# Spreadsheet Reformatting Tool

Report #2

## 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

| Project Part | | Project Category | Team Member | | | |
|---|---|---|---|---|---|---|
| | | | John Mullane | Jeremy Pogue | Taylor Zwiebel | Josh Lewis |
| Responsibility % | 1 | Section 1a Customer Problem Statement | 100% | | | |
| | | Section 1b Glossary of Terms | 100% | | | |
| | | Section 2a Enumerated Functional Requirements | | 100% | | |
| | | Section 2b Enumerated Non-Functional Requirements | | | 100% | |
| | | Section 2c User Interface Requirements | | | | 100% |
| | | Project Management | 100% | | | |
| Respon | 2 | Section 3.a Stakeholders | 100% | | | |
| | | Section 3.b Actors and Goals | 100% | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| sibility % | | **Section 3.c.i Use Cases - Casual Description** | **100%** | | | |
| | | **Section 3.c.ii Use Cases - Use Case Diagram** | | | **100%** | |
| | | **Section 3.c.iii Use Cases - Traceability Matrix** | | **100%** | | |
| | | **Section 3.c.iv Use Cases - Fully Dressed Description** | | **100%** | | |
| | | **Section 3.d - System Sequence Diagrams** | | | **100%** | |
| | | **Section 4.a - User Interface Specification - Preliminary Design** | | | | **100%** |
| | | **Section 4.b - User Interface Specification - User Effort Estimation** | | | | **100%** |
| | | **Project Management** | **100%** | | | |
| Respon sibility % | 3 | **Section 5.a.1 - Concept Definitions** | | | | **100%** |
| | | **Section 5.a.2 - Association Definitions** | | | | **100%** |
| | | **Section 5.a.3 - Attribute Definitions** | | **100%** | | |
| | | **Section 5.a.4 - Traceability Matrix** | | **100%** | | |
| | | **Section 5.b - System Operation Contracts** | **33.3%** | **66.6%** | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | **Section 6 - Project Size Estimation** | **100%** | | | |
| | | **Section 7 - Plan of Work** | **100%** | | | |
| | | **Project Management** | **33%** | **33%** | | **33%** |

## Individual Contributions Breakdown REPORT #2

| Project Part | | Project Category | Team Member | | | |
|---|---|---|---|---|---|---|
| | | | John Mullane | Jeremy Pogue | Taylor Zwiebel | Josh Lewis |
| Responsibility % | 1 | Section 1 - interaction diagrams | 33% | | 33% | 33% |
| | | References | 100% | | | |
| | | Project Management | 33% | 33% | | 33% |
| | 2 | 2.a Class Diagram | | | 50% | 50% |
| | | 2.b Data Types and Operation Signatures | | | 100% | |
| | | 2.c Traceability Matrix | | 100% | | |
| | | 3.a Architectural Styles | 100% | | | |
| | | 3.b Identifying Subsystems | 100% | | | |
| | | 3.c Persistent Data Storage | 100% | | | |
| | | 3.d Global Flow Control | 100% | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | 3.e Hardware Requirements | 100% | | | |
| | | Project Management | 33% | | 33% | 33% |
| | 3 | 5. User Interface Design and Implementation | 100% | | | |
| | | 6. Design of tests | | | 100% | |
| | | 7.a Merging contributions from Individual Team Members | 100% | | | |
| | | 7.b Project Coordination and Progress Report | 100% | | | |
| | | 7.c Plan of Work | 100% | | | |
| | | 7.d Breakdown of Responsibilities | 100% | | | |
| | | 8. References | 100% | | | |
| | | Changes from 1st Report | | | | 100% |
| | | Traceability of classes to domain concepts | | 100% | | |
| | | Project Management | 33% | 33% | | 33% |

# Contents

# 1. 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.

# 2. System Requirements

## a. 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-11 | 4 | User can schedule templates to be applied to reports automatically |
| REQ-12 | 3 | User can group data by columns, similar to a “Pivot Table” in Microsoft Excel |
| REQ-13 | 3 | The system will allow users to compare two reports formatted from the same template. |
| REQ-14 | 3 | Formatted file can be emailed automatically upon completion by system |
| REQ-15 | 3 | User can choose to keep a copy of the original, or save only the newly formatted report. |
| REQ-16 | 3 | Formatted file can be exported to PDF by user. |
| REQ-17 | 3 | User can schedule the templates to run on a recurring basis (i.e. everyday, every Monday, first of each month, etc.) |
| 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. |
| REQ-20 | 1 | The system should store information in a report after a user chooses to delete it, so it may be regained after a template has been run. |
| REQ-21 | 1 | User can email their edited reports as a download link to other people or groups of people. |
| REQ-22 | 1 | User can create visual charts and data representations in the template. |

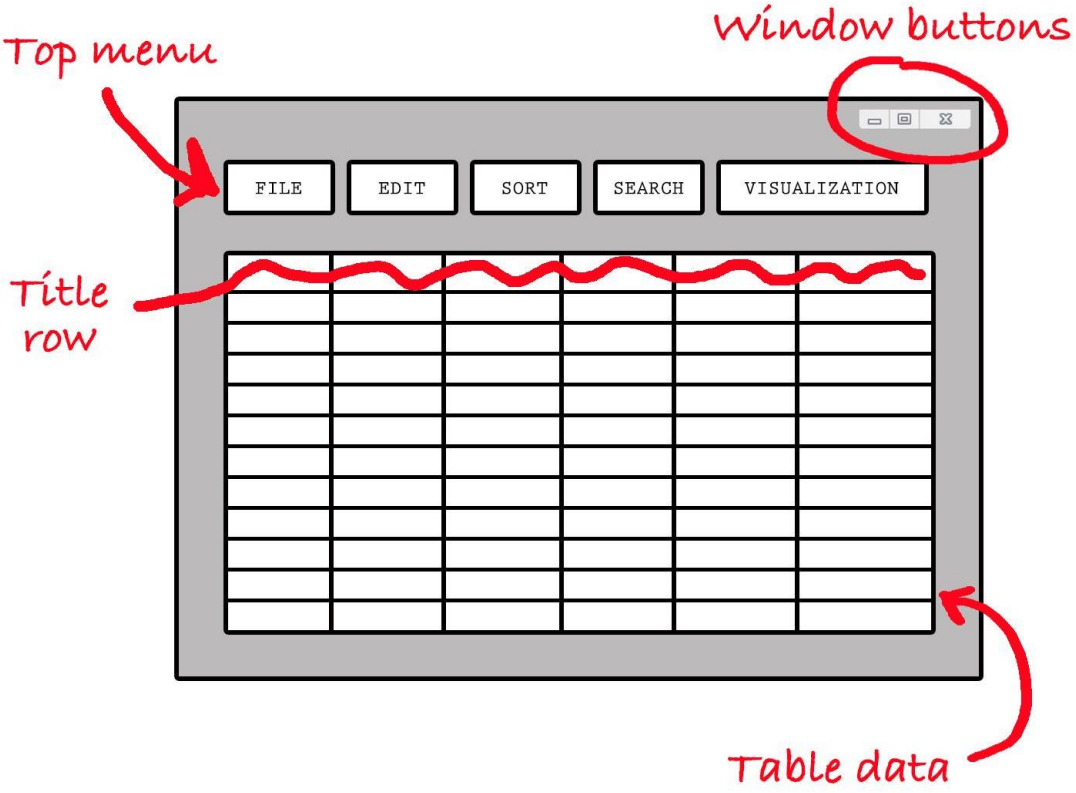
## b. Enumerated Non-Functional Requirements

| 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 to scheduling |

## c. User Interface Requirements

| Requirements | Priority | Description |
|---|---|---|
| REQ-29 | 3 | Initial screen features simple step-by-step instructions for how to use the application. |
| REQ-30 | 4 | Help text should be available for all non-self-explanatory user interactions with the program |
| REQ-31 | 5 | Program should guide the user through various steps (i.e. a “Wizard” type of program) |
| REQ-32 | 3 | Preview of the file/data will be available to the user when setting up or modifying the template |
| REQ-33 | 3 | Column headers can be renamed directly in the preview of the data/file |
| REQ-34 | 2 | Filtering and sorting is done interactively via user interface, similar to Microsoft Excel |
| REQ-35 | 2 | Columns can be dragged and dropped to re-order |
| REQ-36 | 5 | The Visualization Wizard tool will allow for a preview of the visual before exporting the image/document |
| REQ-37 | 5 | User should have a “template library” where they can view all their templates and the reports to which they apply |

