BNP PARIBAS

BNP Paribas: Budget Automation

A Major Qualifying Project Report

Submitted to the Faculty of
Worcester Polytechnic Institute
In Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science

Submitted By:
Jacqueline Nancy Ngo, Business and Economic Science
Nan Zhang, Computer Science and Industrial Engineering

Submitted to:
On-site Liaison:
Andrew Clark

Project Advisors:
Professor Michael Ciaraldi, Department of Computer Science
Professor Renata Konrad, Foisie School of Business
Professor Kevin Sweeney, Foisie School of Business

Submitted on:
January 25, 2017

Abstract

To reduce the time and effort required to complete the semi-annual budgeting process for the Global Markets IT division of BNP Paribas, our team designed an automated process consisting of a Python server, SQL database, and web-based front-end. Currently, to budget and forecast headcount and discretionary costs, the process requires hundreds of hours among the ITO Business Management team, team leads, and senior management. The process is complex, time-consuming, and labor intensive; it lacks a single system to input allocations, an easy way to audit activity, and a way to track multiple versions of the data sources. The new process our team designed addresses the identified bottlenecks within the current process and aims to lessen the complexity and time of the overall process.

Acknowledgements

Our team received tremendous support throughout the duration of the project from our sponsor - BNP Paribas - and from our faculty advisors at Worcester Polytechnic Institute. We would like to especially thank the following people at BNP Paribas who guided us during our eight weeks on-site and made the success of our project possible:

Andrew Clark
Val Ryjouk
Kunal Changela
Artun Koktug
Wassim Najjar
Robert Kraft
Jacques Dillies.

Second, we express our gratitude to our advisors at WPI who supplied us with resources and advice and helped us along every step of this project:

Kevin Sweeney
Michael Ciaraldi
Renata Konrad.

Executive Summary

Semi-annually, BNP Paribas's Global Markets (GM) IT division conducts a budget allocation process. In January and again mid-year, senior management and team leads allocate their headcount and discretionary costs to profit centers (also referred to as front-desks, which are money-making divisions in the front office). In January, team leads and management forecast the percentages of their expenses to charge to specific profit centers, then team leads and management reforecast the percentages (referred to as allocation keys) in June when they have a better idea of their costs. The identified bottlenecks that needed to be addressed were: the lack of a single system for team leads and management to submit their budget allocations, the lack of a process to track revision history, and the complexity associated with tracking multiple versions of headcount and discretionary costs data.

The objective of our project was to design an automated process to reduce the complexity of this budget allocation process which includes creating the budget, mapping the data together, creating the allocation template, and the actual allocations. We identified the bottlenecks in the system and aimed to address them with our new budget allocation process design which consists of a Python Server, SQL database, and front-end web-based application.

The creation of the SQL database was a top priority in this project; it was critical to have a database that is easy to maintain. Therefore, over a dozen database designs were created before finalizing the design – illustrating the complexity of the data. The final database design has 44 normalized tables; by normalizing the database, we were able to avoid having redundant data. The data that makes up the database is from the extracts pulled from the project management tools, from mapping files provided by the ITO Business Management team, and from users' inputs in the front-end.

In our front-end design, not only are team leads and management able to allocate their expenses, but they are also able to view recent activity which eases the auditing process (see Appendix D for possible use cases). Not being able to track edits is a major concern in the current process, but in our application design, the ITO Business Management team and management will be able to track who's making what edits. The other problem that needed to be solved was versioning the data sources, and that is also addressed in our new design.

We conducted significant data and system analyses throughout the duration of this project, and we hope that the progress we had made with our designs and implementation will simplify the process and significantly reduce the amount of time required to complete this semi-annual process.

Authorship

Section	Author
Title Page	Jacqueline Ngo
Abstract	Jacqueline Ngo
Acknowledgements	Jacqueline Ngo
Executive Summary	Jacqueline Ngo
Table of Contents	Jacqueline Ngo
I. Introduction	
1.1 BNP Paribas Background	Jacqueline Ngo
II. Background	
2.1 Current Budgeting Process	Jacqueline Ngo
2.2 Problem Statement	Jacqueline Ngo
III. Methodology	
3.1 Objectives	Jacqueline Ngo
3.2 New Process Overview	Jacqueline Ngo
3.3 Technology Tools	Nan Zhang
IV. SQL Database	
4.1 Database	Jacqueline Ngo
4.2 Data Loading Complexities	Jacqueline Ngo
4.3 Test Cases	Jacqueline Ngo
V. System Analysis	
5.1 System Components	Nan Zhang
5.2 Front End Design	Nan Zhang
VI. Results	
6.1 Progress We Made	Nan Zhang
6.2 Special Features	Nan Zhang
VII. Recommendations	
7.1 Data clean-up	Jacqueline Ngo
7.2 Usage Clarity API	Nan Zhang
VIII. Future Extensions	Nan Zhang
IX. Conclusion	Jacqueline Ngo
Bibliography	Nan Zhang & Jacqueline Ngo
Appendix A: Current General Budget Allocation Process	Jacqueline Ngo
Appendix B: Deliverables and Corresponding Objectives	Jacqueline Ngo
Appendix C: New General Budget Allocation Process	Jacqueline Ngo
Appendix D: Use Case Diagram	Jacqueline Ngo
Appendix E: Sequence Diagram	Nan Zhang
Appendix F: Data Flow Diagram	Nan Zhang
Appendix G: Final Database Design	Jacqueline Ngo
Appendix H: Un-Normalized Database Design	Jacqueline Ngo

Table of Contents

Abstract i
Acknowledgements ii
Executive Summary iii
Authorship iv
Chapter I: Introduction 1
1.1 BNP Paribas Background 1
Chapter II: Background 2
2.1 – Current Budget Allocation Process 2
2.1.1 – Current General Process 2
2.1.2 – Current Blended Keys Process 3
2.1.3 – Current Re-Allocation Process 5
2.1.4 – Allocations Aggregated by Program 7
2.2 – Problem Statement 7
Chapter III: Methodology 9
3.1 Objectives 9
3.1.1 Deliverable 1 Automating General Allocation Process and Corresponding Objectives 9
3.1.2 Deliverable 2 Blended Keys Automated Allocation Process and Corresponding Objectives 9
3.1.3 Deliverable 3 Automating Reallocation Process and Corresponding Objectives 10
3.2 New Process Overview 10
3.3 Technology Tools 11
3.3.1 Microsoft SQL Server 11
3.3.2 Python (Flask Framework) 11
3.3.3 MVC Architectural Pattern 12
3.3.4 Front End 12
3.3.5 Overview 13
Chapter IV: SQL Database 14
4.1 Database 14
4.1.1 Un-Normalized Database Design 14
4.1.2 Normalized Database Design 16
4.2 Data Loading Complexities 18
4.3 Test Cases 19
Chapter V: System Analysis 20
5.1 System Components 20
5.1.1 Startup 20
5.1.2 Authentication 20
5.1.3 Authorization 21
5.1.4 Menu 21
5.1.5 File Upload 22
5.1.6 File Read 22
5.1.7 Edit Profit Center 22
5.1.8 Generate Allocation Template 23
5.1.9 Edit Allocation Percentage 23
5.1.10 Save Allocation Percentage 24
5.1.11 Revision 24

5.1.12 Permission Edit 24
5.2 Front End Design 25
Chapter VI: Results 30
6.1 Progress We Made 30
6.2 Special Features 30
6.2.1 Versioning 30
6.2.2 Pandas Library 31
6.2.3 Bootstrap 31
6.2.4 Master Detail Grid 31
Chapter VII: Recommendations 32
7.1 Data Clean up 32
7.2 Usage of Clarity API 32
Chapter VIII: Future Extensions 33
Chapter IX: Conclusion 34
Bibliography 35
Appendix A: Current General Budget Allocation Process 36
Appendix B: Deliverables and Corresponding Objectives 38
Appendix C: New General Budget Allocation Process 39
Appendix D: Use Case Diagram 41
Appendix E: Sequence Diagram 42
Appendix F: Data Flow Diagram 43
Appendix G: Final Database Design 44
Appendix H: Un-Normalized Database 45

List of Figures

Figure 1 – An Example of a Minimal Flask Application 12
Figure 2 – Structure of this Application 13
Figure 3 – Authentication Process 21
Figure 4 – Authorization Process 21
Figure 5 – Major Fields that Form Allocation Template 23
Figure 6 – Overview of Revision 24
Figure 7 – User Log In Page 25
Figure 8 – Main Menu Page 26
Figure 9 – File Upload Page 26
Figure 10 – Edit Profit Center Page 27
Figure 11 – Permission Edit Page 27
Figure 12 – Allocation Keys Edit Page 28
Figure 13 – Allocation Keys View Page 28
Figure 14 – Revision History Page 29

List of Tables

Table 1 – Example of Headcount Allocation 2
Table 2 – Example of Discretionary Cost Allocation 3
Table 3 – Example of Part 1 of the Blended Keys Process 4
Table 4 – Example of Part 2 of the Blended Keys Process 4
Table 5 – Example of Part 3 of the Blended Keys Process 4
Table 6 – Example of Allocations to Dummy Profit Centers from Step 2 5
Table 7 – Example of Re-allocation from Dummy Profit Center 1 to Real Profit Centers 5
Table 8 – Example of Blended Keys for Re-allocation from Dummy Profit Center 1 to Real Profit Centers 6
Table 9 – Example of Blended Keys for Re-allocation from All Dummy Profit Centers to Real Profit Centers 6
Table 10 – Example of Reallocated Keys in Step 4 6
Table 11 – Example of Allocations with Corresponding Programs 7
Table 12 – Example of Allocations Aggregated by Program 7
Table 13 – Example of Rows from Headcount Extract 14
Table 14 – Projects with Multiple Cost Centers and Charge Codes 16
Table 15 – Example of an Independent Table in the Database 16
Table 16 – Example of a Dependent Table in the Database 17
Table 17 – Example of a Table with Redundant Data 17
Table 18 – Missing Data in Clarity Extract 18
Table 19 – Clarity Extract without Missing Data 18
Table 20 – Examples of Test Cases 19
Table 21 – All the System Components 20
Table 22 – Basic Elements of Startup 20
Table 23 – Basic Elements of Menu 21
Table 24 – Basic Elements of File Upload 22
Table 25 – Basic Elements of Edit Profit Center 22
Table 26 – Basic Elements of Editing Allocation Percentage 24
Table 27 – Basic Elements of Permission Edit 25
Table 28 – An Example of the Versioning Mechanism 30

Chapter I: Introduction

BNP Paribas, a multi-national French investment bank, encourages innovation (Innovation: our responses to a changing world). This is exemplified by our project to automate the budget allocation process for Global Markets (GM) IT. With guidance from multiple departments across the division, our team was given the opportunity to develop a solution to address the complexity of the current budget allocation process. By simplifying and automating the semi-annual process, hundreds of man hours can be saved annually and this time can be spent further improving the customer experience. As a result, BNP staff can focus on projects to improve the bank.

1.1 BNP Paribas Background

Banque Nationale de Paris (BNP) Paribas has history dating back to the 19th century, when its two forerunners, Comptoir National d'Escompte de Paris (CNEP) and Comptoir National d'Escompte deMulhouse, were established in 1848. These "comptoirs d'escompte" (discounting houses) were formed to facilitate credit circuits in France while the country was undergoing an economic meltdown and political revolution which had destroyed the country's former credit system. In 1872, European bankers wanted to "raise funds to borrow to free up regions and, on a longer term, to acquire shareholdings in companies and acquire a stake on capital markets (BNP History);" thus, Banque de Paris et des Pays-Bas (Paribas) was established.

After several more European banks were created and mergers occurred, a merger of Banque Nationale de Paris and Paribas in 2000 led to the creation of BNP Paribas which has remained a strong European leader since. Today, BNP Paribas has a presence in over 70 countries on 5 continents (BNP History; i.e., North America, South America, Europe, Africa, and Asia) and employs approximately 189,000 employees as of 2015.

BNP Paribas's key activities include retail banking (i.e., corporate vehicle leasing, rental and financial solutions, and online savings and brokerage) and financial services (i.e., private banking, asset management, and real estate services) as well as corporate and institutional banking (e.g., solutions across capital markets, securities services, financing, treasury, and financial advisory).

BNP Paribas recognizes that it is a changing world, and encouraging innovation is the firm's response to better serve their customers. An example of BNP Paribas's commitment to innovation is through hosting "International Hackathon" where eight countries simultaneously compete to create solutions for a given theme (BNP Paribas International Hackathon). In 2016, at the second International Hackathon, the theme was "Streamlining the customer journey through co-creation with startups;" 96 startups participated in the 51-hour long event and 18 winners were chosen. BNP Paribas then supports and collaborates with the winning startups (BNP Paribas International Hackathon). This event (along with many other events and sponsorships) exemplifies BNP Paribas's commitment to supporting actors of innovation and innovating to serve their customers.

Chapter II: Background

2.1 – Current Budget Allocation Process

According to Andrew Clark, a senior manager within GM IT and our onsite liaison, the current semi-yearly budgeting process requires hundreds of man-hours between GM IT's senior management, team leads, and the ITO Business Management team. At the beginning of the year and again mid-year, the three groups must forecast where GM IT's expenses will be charged. As GM IT does not directly contribute to BNP's revenues, but represents an expenditure, GM IT must charge their expenses to "Profit Centers" which are the front-desks within the front office. These front-desks that GM IT supports bring in money for the bank through trades and other activities. Different managers within GM IT have different allocation processes, and this chapter provides background on each of the processes addressed in our project.

2.1.1 – Current General Process

Appendix A represents the current general budget allocation process used by Andrew Clark and his team.

Clarity, a project management tool, was the primary source of our data for this project. Team leads and management update "Headcount" data quarterly and "Discretionary Costs" data yearly in Clarity. "Headcount" data is information on all resources (employees) in IT and the corresponding projects they are assigned to. Additional information includes country, which programs their projects correspond to, corresponding cost centers, corresponding charge codes, etc. Each resource has a limited number of "man-days;" the resource's time must be budgeted since each resource has a cost (e.g., salary and benefits). Team leads and management allocate their resources' monthly "man-days" in Clarity to track the expected headcount costs for each project. "Discretionary Costs" are fees that must be paid to outside vendors for licenses and other services. Parameters used to track these costs are: vendor name, budget source, cost center, charge code, etc. The ITO Business Management team pulls two extracts from Clarity – one for Headcount and one for Discretionary Costs – and uses these extracts to determine the budget for the upcoming year.

The data from these two extracts is mapped with other parameters and used to create a single allocation template. An allocation template is an Excel sheet where team leads and managers assign percentages of their costs to the profit centers. For example, as shown in Table 1, a team lead is responsible for Project A. This team lead must assign allocation keys (percentages of their costs) to profit centers, summing to 100%.

Project/Vendor Name	Man-Days	Cash	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
Project A	50	$50,000	20%	10%	70%	0%

Table 1 – Example of Headcount Allocation

In this simplified example, a team lead must allocate costs for Project A's headcount costs. For 2016, the team lead has a budgeted amount of 50 man-days (e.g., one employee working for 50 days, five employees working 10 days each, or fifty employees working one day each, etc.) which is equivalent to $50,000. The team lead must charge these costs to the profit centers since IT does not directly make any money; in this scenario the team lead is charging 20% of Project A's costs ($10,000) to Profit Center 1, 10% of the costs to Profit Center 2 ($5,000), and 70% of the costs to Profit Center 3 ($35,000). Because 100% of the costs have already been charged,

Profit Center 4 is not charged any amount. Note that the percentages assigned to each profit center are the "allocation keys."

A similar process is followed for discretionary costs as illustrated in Table 2.

Project/Vendor Name	Man-Days	Cash	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
Vendor A	0	$100,000	100%	0%	0%	0%

Table 2 – Example of Discretionary Cost Allocation

In this example, a manager has to assign allocation keys for this $100,000 fee for Vendor A (note that because this is a discretionary cost and not a headcount cost, the number of man-days is zero because man-days do not apply to vendors and licenses). The manager assigned all the costs to Profit Center 1 because the other three profit centers did not use the license from the vendor, and thus do not need to be charged. The ITO Business Management team has to create the allocation template where all headcount costs (and their corresponding parameters) and discretionary costs (and their corresponding parameters) are included for team leads and management to input allocation keys.

Once the allocation template is complete, it is sent to senior managers within IT who then email the team leads. The managers and team leads must then sort through the 550+ projects/vendors and filter out the ones that are irrelevant to them and must do the same for the 74 profit centers. Because managers and team leads are not assigned rows in the template to input their allocations and because there is no defined list of which profit centers relate to which projects and vendors, there can often be confusion when assigning allocation keys.

Next, each team lead sends back their completed allocation keys to their manager. Each team lead filled out the allocation keys in their own version of the template because there is not a system where multiple people can edit the allocation template. Thus, the manager must consolidate all the different allocation keys from all the team leads to make one overall file. This file is then sent back to the ITO Business Management team.

This process takes place twice a year, once in January to forecast where all the charges will be allocated for the entire year and once again in June when managers have a better idea of what costs they have used and which profit centers those costs corresponded to.

