Capacity Plan Database Capstone Project

Erin Scheunemann, Brendan Verbrugge, Man Ho Yuen

May 1st, 2023

Introduction

Our team worked with Miltech to create a financial management tool to help cut down on the time it takes their financial team to balance their spending between employees and projects as well as to eliminate the majority of the human error in their current process. Miltech is a company that was formed as a partnership between Montana State University and the Office of the Secretary of Defense and acts as an intermediary between different private companies and the Department of Defense (DoD), the Department of Homeland Security (DHS), and other government agencies in order to provide the government with the latest technology quickly and reliably. They have three different categories of expertise: information research and analysis, design and prototyping, and manufacturing, process improvement and management. These three areas allow them to find and help advance technology to meet the government's needs. How this works is that the government provides funding for different projects and it is Miltech's job to get the project completed in a timely manner while spending out the entirety of the funding.

Managing the spending for different projects is a difficult task. There is a lot of information that Miltech has to keep track of when managing their spending for different projects. The first task that they encounter is planning out the amounts they will pay in benefits and salary total for each pay period up to the last one for each project. These amounts are also able to fluctuate and change based on their actual spending for the current pay period or based on the needs of a project for a certain pay period. An important aspect of project spending is keeping track of salary and benefits. It is important to meet the projected target spending for a certain pay period but it is also equally as important to make sure employees are being paid their full wages for a pay period. Miltech needs to keep track of what employees are working on what projects and what percentage of their time is being spent on their different projects. These

percentages need to add up to 100%. This means that Miltech must balance the amount of time their employees spend on different projects so that they receive their full wages and Miltech reaches the target spending for salary and benefits for their different projects.

Their current solution is slow and cumbersome and leaves them prone to human error. The main reason their process is so slow is that they have to use three disconnected tools: CatBooks, Excel Spreadsheets, and Banner. CatBooks is an accounting software that MSU created and that MSU and MSU related businesses use. CatBooks acts as their checkbook and helps them keep track of their current and future spending for the current fiscal year. They have two main Excel Spreadsheets that they manage; their initial target spreadsheet and their capacity plan spreadsheet. Both of these spreadsheets have to be kept up to date with every pay period. In the case of the Capacity Plan spreadsheet, it has to be remade for every pay period. The initial target spreadsheet is where they keep track of their planned spending for the allotted salary and benefits budget for each project until the end project date. For every project this spreadsheet records its total budget, what they have spent of the total budget so far, and the total amount of the budget that is set aside for salary and benefits. This is further broken down into what they have spent on salary and benefits in total, what they have encumbered in total, and their remaining salary and benefits budget. In the spreadsheet there is also a distinction made for each category between what the numbers actually are in CatBooks and what they are in the plan. Then the spreadsheet records the planned amount that needs to be spent in salary and benefits for each project until the end date for the project. The Capacity Plan spreadsheet is used to determine the distribution of salary and benefits spent for each employee across the different projects so that the target amount is spent for that project and pay period and so that employees are paid their full wages. The spreadsheet records the percentage of each employee's wages spent on each project.

It is important that the percentages for each employee add up to 100% so that the entirety of that employee's wages are accounted for. The salary and benefits payment for each project is calculated from the total wages spent for each employee in that pay period and the total amount spent must be close to the bi-weekly target amount from the initial target spreadsheet. Banner is their bank. They use this to both pay their employees each pay period and to keep track of what they have spent in pay periods past.

The current solution they have been using leaves a lot to be desired in terms of error checking and time saved. Navigating these three tools is a slow process and the financial team must enter data in a time sensitive manner. The capacity plan alone can take upwards of two days to complete and they need those two days because they have to wait until after employees have been paid for the last pay period, which does not happen until midway through the current pay period, and gives them exactly two days to complete the capacity plan for the current pay period. Cutting down on the time it takes for them to update these plans would give them more free time to work on more important matters and give them time to double check their work. Another problem with their current solution is the lack of error checking involved. When the team has to import multiple spreadsheets from catbooks and place the information onto their initial target spreadsheet there is nothing checking that they put the right information in the correct spot. Excel does not check if they are entering information for pay period 22 in the information for pay period 23 table. Along with this they have to rely on excel functions to double check that all their values add up correctly. This opens the door for human error to slip in and mess up their reports. If their data is off or incorrect then that could have disastrous consequences for the company. For instance, if they think that they have planned out the amount they need to spend in their salary and benefits budget for a project by the end date for that project and because of an

error in one of their spreadsheets they do not spend out the full budget then that could result in them not receiving as much money for their next projects. Another consequence could be that they end up incorrectly entering the amount of salary spent from a specific employee and that causes a cascade wherein every employee's pay distributions for that pay period is off. Tiny errors like these can have big consequences for Miltech and their current solution leaves room for these errors to take place.

Our solution was to build a web-app tool that cuts down on the time this process takes along with reducing the amount of human error introduced. A main component of reducing their time will be with importing data from a CSV file and having the web-app check their imputed data for errors or inconsistencies. Originally they had to take the CSV file that they downloaded from CatBooks and copy and paste specific chunks to get the right data into their own excel files. In our web app they will be able to import the CSV files that they get from CatBooks and our web app will parse them and put the data into the database for them. Then they will be able to view and update this data from the different views we will provide them. With their original solution, if they wanted to view the data differently, or to only view specific pieces of the data, they would have had to create their own new excel spreadsheets which takes time away from their already busy days. With our web app they will be able to access a multitude of different views of the data instantly. The different views will also allow them to check their own work more easily. The final thing our web app has to offer over their original solution is more accurate and faster error checking. With their original solution a lot of the error checking was done either by hand or with excel functions. Our web app automatically checks their different imputed data against the data in the database to prevent inconsistencies in the data. Our overall objective is to

ensure that our solution is able to help them reduce the amount of time that the process takes and how much error is introduced into the process.

Background:

The database our team has created for Miltech is tailor-made to fit their very specific needs. As such, there isn't any research available to the public that could better inform or provide more background to the reader as to the nature of this project. That being said, there are a few terms/tools used in our solution that are worth explaining in more detail. The first is the software bundle known as the LAMP stack. As the foundation of our database, it is important to understand exactly what the LAMP stack is. The LAMP stack is a bundle of four open-source software components that combine to provide a means to create interactive web-apps capable of communicating with custom-made databases. Each letter in the LAMP stack stands for one of these four components: Linux, Apache, MySQL and PHP/Python. Linux serves as the operating system that supports the other three components. Apache stores website files and gives LAMP stack web apps a means to communicate with web browsers via HTTP. MySQL is a relational database management system, allowing not only for the creation of an underlying database used by the LAMP stack, but for the manipulation and querying of that database in whichever way the developer of the LAMP stack web app wishes. Finally, the PHP/Python layer of the LAMP stack gives a means to customize the UI of a web app as well as allowing a web app to run dynamic processes.

The terms "pay period" and "pay number" are used many times in this document, often somewhat interchangeably. The difference between them is minor but important. A "pay period" is a two week period in between salary distribution for Miltech employees. The last Friday of a pay period is when all the employees are paid, and new actuals are generated, although the exact numbers for these actuals is not clear until the day before the next pay day for the next pay period. Actuals are the money that Miltech has already spent, and it is very important

information to keep track of as it directly influences the quantity of money remaining in their active grants, and thus is a deciding factor in the employee pay percentages that must be calculated on a bi-weekly basis. The Miltech finance team always has to think two weeks ahead, as they won't receive the actuals for the previous pay period until the current pay period is almost over. One of the reasons this database is so important to the finance team is that it will allow them to take the last pay period's actuals and very quickly see if those actuals conflict with their projections/current planned pay percentages. If there is a conflict, the database will help the team update the pay percentages much faster. By contrast, the term "pay number" refers to a simple numerical identifier for each pay period during the year. The year begins with pay number 1, and ends with pay number 26. The pay numbers give the Miltech finance team a convenient way of keeping track of where they are in the year, and provides an identifier for pay periods so that previous pay periods can easily be referenced.

Work Schedule:

During the first semester of this capstone, our team met consistently two to three times a week (including weekly meetings with Miltech) to produce a formal project proposal that would lay the groundwork for the production of an end product capable of meeting the needs of our client company Miltech. Each member of our team had various responsibilities concerning the production of the proposal. Erin handled most of the UML diagrams, the introduction, proposal, design pattern and design tradeoff components of the proposal. Man Ho worked on some of the UML diagrams and performance requirements as well as the functional and non-functional requirements components of the proposal. Brendan focused on the work schedule, background, UI and expected results components of the proposal. Below is a gantt chart detailing our progress on this portfolio over the course of the semester.

Tasks	Dates of Tasks															
1. Introduction																
2. Proposal Statement																
3. E/R Diagram																
4. UML Diagram																
5. SDLC																
6. Functional/Nonfunctional Requirements																
7. Component Diagram																
8. Design Pattern																
9. Work Schedule																
10. Finishing Touches																
	Wk. 1	Wk. 2	Wk 3.	Wk. 4	Wk. 1	Wk. 2	Wk 3.	Wk. 4	Wk. 1	Wk. 2	Wk 3.	Wk. 4	Wk. 1	Wk. 2	Wk 3.	Wk. 4
	Month 1				Month 2				Month 3				Month 4			

For the first few weeks of the semester, we focused on selecting our Capstone project and learned the very basics of what our project was going to be. From there we started with the introduction and proposal statement. After producing a draft of these, we shifted our attention to the E/R and UML diagrams. Once we had created most of our diagrams, we moved on to planning out our software development life cycle (SDLC) as well as thinking about our functional/nonfunctional requirements. This in turn led to thinking about our components, which led to the creation of our component diagram. From here we were able to identify a design pattern that would work well with our plan, and we updated the relevant UML diagrams to include this pattern. Finally, we spent the last few weeks before the presentation of this portfolio working on our planned work schedule for the second semester, as well as filling in anything else that was missing.