**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.

# 3. Functional Requirements Specification

## a. 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)

## b. 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

## c. Use Cases

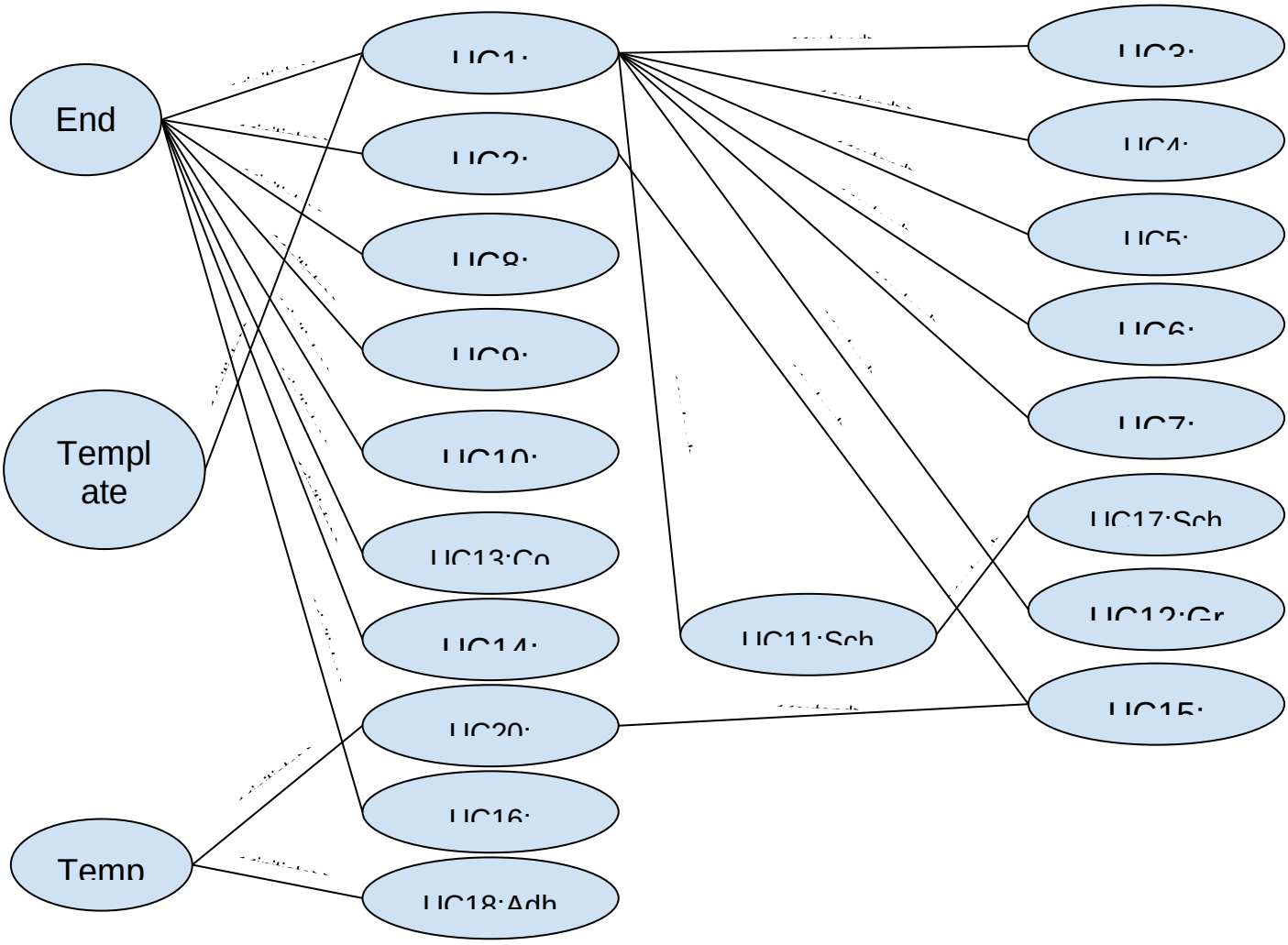
### i. Casual Description

| 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 | Rearrange 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. |
| 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 |
| 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. |
| 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 |
| 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. |
| 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. |
| 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 |
| 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. |

| | | |
|---|---|---|
| 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. |

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-11 | UC-12 | UC-13 | UC-14 | UC-15 | UC-16 | UC-17 | UC-18 | UC-19 | UC-20 | UC-21 | UC-22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REQ1 | 5 | X | | | | | | | | | | | | | | | | | X | | | | |
| REQ2 | 5 | | X | | | | | | X | | | | | | | 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 | | X | | | | | | | | | | | |
| REQ10 | 4 | X | | | | | | | | | X | | | | | | | | | | | | |
| REQ11 | 4 | X | | | | | X | | | | | X | | | | | | X | | | | | |
| REQ12 | 3 | | | | | | | | | | | | X | | | | | | | | | | |
| REQ13 | 3 | | | | | | | | | | | | | X | | | | | | | | | |
| REQ14 | 3 | | | | | | | | | | | | | | X | | | | | | | | |
| REQ15 | 3 | X | X | | | | | | | | | | | | | X | | | | | | | |
| REQ16 | 3 | | | | | | | | | | | | | | | | X | | | | | | |
| REQ17 | 3 | X | X | | | | | | | | | X | | | | | | X | | | | | |
| REQ18 | 2 | X | | | | | | | | | | | | | | | | | X | | | | |
| REQ19 | 2 | | | | | | | | | | | | | | | | | | | X | | | |
| REQ20 | 1 | | | | | | | | | | | | | | | X | | | | | X | | |
| REQ21 | 1 | | | | | | | | | | | | | | | | | | | | | X | |
| REQ22 | 1 | X | | | | | | | | | | | | | | | | | | | | | X |
| **MAX PW** | | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 3 | 3 | 3 | 5 | 3 | 5 | 5 | 5 | 1 | 1 | 1 |
| **Total PW** | | 42 | 11 | 10 | 10 | 10 | 9 | 5 | 10 | 5 | 4 | 12 | 3 | 3 | 3 | 9 | 3 | 12 | 7 | 7 | 1 | 1 | 1 |

### 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)*.

| **Use Case UC - 17:** | **USE CASE Schedule Template Options** |
|---|---|
| **Related Requirements:** | REQ-2, REQ-11, REQ-17 |
| **Initiating Actors:** | End-User Employee |
| **Actor's Goals:** | Set up a scheduled run for a specific template |
| **Participating Actor:** | End-User Employee, Template Storage |
| **Preconditions:** | 1. Template has been saved in template storage<br>2. Template's unformatted file is saved in a place accessible to the system. |
| **Postconditions:** | Formatted report is saved in a location known to End-User Employee, or emailed to employee(s) |

**Flow of Events for Main Success Scenario:**

→ 1. User chooses to set up run from menu in system.

← 2. System displays a list of templates for user to choose from for the scheduled runs.

→ 3. User selects desired template.

← 4. System prompts for user settings regarding the run. This include: frequency, time of scheduled run, source file location, and output location.

→ 5. User selects save settings.

← 6. System runs the template at the desired time and creates output file as the user wishes.

**Flow of Events for Extensions (Source File not Found):**

← 1. System prompts reaches for the source file and cannot find it. The system will log this attempt and alert the user the next time the user opens the application.

| | |
|---|---|
| Use Case UC - 14: | USE CASE Email File |
| Related Requirements: | REQ-14 |
| Initiating Actors: | End User Employee |
| Actor’s Goals: | Email a formatted file to himself/herself and/or other employees |
| Participating Actor: | End User Employee |
| Preconditions: | A formatted file has been created from a templated. |
| Postconditions: | The file is emailed to the destination users. |

Flow of Events for Main Success Scenario:

→ 1. User chooses email formatted file from the menu.