2.1.2 – Current Blended Keys Process

Another senior manager within GM IT has automated his allocation process, and he has done so by using "Blended Keys." The steps prior to his allocation process are similar to the general process, except his extracts are pulled from a tool called Bam+, instead of Clarity, and he creates his own allocation template that is used in addition to the template provided by the ITO Business Management team. After extracting his data and setting up his template, he and his team leads use weighted averages for their allocations:

Part 1 - Allocation keys are input for "applications" instead of for projects; each of these applications is assigned a weight, and each of these applications belongs to a "stream" as exemplified in Table 3.

Global Stream Name	Application Name	Stream Weight	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
Stream 2	App A	0.10	25%	10%	30%	35%
Stream 2	App B	0.20	50%	30%	0%	20%
Stream 2	App C	0.30	100%	0%	0%	0%
Stream 2	App D	0.40	10%	40%	5%	45%

Table 3 – Example of Part 1 of the Blended Keys Process

Team leads input values under the "Stream Weight" column for their corresponding applications and then inputs allocation keys for the Profit Centers.

Part 2 – The "Blended Keys" for each stream is calculated as exemplified in Table 4.

Global Stream Name	Application Name	Stream Weight	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
Stream 2	App A	0.10	25%	10%	30%	35%
Stream 2	App B	0.20	50%	30%	0%	20%
Stream 2	App C	0.30	100%	0%	0%	0%
Stream 2	App D	0.40	10%	40%	5%	45%
Stream 2	**Blended Keys**		**46.5%**	**23%**	**0.05%**	**25.5%**

Table 4 – Example of Part 2 of the Blended Keys Process

The "SUMPRODUCT" Excel function is used to calculate the blended keys; it "multiplies the corresponding items in the arrays and returns the sum of the results (TechontheNet)." For example, the blended key of 46.5% under Profit Center 1 is the sum of: App A's allocation key of 25% multiplied by the weight of 0.10, App B's allocation key of 50% multiplied by the weight of 0.20, App C's allocation key of 100% multiplied by the weight of 0.30, and App D's allocation key of 10% multiplied by 0.40. Note that the allocation keys for each application across the profit centers must total 100% and the calculated blended keys must also total 100%.

Part 3 – The blended keys are populated in the general allocation template as exemplified in Table 5.

Stream	Project/ Vendor Name	Man-Days	Cash	Methodology	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
Stream 2	Project B	50	$50,000	Stream 2	46.5%	23%	0.05%	25.5%
Stream 2	Project C	10	$10,000	Stream 2	46.5%	23%	0.05%	25.5%
Stream 2	Project D	25	$25,000	Stream 2	46.5%	23%	0.05%	25.5%
Stream 2	Project E	5	$5,000	Stream 2	46.5%	23%	0.05%	25.5%

Table 5 – Example of Part 3 of the Blended Keys Process

Additional columns in the general allocation template that were not shown in Table 2 and 3 are "Stream" and "Methodology." Each project corresponds to a stream which is provided in the allocation template by the ITO Business Management team through their mapping process. In the general process, team leads or managers will type up their methodology, which is how they decided upon their allocation keys, but in this process, the senior manager selects a methodology which is the name of one of his corresponding streams, and populates the blended keys. Thus, all projects belonging to that stream will have the same set of percent allocations to the profit centers. By populating these numbers, he does not have to assign allocation keys to each profit center for each project; instead, he finds weighted averages for his applications and populates the template with these blended keys.

2.1.3 – Current Re-Allocation Process

Out of the 74 profit centers, 7 are "dummy" profit centers. These belong to the third manager who we met with regarding this project. Allocation keys are assigned to these "dummies" that fall under "Operations." Because these are not real profit centers, the charges allocated to these dummy profit centers must be reallocated to the actual front-desks (the remaining 67 profit centers) in order for the costs to be accurately accounted for. This reallocation process is called "Step 4." The process we have discussed so far where team leads and managers assign allocations to all 74 profit centers is Step 2 (i.e., Step 4 is reallocating the assigned percentages to the dummy profit centers from Step 2); Step 3 is not addressed in this project.

For example, suppose Table 6 represents the allocations for "Project F" made in Step 2.

Project/Vendor Name	Man Days	Cash	Dummy Profit Center 1	Dummy Profit Center 2	Dummy Profit Center 3	Dummy Profit Center 4
Project F	10	$10,000	10%	20%	30%	40%

Table 6 – Example of Allocations to Dummy Profit Centers from Step 2

The 10% allocated to Dummy Profit Center 1 ($1,000 out of the $10,000 for Project F) has to be reallocated to real profit centers. The process used to calculate the allocation keys for Step 4 is similar to the Blended Keys process where a weighted average is calculated. Team leads input weights and allocation keys to the real profit centers by traits instead of by applications, as shown in Table 7.

Trait	Weight	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
Volume	25%	0%	50%	50%	0%
HC	10%	20%	10%	60%	10%
KPI	55%	10%	10%	80%	0%
Trades	10%	10%	40%	40%	10%

Table 7 – Example of Re-allocation from Dummy Profit Center 1 to Real Profit Centers

In Table 7, these are the inputs of a team lead to reallocate the $1,000 allocated to Dummy Profit Center 1 back to real profit centers. "SUMPRODUCT" is used again to find the weighted averages of these allocation keys, as shown in Table 8.

Project/Vendor Name	Dummy Profit Center	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
Project F	Dummy Profit Center 1	8.50%	23.00%	66.50%	2.00%

Table 8 – Example of Blended Keys for Re-allocation from Dummy Profit Center 1 to Real Profit Centers

The allocations must always total 100%. According to the calculated blended keys seen in Table 8, 8.50% of the $1,000 ($85) allocated to Dummy Profit Center 1 is being reallocated to Profit Center 1, 23% of the $1,000 ($230) allocated to Dummy Center 1 is being reallocated to Profit Center 2, 66.5% of the $1,000 ($665) is being reallocated to Profit Center 3, and 2% of the $1,000 ($20) is being reallocated to Profit Center 4.

This process is repeated for each of the allocation keys assigned to the dummy profit centers for each project.

Project/Vendor Name	Dummy Profit Center	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
Project F	Dummy Profit Center 1	8.50%	23.00%	66.50%	2.00%
Project F	Dummy Profit Center 2	25.50%	58.00%	8.00%	8.50%
Project F	Dummy Profit Center 3	25%	9%	38%	28%
Project F	Dummy Profit Center 4	30%	6%	47%	18%

Table 9 – Example of Blended Keys for Re-allocation from All Dummy Profit Centers to Real Profit Centers

From Table 9, we have 16 allocation keys, but we only want one allocation key to each profit center for Project F. Thus, we sum the products of the blended keys for each profit center by the original allocation keys from Step 2. For example, to find the final allocation for Profit Center 1 for Project F, we multiply 8.50% (the blended key calculated from Dummy Profit Center 1) by 10% (the amount allocated to Dummy Profit Center 1 in Step 2), 25.50% (the blended key calculated from Dummy Profit Center 2) by 20% (the amount allocated to Dummy Profit Center 2 in Step 2), 25% (the blended key calculated from Dummy Profit Center 3) by 30% (the amount allocated to Dummy Profit Center 3 in Step 2), and 30% (the blended key calculated from Dummy Profit Center 4) by 40% (the amount allocated to Dummy Profit Center 4 in Step 2); we then sum these numbers together, as shown in Table 10.

Project/Vendor Name	Man Days	Cash	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
Project F	10	$10,000	25.25%	18.80%	38.38%	17.58%

Table 10 – Example of Reallocated Keys in Step 4

Finally, these new allocation keys for the Step 4 re-allocation are charged to the real profit centers and the dummy profit centers no longer have any charges in them because they have been reallocated.

2.1.4 – Allocations Aggregated by Program

Once the allocation keys are finalized, the ITO Business Management team must submit the charges to the finance department. The charges are aggregated by program (each headcount and discretionary cost corresponds to a program). Suppose the following headcount and discretionary costs in Table 11 belong to Programs A and B.

Program Name	Project/Vendor Name	Man-Days	Cash	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
A	Project G	20	$20,000	20%	10%	60%	10%
A	Vendor B	0	$100,000	50%	0%	50%	0%
B	Vendor C	0	$40,000	15%	30%	20%	35%
B	Vendor D	0	$70,000	55%	20%	10%	15%
B	Project H	35	$35,000	5%	0%	90%	5%

Table 11 – Example of Allocations with Corresponding Programs

To calculate the cash amounts that will be charged to profit centers, we use the "SUMPRODUCT" function again, as shown in Table 12.

Program Name	Cash	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
A	$120,000	$54,000	$2,000	$62,000	$2,000
B	$145,000	$46,250	$26,000	$46,500	$26,250

Table 12 – Example of Allocations Aggregated by Program

For example, the $54,000 being charged to Profit Center 1 from Program A is the summation of the 20% of Project G's $20,000 and the 50% of Vendor B's budgeted $100,000. This is repeated for all programs and all profit centers to be submitted to finance.

2.2 – Problem Statement

The current budget allocation process is complex, time-consuming, and labor intensive; the process lacks a single system to input allocations, an easy way to audit the activity, and a way to track multiple versions of the data sources.

Currently, the budgeting process takes hundreds of man-hours each year as a single system does not exist in which everyone can simultaneously input their allocation keys. Rather, team leads only have the allocation template sent from their manager and little other information. The current process is also an inconvenient and confusing process for team leads and managers to have to filter through the 500 rows of data in the allocation template to find the costs that only correspond to them.

In addition to the inconvenient process for the team leads and managers, the ITO Business Management team identified two other critical problems with the current process. First, keeping track of different versions of the headcount and discretionary costs files is difficult. For example, if there are new hires, employees who leave the bank, or changes in vendors, the ITO Business Management team must update the budget and the allocation template, affecting

everyone's allocation keys as their budgets could have changed with the new versions of these extracts. Tracking these versions and adjusting the budgets, templates, and allocations to the updated data requires a lot of extra time, and there is currently no efficient way to handle these different versions of the data sources (the extracts).

Second, a process does not exist to log the edits made to the allocation template. With the confusion of which profit centers and costs correspond to which team lead or manager, there exists a possibility that team leads and managers charge costs incorrectly. For example, if a team lead were to input allocation keys on the wrong row (e.g., the cost does not belong to her) in the allocation template, this would mean the profit centers are not being charged accurately for that project. A way to track who was making which edits is necessary for the ITO Business Management team, team leads, and senior management to be able to inquire about any concerns or uncertainties they may have.

Chapter III: Methodology

To design the new process, we collaborated with multiple people. We met regularly with the ITO Business Management team to understand their processes, with team leads to understand what their experiences have been with the current system, and with senior management to discuss what requirements they had for the new tool. This chapter describes the steps taken to design and implement the application that was created to address the requirements from each of the parties involved in the budgeting process.

3.1 Objectives

During the first two weeks of the quarter, our team had primarily addressed the concerns and requirements of Andrew Clark, his team, and the ITO Business Management team. But the next few weeks consisted of meetings with other managers, and their requirements significantly added to the scope and complexity of the project. Thus, the project was organized into two phases which was comprised of three deliverables. Phase 1, consisting of Deliverable 1, was what we aimed to accomplish within the eight weeks on-site. Phase 2, consisting of Deliverables 2 and 3, would be begun but documented for a future extension (see Appendix B).

3.1.1 Deliverable 1 Automating General Allocation Process and Corresponding Objectives

The first deliverable was to address the original scope of the project, which were the requirements of Andrew Clark and the ITO Business Management team. This deliverable was to automate the general allocation process described in Section 2.1.1 – create an application for team leads, managers, and the ITO Business Management team to enter allocation keys to charge costs to profit centers and for them to track the edits and versions.

Objective 1: *Create a SQL database consisting of the normalized data and mapping files.*

Objective 2*: Create a Python Server that reads the data sources and inputs the data into the SQL database and maps the data together.*

Objective 3*: Create a front-end where users can input allocation keys, view revision history, and compare different versions.*

Deliverable 1 is the foundation of this budget automation project and the following deliverables are additions to the database and additional features in the front-end.

3.1.2 Deliverable 2 Blended Keys Automated Allocation Process and Corresponding Objectives

The second deliverable was to address the Blended Keys automated allocation process.

Objective 1: *Include functionality in the front-end for the team to assign weights and allocation keys by application.*

Objective 2*: Include functionality in the database to calculate the blended keys based on the user inputs.*

Objective 3*: Include functionality in the front-end to populate the blended keys to projects when a stream is selected under the methodology column.*

The manager of this Blended Keys process had indicated that to allow for flexibility, "Applications" would be referred to as "Keys" and "Stream" would be referred to as "Blended Keys" in the database. This makes it possible for the manager and his delegates to apply this blended key process to other parameters (e.g., if he wanted to allocate by traits instead of applications like in the reallocation process, the database would not restrict him to only applications).

3.1.3 Deliverable 3 Automating Reallocation Process and Corresponding Objectives

The third deliverable was to address the reallocation process for Step 4.

Objective 1: *Include functionality in the front-end for the Operations team to assign weights and allocation keys by traits.*

Objective 2*: Include functionality in the database to calculate the reallocation percentages to the real profit centers based on the user inputs.*

Objective 3*: Include functionality in the database to reallocate the charges from the dummy profit centers to the real ones.*

Objective 4: *Create a view of the allocations by GBLs (a parameter corresponding to profit centers) and include the functionality to take snapshots (baselines) of the data to be able to compare allocations through time.*

3.2 New Process Overview

The new process developed in this project consists of a Python Server, SQL Database, and web-based front-end application. See Appendix C for a visual representation of the new general budget allocation process.

Headcount and discretionary costs extracts will still be pulled from Clarity, but during one of the discussions with the ITO Business Management team, it was suggested that managers update their headcount numbers in Clarity monthly as opposed to quarterly (as done in the current process). With this change, the ITO Business Management team will no longer have to create the budget but can duplicate the monthly forecasts that managers input to become the upcoming year's budget (e.g., if Project A's forecasted headcount cost for January 2016 was $50,000, Project A's budget for January 2017 will be $50,000). After logging into the web-based application, the ITO Business Management team will upload these extracts and mapping files into the database through the front-end. The server will read the data sources and input the mapped data into the SQL database. In addition to no longer manually creating the budget, having a server that automatically maps the data together and creates the allocation template removes two time-consuming steps from the process, significantly reducing manual work required from the ITO Business Management team.

When a user logs in, the application reads the credentials that are specified in the permission table to fetch corresponding data from the database through the server. The permission table defines the access for each user of the application. Previously, users saw all expenses without a defined list of which costs to allocate, which created the risk of allocating incorrectly, but limiting users' access and only showing relevant fields diminishes confusion because users no longer have to sort through hundreds of rows. These permissions are assigned by the ITO Business Management team (given the role of "Admin" in the permission table) and senior management (given the role of "Managers" in the permission table).

Team leads (given the role of "User" in the permission table) will be able to allocate their headcount costs and managers will be able to allocate both headcount and discretionary costs – as well as override their team leads' allocations. The application will verify that all allocations total to 100%; if not, there will be an alert to re-input the allocation keys. Once the allocations are complete, the server will store the allocations into the SQL database.

As team leads and managers input allocations, the edits are tracked in the "Revision History" table which logs all edits to the allocation keys as well as any other activity in the application: updates to the permission able, data source uploads, or edits to the profit centers. Having a revision history table allows for the ITO Business Management team and management to refer back to changes and eases any necessary auditing.

The application will also create a view for the ITO Business Management team to view the allocations by program. This view will be exportable into Excel format in order for ease in submitting the charges to finance.

This application significantly decreases the amount of manual work required from the people involved in this semi-annual process. With the creation of this new system: the complex steps of creating the budget, mapping the data, and creating the allocation template is no longer required from the ITO Business Management team, sorting through hundreds of rows of irrelevant expenses is no longer necessary for team leads and managers, and a simple way to view recent activity is no longer lacking. The server will now replace the labor-intensive steps of the budget allocation process and the ITO Business Management team, team leads, and management will have a single system for this process.

3.3 Technology Tools

To achieve the new process that our team designed for budget allocations (see Appendix E for the sequence diagram of the new process), several technology tools are needed. Our team planned to use Microsoft SQL Server, Python (Flask framework), Model View Controller (MVC) Architectural Pattern, and a front-end (AngularJS, HTML, and CSS) to achieve project objectives. Each of the technology tools covers a major functionality. In this section, we are going to elaborate on each tool we planned to adopt and analyze how they coordinate with each other.

3.3.1 Microsoft SQL Server

Microsoft SQL Server was used as the database engine. SQL Server is a stable, popular, and fast platform for building databases, and it also provides security for the company's data. We were provided with SQL Server (on our desktops) at the beginning of the project.

To build a comprehensive database in Microsoft SQL Server, our team first wrote queries to create tables. These tables capture the parameters that are used in the budget allocation process. Each table contains several column headings, and we specified the data structure of each column heading. We also specified the primary key and the foreign key for every table. By doing so, mapping relationships among tables were properly taken care of. Then, we used SQL Server to see the graphical representation of each table and its corresponding mapping relationships. Finally, we manually wrote queries to insert data into the database.

