatted version of the Command pattern, and is more of a functional pattern than object oriented; however, it works within the scope of the user’s input from the UX and interacting with the data underneath (the file). Essentially our functional commands are built within the “facade”.

e. Object Constraint Language (OCL) Contracts

1. delete_field
    a. context Window:: delete_field pre: self.df not empty
    b. context Window:: delete_field post: self.df.size equals self.pre.df.size - 1
2. filter_field
    a. context Window:: filter_field pre: self.df not empty
    b. context Window:: filter_field post: self.df[field_name] == filter_value
3. rename_field
    a. context Window:: rename_field pre: self.df not empty
    b. context Window:: rename_field post: rename_field is in self.df
4. sort_field
    a. context Window:: sort_field pre: self.df not empty
    b. context Window:: sort_field post: self.df equals self.df.sort()
5. rearrange_field
    a. context Window:: rearrange_field pre: self.df not empty
    b. context Window:: rearrange_field post: filed_name equals new_list.index(before_field + 1

10. System Architecture and Design

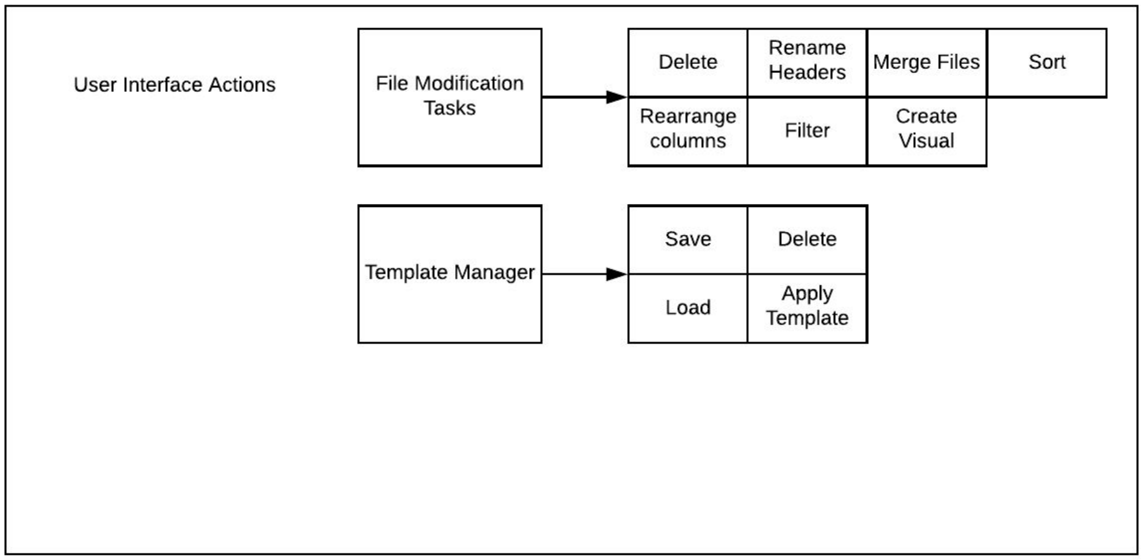
a. Architectural Styles

Our system is designed with the architectural style called “Component-based software engineering”. The program is mainly a collection of different functions that are to be applied to a report object (spreadsheet/table). As such, the different functions are implemented in isolation, as components. They all work to modify the given report object, upon the user’s discretion. Information from each modification task is stored temporarily in the program, and then later compiled with other modifications (i.e. deleted columns and the filters applied) and stored into a template for later use. Upon use of the template (or a scheduled run of a template), these individual parts are dispersed to the appropriate modules for altering the file.

b. Identifying Subsystems

Within our system, there is one route for the main application: user-directed actions via the interface.

The file modification tasks sub-system will make available to the user all the possible modifications to a file, to be performed one at a time by the user.



c. Persistent Data Storage

Template and schedule objects will need to be saved outside of the system for later access. Template objects will be stored in a flat file format. For easy storage and retrieval, the template files will be serialized and stored in the ".pickle" format. Schedules will also be stored for later access in the pickle format. The main program will load the schedule file, and subsequently, the template files will be opened as applicable.

d. Global Flow Control

Overall, the system can be seen as a linear flow, in viewing the procedure as "load file", "edit file", "save template". However, within the "edit file" step, there is an event-driven flow. Although the user is guided through the same sets of steps in setting up a file format, each user may perform the same task type multiple times (i.e. deleting multiple fields rather than just one). Additionally, users may freely navigate from one part of the guided editor to another, so there is no particular order for most of the file formatting. The system waits to respond to various user input.

e. Hardware Requirements

The program requires the following of the end-user's computer at a minimum. Larger files will require more working memory and the program may run slowly if only the minimum requirements are used.

i. CPU: 1GHz or faster
ii. Memory: 2GB
iii. Hard Disk: 100MB for installation
    1. Additional hard disk space if storing output files on personal computer and not network location
iv. Display: Any display supported by the individual computer

11. User Interface Design and Implementation

- Rather than a sequence of tabs, the core piece of the user interface for file modification (delete, rename, etc.) will be held in one screen, allowing the user to manipulate each column with the modification options applying to each column. Upon further work with the client, it was determined that it was very beneficial to allow the end user to see all the pending modifications to each column in the same view.