Each member of our team had a different responsibility when it came to writing the code for our project during the second semester. Man Ho focused on development of the frontend, including the web app and the GUI, Brendan focused on development of the backend (the database itself) and Erin assisted with both frontend and backend as needed. Our team met two to three times a week including a weekly meeting with the Miltech team in which we discussed our progress as we moved forward with development. The other meetings were group meetings in which we discussed as a team the progress we were making, potential issues that had arisen during the coding process, solutions to these and any other issues and in general, our current

strategy to ensure development of a satisfactory end-product. We developed the database using a SDLC known as the iterative model. This means that we focused on creating "low functionality" prototypes of various database components that we could present to the Miltech team during our weekly meetings. (Tutorials Point) For example, one prototype might involve an operational "create" function for new projects and employees, but not much else. Another prototype example might be a prototype in which the "import" function is operational but not much else. The purpose of these prototypes was to give the Miltech team something tangible to interact with as we moved forward with development of the project, so that we could get feedback from the prospective users of the database step by step during development without having to walk these prospective users through the code itself in extreme detail. We were able to incorporate this feedback into new, increasingly robust prototypes over the course of development, and eventually were able to hook these prototypes together into our eventual end-product. Each prototype created over the course of this process was considered to be a development "milestone," or a tangible measurement of our progress made towards delivery of a satisfactory end-product. Below is pictured the Gantt Chart we followed during the development process.

Tasks	Dates of Tasks															
1. Database Skeleton																
2. CRUD Functionality																
3. Import Functionality																
4. Export Functionality																
5. Web App Basic Functionality Implementation																
6. Web App Access Stratification																
7. Web App User Confirmation Implementation																
8. Web App Extra Security Implementation																
9. Stretch Goals																
	Wk. 1	Wk. 2	Wk 3.	Wk. 4	Wk. 1	Wk. 2	Wk 3.	Wk. 4	Wk. 1	Wk. 2	Wk 3.	Wk. 4	Wk. 1	Wk. 2	Wk 3.	Wk. 4
	Month 1				Month 2				Month 3				Month 4			

The first two weeks of development were focused on the construction of the database "skeleton," or the basic underlying structure of the database itself. The next two weeks focused on CRUD functionality, or create, read, update and delete functionality. The next two weeks focused on the import functionality - making sure that the database was capable of reading csv

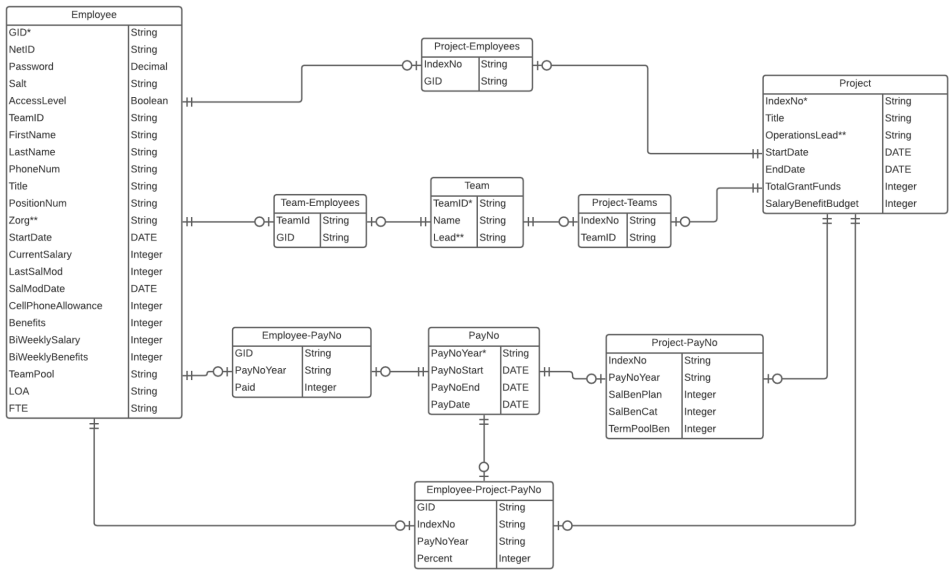
files exported from Catbooks and properly populating the database with relevant information. After import, the next two weeks focused on implementing export functionality. After this, focus shifted to the web app. Two weeks were devoted to the development of basic functionality of the web app - making sure that all the buttons and displays were working as intended. After this, a week was devoted to implementing user access stratification, or separating what features of the web app users could access based on their security clearance. A week was then spent on implementing user confirmation - whenever a user makes an input that will cause a change in the database, an extra confirmation step was implemented to help minimize human error. Next, a week was spent on implementing extra security features to ensure the PII contained in the database remained secure. The last two weeks before presentation and submission of this project were spent on refining all features of the database and web app, as well as working on stretch goals such as entry logging.

Proposal statement:

Our team has created a LAMP stack based web-application to better solve Milech's current financial management problem. For our LAMP stack we choose to specifically be working with Python and using the Django web-programming framework. Our team decided to work with Django because it follows the MVT, or Model-View-Template, structure. (DjangoProject) This was beneficial to our team because the backbone of our solution is the web-app's Database. This database contains all the financial information needed to create the reports displayed on the web-app. The MVT structure of Django has allowed us to create a web-app that will let us do CRUD operations on the database easily and efficiently. Our web-app allows them to create multiple different views, or reports, of the data from the database. These views may be new, such as displaying employee information, or be analogous to their current

reports, like the capacity plan. On these different pages the user is able to update the information on the page by modifying the different cells of the report and then committing the changes. The user is also able to export these reports to CSV files for archival and comparison purposes. A big part of our web-app that has sped up their current process is the ability for employees to upload CSV files to automatically update the database. Previously, employees would have had to download the reports from CatBooks and manually copy and paste the information from the CSV into specific places in either the initial target plan or capacity plan spreadsheet. But with the web-app they simply have to upload the report and the web-app will parse the file and update the information in the database accordingly which then allows them to view the information in different ways through the front-end. Another way our web-app helps them to better solve their problem is helping them with verification of information. In their original solution they have to rely on excel functions and visual checking to make sure that their information adds up to the correct amount. Our web-app does the verification for them and lets them know when values are not adding up correctly or if they try to enter incompatible data into the system.

E/R Diagram:



This is our E/R diagram which is a graphical representation of our database. Our database is the backbone of our web-app. There are three tables that directly correspond to Miltech's current solution: Employee-Project-PayNo, Project-PayNo, and Employee-PayNo. Employee-Project-PayNo contains the information that corresponds to the capacity plan so it holds the percentage of an employee's wages that were spent on a project for a specific pay period. Project-PayNo corresponds to the initial capacity plan and contains what was planned to be spent on a project for a certain pay period and what was actually spent for that pay period. Employee-PayNo corresponds to what an employee's salary is for a specific pay period. There are four tables corresponding to different elements of Miltech's financial system: Employee, Project, Team, and PayNo. These tables all contain the information pertaining to their subjects. The last three tables, Project-Employees, Project-Teams, and Team-Employees are there to act as connectors between the different tables. Another thing to note is that we are storing our financial information using the Integer type. Thus us because we are choosing to store the information in terms of US cents and not USD. We choose to store the information this way so that we could maintain as much precision as possible which we would not have gotten if we stored the data in USD using the Decimal or Floating Point type.

Functional and non functional requirements

For this software, there are six main functional requirements. Update database with user change is the first main functional requirement. Only users with elevated privileges can change the data in the software and all the changes should be read and updated to the database. This should make sure the database is updated with user input. If not, the web-app may need to make up new rows to store that new data. The second functional requirement is appending data/rows to the data tables in the database. In the data table, there is an initial number of rows from when the

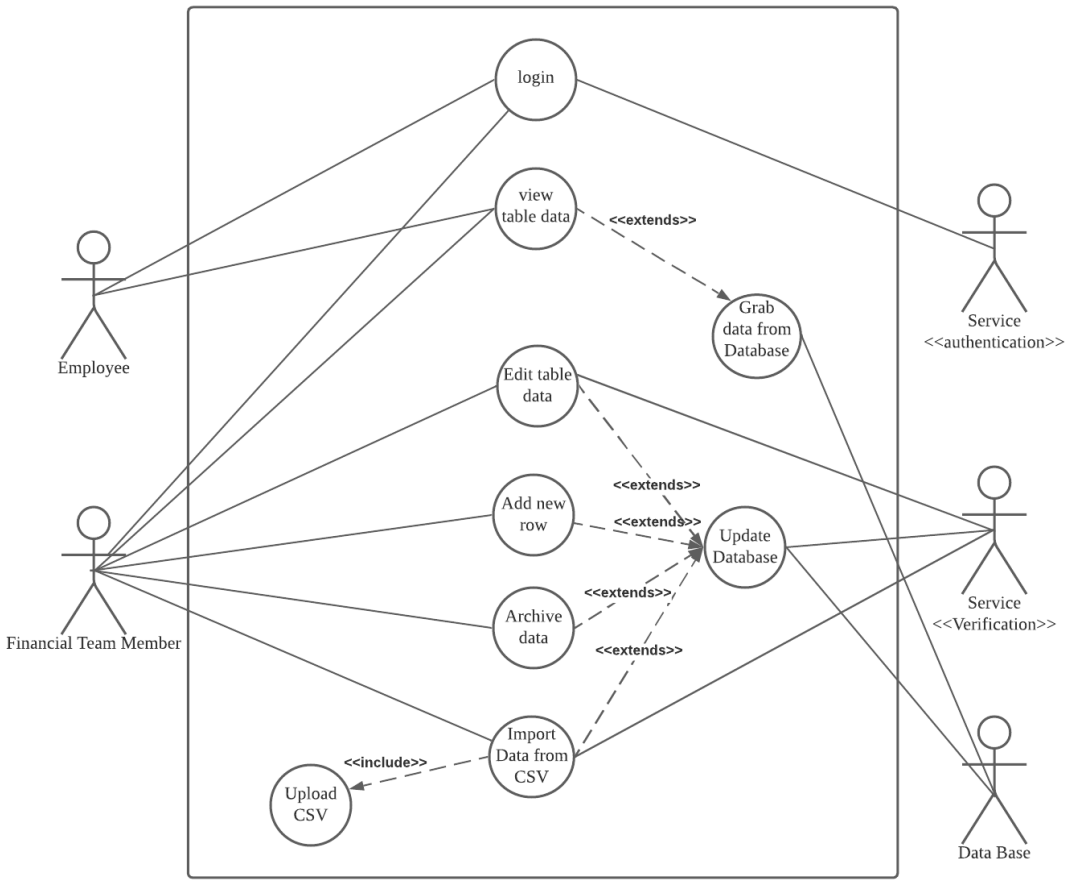
data table was created. This function allows users to add up more data/rows to data tables, so it gives users some flexibility to add new data to the database. This function helps not limit users in data input and the ways to express data tables. A third functional requirement is the ability for users to archive data in the database, we have to make sure the data have archived into the database when users are making changes or inputting data. Otherwise all users input will not store or archive in the database. Confirming every change is one of the important functions. Because when a user makes a change in software, we have to make sure the user knows they are making changes to the data. If a user doesn't know that they have elected to change the data, it could cause a serious problem. Most data is connected to each other, one small change could make other data change also. And confirming change could prevent users accidentally changing data, that could help users know they have changed the data. The last two functional requirements are importing CSV files. We are looking for a function that could allow users to upload a CSV file and create/update the database by the CSV file. This function allows users to not need to manually input data, users can import CSV files to automatic input data.