3.3.2 Python (Flask Framework)

Due to the complexity of this project, we planned to build a back-end for this web-based application. Because we decided to code the back-end in Python, we found that using Flask framework could quicken the development process. Flask framework is written in Python. Figure

1 below shows the simplicity of building a small Flask application (Quickstart). Our project size is much bigger than this small application, but this example is considered as a basic template for the project.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

Figure 1 – An Example of a Minimal Flask Application

Although Figure 1 only represents a minimal Flask application, it displays the most basic structure of a Python server. It shows how the Flask framework gets imported and how the framework renders a web page. That is why we consider it as a template and a starting point for our project. According to our research on this framework, we found that it can be easily maintained and there are numerous extensions that can be utilized. Another benefit of using Flask framework is that it is more compatible with SQL Server than Django framework is. Admittedly, Django is another option for choosing a Python framework. However, based on our team's research, there might be potential problems using Django to talk to the database, such as handling database errors. Because we were new to both Django and Flask, we chose Flask hoping to make the development process easier.

3.3.3 MVC Architectural Pattern

"MVC" represents model, view, and controller. It is possible to let our front-end interact with our Flask back-end directly. However, because we planned to use DevExpress (will be elaborated in the following chapter) to help us build the front-end view of the allocation template, we wanted to adopt MVC Architectural Pattern between the back-end and the front-end. Considering that DevExpress provides user interface control in Visual Studio platform, to realize the MVC pattern, we planned to use ASP.NET.

In this MVC pattern, "Model" grabs data from the Python server, because the server talks to the database directly. "View" binds to the front-end to generate everything a user can see. "Controller" is like a commander of this overall process. It listens to every user request sent from the front-end and talks to the server to find a solution to address it. After the "Controller" figures out how to address the user request, it changes the "Model." The change of the "Model" will update the "View," and the user can see the changes through the front-end. This pattern works as an intermediate role between the server and the front-end, making the data flow easy to maintain.

3.3.4 Front End

This project could not be complete without a front-end. We planned to build the front-end in AngularJS, HTML and CSS. Specifically, AngularJS is a JavaScript framework for building dynamic web pages. Since the web pages we designed are supposed to handle various types of user requests, AngularJS is one of the most convenient tools to address dynamic changes of a web page. As for HTML and CSS, they work together to form the basic structure of a web page. By using HTML, a web page is broken down into many small pieces. Each of these pieces is a building block of the web page, such as a paragraph or an image. By using CSS, we are able to change the style of each piece, like changing colors or specifying font size.

3.3.5 Overview

The four technology tools listed above are the most basic ones we planned to adopt in this project. They are the foundation of this web-based application. Figure 2 provides an overview of these four tools.

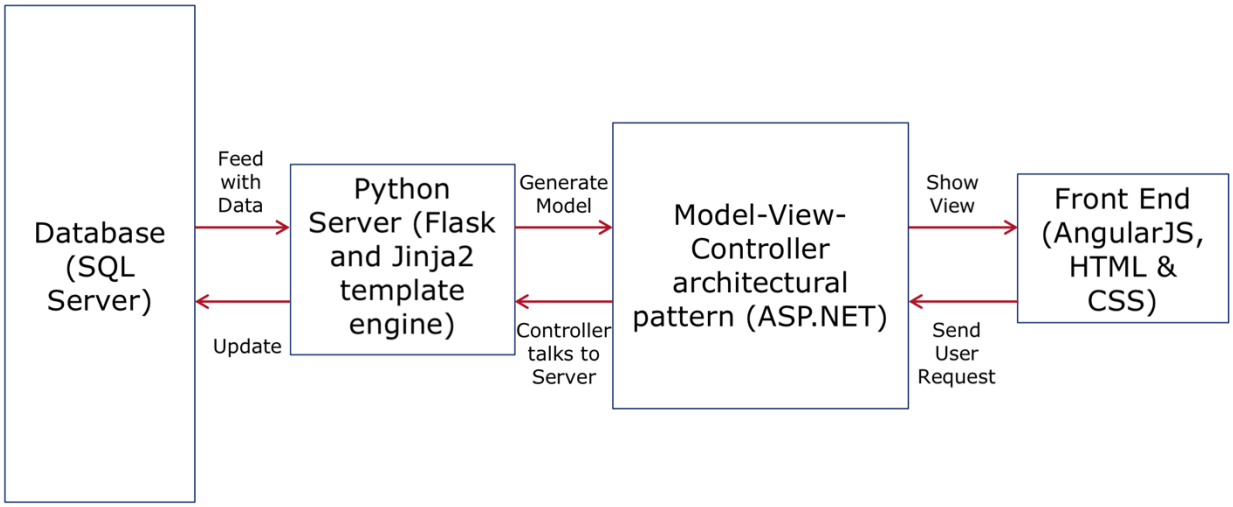


Figure 2 – Structure of this Application

In this application, the database is a pure storage of all the data; the data can come from the Excel files that have been uploaded by the ITO Business Management team or from users' inputs in the front-end. This database feeds the Python server with data, and the server can update the database when a user request comes in. The Model-View-Controller Architectural Pattern serves as an intermediate role between the front-end and the server. Model binds to the server to grab data; View binds to the front-end so the users can see the data; Controller manipulates Model to address different kinds of user requests. Finally, the front-end provides the users with various views and enables them to submit user requests.

Besides these four basic technology tools, there are many more technologies and methodologies that we planned to use. For example, a Python library called Pandas is needed for reading the Excel files. We will talk about the other technologies in Chapter VI.

Chapter IV: SQL Database

Building a maintainable SQL database was the top priority for our sponsor, thus, the majority of the term was spent designing the database and normalizing the data (see Appendix G). In the database was data from our three extracts (headcount costs from Clarity, discretionary costs from Clarity, and applications (keys) from Bam+), the mapping relationships provided from the ITO Business Management team, tables to store users' inputs from the front-end, a permission table, and a logging table.

4.1 Database

The first several versions of the database design were made in Microsoft Publisher, but the final design was created using SQL Server. The tables were also created in the server and the data was manually loaded through INSERT statements.

4.1.1 Un-Normalized Database Design

Originally, our plan was to create exact copies of the data from the extracts into the database (see Appendix H; e.g., there would be one table to store all 43 columns of data from the headcount extract), but we realized this made the database difficult to maintain and it had to be normalized. For example, one project could have multiple resources and each resource's time was allocated for every month of the year. Therefore, for just one project, one resource (identified by a "resource code") could account for 12 rows of the extract for each month. With hundreds of resources and hundreds of projects, many of the values would be repeated in the database.

Area	Country	Cost Center	Project Name	Charge Code	Resource Code	MM/YYYY	Forecast MD
North America	United States	AB	Project A	ABC	1234567	10/2016	14
North America	United States	AB	Project A	ABC	1234567	11/2016	14
North America	United States	AB	Project A	ABC	1234567	12/2016	14
North America	United States	AC	Project A	ABD	2345678	01/2016	10
North America	United States	AC	Project A	ABD	2345678	02/2016	12
North America	United States	AC	Project A	ABD	2345678	03/2016	10

Table 13 – Example of Rows from Headcount Extract

There are 43 columns in the headcount extract and the columns in Table 13 are a few of them. For Project A alone, there can be hundreds of rows corresponding to the project depending on how many resources are working on the project. As exemplified in the table, certain values such as "United States" are repeated multiple times because it corresponds to this one project. In fact, all projects in the headcount extract belong to North America; therefore, "North America" in the "Area" column is repeated over 9000 times in the extract. If we were to copy the extract into the database as is, and "North America" was to be changed to "N. America," then 9000 rows would need to be changed. This illustrates how if we were to stick to our original database designs of

taking pure copies of the extracts, many values would be repeated in our database making it difficult to maintain.

Although each of the tables in our original database designs had to be broken up into independent tables, the relationships represented in Appendix H still apply. The logging table in the database tracks recent activity made in the front-end, and the permission table is a list of all users of the application and what they have access to. As mentioned in Section 3.2, this permission table defines what the user will be able to see and edit in the front-end to avoid the risk of incorrectly allocating costs.

In this version of the database design, there are still pure copies of the extracts from Clarity and Bam+. The "Keys Data" table contains data from the applications used in the Blended Keys allocation process; this data is from Bam+ and from team leads. The "Headcount Data" and "Discretionary Costs Data" tables are pure copies of the headcount and discretionary costs extracts from Clarity.

There are four mapping files, provided by the ITO Business Management team, to represent the mapping relationships between parameters in the data. The headcount extract does not have "Nature," "Stream," "SILO," "WIP/NWIP," or "Program Code" as columns; therefore, the ITO Business Management team had to manually input these fields each time they created the budget and allocation template.

Using the cost centers from the headcount extract, we were able to find the Nature (e.g., Regulatory); each cost center could only correspond to one Nature. Given the Resource Codes in the headcount extract, we were able to identify the corresponding SILOs (in later versions of the database design we used Cost Centers instead of Resource Codes to identify both Nature and SILO). Given the Project Codes, we found the corresponding Streams – unlike the relationships between Cost Center and Nature and SILO, a Project Code could correspond to more than one stream. If this was the case, we used the Stream with the highest number of "man-years" which is the number of collective years that the resources are expected to work. Once we knew these streams, we were able to identify the corresponding applications used in the Blended Keys process. The final mapping relationship was linking Charge Codes (codes to identify where to charge) from the headcount extract to WIP/NWIP (used to identify whether a project can be capitalized or not) and Program Code by using the VLOOKUP Excel function in the provided IT Masterfile which contained all projects and the corresponding parameters. Also provided from the ITO Business Management team is a file with all Profit Centers (the front-desks) and the corresponding parameters (e.g., descriptions and owners); this is another table in the database.

The three other main tables in the database design are the Templates – the "HC Allocation Template," "Blended Keys Template," and the "DC Allocation Template." These store users' inputs from the front-end. Notice there are column headings underlined in each table – these identify the primary keys for the tables. As exemplified in each of the three Template tables, multiple column headings are underlined because the combination of these parameters makes the value unique. For example, one project can have multiple corresponding budgets if it belongs to more than one cost center or has multiple charge codes.

Project/Vendor Name	Project Code	Cost Center	Charge Code	Man-Days	Cash	Profit Center 1	Profit Center 2	Profit Center 3	Profit Center 4
Project I	PROJI	AB	ABC	20	$20,000	10%	90%	0%	0%
Project I	PROJI	AB	ABD	10	$10,000	25%	25%	25%	25%
Project I	PROJI	AC	ABC	5	$5,000	30%	20%	20%	30%
Project I	PROJI	AC	ABD	5	$5,000	50%	50%	0%	0%

Project J	PROJJ	AD	ABC	25	$25,000	0%	0%	100%	0%
Project J	PROJJ	AD	ABD	15	$15,000	5%	25%	40%	30%

Table 14 – Projects with Multiple Cost Centers and Charge Codes

Project I in Table 14 has four different budgets because it corresponds to multiple cost centers and charge codes. Therefore, it requires the combination of Project Code – Cost Center – Charge Code to be unique in the allocation template. Suppose a user inputs an allocation key of 10% for Project I, this is not enough information to know which budget Profit Center 1 is being charged 10% of. We need to have "PROJI-AB-ABC" to identify the budgeted $20,000 to appropriately charge the profit centers. As headcount costs require the combination of Project Code – Cost Center – Charge Code to be unique, discretionary costs require the combination of Vendor Name – Cost Center – Charge Code – Nature, and keys only required Key Name.

In addition to needing these combinations to identify the row in the allocation template (which identifies the budget), when team leads and managers allocate their costs, they allocate for both the current year and the upcoming year; therefore "Year" must also be a part of the primary key combination to identify the row and budget. Because we are storing the allocation keys that are being assigned to the profit centers, "Profit Center" must be included in the primary key as well. Therefore, if we wanted to know what the team lead of PROJJ-AD-ABD allocated to Profit Center 3 for 2016, the primary key we would need is 2016-PROJJ-ADABD-Profit Center 3. Supposing Table 14 is for 2016, we would be able to identify "40%" as the allocation key. Therefore, the combination of columns required to be the primary key in the headcount allocation template is Year – Project Code – Cost Center – Charge Code – Profit Center; the combination required to be the primary key in the discretionary costs allocation template is Year-Vendor Name – Cost Center – Charge Code – Nature; the combination required to be the primary key in the Blended Keys allocation template is Year – Key Name – Profit Center. (In Appendix H "Step" is also an underlined column heading in the Template tables, however, this pertains to Deliverable 3 which was not addressed during this phase.)

4.1.2 Normalized Database Design

As mentioned in Section 4.1.1, the database needed to be normalized for easy maintenance. Normalization is the process of efficiently organizing data to avoid redundancy in data (Chapple, 2016; e.g., repeating "North America" 9000 times). This required understanding the relationships between all the parameters in the extracts to find out which parameters were independent and which were dependent. This required studying the data in great detail.

It can be seen in Appendix G that there are significantly more tables in the new database design (there are now 44 tables) compared to the original database design in Appendix H, but each table now has fewer columns compared to the tables that were pure copies of the extracts. When normalizing data, values are given a unique ID, as exemplified by Table 15.

ID	Area
1	North America

Table 15 – Example of an Independent Table in the Database

Area, a column in the headcount extract, is an independent parameter. Because there was only one "Area" (i.e., all projects belonged to "North America") there was only one value in the table. If there were other "Areas" they would also been given an ID (e.g., Europe could have an ID of 2 and Asia could have an ID of 3). Note that the "ID" columns in each of these tables are the

primary key. Although "Area" is an independent value, there were parameters that were dependent on the field. As illustrated in the database design, the Region (Area) table is linked to "CountryDepartmentLvl" which is one of its dependencies.

ID	Name	Department Level	Area ID
1	United States	United States	1
2	Canada	Canada	1

Table 16 – Example of a Dependent Table in the Database

Table 16, is the "CountryDepartmentLvl" table in the database which includes the country that the projects belonged to and the corresponding Department Level 1. If we first look at the "Name" column, which is for Country Name, there are only two possible values because United States and Canada are the only two countries the projects in this extract belonged to (since the only "Area" all projects belonged to is North America). "Department Level," another parameter in the headcount extract, was dependent on the Country. There was a one-to-one relationship between the parameters (i.e., a country could only have one corresponding department level; United States under "Country" can only correspond to "United States" under "Department Level" and nothing else). Because of this one-to-one relationship, we are able to put the two columns under the same table and assign them the same ID number. Thus, if we know the ID is 1 for the "CountryDepartmentLvl" table, we automatically know that the country name is United States and that the Department Level is also United States.

Because both Country and Department Level are dependent on the Area, "Area ID" is a column in this table as well. This means that for each country and department level, there can only be one corresponding Area. Note that Area ID is a foreign key in the CountryDepartmentLvl table because the ID is used to refer to a value in another table.

If there was not a one-to-one relationship between either Country Name or Department Level with Area, then Area ID would not be a column in the table. Suppose Europe (given an ID of 2) and Asia (given an ID of 3) were other Areas listed in the extract and there was not a one-to-one relationship between the Country Name or Department Level values with Area (note this is not a realistic example given United States and Canada would only belong to North America).

ID	Name	Department Level	Area ID
1	United States	United States	1
2	United States	United States	2
3	United States	United States	3
4	Canada	Canada	1

Table 17 – Example of a Table with Redundant Data

Table 17 exemplifies the table if United States corresponded to more than one Area. "United States" would have to be listed multiple times to accommodate for the multiple corresponding Areas. This repetition is an example of the redundant data we are trying to avoid through normalization; therefore, when normalizing these tables, the columns should have one-to-one relationships.

All the data from the extracts were analyzed to confirm which parameters had one-to-one relationships in order to normalize the database to a reasonable level. The database design in Appendix G was the final design of the SQL database. As mentioned, the relationships from the

Un-Normalized Database still apply for the normalized version; however, there are minor changes such as the use of the IDs and the Template tables now refer to additional tables.

4.2 Data Loading Complexities

As mentioned, the data in the database was manually loaded using INSERT statements. However, this process had proven to be difficult for some tables because the Clarity extracts had missing data that needed to be accounted for. For example, not all charge codes are printed in the extract, as exemplified in Table 18.

Project Code	Cost Center	Charge Code	MM/YYYY	Forecast MD	Actual MD	Forecast MY	Actual MY
PROJK	AB	7CW4381	11/2016		4.86		0.022086
PROJK	AB	NA	11/2016	1.63934		0.007452	
PROJK	AB	7CW4381	12/2016		2.7		0.01227
PROJK	AB	NA	12/2016	1.69399		0.0077	
PROJK	AB	NA	01/2017	1.63934		0.007452	
PROJK	AB	NA	02/2017	1.69399		0.0077	
PROJK	AC	7CW4382	11/2016		14.6		0.06636
PROJK	AC	NA	11/2016	16.93989		0.077	

Table 18 – Missing Data in Clarity Extract

"NA" is printed in the "Charge Code" column in a majority of the headcount extract's 9000 rows. For months that have already passed, instead of printing the Forecast MD and Forecast MY values on the same row as the Actual MD and Actual MY values (even though it corresponds to the same resource for the same Project Code – Cost Center – Charge Code), the forecasts are printed on a separate row without the charge code, and future months without Actuals also receive a charge code of "NA." If the data was printed accurately, the data would look like Table 19:

Project Code	Cost Center	Charge Code	MM/YYYY	Forecast MD	Actual MD	Forecast MY	Actual MY
PROJK	AB	7CW4381	11/2016	1.63934	4.86	0.007452	0.022086
PROJK	AB	7CW4381	12/2016	1.69399	2.7	0.0077	0.01227
PROJK	AB	7CW4381	01/2017	1.63934	-	0.007452	-
PROJK	AB	7CW4381	02/2017	1.69399	-	0.0077	-
PROJK	AC	7CW4382	11/2016	16.93989	14.6	0.077	0.06636

Table 19 – Clarity Extract without Missing Data

To find the charge code, we had to use the charge code in the row above and sometimes the row below. Thus, to load the data, we had to write a script that would print the appropriate charge code.