12. Design of tests

- Select a file from the repository
    - A specific file will be selected via system dialog and loaded into the program for modification.
- Delete field
    - A specific field will be selected to be deleted and then the file will be checked to be sure that the desired field has been deleted.
- Rename field
    - A specific field will be selected to have the header renamed and then the file will be checked to be sure that the outcome is correct.
- Sort field
    - A specific field will be selected to be sorted ascending/descending and then the file will be checked to be sure that the outcome is correct.
- Rearrange field
    - A specific field will be selected to be moved in the column order and then the file will be checked to be sure that the outcome is correct.
- Filter field
    - A specific field will be selected to be filtered on user input and then the file will be checked to be sure that the outcome is correct.
- Group by

    - A specific field will be selected to be used to perform a grouping and then the file will be checked to be sure that the outcome is correct.
        - I.e. “group by” location will group all elements by the field location, and count the resulting members
- Pivot
    - Specific fields will be selected to perform a “pivot” and then the file will be checked to be sure that the outcome is correct.
        - I.e. “pivot” location and gender will group by both location and gender and count the resulting members, showing the breakout of each gender for each location.
- Merge with another file
    - Two separate files will be merged based on a “key” field to be selected by the user. The resulting file will have all of the fields from both of the reports, merged on this “key” field.
- Save a template
    - The program will be able to load a template to be applied to a specified report file. This template will apply all changes specified as if the user had entered the changes manually via interface.

13. History of Work, Current Status, and Future Work

a. History of work

All items showing with strikethrough text were determined out of scope for this project following the first demonstration due to time constraints.

Our first demonstration was noted to include the core feature of template saving. However, the user interface design consumed a larger portion of our build time than anticipated. Therefore, only the core features of file manipulation were implemented at the time of the first demonstration.

For our first demonstration, a “working interface” with simplified modification inputs from the user was used. Final implementation will see a more robust and visually interactive interface as was displayed during demonstration one.

Below is our prior plan, with notes about actual development in bold/italic text below

Week 1: Build Core Functionality

Create a minimum viable product consisting of these features. The proximal features will not be addressed until these core features are operating. Items 1 and 5 are reused throughout all other features. Items 2-4 are standalone, but comprise the most base functionality, and without these features the system will not fulfill the customer's needs.

Saving a template is critical functionality, and so it will be demonstrated in Demo 1. However, there are many settings that will eventually be applicable to the template. Therefore, the most basic functionality will be created and then the template functionality will be applied to these most basic settings before moving to higher-level functionality.

1. Import File
2. Delete Fields
3. Sort Data
4. Group by Fields
5. Save new report
6. Save these settings to a template

Time permitting, move on to Core+ Functionality

Week 1 results: items 1 through 5 were completed during week 1, using the "working interface". Week 1 should have included items regarding the core user interface that houses all of these functions. Item 6 has yet to be implemented at time of first demonstration due to complexity and reliance on interface to determine implementation of template creation. Building for week 1 carried over into week 3, pushing timeline into test time.

Week 2: Build Core+ Functionality

Expanded basic features

7. Filter Data
8. Rename Column Headers
9. Merge two reports
10. Rearrange columns
11. ~~Create visuals~~
12. ~~Export to PDF~~
13. Add these Core+ features to the template save feature

Time permitting, move on to Final Functionality (week 7)

Week 2 results: items 7,8,10 were completed during week 2. Merging reports was repositioned for final functionality.

Week 3: Complete Testing

Complete test cases covering Core and Core+ functionalities above

Week 3 results: week 3 items started late due to pushed timelines in weeks 1 and 2. However, as these items were simple to test once inserted into the interface, testing was able to be completed.