← 2. System displays files from templates saved within the system.

→ 3. User selects desired file.

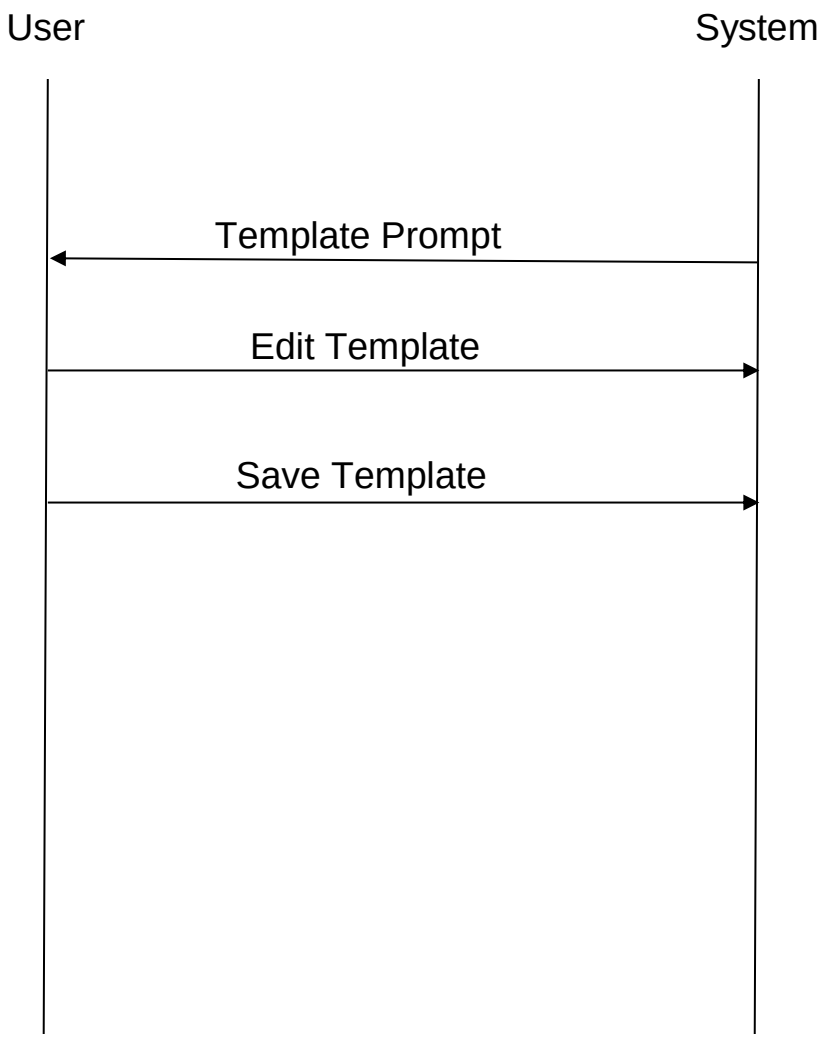
← 4. System prompts for email address(s)

→ 5. User enters appropriate email(s)

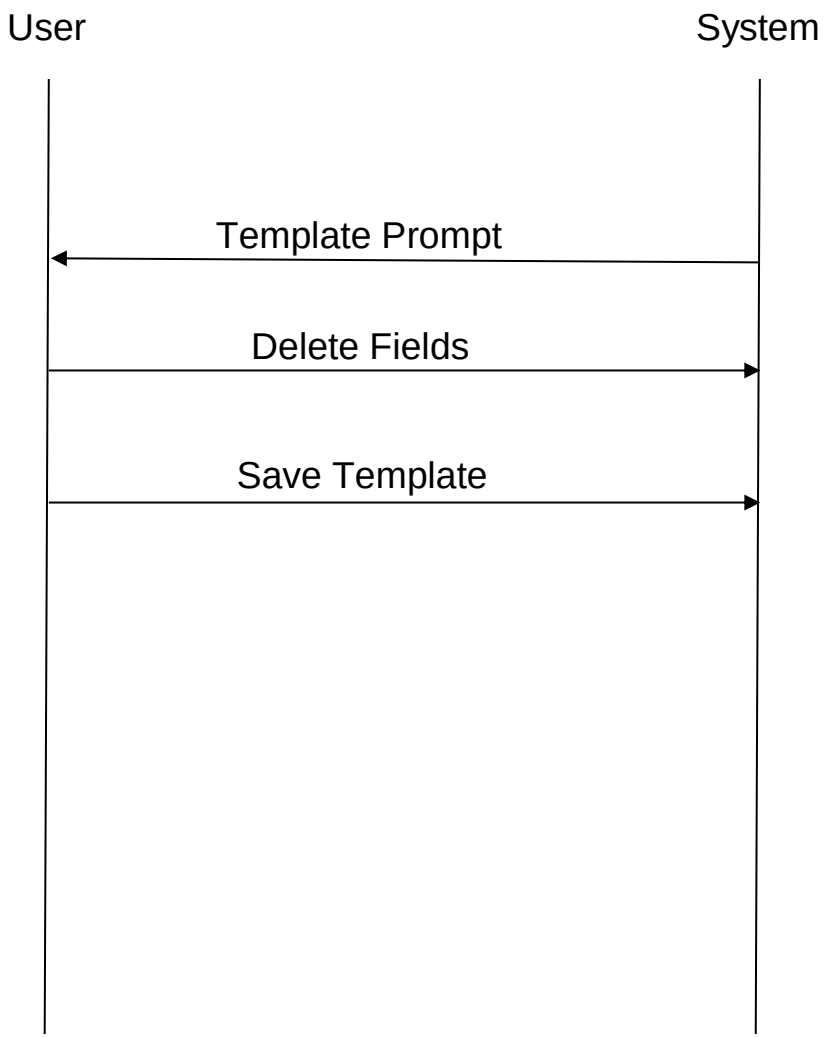
← 6. System sends emails containing the report.