Besides missing charge codes, there were other fields that were missing such as values in the Org Chart Level columns. These are used to identify resources' corresponding business units. Many missing fields had to be manually input.

4.3 Test Cases

To ensure the data was loaded correctly and that the script we wrote for the missing charge codes was accurate, we created test cases to check our data. We tested that mapping relationships were accurate (e.g., cost centers were linked to the correct natures, streams were mapped to the correct project code, etc.) and that budgets were aggregated accurately (i.e., the amounts that are required for the templates). Table 20 provides a sample of these cases.

Table(s)	Year	Test	Data Sources	SQL
HC Allocation Template	F16	MYs for ProjectA-ChargeCode1-CostCenter2	0.4875	0.4875
HC Allocation Template	B17	MDs for ProjectA-ChargeCode2-CostCenter3	220	220
HC Allocation Template	F16	MDs for ProjectB-ChargeCode1-CostCenter2	196.058	196.058
HC Allocation Template	B17	MYs for ProjectC-ChargeCode4-CostCenter4	0.4227	0.4227
HC Allocation Template	F16	MDs for ProjectD-ChargeCode5-CostCenter5	70.973	70.973
HC Allocation Template	F16	MYs for ProjectE-ChargeCode6-CostCenter2	0.4425	0.4425
HC Allocation Template	F16	MDs for ProjectE-ChargeCode3-CostCenter1	260.02	260.02
HC Allocation Template	B17	MYs for ProjectE-ChargeCode4-CostCenter8	0.7499	0.7499
HC Allocation Template	B17	MYs for ProjectF-ChargeCode1-CostCenter7	0.9999	0.9999
HC Allocation Template	F16	MDs for ProjectF-ChargeCode4-CostCenter5	220	220

Table 20 – Examples of Test Cases

We created 70 test cases to check the data in our SQL database compared with the data from the extracts (the data sources). All the tests that were completed passed.

Chapter V: System Analysis

We consider the whole application as a giant system. This chapter is an analysis of the whole system. We want to focus on data flow (see Appendix F for the data flow diagram), as everything we designed is in consideration of data transfer from the front-end to the database or vice versa. First, we discuss the system components. To address various user requests, we broke the whole system down into several key components. Second we discuss the front-end design as it is based on the system components.

5.1 System Components

System components are the key elements that form the whole system. Our team designed these components in consideration of all the user requests anticipated by the stakeholders. The system components specify the data flow in this web-based program. Since data binding between the server and the front-end was not the major concern in this project, the data flow only concentrates on the communication between the database and the front-end.

There are twelve system components that we created. Out of these twelve system components, some of them are back-end based. They represent back-end procedures. The rest are front-end based. Each of them specifies the design of a single web page. Table 21 below shows all the system components. The table breaks all the system components into front-end based components and back-end based ones.

Front End Based	Back End Based
Startup	Authentication
Menu	Authorization
File Upload	File Read
Edit Profit Center	Generate Allocation Template
Edit Allocation Percentage	Save Allocation Percentage
Permission Edit	Revision

Table 21 – All the System Components

We provide details on each system components in the order of process.

5.1.1 Startup

Startup is a front-end based system component. There is a web page that covers this component. Table 22 shows the basic elements of this web page.

Elements	Details
User ID	The company employee ID such as AB6666.
Password	A string which consists of numbers, uppercase and lowercase characters, and all special characters.
Button	The action is clicking on the button to submit the user ID and password for authentication.

Table 22 – Basic Elements of Startup

5.1.2 Authentication

Authentication is a back-end based system component. It tells the system whether the login information refers to a valid user or not. For testing purposes, we store the (fake) password

into the Permission Table in the database. Figure 3 displays all the processes that belong to Authentication.

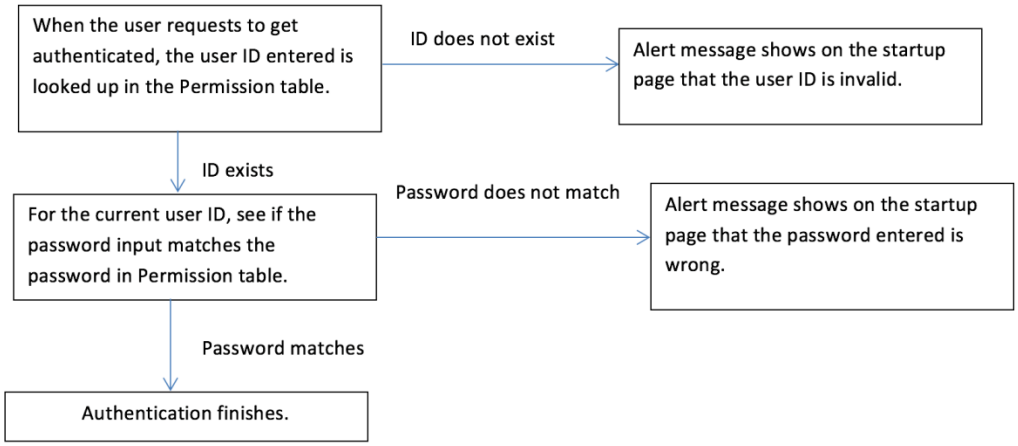


Figure 3 – Authentication Process

5.1.3 Authorization

Authorization is a back-end based system component. Authorization starts when the system checks Employee ID in the Permission table in the database. Figure 4 shows the processes that Authorization contains.

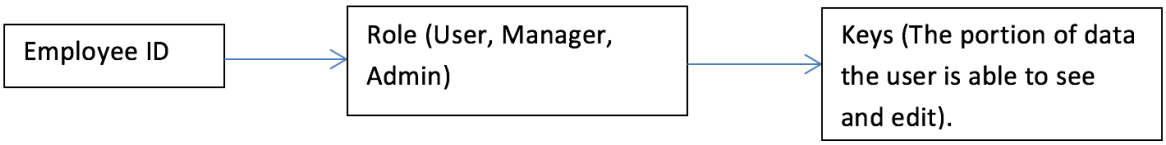


Figure 4 – Authorization Process

5.1.4 Menu

The menu is a front-end based system component. It contains two major elements, and there is a web page that covers this component. Table 23 shows the elements that the menu contains.

Elements	Details
Tab	A group of tabs on top of the website contains a menu of "Main Menu", "File Upload", "Allocation Table", "Profit Center", "Permission Change", and "Revision History".
Button	A group of buttons contains a menu of "Main Menu", "File Upload", "Allocation Table", "Profit Center", "Permission Change", and "Revision History".

Table 23 – Basic Elements of Menu

5.1.5 File Upload

File Upload is a front-end based system component. This system component is only for admin users. There is a web page that covers this component. Table 24 shows its major elements.

Elements	Details
Upload box	There are 9 upload boxes in total for 9 files to upload (Headcount Data, Discretionary Cost Data, Keys Data, Cost Center & Nature, Project Code & Stream, Silo & Cost Center, Charge Code & WIP/NWIP & Program Code, Activity Owner & Cost Center Owner, and Vendor & Stream)
Button	The action is clicking on the button to submit the file. An error message is shown when the button is clicked but no file been selected.

Table 24 – Basic Elements of File Upload

5.1.6 File Read

File Read is a back-end based system component. The functionality of this system component is to read the files that have been uploaded by admin users. All the columns from the Excel files are assigned to a specific variable (as an object) and these variables are passed into the database.

5.1.7 Edit Profit Center

Edit Profit Center is a front-end based system component, and it is only for admin users. There is a web page that covers this system component. Table 25 shows its major elements.

Elements	Details
Existing Profit Center Table	A table contains all the existing profit centers from the Profit Center table in the database.
Field edits button	There are six buttons in total. Click on one of the buttons to trigger a pop out area (consisting of a field table, two text boxes and a submit button) for users to input any possible changes for one of the fields.
Field table	This table is for admin users to see all the values of a certain field and look up a certain value if any necessary changes needed to be made. This field table is obtained by look up relevant table in the database based on users' selection.
Text box	There are 2 text boxes for admin users to input both old value and new value of a certain point in the field table.
Submit button	Action is clicking on the button to submit changes on a certain field. After the action is made, the new value get updated (update statement) in the database, and the existing profit center table gets refreshed.

Table 25 – Basic Elements of Edit Profit Center

5.1.8 Generate Allocation Template

Generate Allocation Template is a back-end based system component. It provides a user with the allocation data that corresponds to his or her permission, and it combines data for both headcount costs and discretionary costs. By sending queries to the database, this component is able to generate views to the front-end. Figure 5 displays the major fields that form the allocation template.

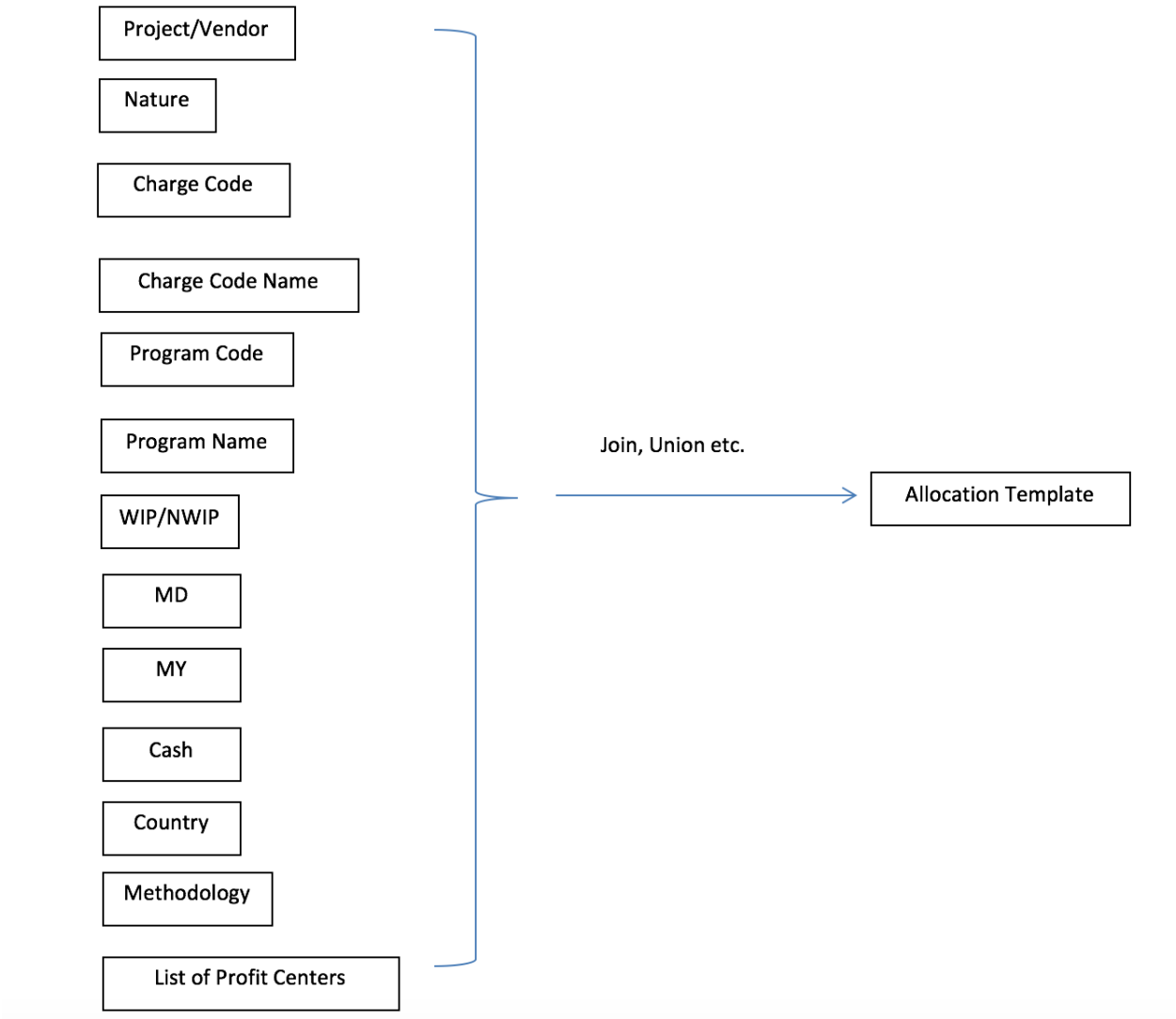


Figure 5 – Major Fields that Form Allocation Template

5.1.9 Edit Allocation Percentage

Edit Allocation Percentage is a front-end based system component. It allows users to make changes to allocation keys. When the percentages allocated across the same "Project Code – Cost Center – Charge Code" does not add up to 100%, an alert box will appear indicating the allocation keys need to be edited. Table 26 shows the major elements of this system component.

Elements	Details
Allocation Template	A view that is from "Generate Allocation Template" system component.
Text box	A text box that allows user to input the allocation percentage.
Button	Action is clicking on the button to submit the change of percent allocated. Refresh the master detail grid after the action is finished.

Table 26 – Basic Elements of Editing Allocation Percentage

5.1.10 Save Allocation Percentage

Save Allocation Percentage is a back-end based system component. To save the allocation percentages entered by the user, the application needs to dive into the database and check whether the allocation percentage is empty based on the key given by the front-end. To be more precise, the job of the application now is to see whether there was a previous input. If this is a first-time input, the Headcount Allocations and Discretionary Cost Allocations tables in the database will be updated. If there was a previous input, a new row will be inserted with the same local key.

5.1.11 Revision

Revision is a back-end based system component. Whenever there is an edit (no matter whether it is for changing an allocation percentage or editing Profit Center), the revision history is stored into the Logging table in the database. Figure 6 shows the overview of Revision.

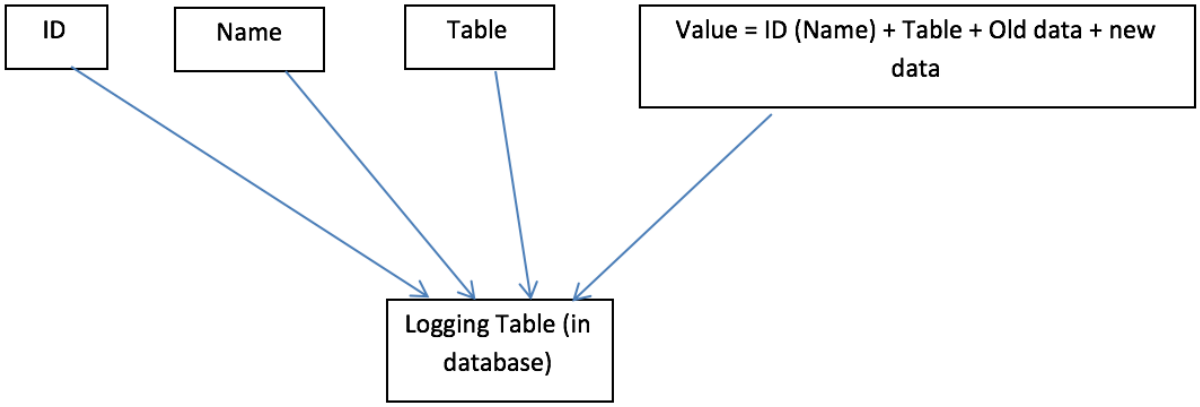


Figure 6 – Overview of Revision

5.1.12 Permission Edit

Permission Edit is a front-end based system component. This system component can only happen when the user is an admin or a manager account. For testing purposes, the admin user is supposed to add the password for a new user (a complete user log in system needs to be built based on the company's security rules). Table 27 shows the basic elements of Permission Edit.

Elements	Details
Current user list	A table contains all the existing users' data from the Permission table in the database except password column.
Text box	There are 6 text boxes in total (for ID, Password, Name, Cost Center, Role, and Keys) for users to input.
Delete button	Action is clicking on the button to delete a current user. The system selects the user ID based on the input specified by the admin account, and then it deletes the user and refresh the current user list.
Add button	Action is clicking on the button to add a new user. The system inserts the user based on the input specified by the admin account, and then it refreshes the current user list.

Table 27 – Basic Elements of Permission Edit

For the delete button, an error message will appear when the user ID text box is empty. For the add button, an error message will appear when any of the text boxes are empty.

5.2 Front End Design

Based on the front-end based system components, designing the front-end interface is straightforward. The following figures illustrate the views for our front-end design.

Figure 7 is the user log in page.