Non functional requirements

For users they have different user levels to managers and employees. Managers can input data, import data and change data, but regular employees only can view data on the software. In the login system, we require user login with their ID. Because the database stores what level of access a user has, it makes sure that that employee only has access to the features they have permission to use. It makes sure regular employees can’t input data or change data, so regular employees can not input or change data accidentally. The user's ID is an important place for system safety. Because all the payment for contract and employee payment is stored in this software, we have to prevent regular employees from changing the payment. Also for the

software we have to make sure we have enough memory to store all the data. Because Miltech will store all the data including employee information and payment etc, there will be a lot of data that needs to be stored.

Use Case Diagram:

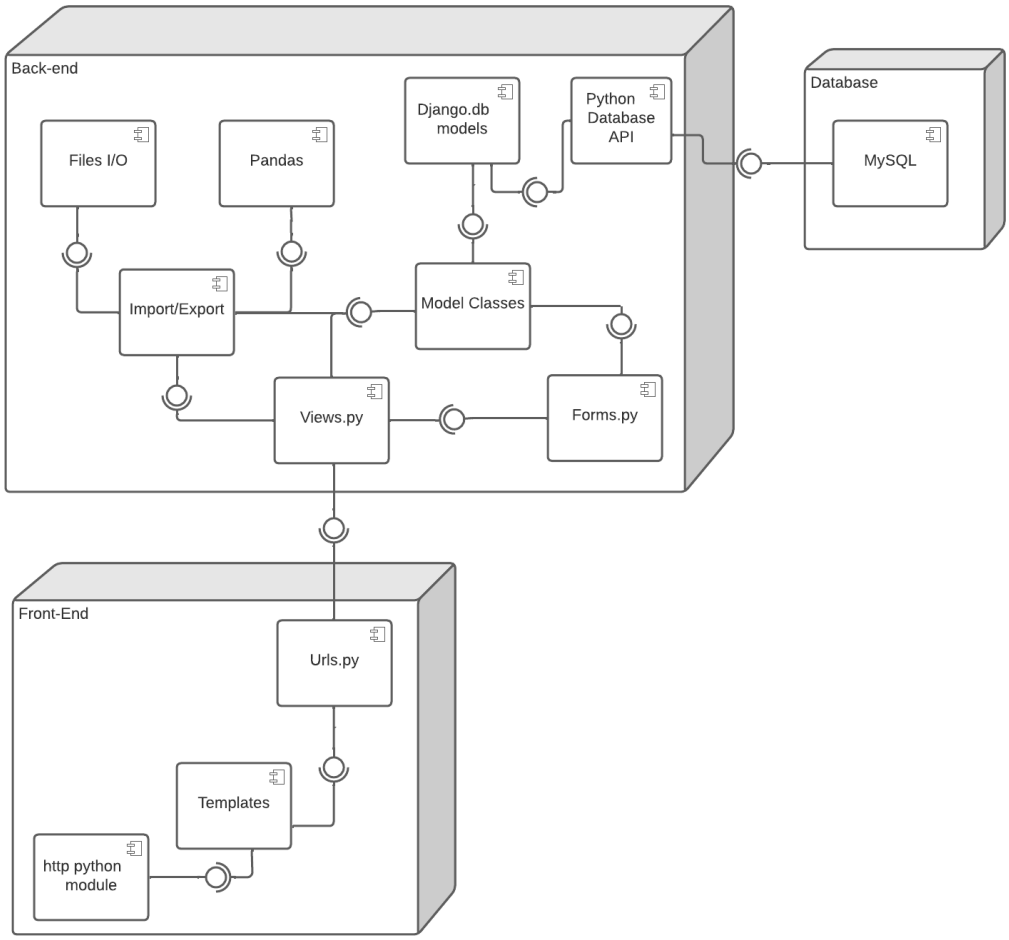


Performance requirements

For the performance requirements, the data workload should not be too long. Because in the software users always need to read or input data, users should not have to wait for a long time to get the data. We have to make sure the user will not wait more than ten seconds to get the data they need. Otherwise, every time users have to wait when the system reads data or when users input data. It could happen multiple times in a minute, so it is important to make sure data

workload can’t be too long. Another performance requirement is that data should be easy to read. For example, when users take the data form the database, the way that system shows the data should be easy to read. It could increase the speed of users finding the data that they want. Miltech has a great amount of data on the data table. So we are going to use the list to show the data and try to not show too much data in the same page. Make sure to only show the data simple, not too complex. That could help the user easily read and easily find out the data he wants. And keeping data simple could help programmers to easily find out the problem when the system goes wrong.

Software Interfaces:



Component Diagram

There are three major components that have to communicate with each other: the database, the back-end, and the front-end. These three major components are made up of smaller components that communicate with each other in order to connect the major three together. The database component is composed of just the MySQL component. MySQL is a part of our LAMP stack and is where we plan to host our database. The back-end is composed of a few premade python interfaces combined with our created classes. This is the portion of the web-app where all our logic will be stored and is the component that connects the front-end to the database. This allows the user to read from the database or make modifications to the database via the front-end. The front-end contains the components that allow the user to interact with our web-app. Namely the HTML templates, Urls.py, and the HTTP python module that allows us to make use of HTTP protocols.

The back-end of our web application is where our logic is stored. It contains the classes we made and their required python libraries. Import, as discussed previously, allows the user to update to or read from the database via the use of CSV files. In order to write or read files python requires the use of the files I/O module. These classes also require Pandas which is a python package that makes data manipulation and analysis from data frames easier. The reason that we are using Pandas here is because it has excellent formatting capabilities for both reading from and writing to excel or CSV files. This will help the web-app to decipher the files that the user gives us. The model classes component represents all of our model classes for the database; each corresponding to one table in the database. Each class will implement the Django Models interface. This interface provides us the skeleton of our model classes for the CRUD operations that will be needed. Views.py and Forms.py work together to create the pages on the front-end. Forms in the forms.py file are predefined by us to display forms on the front end that allow us to

modify model objects in the back-end. They are displayed by the view functions in the views.py file, which render HTML templates to create a dynamic and unique web-page for the user.

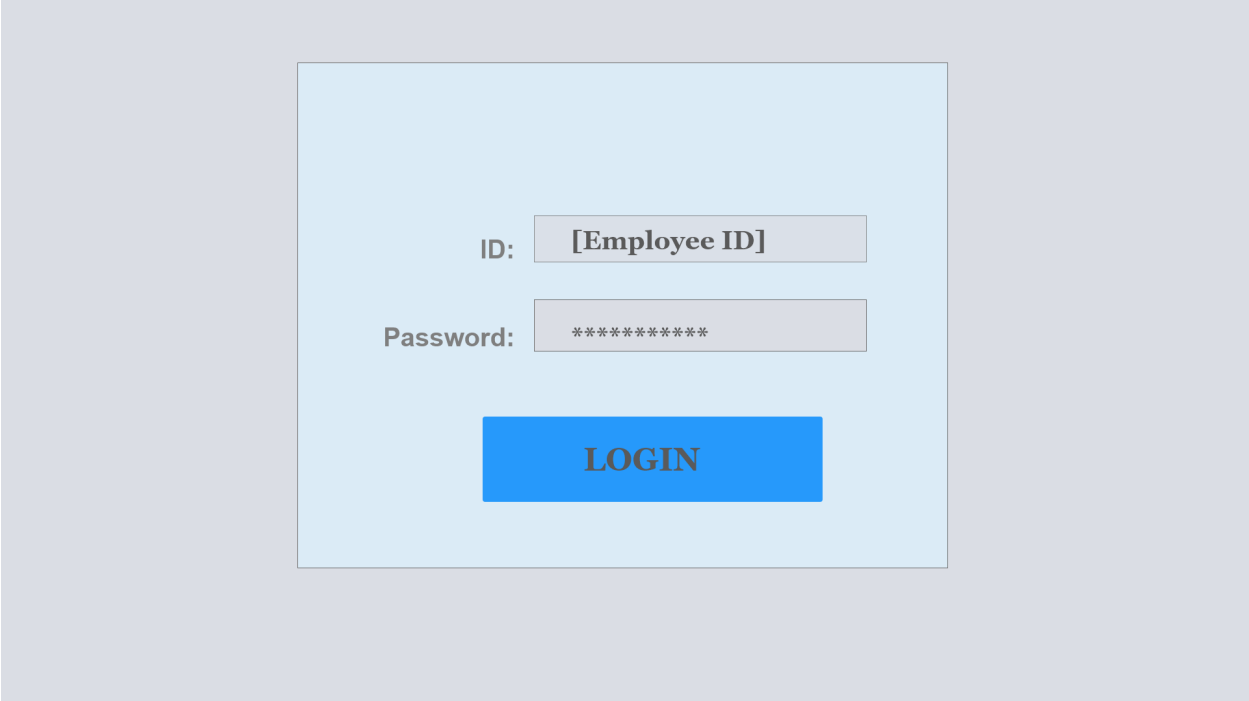
Software Interface Requirements:

These interfaces have to interact in certain ways for our web-app to be functional. The model classes component shall write, read, update and delete from the database stored in MySQL through the use of the Python DataBase API and the Models interface provided by Django. The Import class shall read from files, in our case mainly CSV files, using the Files I/O module provided by Python. The import class shall read from incoming files using the Pandas Library provided by Python. The HTML templates shall send HTTP requests and posts using the HTTP Module provided by Python. All of these ways of communication via the different components must be working in order for our web-app to be functional.