## d. System Sequence Diagram

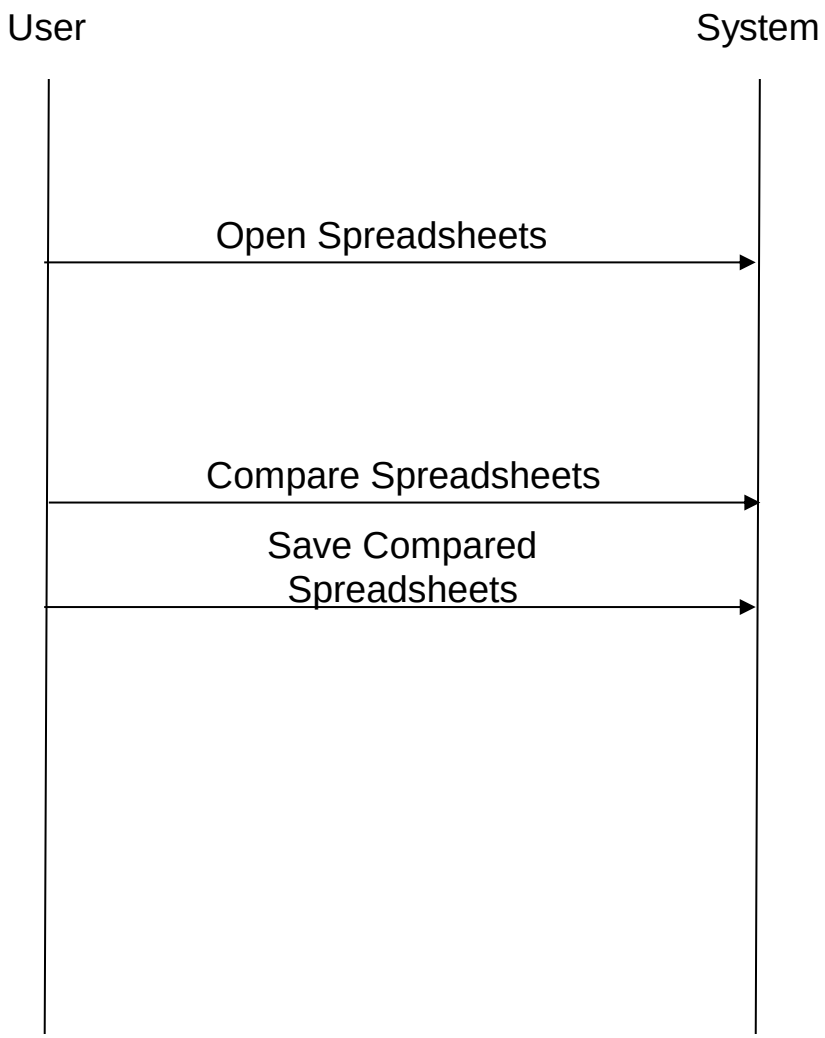
1. Save Template




2. Delete Fields



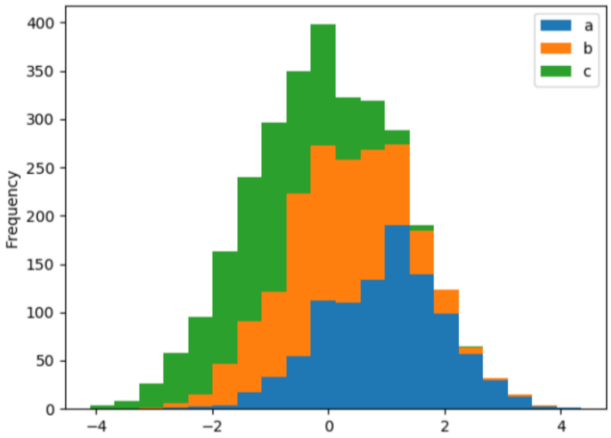

3. Compare Reports




## 4. User Interface Specification

## a. Preliminary Design

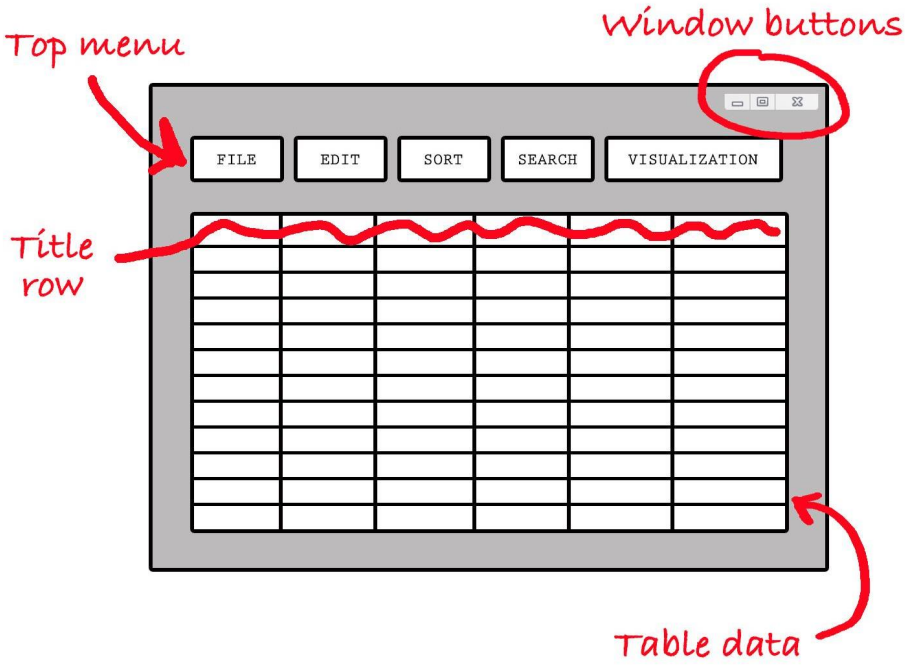
- Initial screen features simple step-by-step instructions for how to use the application.
- The user's first action will be to select the "import" option.
- A file selection window will then open, allowing the user to browse their local machine for a file to import.
- Once imported, the spreadsheet's data will be ready for modification. The data will be searchable, organizable by column, and able to be deleted, copied or manipulated from within the interface.
- After the spreadsheet has been organized to the user's liking, a variety of chart visualization options will be selectable.
- The final product will then be exportable, with an option to either save the file locally or email it as an attachment.
- Sample data visualization:




## b. 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
- With a selection of columns or cells highlighted, click **Visualization** from the top menu to open the Chart Wizard tool, which will list visualization options including pie charts, line graphs, bar graphs, etc.
- In the top menu, click **File -> Export -> Save As** to save a copy of the report to the local machine as a PDF
- Alternatively, click **File -> Export -> Email File** to send a copy of the report to an email address

- In the top menu, click **File -> Close** OR click the “x” button in the top right corner to close the application