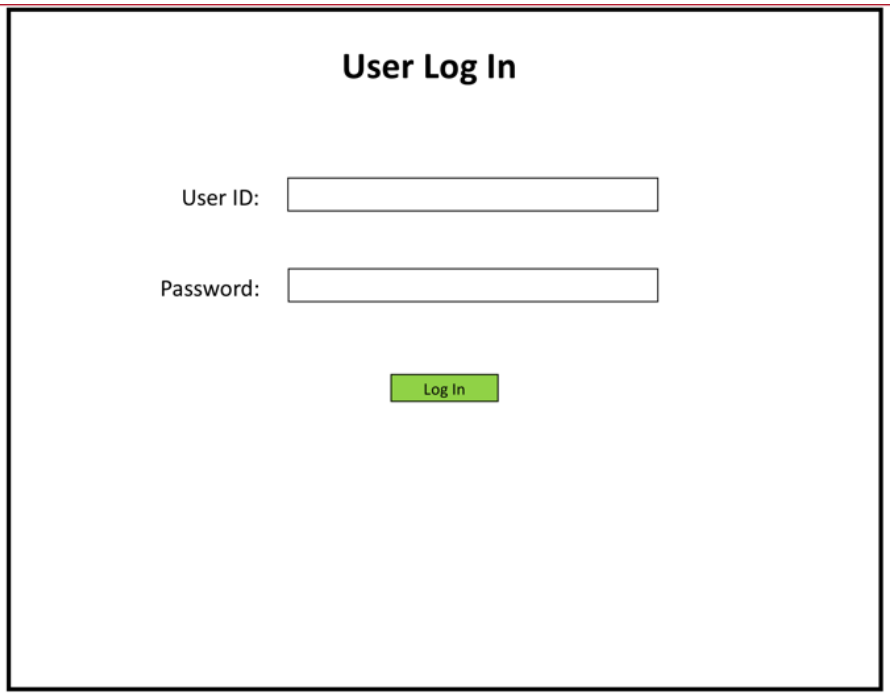


Figure 7 – User Log In Page

Figure 8 is the main menu page. The main menu page is the page that users will see after they successfully log in. It serves as a redirection to all the functionalities we designed. In addition to the tabs on the top of the page, we also display the options in the center to better present all the key features of this application.

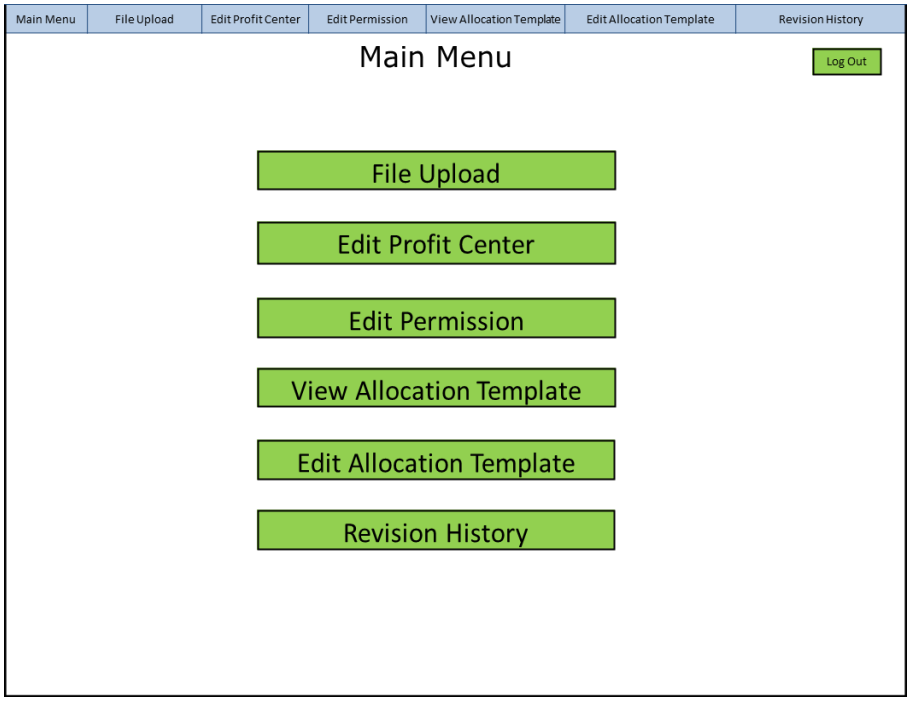


Figure 8 – Main Menu Page

Figure 9 is a web page for admin users to upload files. The first three files on top are data source files. The remaining six files are for the mapping relationships.

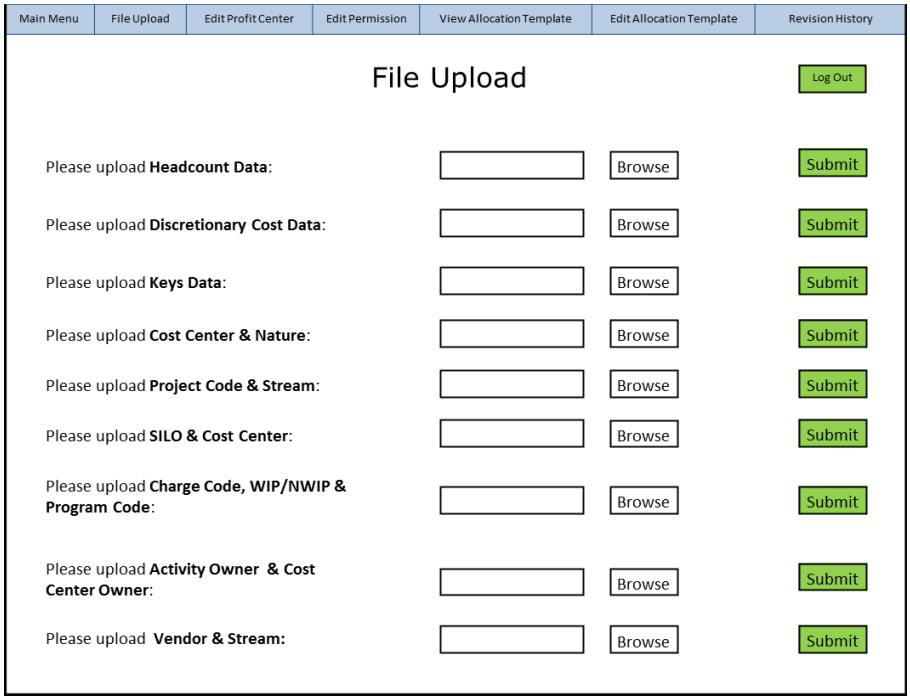


Figure 9 – File Upload Page

Figure 10 is a web page for editing profit centers. When an admin user clicks on a parameter to make an edit, text boxes pop out for the user to input the edits.

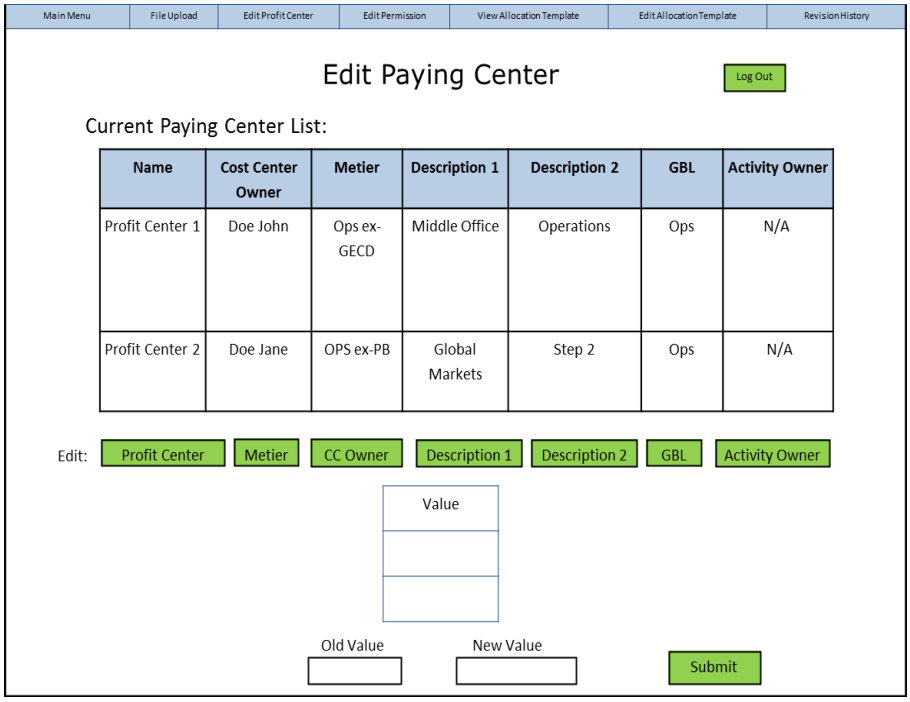


Figure 10 – Edit Profit Center Page

Figure 11 is the Permission Edit Page. The ITO Business Management team and management are able to add a user, delete a user, or edit a user's permissions.

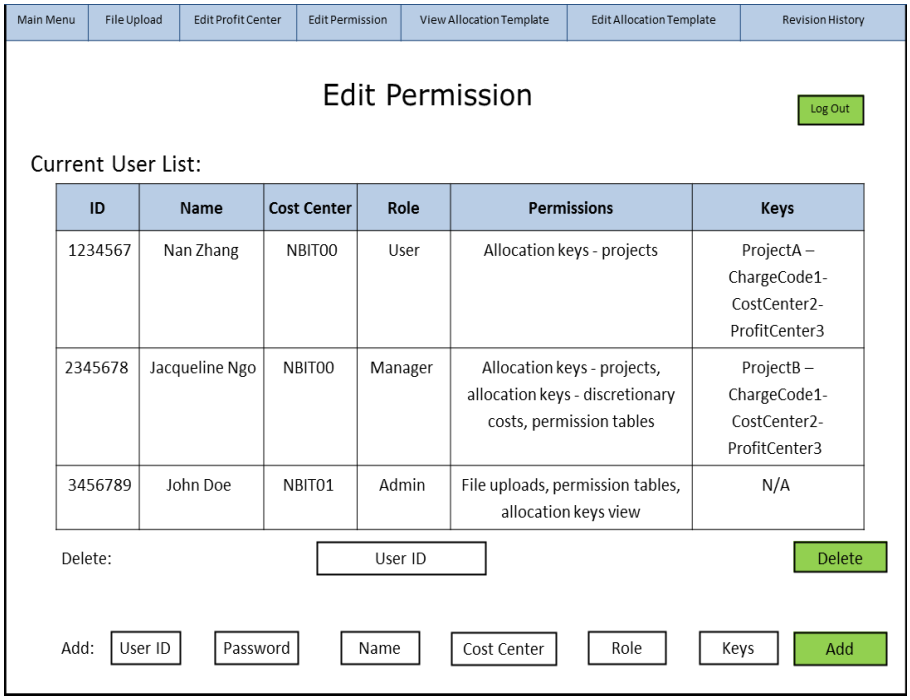


Figure 11 – Permission Edit Page

Figure 12 is the Allocation Keys Edit Page. There are several filters on top for selecting appropriate data. Notice that our team uses master detail grid to structure the rest of the allocation template. Master detail grid is a type of view that contains a parent and a child

category. When a parent category is clicked, the whole view is expanded and an additional view gets displayed. This additional view is the child category. Master detail grid uses the hierarchical relationship to present data.

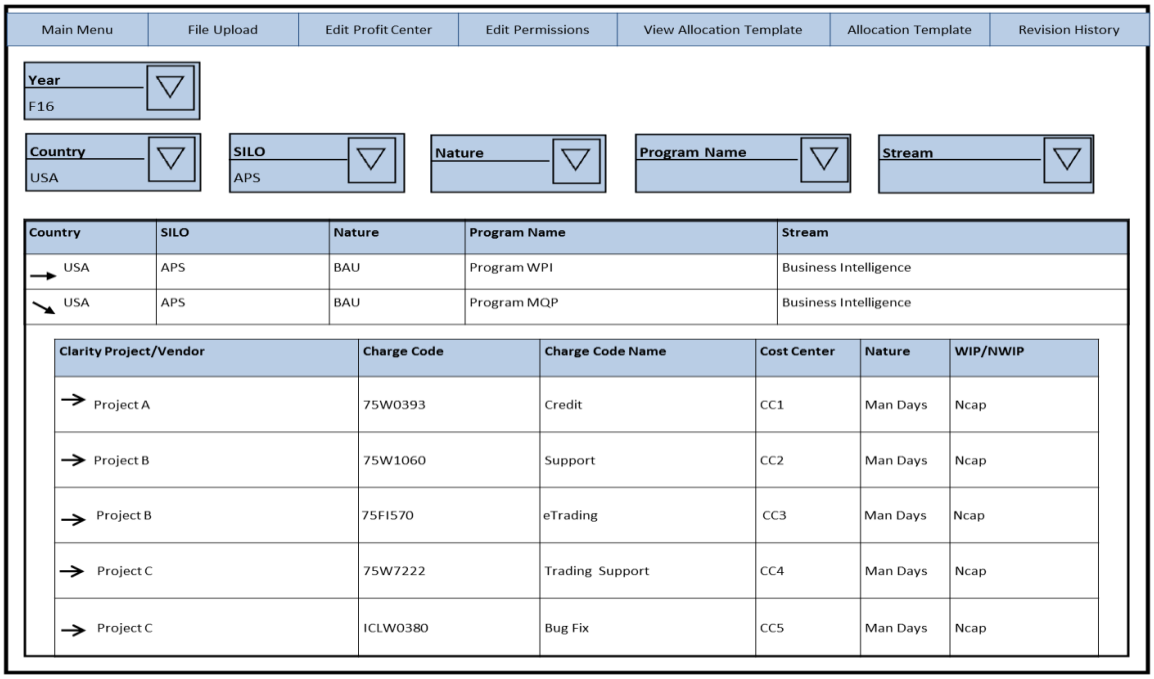


Figure 12 – Allocation Keys Edit Page

Figure 13 is the Allocation Keys View Page. It is still presented by using master detail grid. This view is for the ITO Business Management team to view the numbers they will be submitting to finance.

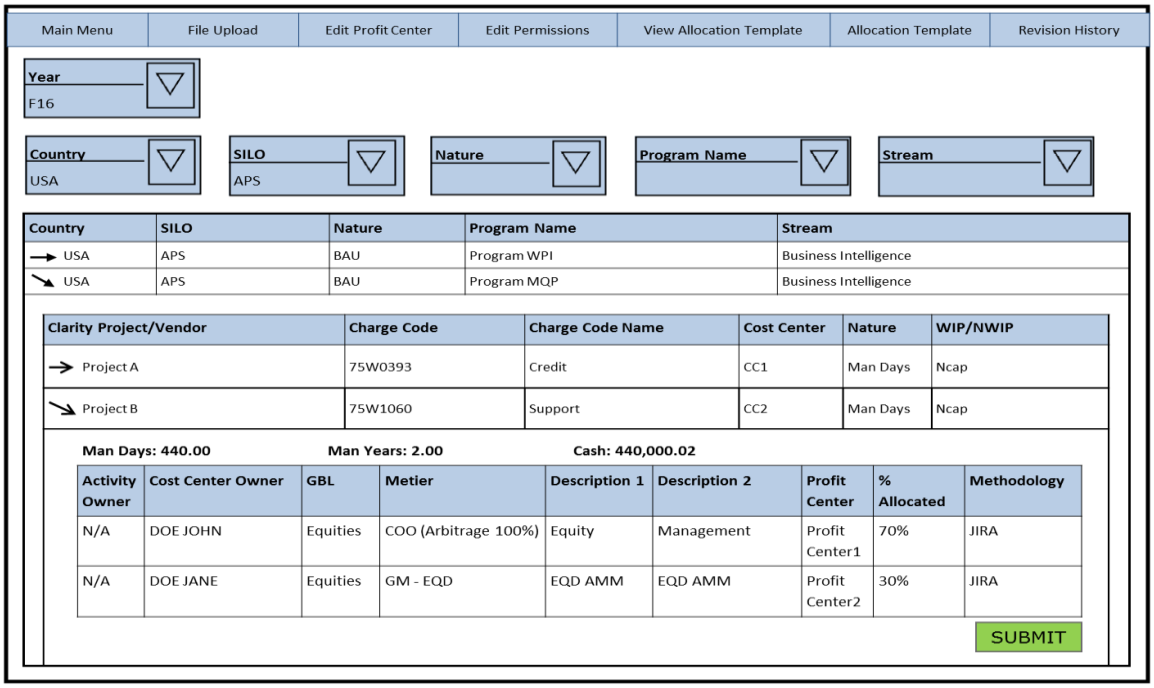


Figure 13 – Allocation Keys View Page

Figure 14 is the Revision History Page. Any edits will be documented here.