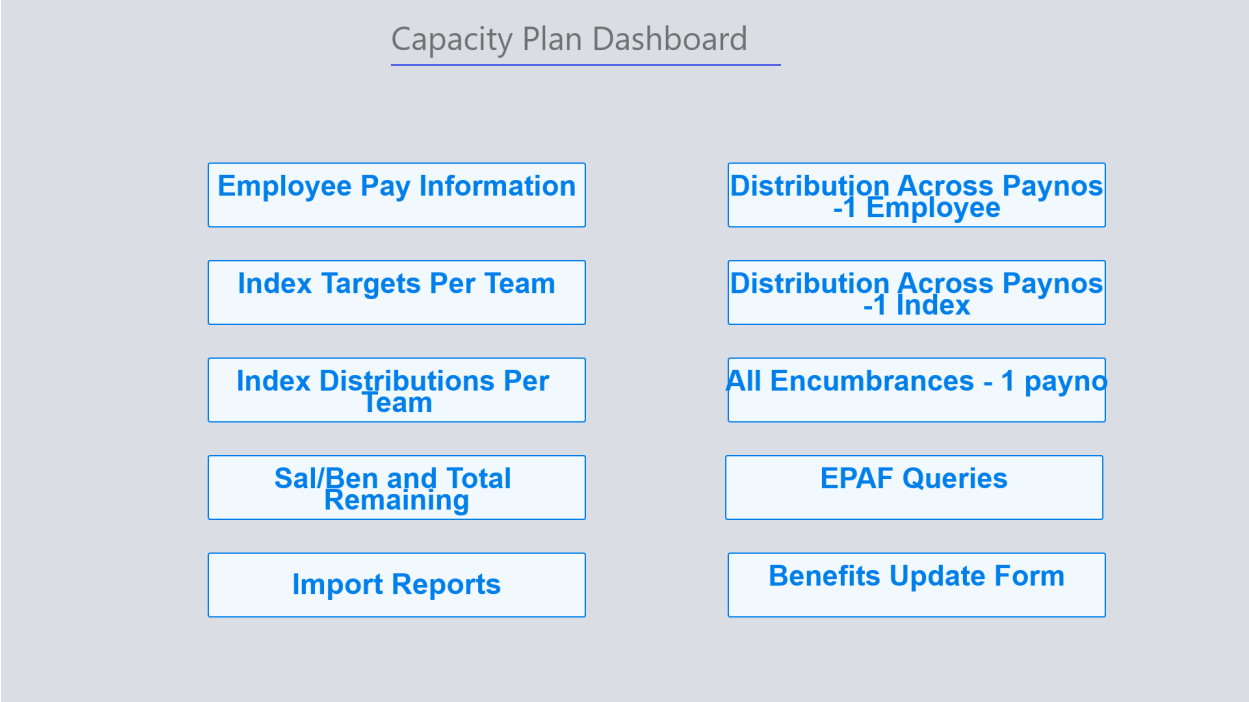
User Interface:

Our user interface was designed with the help of Miltech. The UI prioritizes functionality over appearance - it is designed to be intuitive to our prospective user base without being visually cluttered. The objective is that anyone who understands Miltech's financial system would be able to make use of this web app with relatively minimal training. Below are numerous images of our UI followed by brief descriptions of the features on each screen, as well as an explanation of how UI features differ depending on user access level.

Login and Dashboard:



This first screen displays what a user would first see when accessing our web app. It is a simple login screen that will use the Miltech employee's netID and netID password as the login parameters. When the user logs in, the database does an automatic background check to determine the access level of the user.

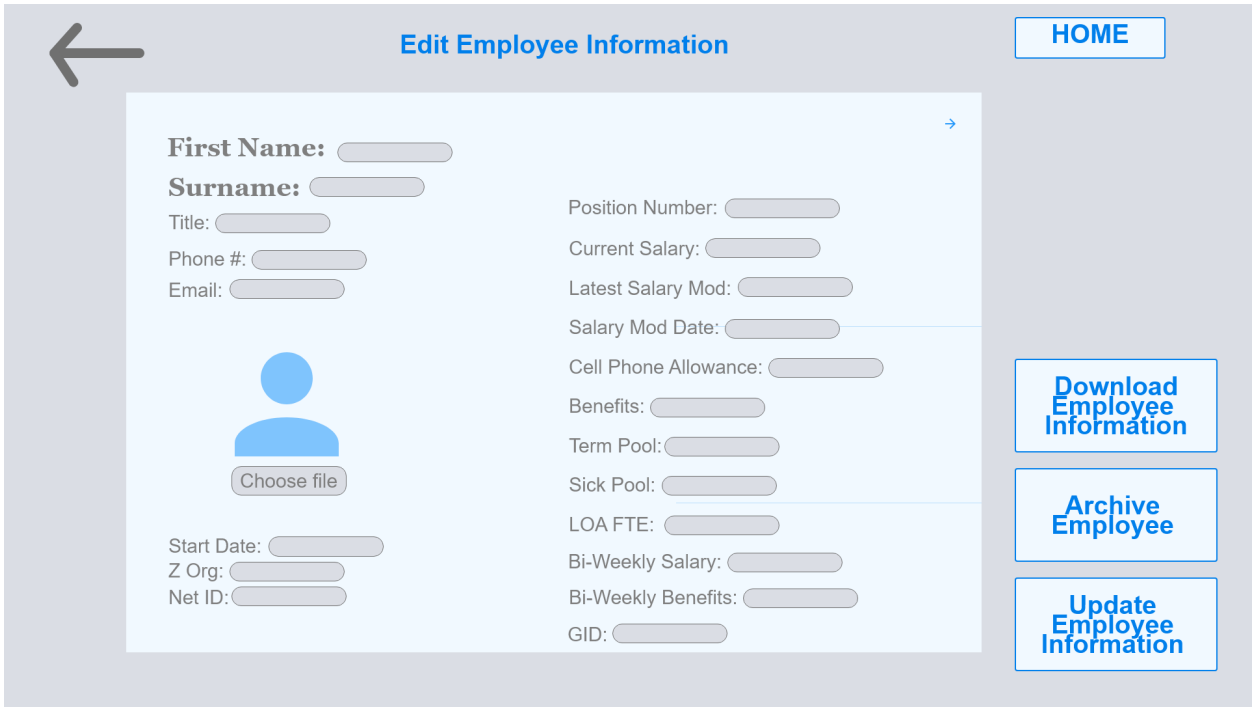


This dashboard acts as a main menu, or "central hub" for our web app from which all other features of our web app can be accessed. Here the user can access whichever feature of the web app they desire to utilize.

Employee Pay Information:

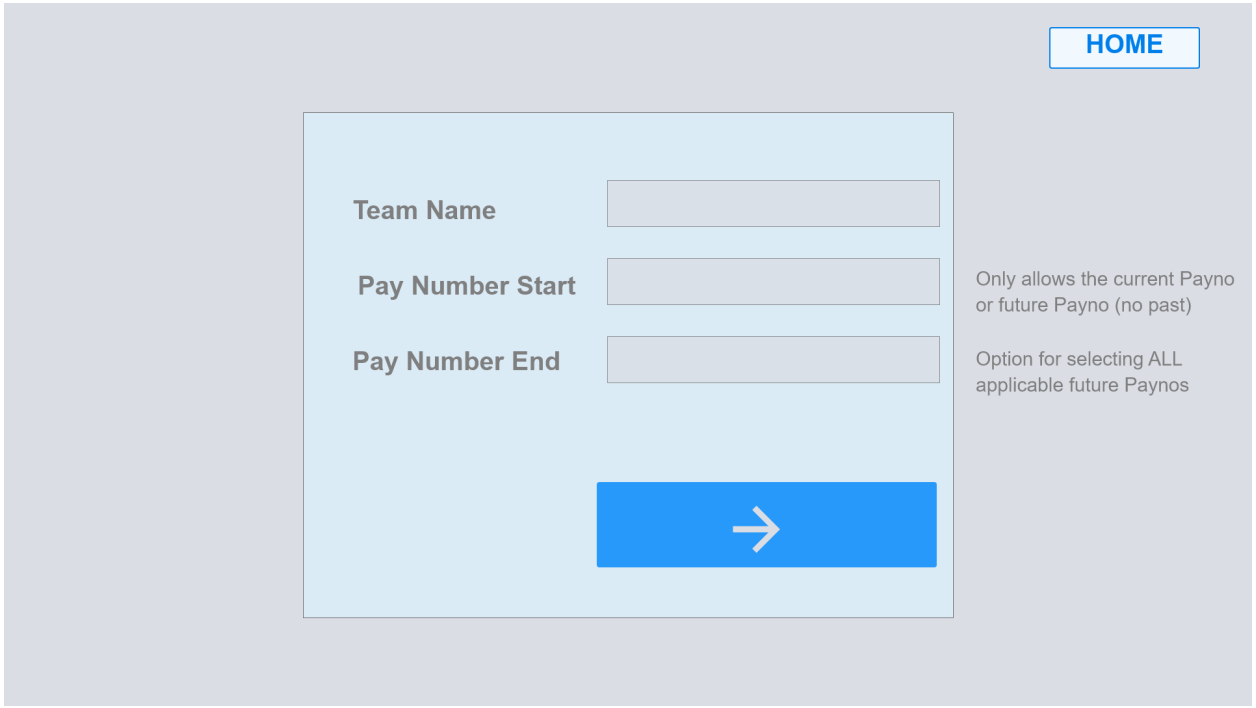
This screen allows a user to view all the relevant information for a certain employee in the database. It is important to note that this feature will change depending on the user's access level.