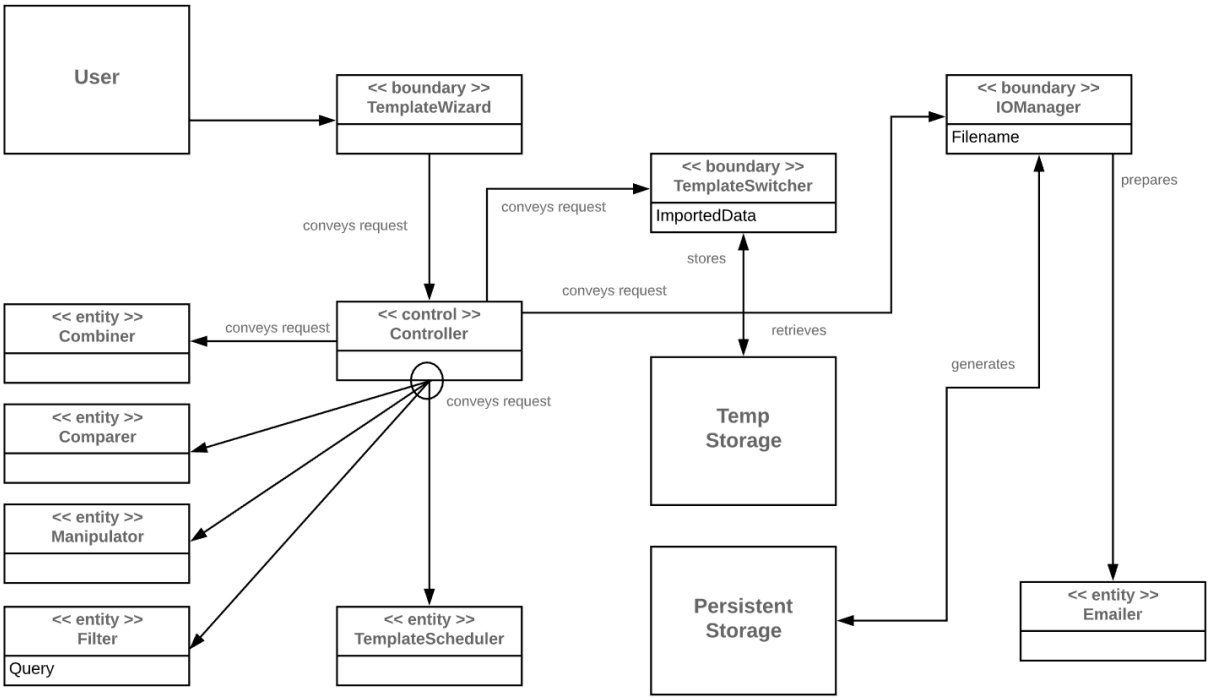



## c. Data Manipulation

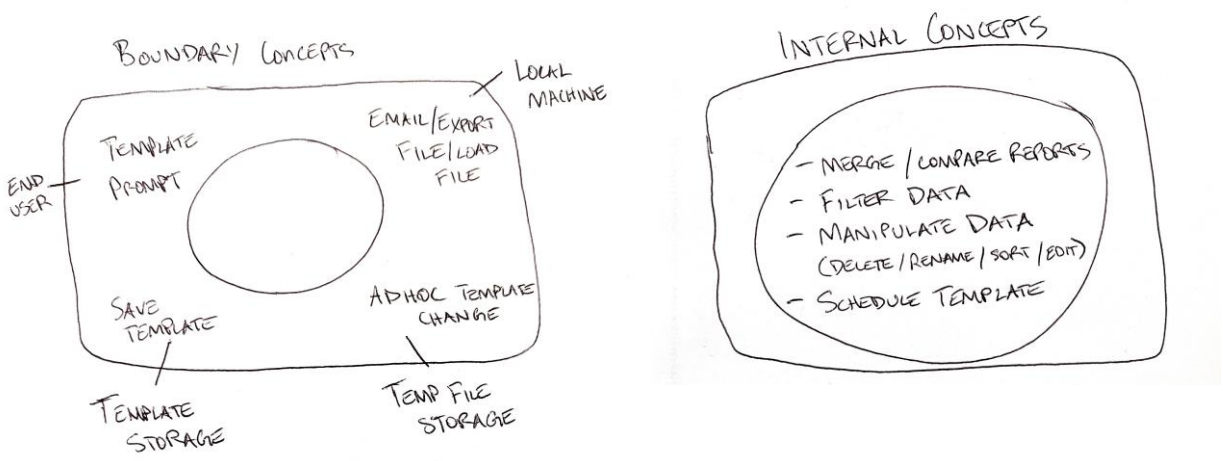
- Once the imported file has been loaded, right-click the database cell to be manipulated, then from the dropdown menu, select from the available options to edit, format, cut, copy, paste, or delete.
- Alternatively, drag and drop columns to reorganize their order
- Select the column’s name cell to select the entire column for dragging, sorting, deletion or cutting/copying
- Double-click a table cell to modify its contents

# 5. Domain Analysis

## a. 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.




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 |
| Guide the user through various steps (Template prompt → Merge/Compare Reports OR Sort/Filter/Modify Fields OR Schedule Template → Save Template AND/OR Export/Send File). | D | TemplateWizard |
| Schedule saved templates to automatically reformat the reports to which they are assigned. The user must first indicate the save location, filetype, and the original filename which is to be modified. | D | TemplateScheduler |
| Container for temporary data storage of template being modified by user. | K | TempStorage |
| Compare contents of selected reports and output the differences discovered, line-by-line. | D | Comparer |
| 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 | D | Manipulator |

| | | |
|---|---|---|
| and paste. | | |
| 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 |
| Handles the sending of reports via email to designated address(es) once they have been modified to the user's needs. | D | Emailer |
| Supports the ad-hoc changing of templates for a data set to display the data in a different, predefined way depending on the selected template's configuration. | D | TemplateSwitcher |

## 2. Association definitions

| **Concept Pair** | **Association Description** | **Association Name** |
|---|---|---|
| TemplateWizard ←→ Controller | TemplateWizard passes requests to the Controller | Conveys requests |
| Controller ←→ Comparer | Controller passes comparison requests to Comparer | Conveys requests |
| 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 ←→ TemplateScheduler | Controller passes scheduling requests to TemplateScheduler | Conveys requests |
| Controller ←→ TemplateSwitcher | Controller passes data re-configuration requests to TemplateSwitcher | Conveys requests |
| TemplateSwitcher ←→ TempStorage | TemplateSwitcher stores data in TempStorage for ad-hoc changes to template, then retrieves it | Stores and retrieves data |
| Controller ←→ | Controller passes file I/O requests to IOManager, which imports data or generates a file export | Generates |

| | | |
|---|---|---|
| IOManager | | |
| IOManager ←→ Emailer | IOManager prepares file for sending via the Emailer | Prepares |

3. Attribute definitions

4.

| Concept | Attributes | Attribute Description |
|---|---|---|
| Filter | Query | A given set of data for the Filter to drill down and focus on. |
| Template Switcher | ImportedData | Data brought into the template switcher to temporarily be formatted bases upon the templates ad hoc settings. |
| IOManager | Filename | The name and path of the file to both attach to an email or save into persistant storage. |

5. Traceability matrix

| | | Domain Concepts | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Use Case | PW | Controller | TemplateWizard | TemplateScheduler | TempStorage | Comparer | Combiner | Filter | Manipulator | IOManager | Emailer | TempateSwitcher |
| UC1 | 42 | X | X | X | | | | | X | X | X | |
| 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 | | | | |
| UC8 | 10 | X | X | | | | X | | | | | |
| UC9 | 5 | X | X | | | | | | | X | | |
| UC10 | 4 | X | X | | | | | | | X | | |
| UC11 | 12 | X | X | X | | | | | | | | |
| UC12 | 3 | X | X | | | | | | X | | | |
| UC13 | 32 | X | X | | | X | | | | | | |
| UC14 | 3 | X | X | | | | | | | X | | |
| UC15 | 9 | X | X | | | | | | | X | | |
| UC16 | 3 | X | X | | | | | | | X | | |
| UC17 | 12 | X | X | X | | | | | | | | |
| UC18 | 7 | X | X | | | | | | | | | X |
| UC19 | 7 | X | X | | | | | | | | | |
| UC20 | 1 | X | X | | | | | | | | | |
| UC21 | 1 | X | X | | | | | | | X | X | |
| UC22 | 1 | X | X | | | | | | X | | | |

## b. 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:**<br>- User has selected a template to edit or is creating a new template<br>- User has a report or spreadsheet to import into the program |
| **Postconditions:**<br>- User has created a custom template using settings allowed by system |

| **Operation:** |
| --- |
| Save Template |
| **Preconditions:**<br>- User has created a custom template using settings allowed by system<br>- User has given this template a unique identifier (A name) |

**Postconditions:**

- Program stores template settings within persistent storage.

**UC-14 Email File**

| **Operation:** |
| --- |
| Email File |
| **Preconditions:**<br>Program is running and has produced a formatted file<br>Formatted file is still in memory (does not need to be retrieved)<br>Local Machine is connected to email program to which program has access<br>User has provided email address of recipient (or it is saved in template) |
| **Postconditions:**<br>Formatted file is emailed to recipient via Local Machine's access to email server |

**UC-17 Schedule Template Options**

| **Operation:** |
| --- |
| Choose Template |

**Preconditions:**

- User has at a template he or she would like to create schedule time to run

**Postconditions:**

- The template is chosen and brought to the TemplateScheduler

**Operation:**

Choose Schedule Settings

**Preconditions:**

- A template has been chosen and passed to TemplateScheduler

**Postconditions:**

- User has chosen scheduled settings
- Settings around the scheduled run are saved within the system

**Operation:**

Perform Scheduled Run

**Preconditions:**

- User has defined and saved a specific timeframe for the scheduled run
- The time matches user's defined timeframe
- The input file is located by the program

**Postconditions:**

· The output report is created and saved in the user-designated location by the system

# 6. 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-11 | Schedule Template | Average | >7 | 3 - User, Template Storage, Local Machine | Complex | 15 |

| | | | | | | |
|---|---|---|---|---|---|---|
| UC-12 | Group Fields | Simple | 1 | 1 - User | Simple | 5 |
| UC-13 | Compare Reports | Complex | 1 | 1 - User | Complex | 15 |
| UC-14 | Email File | Average | 2 | 2 - User and Local Machine | Complex | 15 |
| UC-15 | Save Original | Simple | 1 | 2 - User and Local Machine | Simple | 5 |
| UC-16 | Export to PDF | Simple | 3 | 1 - User | Average | 10 |
| UC-17 | Schedule Template Options | Average | 3 | 3 - User, Template Storage, Local Machine | Complex | 15 |
| 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 |
| UC-21 | Download Link | Simple | 3 | 2 - User and Local Machine | Complex | 15 |
| UC-22 | Create Graph | Complex | 3 | 1 - User | Complex | 15 |

Unadjusted Use-case Weight (UUCW) = 205 = 10*5 + 5*10 + 7*15

Unadjusted Use-Case Points (UUCP) = 211 = 6+205

# 7. Plan of Work

Beginning 9/24/19 - All weeks end on Monday, and all items are to be accomplished by end of week.

Week 1: Interaction Diagrams

Interaction diagrams will be created to describe the interactions of the actors and the system for the main use cases. This will begin the architecture of the components comprising the system. The interaction diagrams will be created by the respective product owners (listed below in “Product Ownership”. Where overlap occurs, product owners will work together.