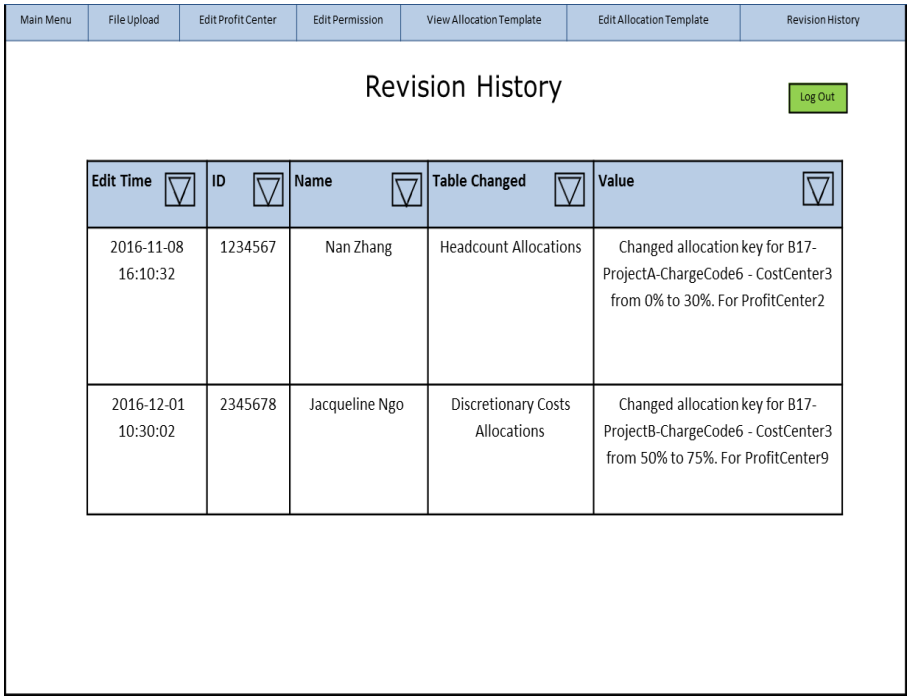


Figure 14 – Revision History Page

Chapter VI: Results

6.1 Progress We Made

Due to the relatively short time frame we had on site, project management was critical to the completion of our project. We wanted to set expectations with our managers and mentors on the deliverables we planned to finish during our time on site and what we planned to be future extensions. Accordingly, this project can be divided into two parts. One part is to design the big picture and to create a database. The other part is towards data binding which requires a solid back-end implementation. As mentioned previously, based on managers' expectations and limited time, data binding was not our major concern for this project. As a result, we concentrated more on the first part, which is to conduct system analysis and build a database. We finalized our progress into three points listed below:

1. Built the database with normalized tables to capture all the required parameters of the process. Designed the tables to fit different kinds of allocation processes. Filled the database with data from Clarity, the project management tool.
2. Delivered the front-end design to properly handle different kinds of user requests.
3. Conducted system analysis on how the application provides various services. Documented the data flow between the front-end and the database.

6.2 Special Features

Besides the three major points we achieved, there are some special features that we implemented or designed. These features are not major technologies or methodologies to this project, but they are very important to this web application.

6.2.1 Versioning

Addressing versioning of the data sources was one of the most important problems we needed to handle. The application we designed is supposed to receive continuous file uploads and should be able to compare the data between different uploads. Because comparing data requires historical data, the database should keep inserting data when a new upload comes in. Instead of updating the rows in the database, the application does not lose history by inserting data. However, that is not enough. In the database, for a row that comes from a past file upload, there is no way that we can figure out its "version." There should be some parameters that work as identifiers of a row in the database.

Our team developed a versioning mechanism that gives every row in the database an identifier. Table 28 illustrates how the versioning mechanism works.

Primary Key	Version Number	Local Key	Data
1	1	1	AA
2	1	2	BB
3	1	1	CC
4	2	1	DD
5	2	2	AA

Table 28 – An Example of the Versioning Mechanism

In Table 28, the primary key column holds the unique identifiers of every row, so the numbers in the column go from one to infinity. The "Version Number" column tells the "version" of that row. To be more precise, whenever a new file upload comes in, the application increments the

version number by one. For instance, say the current version number is 2. When an admin user uploads a new data source only for Headcount data, all the rows generated by this new upload have version number 3 now. Next time when this admin user uploads two files for both Headcount and Discretionary Cost data, the version number goes to 4.

The "Local Key" column only makes sense within two rows with the same version number. For example, when a user makes a change on an existing allocation percentage, the application inserts a new row with the same local key as the row that should be updated. In Table 28, the second row and the fourth row have the same local key and version number. That means a user makes a change of data from "AA" to "CC".

By using a combination of primary keys, version numbers, and local key, the application is able to compare different versions of data, and the old data does not get lost.

6.2.2 Pandas Library

The files the application is expected to read have relatively large sizes; as such, the application's file reading performance was of concern. To address this concern, we suggested the usage of a Python library called Pandas. It reads an Excel file column by column, and it stores each column as an object. By assigning each object a variable, it is easier for us to put these variables into the database. Moreover, using Pandas will improve the speed of reading Excel files.

6.2.3 Bootstrap

Bootstrap is a front-end framework that we planned for constructing the front-end interface. There are many templates that we can use, so there was not much need for us to think about designing the web page using HTML and CSS. As readers may notice from the front-end design, except for the log in page, all the other pages have tabs on top. Bootstrap provides some colorful templates which can help us realize our front-end interface design. A lot of time will be saved.

6.2.4 Master Detail Grid

As illustrated in Figures 14 and 15, a master detail grid is used for presenting allocation data. Our team believes that this kind of view is the best fit for presenting the allocation template, because the allocation template we received from the ITO Business Management team uses filters in Excel. By using a master detail grid, a kind of hierarchical relationship can be clearly displayed. We also planned to use the template from DevExpress for constructing it. In that case, all we need to do is to write our own controller classes.

Chapter VII: Recommendations

7.1 Data Clean up

As mentioned in Section 4.2, Data Loading Complexities, a lot of data was missing in the extracts (e.g., charge codes, organization chart levels, etc.) which made loading the data very difficult. To ease the data load process, the extracts need to be cleaned up. For example, charge codes need to be printed in each row because although we can write script to accommodate for these missing fields, the more rules we write, the messier the process becomes and there is more risk of error.

Another major problem was that many relationships had exceptions to being one-on-one, sometimes due to historical data, which affects the database normalization. For example, a resource should only have one corresponding Resource Type (e.g., employee, contractor, or consultant) but there are sometimes exceptions because a resource's employment status could have changed during the year. Therefore, if the relationship is not one-on-one, a table such as Table 16 would not work and we would need to link the parameters elsewhere. Fixing the exceptions and having a standard rule/standard relationship between the parameters lessens the complexity of the mapping relationships.

7.2 Usage of Clarity API

Another recommendation we had for this project is to use Clarity API instead of reading the uploaded Excel files. API is an abbreviation of Application Program Interface. It sets a method for different program components to communicate. In this project, the method for communicating with Clarity is called Clarity API. Although it may not exist within the company, developing one can be beneficial. If it exists, we suggest using it directly.

Although we introduced Pandas as our Python library to read the files, processing a big file all at once still consumes a large portion of time. This is something that we cannot avoid. In addition, as long as there is a need to read the files, we need to set up some sort of standardization to structure the Excel files. For example, the order of the columns in the data sources should be fixed and there should not be any missing columns that are out of our expectation. Making this kind of standardization is tedious, and it is possible that people may ignore it when they create the data source. Admittedly, this example of setting up standardization can be compromised by designing more intelligent spell checking. However, even with smarter algorithms, some level of standardization or agreement is still needed for prompting the users to follow the rules.

To avoid the issues above, using Clarity API is a good solution. By doing that, the application only needs to request for specific information from Clarity directly. Although it would be difficult to get access to the Clarity API under the company's policy, getting rid of reading Excel files can significantly improve the application's performance.

Chapter VIII: Future Extensions

After completing our term on site, we want to document the next steps for those at BNP Paribas who continue the project. We strived to automate the budget allocation process as much as possible and serve as a starting point for future development. The following four steps are to be followed, and they are sequential.

1. Blended Keys and Reallocation Process. Although we designed the database for all three managers' allocation processes, there is still work that is needed to be done for the keys' allocations (Blended Keys allocation process) and reallocation for the dummy profit centers (Step 4). These two allocation processes also need the system analysis and the front-end design.
2. Data Binding. As previously mentioned, data binding was not the major concern for this project. However, in order to implement this project, the next step should focus on data binding. The future developer needs to write code to achieve the data flow mentioned in the System Analysis chapter. It is important to make sure that the front-end view and the back-end server work together for data transfer. Corresponding testing work is also needed.
3. Password encryption. Currently, for testing purposes, when an admin user adds a new user into the system, he or she needs to create a password for the new user. This should not happen when this application is used by the company. The permission table in the database should also be fixed, since the table has a column for storing users' passwords. A complete password system needs to be built under the company's security policy.
4. Web-based Deployment. This step ensures that all the users have access to it. Once the application is deployed, this will save significant time for all the parties involved in the budget allocation process.

Chapter IX: Conclusion

The Global Markets IT division of BNP Paribas currently conducts a budget allocation process that is complex, time-consuming, and labor intensive. The process lacks a single system to input allocations, an easy way to audit the activity, and a way to track multiple versions of the data sources; thus, our group designed an automated budget allocation process that addresses these bottlenecks to reduce the amount of time required from the ITO Business Management team, team leads, and senior management to complete the process.

The new process consists of a SQL database – which is normalized for easy maintenance, a Python server – which is the back-end of our application, and a web-based front-end application – which is for users to input their allocation keys and for the ITO Business Management team and management to view allocations and revision history for auditing. The new design addresses each of the problems that were identified by management and the ITO Business Management team; it provides a single system for the semi-annual budget allocation process to take place, replaces the labor-intensive steps that were previously required, and makes tasks such as tracking data source versions less complex. In addition to the designs and database, significant documentation and system analysis was provided for future reference and for future extensions of this project.

Bibliography

"Benefits of Microsoft SQL Server." *Acctivate Help*. N.p., 09 Sept. 2015. Web. 07 Jan. 2017. <https://help.acctivate.com/articles/5914/>.

"BNP Paribas International Hackathon 2016." International Hackathon. N.p., n.d. Web. 15 Nov. 2016. <https://international-hackathon.bnpparibas/>.

Chapple, Mike. "The Basics of Normalizing a Database." About.com Tech. N.p., 03 Aug. 2016. Web. 12 Dec. 2016. <http://databases.about.com/od/specificproducts/a/normalization.htm>.

"History: two centuries of banking." History: two centuries of banking. N.p., n.d. Web. 15 Nov. 2016. <https://group.bnpparibas/en/group/history-centuries-banking>.

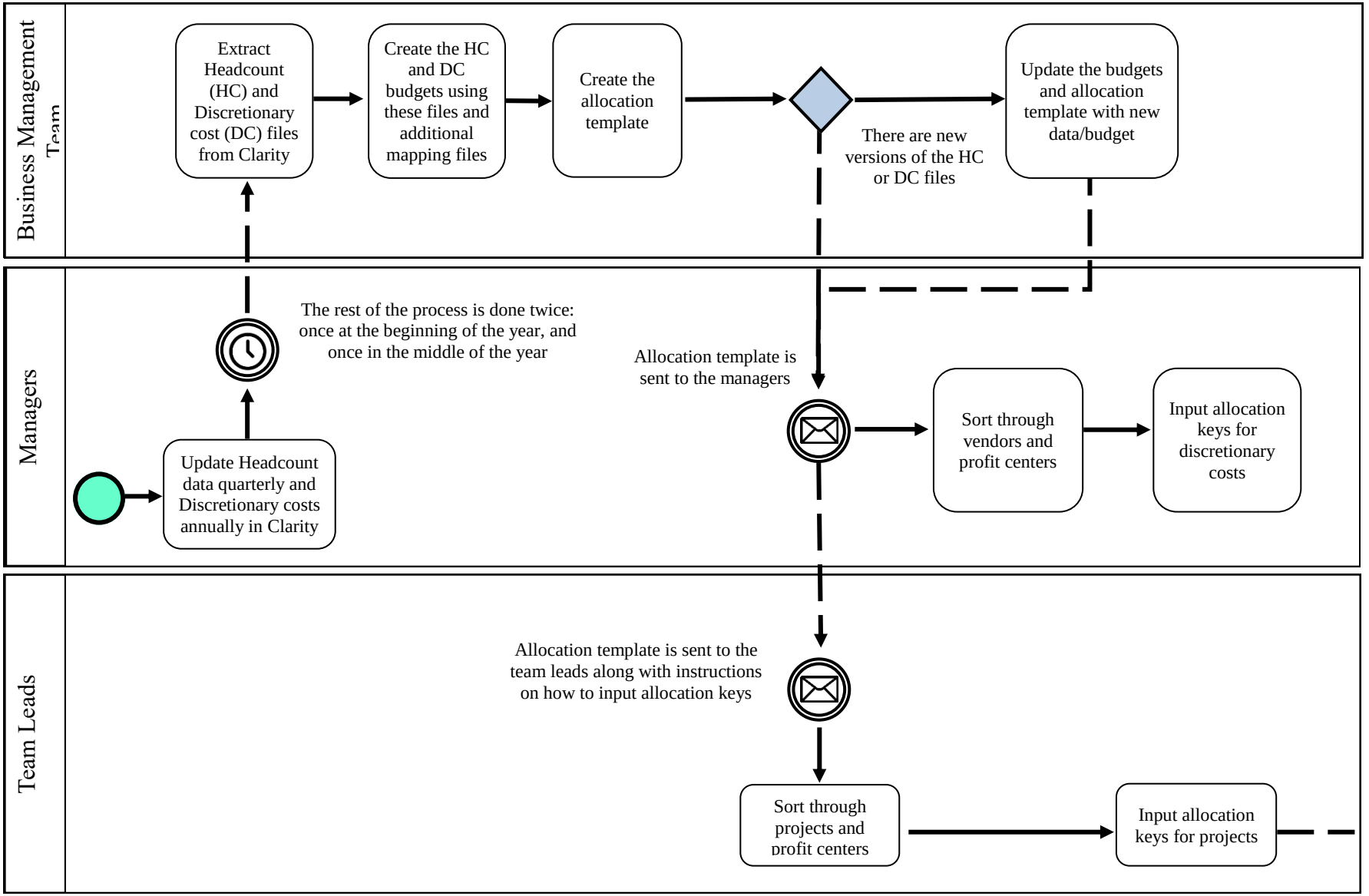
"Innovation: our response to a changing world - BNP Paribas." Innovation : our response to a changing world - BNP Paribas. N.p., n.d. Web. 18 Jan. 2017. <https://group.bnpparibas/en/group/innovation-response-changing-world>.

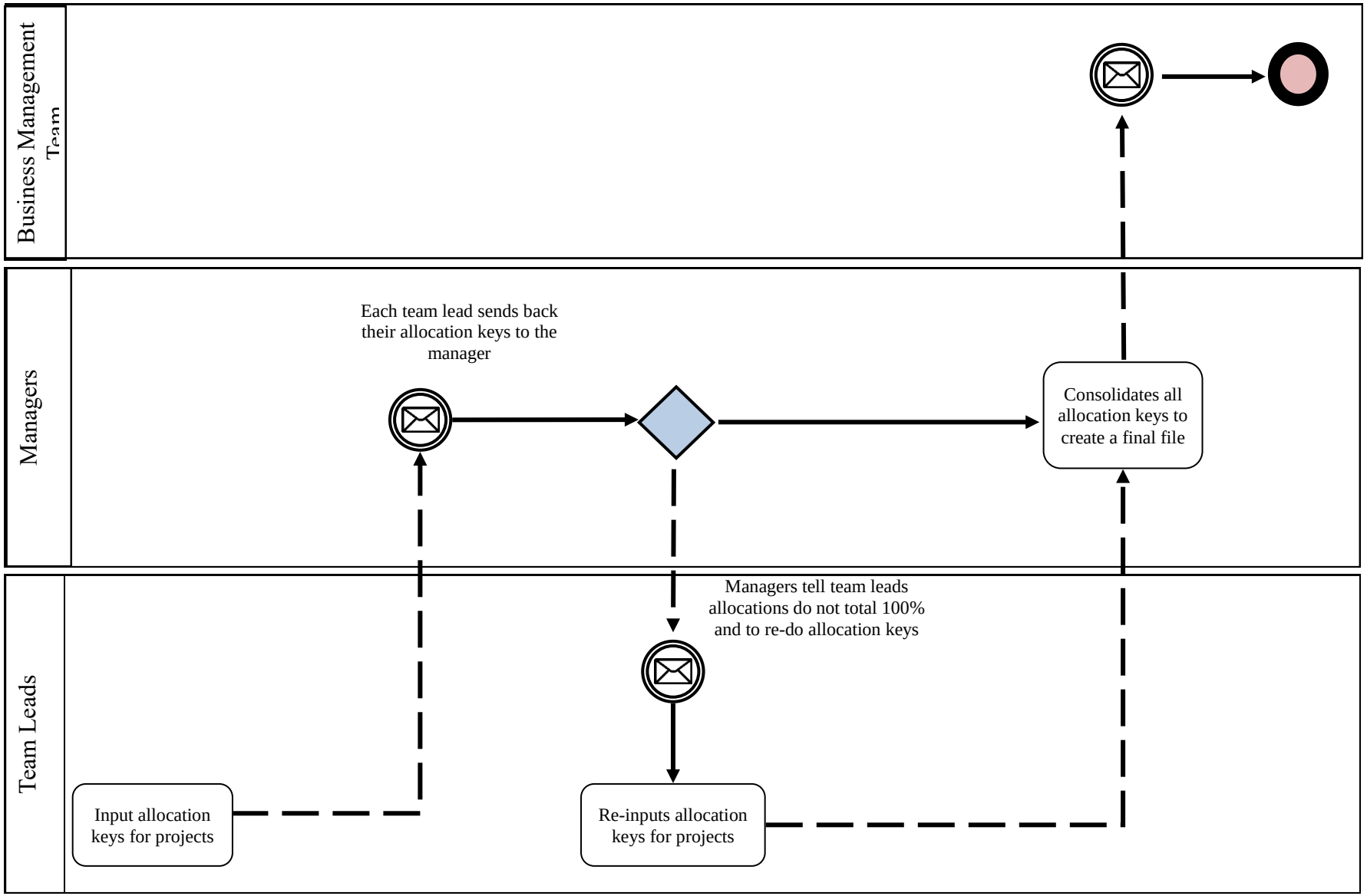
"MS Excel: How to use the SUMPRODUCT Function (WS)." MS Excel: How to use the SUMPRODUCT Function (WS). N.p., n.d. Web. 12 Dec. 2016. <https://www.techonthenet.com/excel/formulas/sumproduct.php>.