Low level employees can only see information for themselves, Operations Managers can see information for their team members and Finance Admins can see information for everyone.

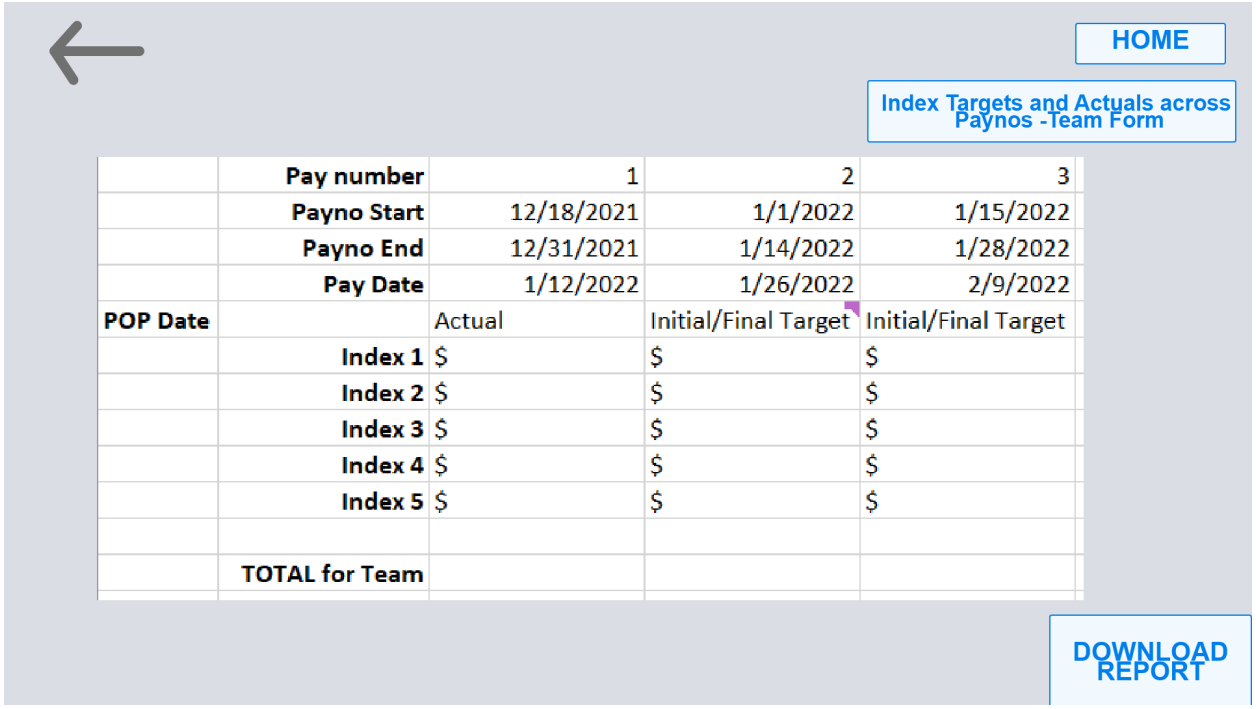


This screen allows a user to update information for an employee. Only Finance Admins are able to edit monetary values like current salary and latest salary mod. Any user should be able to update their contact info (phone # and email) and profile picture if desired.

Index Targets Per Team:



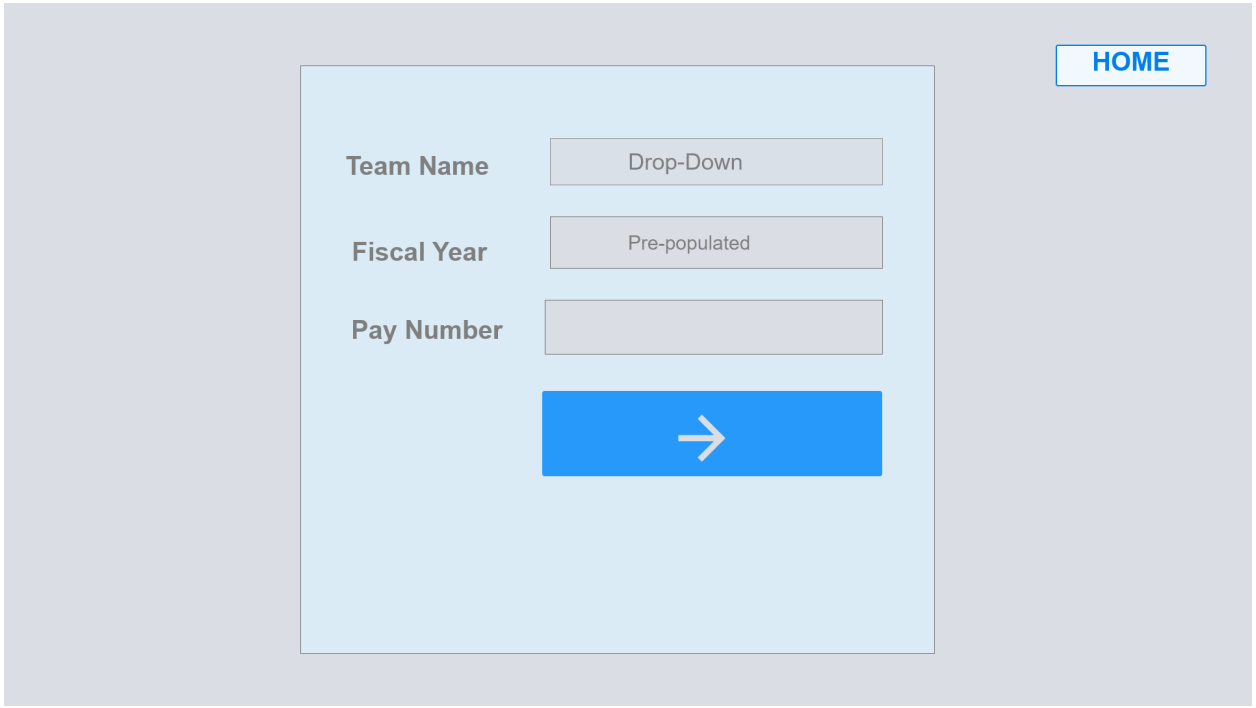
This screen allows a user to create a visualization of actuals and targets for the various indexes managed by a certain team, for a certain range of pay numbers, given the team name, starting pay number and ending pay number.



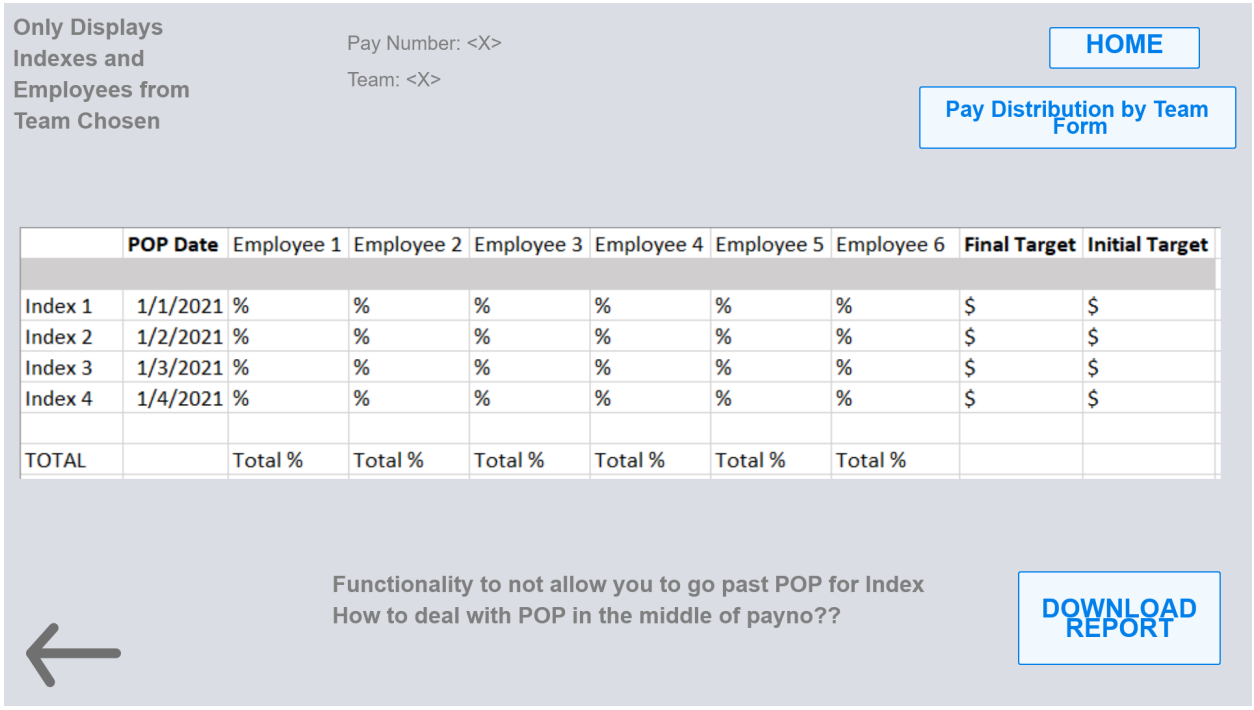
	Pay number	1	2	3
	Payno Start	12/18/2021	1/1/2022	1/15/2022
	Payno End	12/31/2021	1/14/2022	1/28/2022
	Pay Date	1/12/2022	1/26/2022	2/9/2022
POP Date		Actual	Initial/Final Target	Initial/Final Target
	Index 1	$	$	$
	Index 2	$	$	$
	Index 3	$	$	$
	Index 4	$	$	$
	Index 5	$	$	$
	TOTAL for Team			

This is an example of a visualization created by the above screen. A low level access user would only be able to look at this table, but an Operations Manager or Finance Admin would be able to make edits to the targets if they desired to do so.