Week 2: User Interface Design and Design of Tests

Design framework of user interface and create test scenarios to properly address all use-cases. First, we will create a framework for all the interactive pieces, and then product owners will design their own portions within that framework. Where products overlap in the user interface (where multiple functionalities reside in one interface “area”), product owners will combine interface designs in order to create a cohesive and standardized interface.

Week 3: 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 4: Build Core+ Functionality

Expanded basic features

1. Filter Data
2. Rename Column Headers
3. Merge two reports
4. Rearrange columns

5. Create visuals
6. Export to PDF
7. Add these Core+ features to the template save feature

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

Week 5: Complete Testing

Complete test cases covering Core and Core+ functionalities above

Week 6: Revise Report for all changes

Review and edit documentation to align with all changes taken place

Week 7: Design Patterns, Object Constraint Language Contract

Complete design patterns and OCL contract

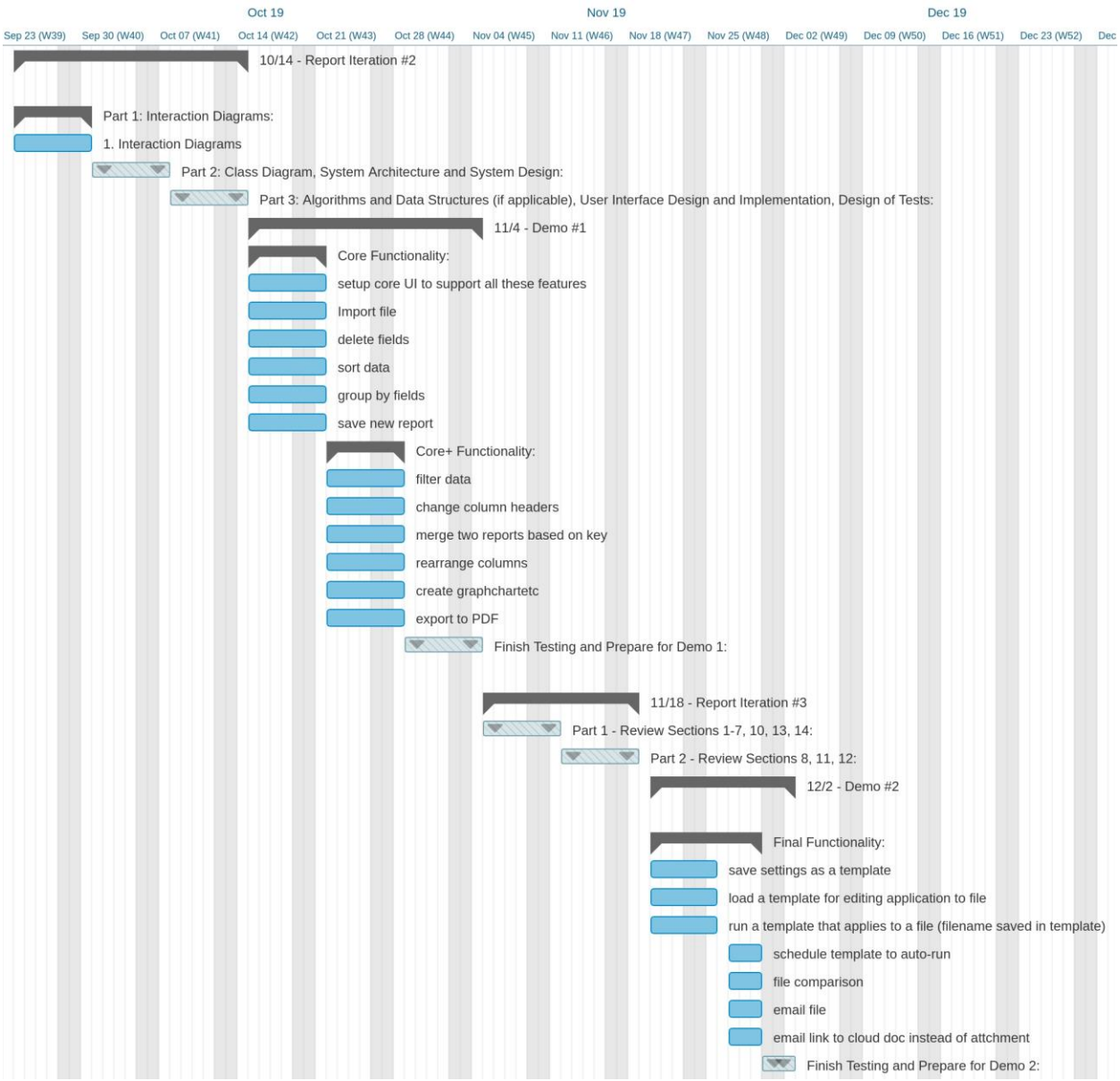
Start work on the final pieces of functionality

1. Including all features into the template
2. Loading a template for editing
3. Applying a template to a file
4. Comparing two files for differences
5. Emailing a file
6. Generating a download link to be emailed

Week 8: 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
4. Comparing two files for differences
5. Emailing a file
6. Generating a download link to be emailed




## Product Ownership

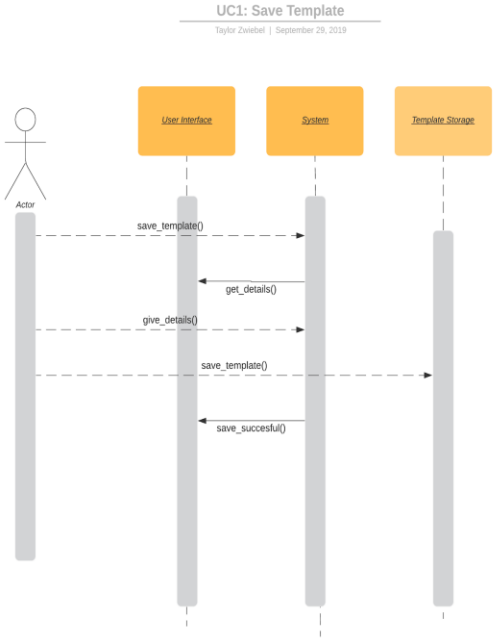
The following list describes the product/feature ownership of this project. Each member will own that specific aspect of the project.

- **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

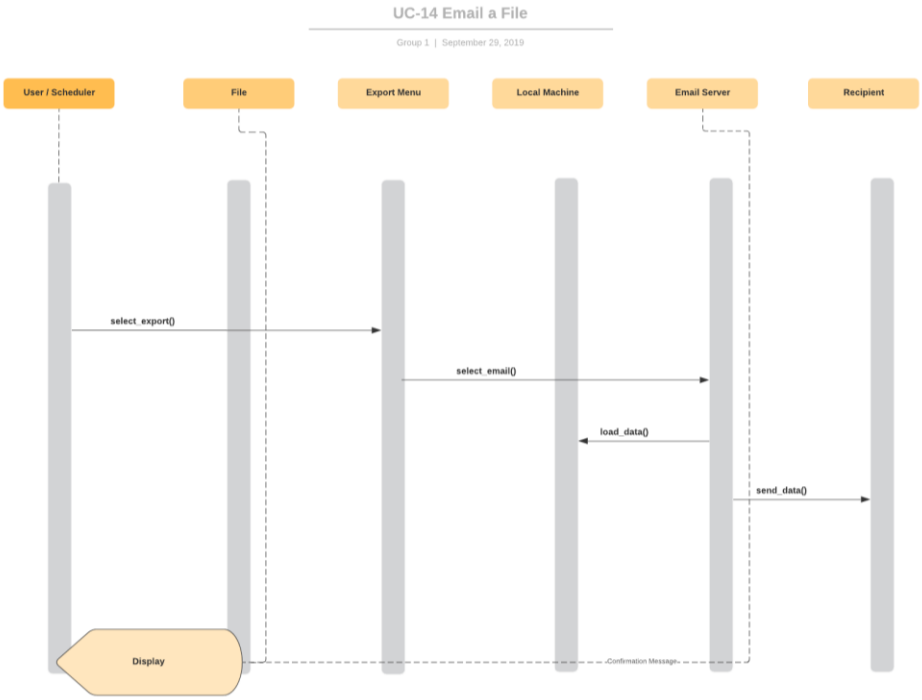
UC-1 Interaction Diagram




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 is 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.