"Quickstart." Quickstart — Flask Documentation (0.12). N.p., n.d. Web. 07 Jan. 2017. <http://flask.pocoo.org/docs/0.12/quickstart/#a-minimal-application>.

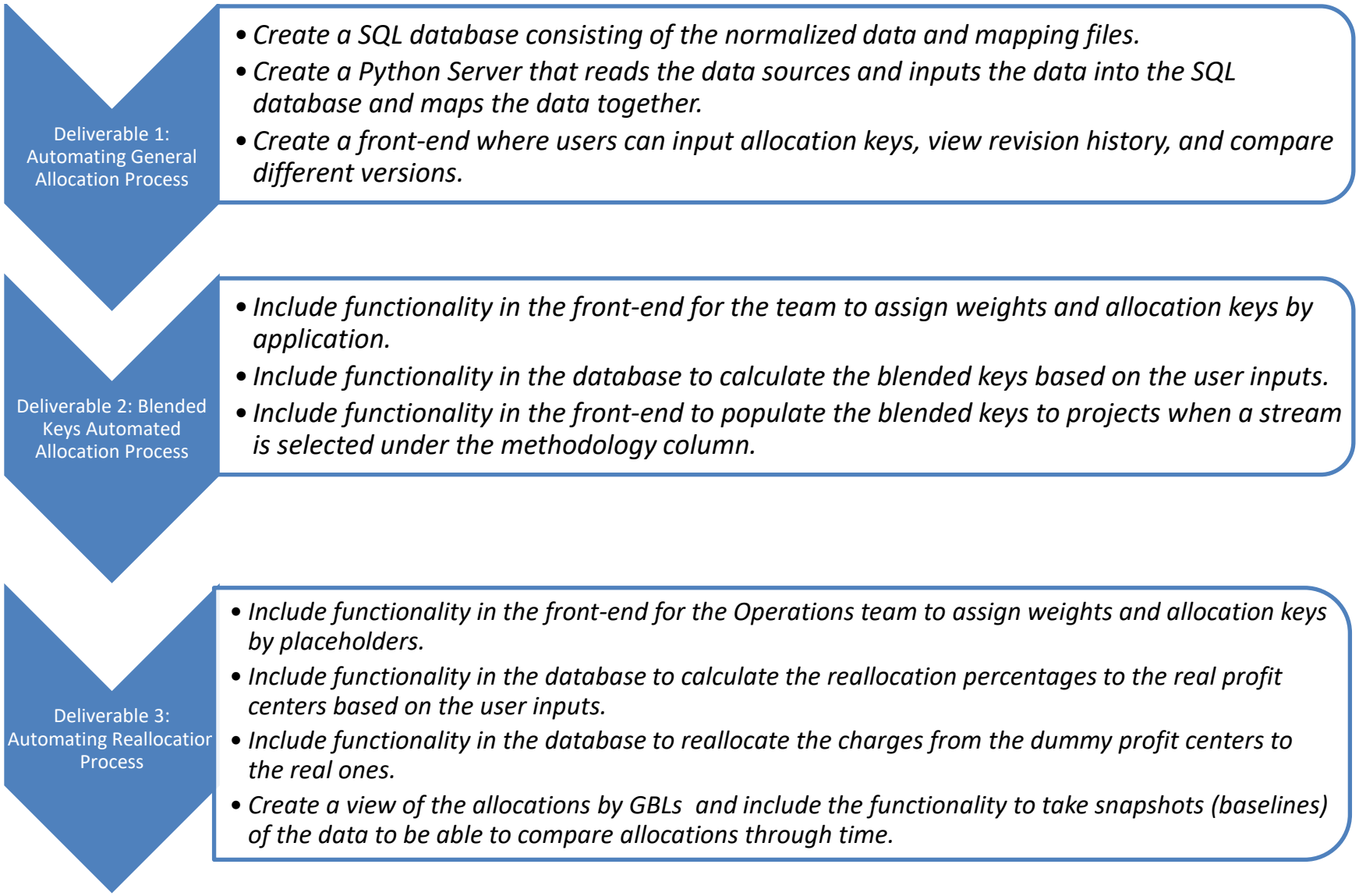
Beal, Vangie. "API - Application Program Interface." What Is API - Application Program Interface? Webopedia. N.p., n.d. Web. 14 Jan. 2017. <http://www.webopedia.com/TERM/A/API.html>.

Appendix A: Current General Budget Allocation Process

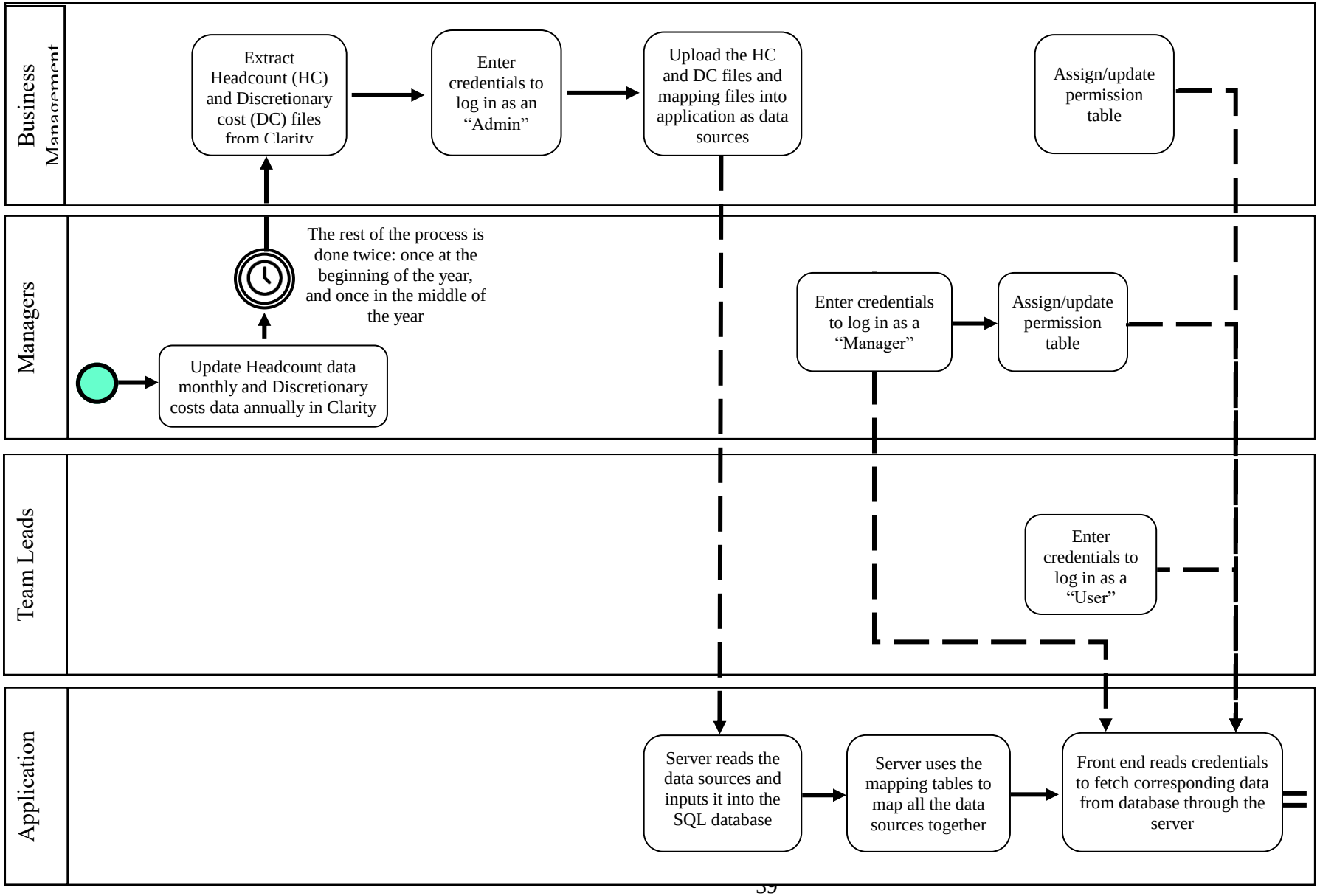


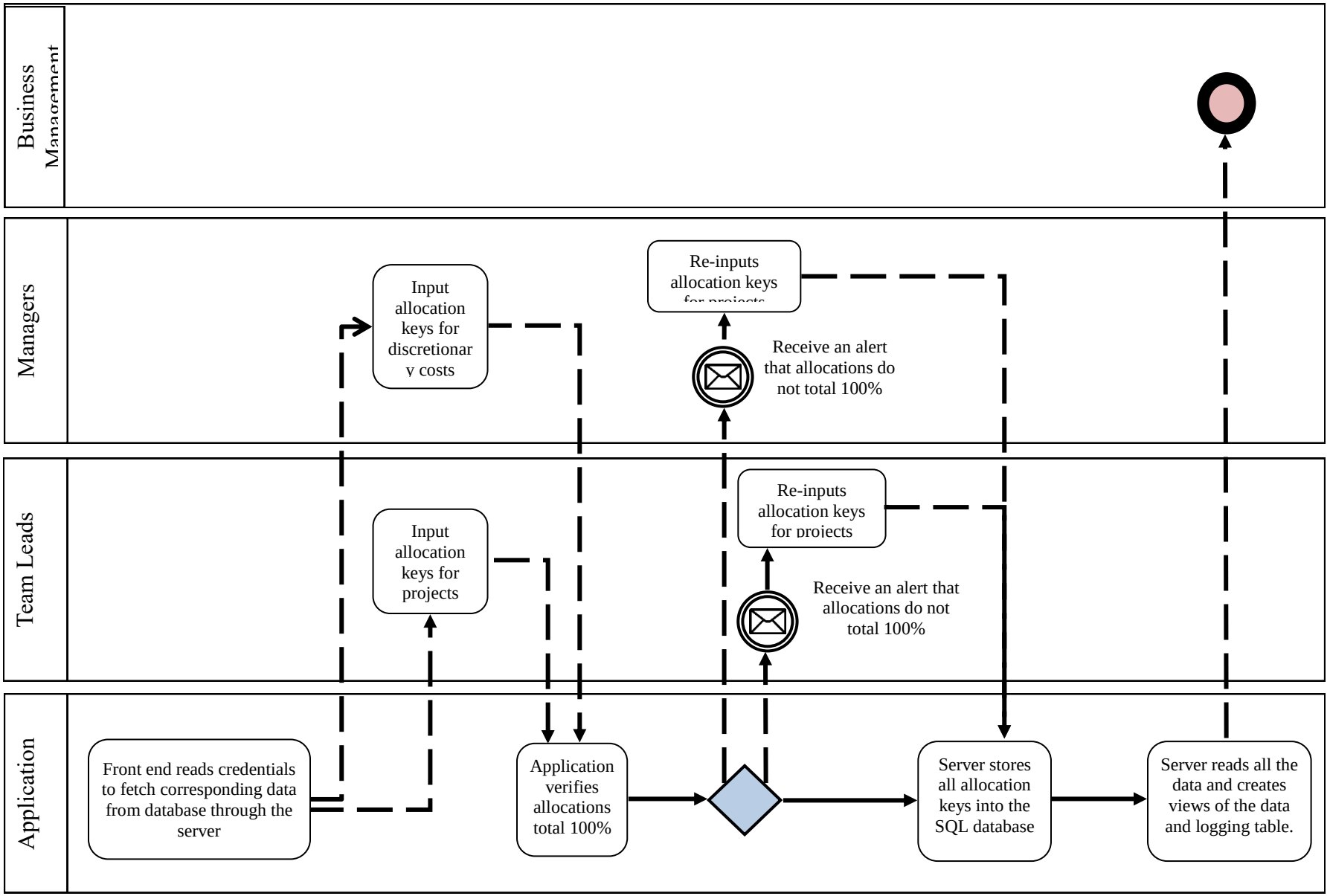

Business Management Team
Managers
Each team lead sends back their allocation keys to the manager
Consolidates all allocation keys to create a final file
Team Leads
Managers tell team leads allocations do not total 100% and to re-do allocation keys
Input allocation keys for projects
Re-inputs allocation keys for projects

Appendix B: Deliverables and Corresponding Objectives



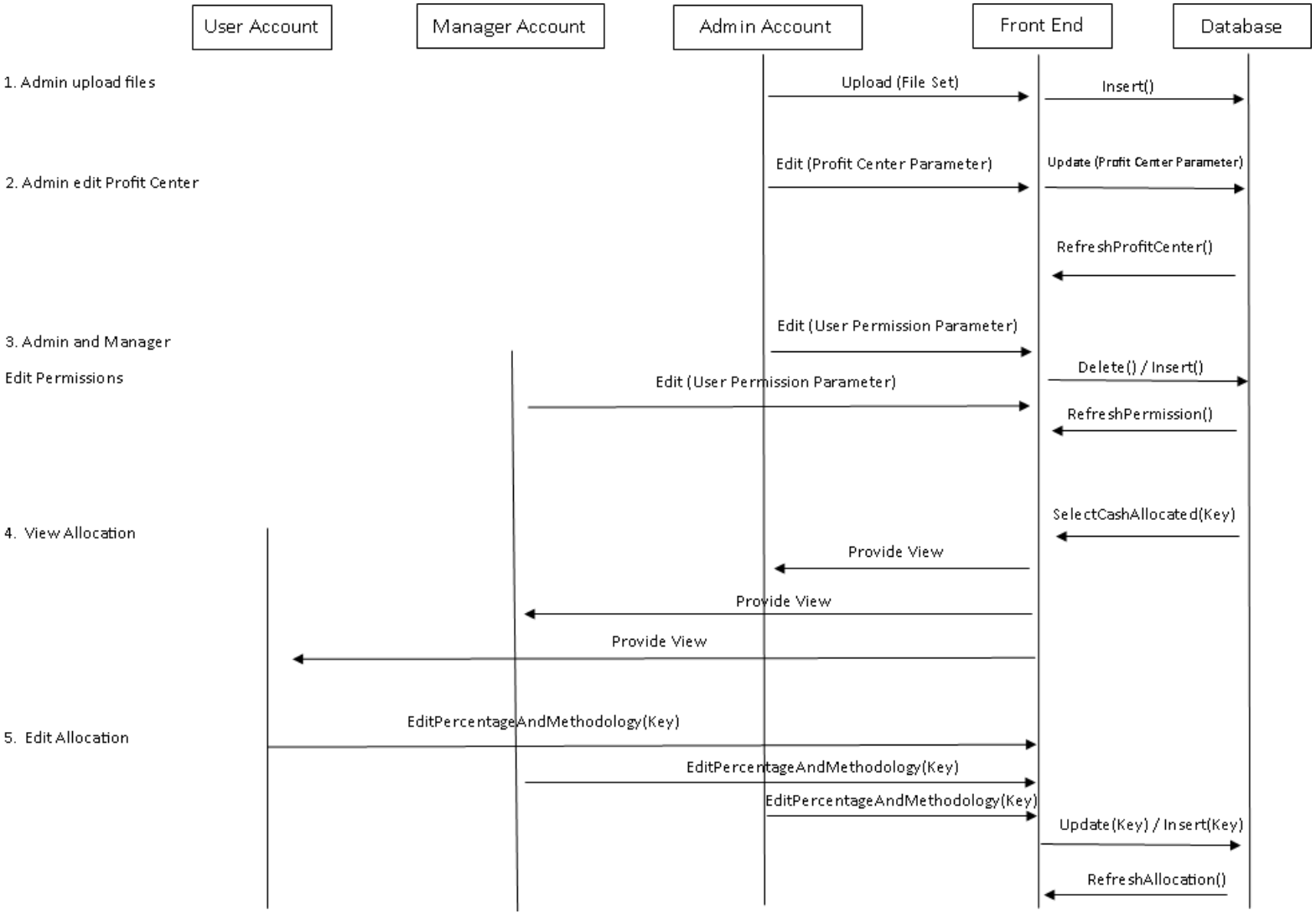
Appendix C: New General Budget Allocation Process