Index Distributions Per Team:



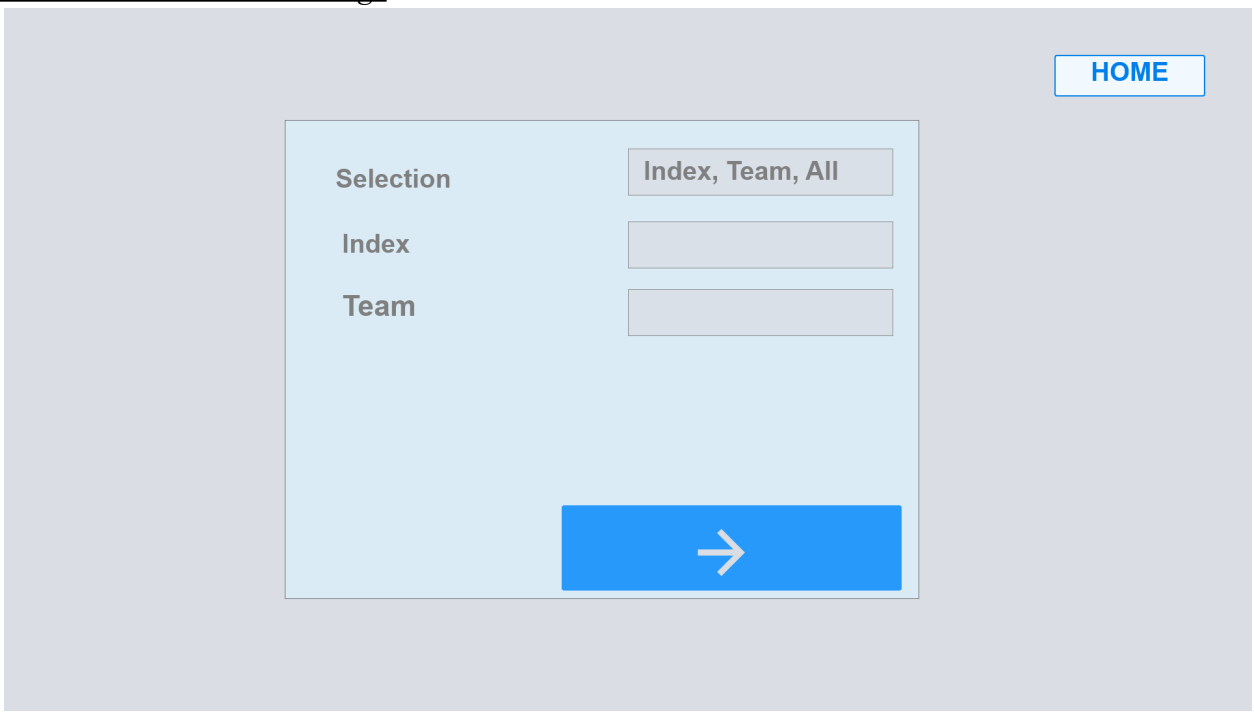
This screen allows the user to produce a visualization of employee pay percentages for a certain pay number from a certain team.



	POP Date	Employee 1	Employee 2	Employee 3	Employee 4	Employee 5	Employee 6	Final Target	Initial Target
Index 1	1/1/2021	%	%	%	%	%	%	$	$
Index 2	1/2/2021	%	%	%	%	%	%	$	$
Index 3	1/3/2021	%	%	%	%	%	%	$	$
Index 4	1/4/2021	%	%	%	%	%	%	$	$
TOTAL		Total %	Total %	Total %	Total %	Total %	Total %		

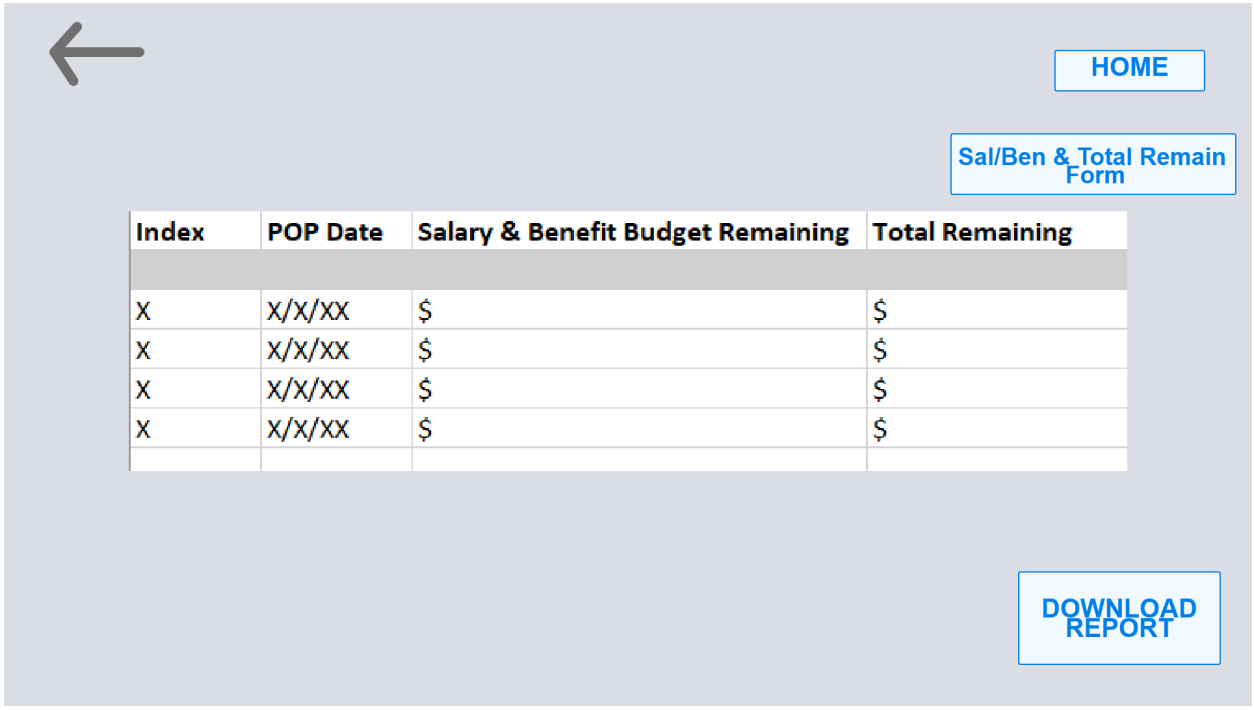
This is an example of a visualization created by the above screen. A low level access user would only be able to look at this table, but an Operations Manager or Finance Admin would be able to make edits to the targets if they desired to do so.

Sal/Ben and Total Remaining:



This screen allows the user to produce a visualization of the remaining salary and benefits budget for the indexes from a certain team or from just one index of choice. Alternatively, the user can

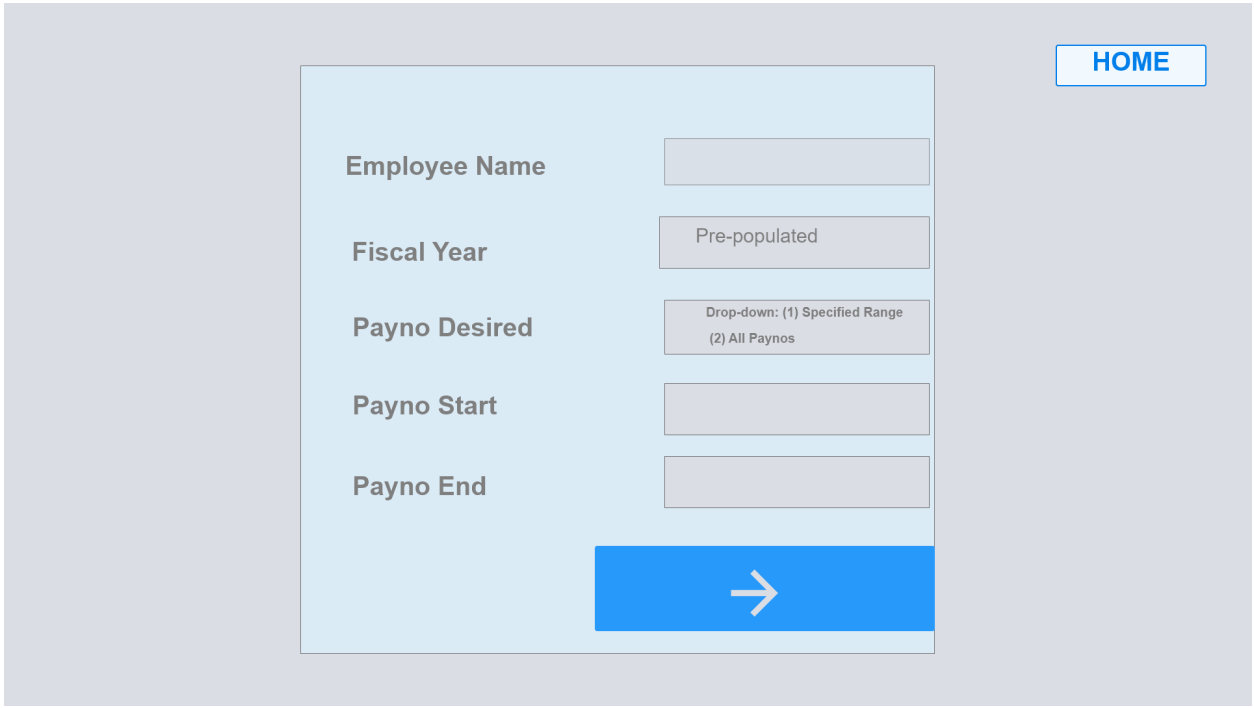
choose to create a visualization of the remaining salary and benefits budget for all indices across all teams.