Week 4: Revise Report for all changes

Review and edit documentation to align with all changes taken place

Week 4 results: this was completed successfully within the time allotted

b. Key accomplishments

- Core functionality has all been demonstrated and it is clear that we will have a viable solution for our client.
- The final form of the user interface has been decided upon. By working through initial iterations of the interface with the customer, it was determined that this final form of the interface will be a much better solution for its workforce.
- All timelines have been maintained when considering the change in scope. Initial requirements list put this as a 6 month project for a team of our size. Revised requirements have cut this down to a 3 month project, staying within the customer's timeline.

c. Future Work

The final two weeks of project building are upcoming, and are outlined below. Beyond this current project, the scheduling and auto-emailing functionalities can be built to bring further benefit to the client.

Week 5: Design Patterns, Object Constraint Language Contract

Complete design patterns and OCL contract

Start work on the final pieces of functionality

1. Form the new user interface to work with the individual modification components
2. Including all features into the template
3. Loading a template for editing
4. Applying a template to a file

Week 6: Final Functionality

Complete and test the following features in preparation for Demo 2

1. Including all features into the template
2. Loading a template for editing
3. Applying a template to a file

d. Breakdown of responsibilities

The following list describes the product/feature ownership of this project. Each member will own that specific aspect of the project.

The integration of the files will be conducted by all members of the team. Coordination of the GitHub repository will be owned by Jeremy Pogue. As modules are completed, they will be integrated into the main set of code by the owner of that particular module, with other team members contributing as necessary to ensure consistency of implementation. For final overall testing of the integrated codeset, we will test one another's functions, rather than our own, in an effort to discover any issues, and to better understand the interaction of the modules.

Select a file from the repository - Jeremy Pogue

Save a template - John Mullane

Merge with another file - Taylor Zwiebel

Delete Fields - Jeremy Pogue

Specify the output name of the file - John Mullane

Save in a desired location - Jeremy Pogue

Change file format to .csv or otherwise - Jeremy Pogue

"Group By" / Pivot - John Mullane

Sort data - Taylor Zwiebel

Filter data - John Mullane

Rearrange columns - John Mullane

Rename columns - Taylor Zwiebel

Run Template - Jeremy Pogue

14. References

1. (28) Tkinter - Create tabs in your GUI interface using Notebook - YouTube
   https://www.youtube.com/watch?v=CUf7fB8hIOU

2. 6 ways to Sort Pandas Dataframe: Pandas Tutorial — Python, R, and Linux Tips
   https://cmdlinetips.com/2018/02/how-to-sort-pandas-dataframe-by-columns-and-row/

3. Autorun a Python script on windows startup - GeeksforGeeks
   https://www.geeksforgeeks.org/autorun-a-python-script-on-windows-startup/

4. Component-based software engineering - Wikipedia
   https://en.wikipedia.org/wiki/Component-based_software_engineering

5. Extracting extension from filename in Python - Stack Overflow
   https://stackoverflow.com/questions/541390/extracting-extension-from-filename-in-python

6. Generating an excel report with python — Mourad Mourafiq
   https://mourafiq.com/2016/01/01/generating-excel-report-with-python.html

7. GUI Programming with Python: Text Widget
   https://www.python-course.eu/tkinter_text_widget.php

8. How to Use Pickle to Save Objects in Python
   https://www.thoughtco.com/using-pickle-to-save-objects-2813661

9. How would you make an 'undo' function in Python? - Quora
   https://www.quora.com/How-would-you-make-an-undo-function-i

n-Python

10. openpyxl-templates · PyPI
    https://pypi.org/project/openpyxl-templates/

11. python - Delete column from pandas DataFrame - Stack Overflow
    https://stackoverflow.com/questions/13411544/delete-column-from-pandas-dataframe

12. python - How do you create different variable names while in a loop? - Stack Overflow
    https://stackoverflow.com/questions/6181935/how-do-you-create-different-variable-names-while-in-a-loop

13. python - How to change the order of DataFrame columns? - Stack Overflow
    https://stackoverflow.com/questions/13148429/how-to-change-the-order-of-dataframe-columns

14. Python | Schedule Library - GeeksforGeeks
    https://www.geeksforgeeks.org/python-schedule-library/

15. Python GUI - tkinter - GeeksforGeeks
    https://www.geeksforgeeks.org/python-gui-tkinter/

16. Scheduling a Python script or model to run at a prescribed time
    https://www.esri.com/arcgis-blog/products/product/analytics/scheduling-a-python-script-or-model-to-run-at-a-prescribed-time/

17. Software architecture - Wikipedia
    https://en.wikipedia.org/wiki/Software_architecture#Architectural_styles_and_patterns

18. Tkinter tkFileDialog module - Python Tutorial
    https://pythonspot.com/tk-file-dialogs/

19. What is Package Diagram?
    https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/

20. What you need to know about Office 2016 before you install it - gHacks Tech News
    https://www.ghacks.net/2015/05/04/what-you-need-to-know-about-office-2016-before-you-install-it/