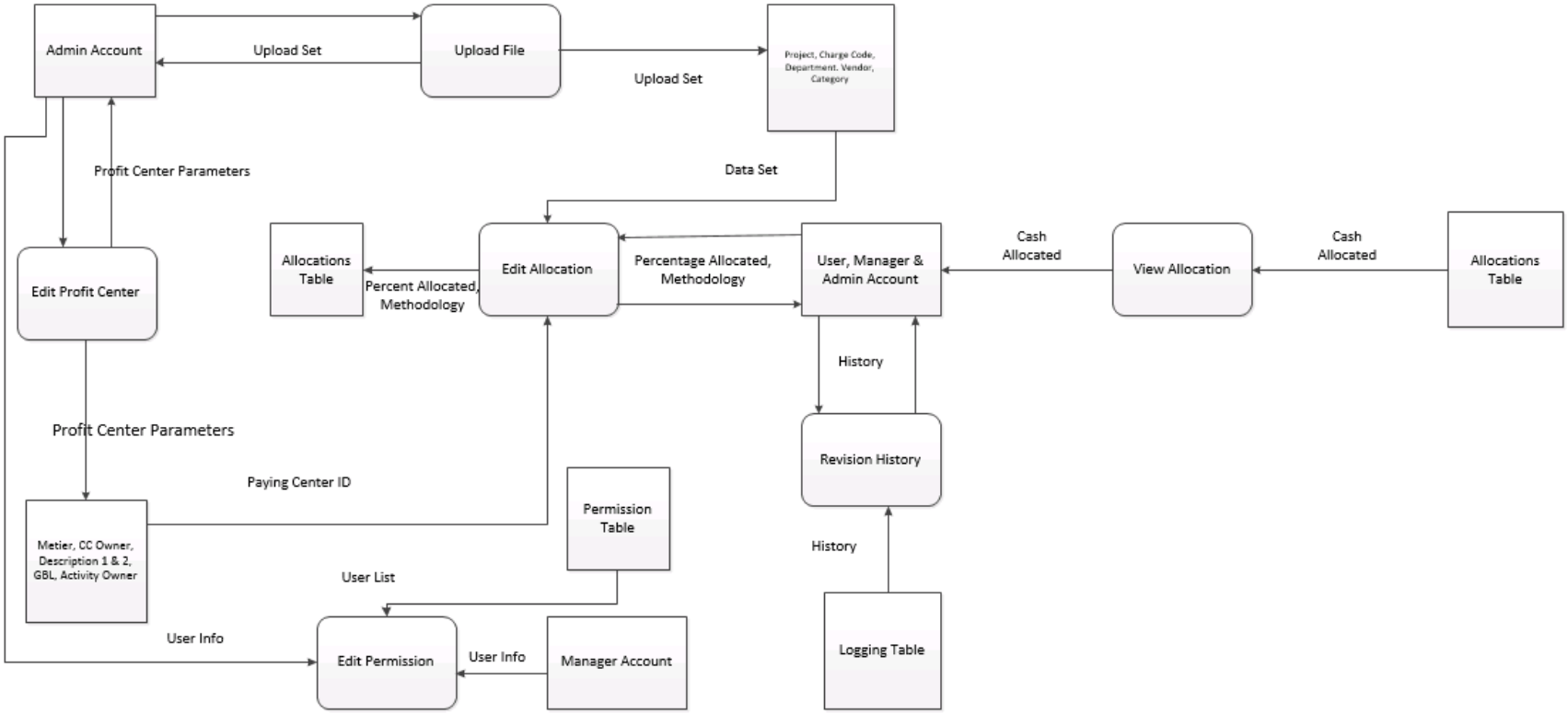

Business Management
Managers
Input allocation keys for discretionar y costs
Re-inputs allocation keys for projects
Receive an alert that allocations do not total 100%
Team Leads
Input allocation keys for projects
Re-inputs allocation keys for projects
Receive an alert that allocations do not total 100%
Application
Front end reads credentials to fetch corresponding data from database through the server
Application verifies allocations total 100%
Server stores all allocation keys into the SQL database
Server reads all the data and creates views of the data and logging table.

Appendix D: Use Case Diagram

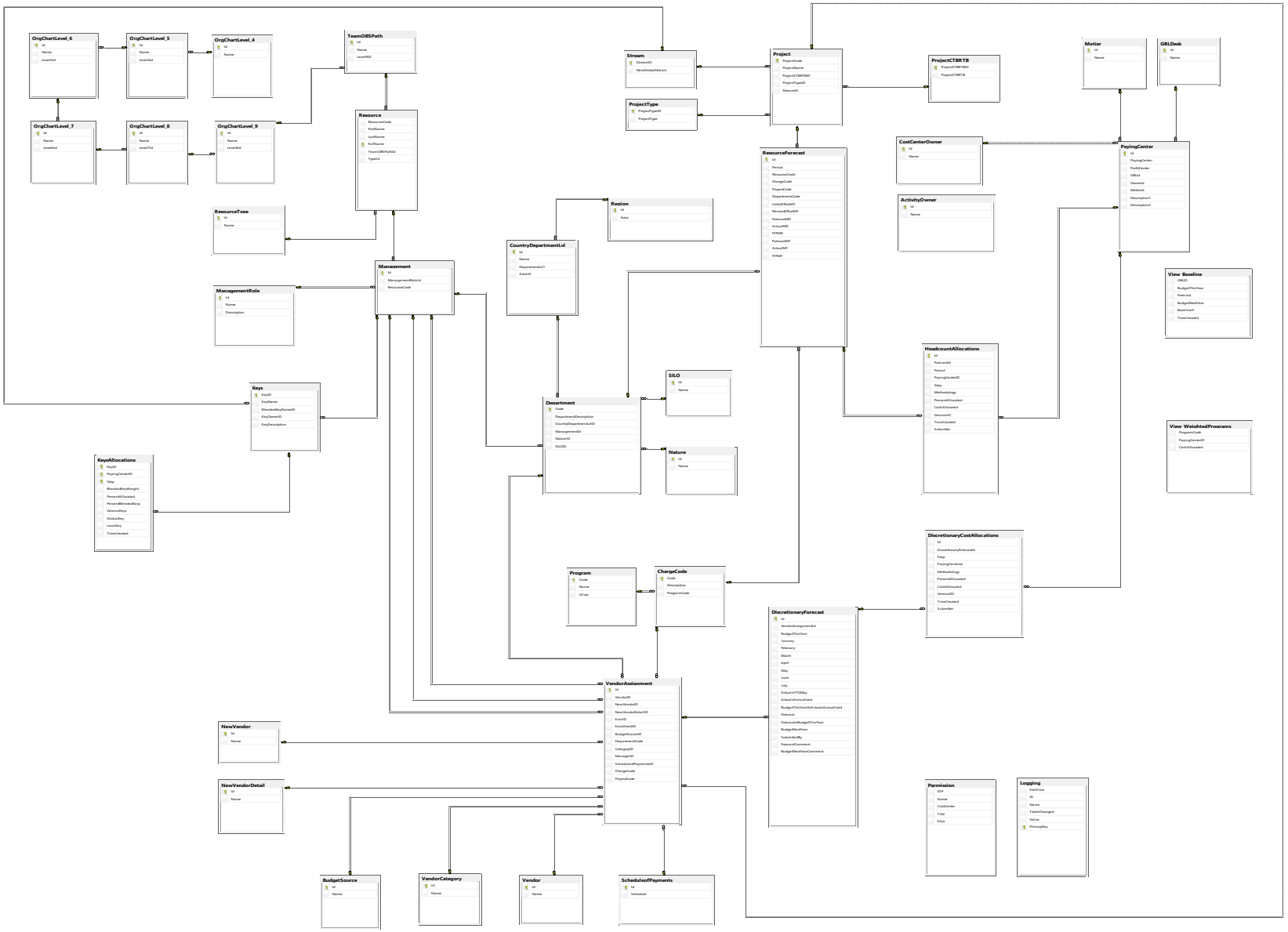
Appendix E: Sequence Diagram



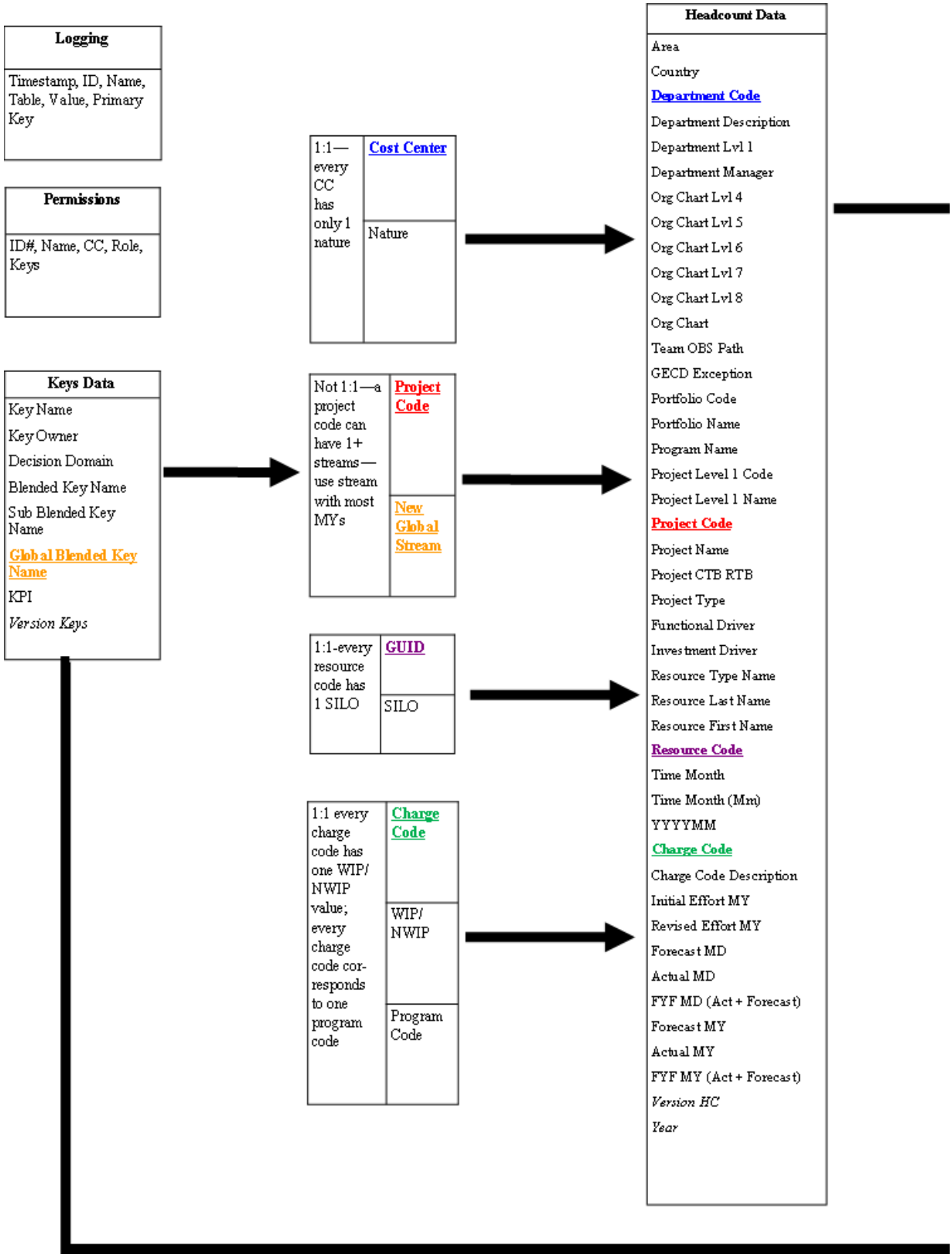
Appendix F: Data Flow Diagram

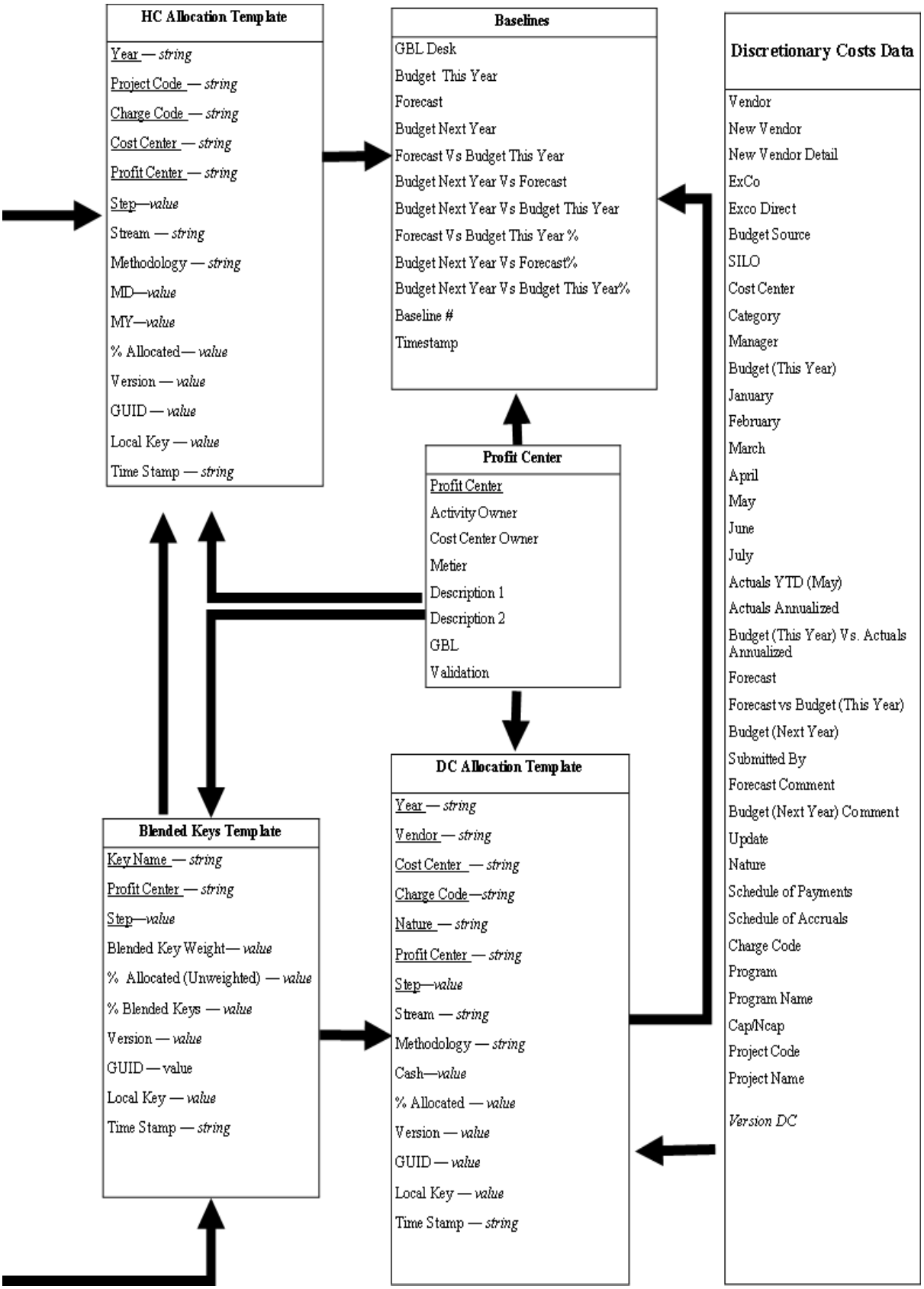


Appendix G: Final Database Design



Appendix H: Un-Normalized Database




HC Allocation Template
Year — string
Project Code — string
Charge Code — string
Cost Center — string
Profit Center — string
Step—value
Stream — string
Methodology — string
MD—value
MY—value
% Allocated— value
Version — value
GUID — value
Local Key — value
Time Stamp — string
Baselines
GBL Desk
Budget This Year
Forecast
Budget Next Year
Forecast Vs Budget This Year
Budget Next Year Vs Forecast
Budget Next Year Vs Budget This Year
Forecast Vs Budget This Year %
Budget Next Year Vs Forecast%
Budget Next Year Vs Budget This Year%
Baseline #
Timestamp
Discretionary Costs Data
Vendor
New Vendor
New Vendor Detail
ExCo
Exco Direct
Budget Source
SILO
Cost Center
Category
Manager
Budget (This Year)
January
February
March
April
May
June
July
Actuals YTD (May)
Actuals Annualized
Budget (This Year) Vs. Actuals Annualized
Forecast
Forecast vs Budget (This Year)
Budget (Next Year)
Submitted By
Forecast Comment
Budget (Next Year) Comment
Update
Nature
Schedule of Payments
Schedule of Accruals
Charge Code
Program
Program Name
Cap/Ncap
Project Code
Project Name
Version DC
Profit Center
Profit Center
Activity Owner
Cost Center Owner
Metier
Description 1
Description 2
GBL
Validation
DC Allocation Template
Year — string
Vendor — string
Cost Center — string
Charge Code—string
Nature — string
Profit Center — string
Step—value
Stream — string
Methodology — string
Cash—value
% Allocated — value
Version — value
GUID — value
Local Key — value
Time Stamp — string
Blended Keys Template
Key Name — string
Profit Center — string
Step—value
Blended Key Weight— value
% Allocated (Unweighted) — value
% Blended Keys — value
Version — value
GUID — value
Local Key — value
Time Stamp — string