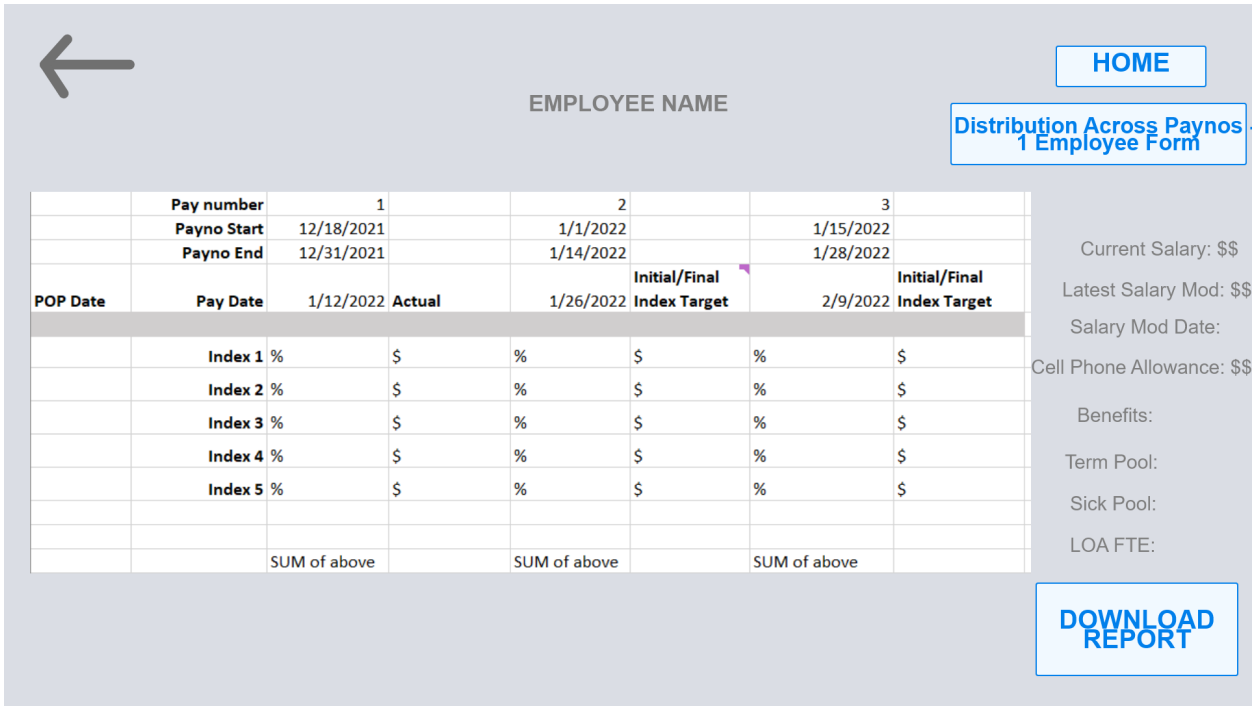
Index	POP Date	Salary & Benefit Budget Remaining	Total Remaining
X	X/X/XX	$	$
X	X/X/XX	$	$
X	X/X/XX	$	$
X	X/X/XX	$	$

This is an example of a visualization created by the above screen. This table is static and doesn't get updated manually by the user.

Distribution Across Paynos -1 Employee:

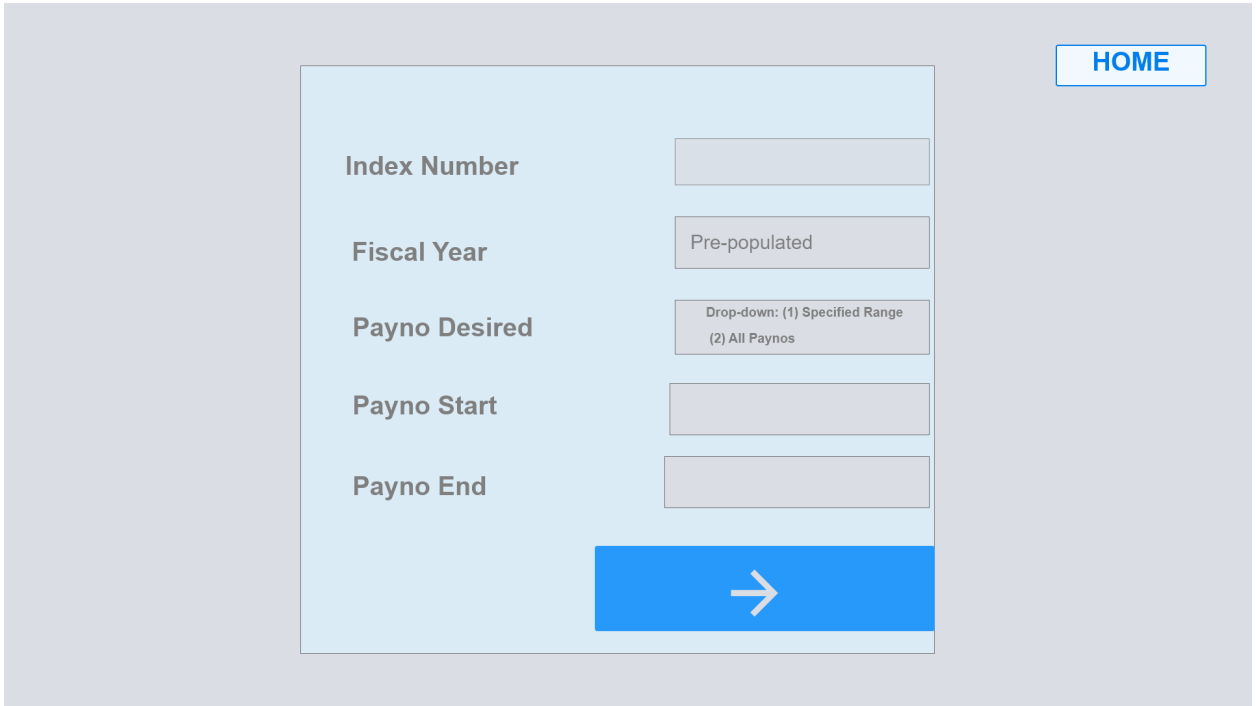


This screen allows the user to create a visualization of employee pay percentages for a single employee for a given range of pay numbers.

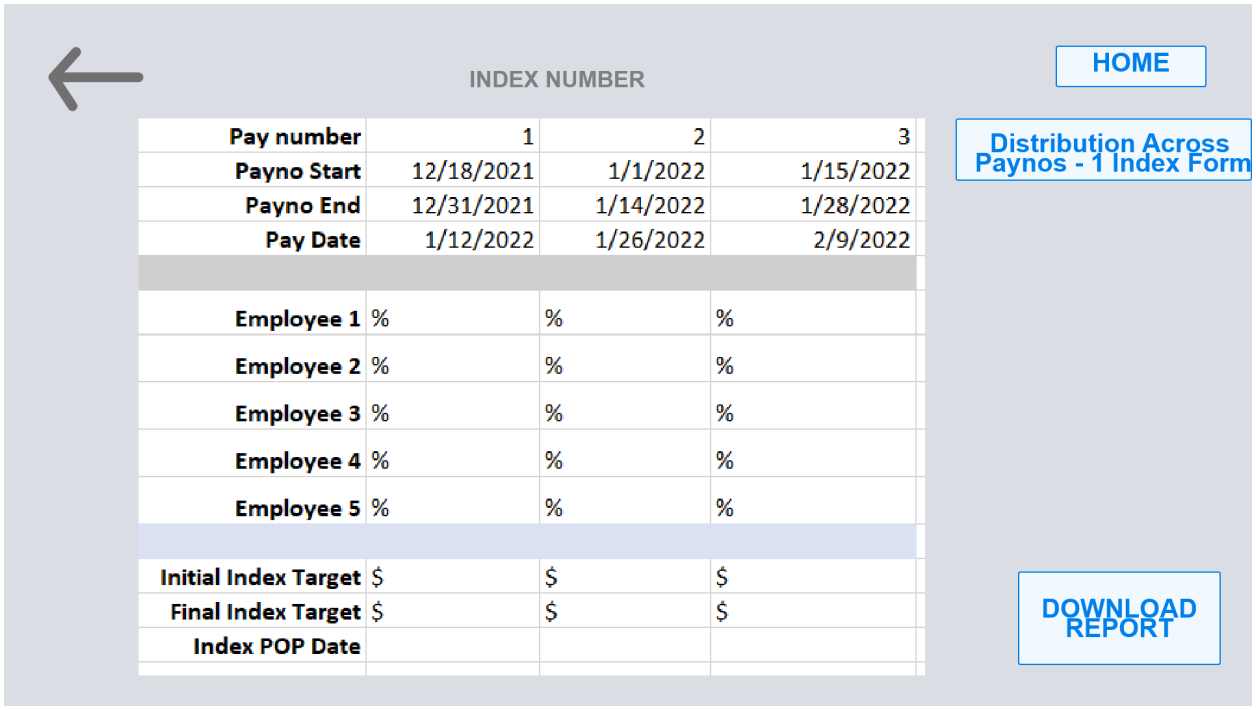


This is an example of a visualization created by the above screen. A low level access user would only be able to look at this table, but an Operations Manager or Finance Admin would be able to make edits to the pay percentages if they desired to do so.