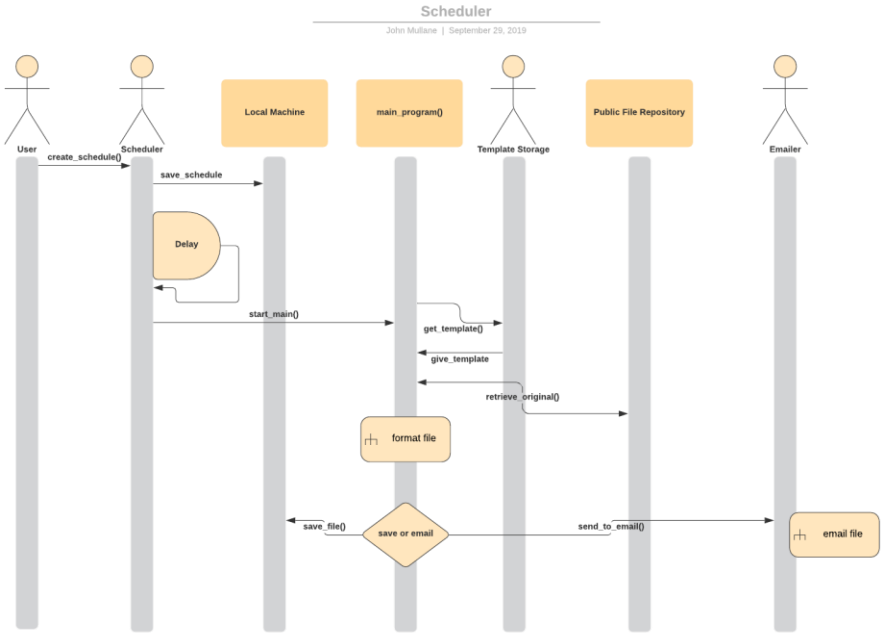
UC-14 Interaction Diagram




Traceability and Interaction Diagram #2

This interaction diagram describes the internal flow of events for UC-14, involving emailing a file after the template has been ran. In this case, there is not a sequence diagram to trace back to; however, we can trace it back to the fully dressed use case found in report #1. This interaction touches on the domain concepts of Emailer, which extended into the Template and TemplateApplicator classes.

UC-17 Interaction Diagram




Traceability and Interaction Diagram #3

This interaction diagram traces back through UC-17 Schedule Template Options. There is no sequence diagram for this use case listed in report #1; however, again there is a fully dressed use-case for this interaction listed in report #1. System Sequence Diagram #2 is a subsequence of this interaction, included in the format file portion of the interaction. This interaction expands the domain concepts: TemplateSchedular, Emailer, and Controller. This interaction will be supported by the classes: Frame, TabClicked, Schedule, TemplateApplicator and Main.

# 9. Class Diagrams, System Architecture and Design

## a. Class Diagrams

i. Window Class
ii. Frame Class
iii. Template Class
iv. Schedule Class

## 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.

**iii. Template Class**

The template class accesses the name of the templates stored in the system. The template then gets the specified template with the filtered, modified, and sorted data and sets it as the active template.

- name: string
  -The name of the templates in storage.
- getFiltered(List): void
  -Gets the current filtered data from the template.
- getModified(List): void
  -Gets any modified data from a template.
- getSorted(List): void
  -Gets the current sorted status from a template.
- setTemplate(): Dictionary
  -Sets the current template being used.

**iv. Schedule Class**

The schedule class accesses the date and time and pulls a template that is specified at the set date and time.

- date: int(mmddyy)
  -This is the current date in the format of mmddyy.
- time: int([1-24][00-59])
  -This is the current time with a 24 hour clock.
- loadName(Object): void
  -The scheduled object to be run based on the date and time

### c. Traceability Matrix

| | Software Clases | | | | | | |
|---|---|---|---|---|---|---|---|
| Domain Concepts | MyWindow | Frame | TabClicked | Template | Schhedule | Main | Template Applicator |
| Controller | | X | X | | X | X | |
| TemplateWizard | X | | X | X | | | |
| TEmplateScheduler | | | | | X | | |
| Comparer | | | | X | | | X |
| Combinner | | | | X | | | X |
| Filter | | | | X | | | X |
| Manipulator | | | | X | | | X |
| IOManager | X | | | | | X | |
| Emailer | | | | X | | | X |
| TemplateSwitcher | | | | | X | | |

## 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.

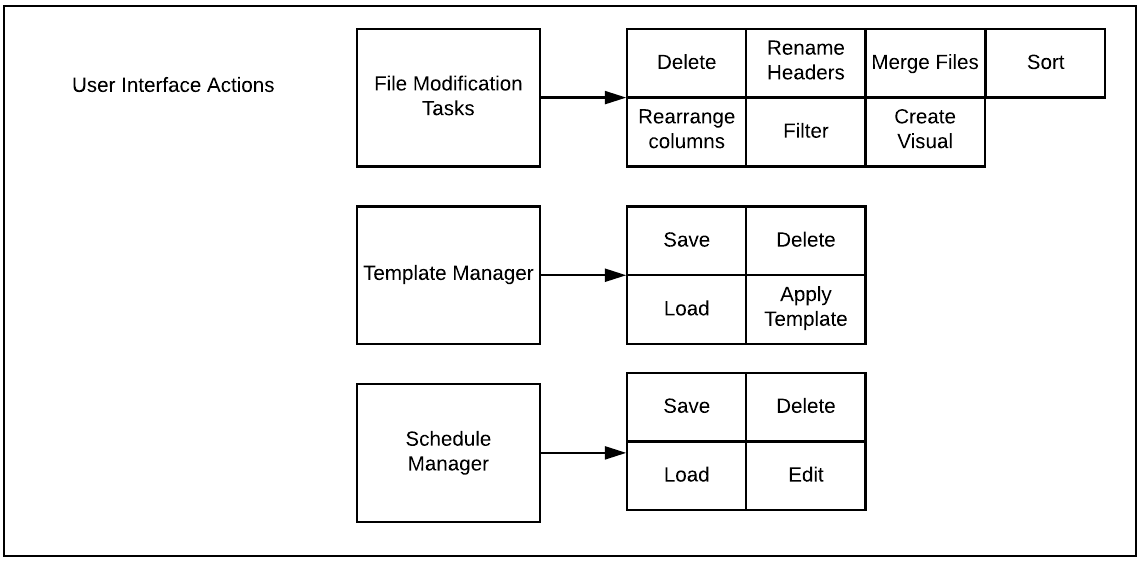
### b. Identifying Subsystems

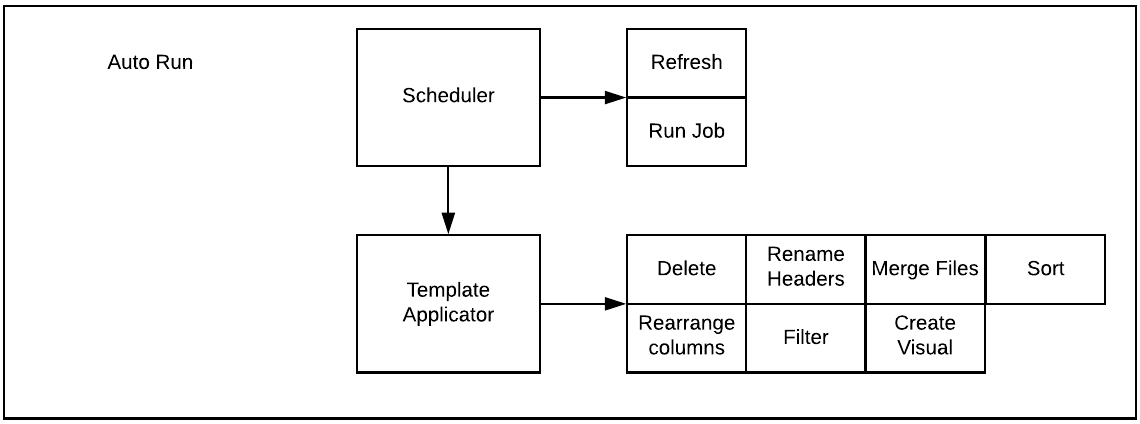
Within our system, there are two routes for the main application: user-directed actions via the interface, and automated runs from the scheduler or taken from a user-directed template application (user selects a stored template and that

template is applied to the file it references, rather than the template application being executed from the result of a schedule).

The file modification tasks subsytem will make available to the user all the possible modifications to a file, to be performed one at a time by the user. The template manager will allow the user to save the current settings as a template, load an existing, delete an existing, or apply a template. When the user chooses to apply a template, the template manager passes control to the Auto-Run subsystem.

The auto-run subsystem handles template applicables generated as the result of the user selection or from a scheduled job. In cases of user selection, the scheduler does not come into play, control is passed directly to the template applicator, and all changes made. The scheduler is responsible for monitoring the pending jobs and passing the applicable template to the template applicator when needed.



## 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.

Additionally, there are time dependencies within our system, as it will incorporate scheduling of events (without interaction for the user). The scheduler will activate the program and supply the template to the program for processing. The scheduler will routinely check for pending jobs and execute when a job is awaiting processing.

## 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

There have been no significant changes to the initial interface designs. The application will feature a “wizard” type setup with tabs for each of the various actions. The user can move freely between the tabs as needed to execute the various changes desired.

# 12. Design of tests

- **Compare to prior report**
  - Take two basic reports and test that the system will compare the two reports and return the correct compared results.
  - This will cover comparing the two reports, but also the system getting both reports to compare from wherever they are stored.
- **Scheduling**
  - By scheduling a report to be ran at anytime preferably within minutes of telling the system we will see that the program will run reports on a schedule put in place by the user.
  - This will also make sure the system can get the reports and run the tasks the scheduler setup.
- **Emailing**
  - Emailing a report to yourself and others is part of the programs emailing task. By putting in my email and those of others will test that the program can email the reports to both parties.
- **Merge with another file**
  - This test case will take two basic test reports and merge the two together. It will make sure the outcome of running a merge on files will give the correct resulting file.
- **Select a file from the repository**
  - This test will tell the program to select specific files and then test that the files pull match the desired outcome.
- **Delete fields**
  - 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.
- **Save a template**
  - The test will have different variables saved to create a template. The template will then be pulled and checked that the specified details of the template were saved.
  - This also makes sure the system can pull templates from where they are saved.

# 13. Project Management and Plan of Work

## a. Merging the contributions from Individual Team Members

We have utilized a shared document throughout our process, which updates to all contributors in real time. Therefore, no compiling was necessary. Formatting was reviewed prior to each iterative submission to ensure uniformity. Notes and comments were shared as well, to ensure all team members are working through the same ideas.

## b. Project Coordination and Progress Report

Basic functionalities such as importing a file, deleting columns, moving columns, sorting, grouping, and exporting have already been built in their most fundamental form. We are currently working on building core user interface functionality in which to house these capabilities. As mentioned previously, the user interface will feature tabs for the different functions the user can execute to change the report. These functions work separately, and therefore implementing them via the interface, with its tabbed format, will also be conducted in a piecework fashion. The proximal step will be to log all of the changes and store them in a template so we can retrieve them for application at a later time.

## c. Plan of Work

Beginning 10/15/19 - All weeks end on Monday, and all items are to be accomplished by end of week.

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 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 3: Complete Testing

Complete test cases covering Core and Core+ functionalities above

Week 4: Revise Report for all changes

Review and edit documentation to align with all changes taken place

Week 5: Design Patterns, Object Constraint Language Contract

Complete design patterns and OCL contract

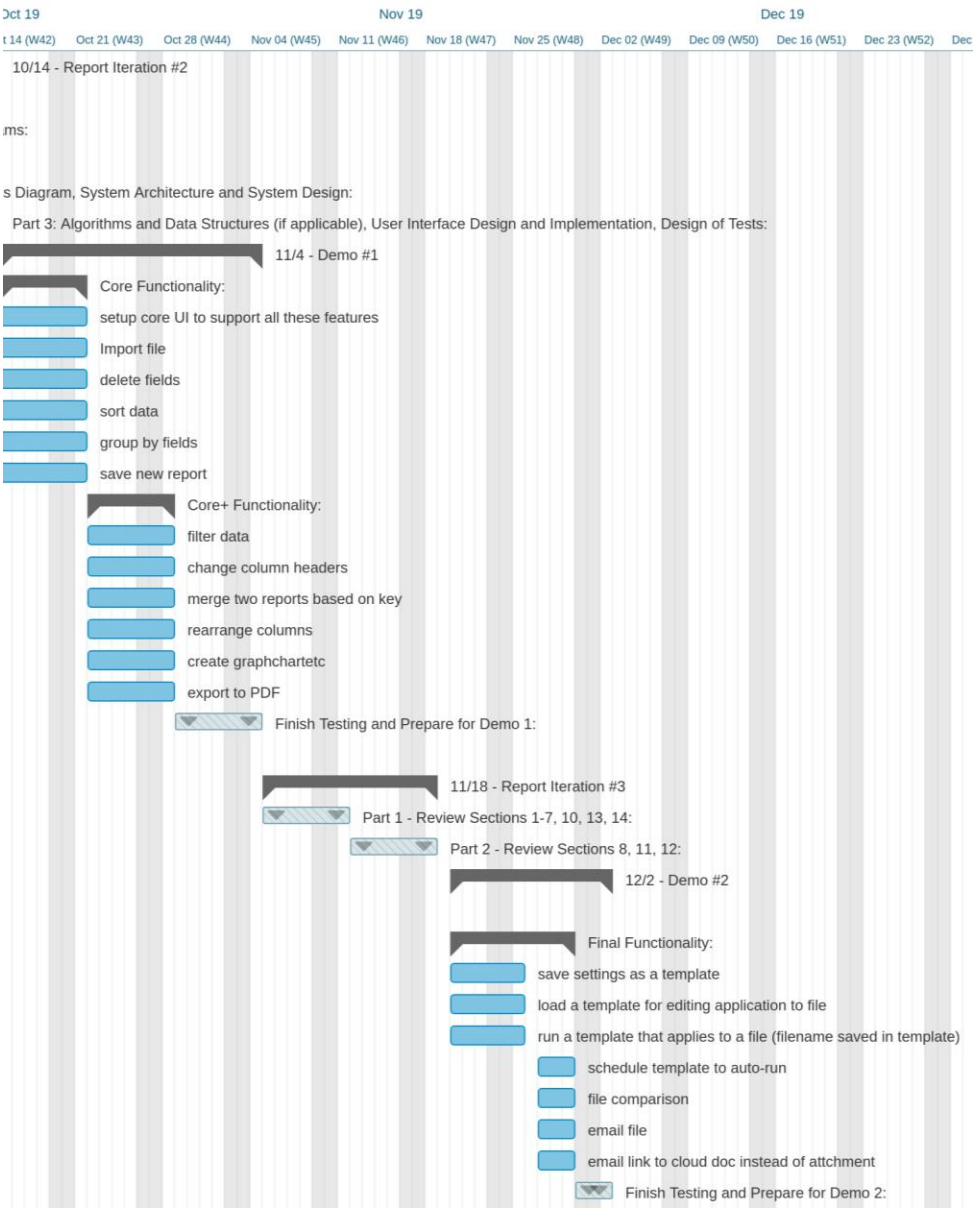
Start work on the final pieces of functionality

7. Including all features into the template
8. Loading a template for editing
9. Applying a template to a file
10. Comparing two files for differences
11. Emailing a file
12. Generating a download link to be emailed

Week 6: Final Functionality

Complete and test the following features in preparation for Demo 2

7. Including all features into the template
8. Loading a template for editing
9. Applying a template to a file
10. Comparing two files for differences
11. Emailing a file
12. Generating a download link to be emailed




## 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

**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

**Sort data** - Taylor Zwiebel

**Filter data** - John Mullane

**Rearrange columns** - John Mullane

**Rename columns -** Taylor Zwiebel

# 14. References

1. (28) Tkinter - Create tabs in your GUI interface using Notebook - YouTube
   https://www.youtube.com/watch?v=CUf7fB8hlOU

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-in-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/