Distribution Across Paynos -1 Index:



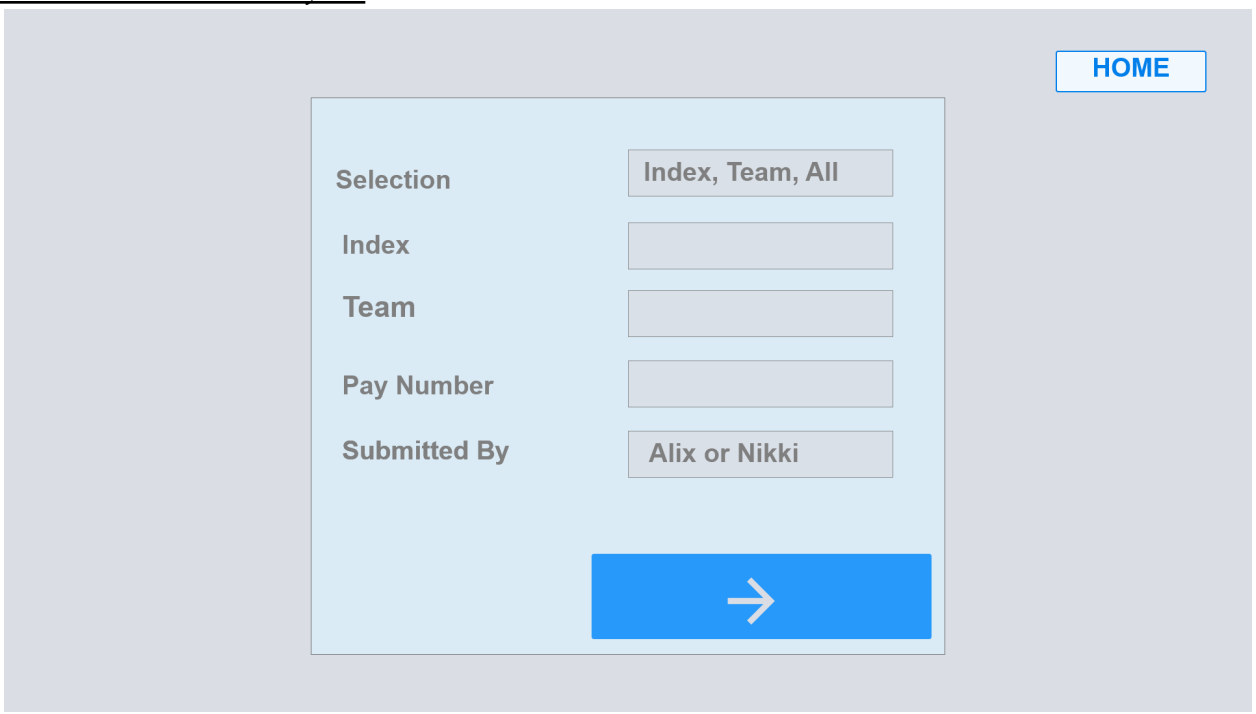
This screen allows the user to create a visualization of employee pay percentages for a single index for a given range of pay numbers.



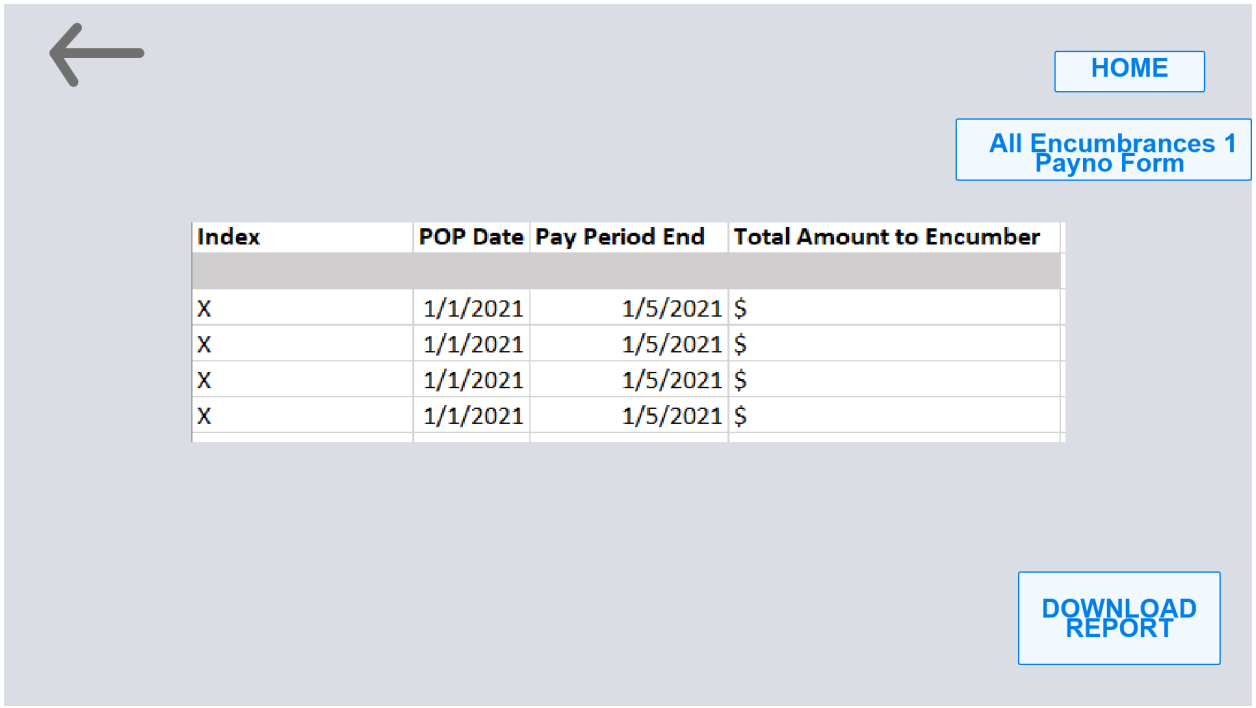
Pay number	1	2	3
Payno Start	12/18/2021	1/1/2022	1/15/2022
Payno End	12/31/2021	1/14/2022	1/28/2022
Pay Date	1/12/2022	1/26/2022	2/9/2022
Employee 1	%	%	%
Employee 2	%	%	%
Employee 3	%	%	%
Employee 4	%	%	%
Employee 5	%	%	%
Initial Index Target	$	$	$
Final Index Target	$	$	$
Index POP Date			

This is an example of a visualization created by the above screen. A low level access user would only be able to look at this table, but an Operations Manager or Finance Admin would be able to make edits to the pay percentages if they desired to do so.

All encumbrances -1 Payno:

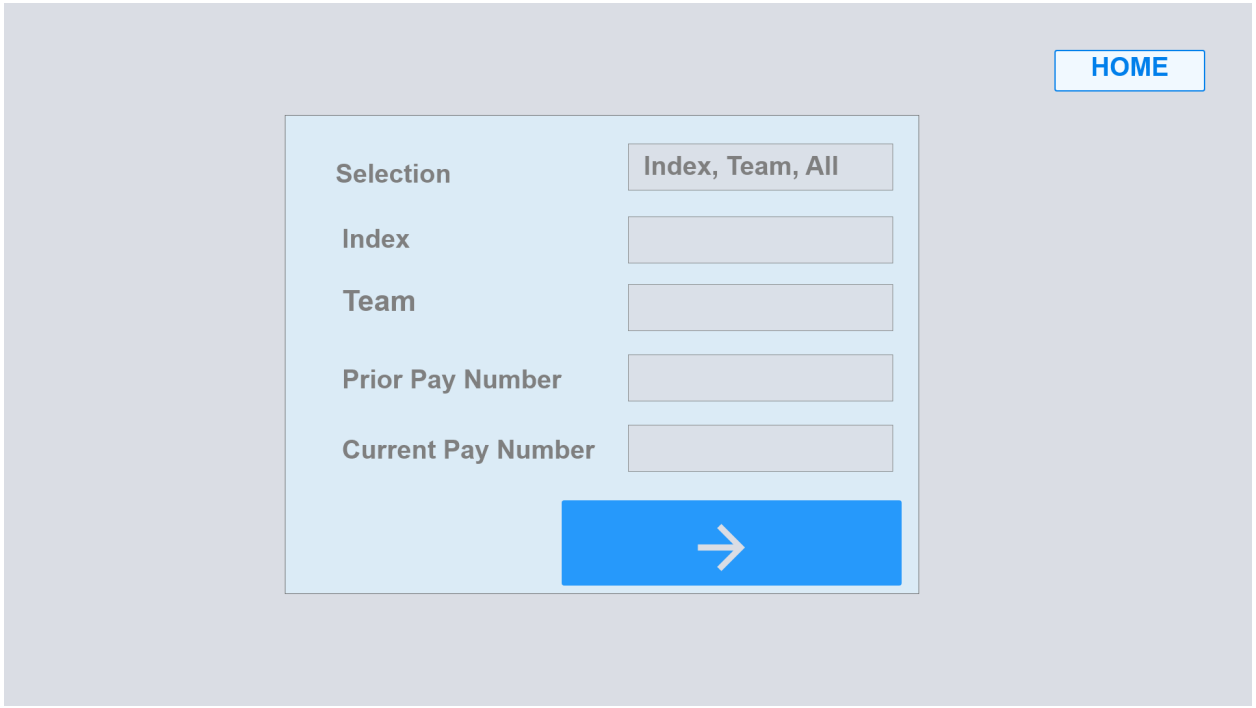


This screen allows the user to create a visualization of encumbrances for the indexes from a certain team or from just one index of choice. Alternatively, the user can choose to create a visualization of encumbrances for all indices across all teams.



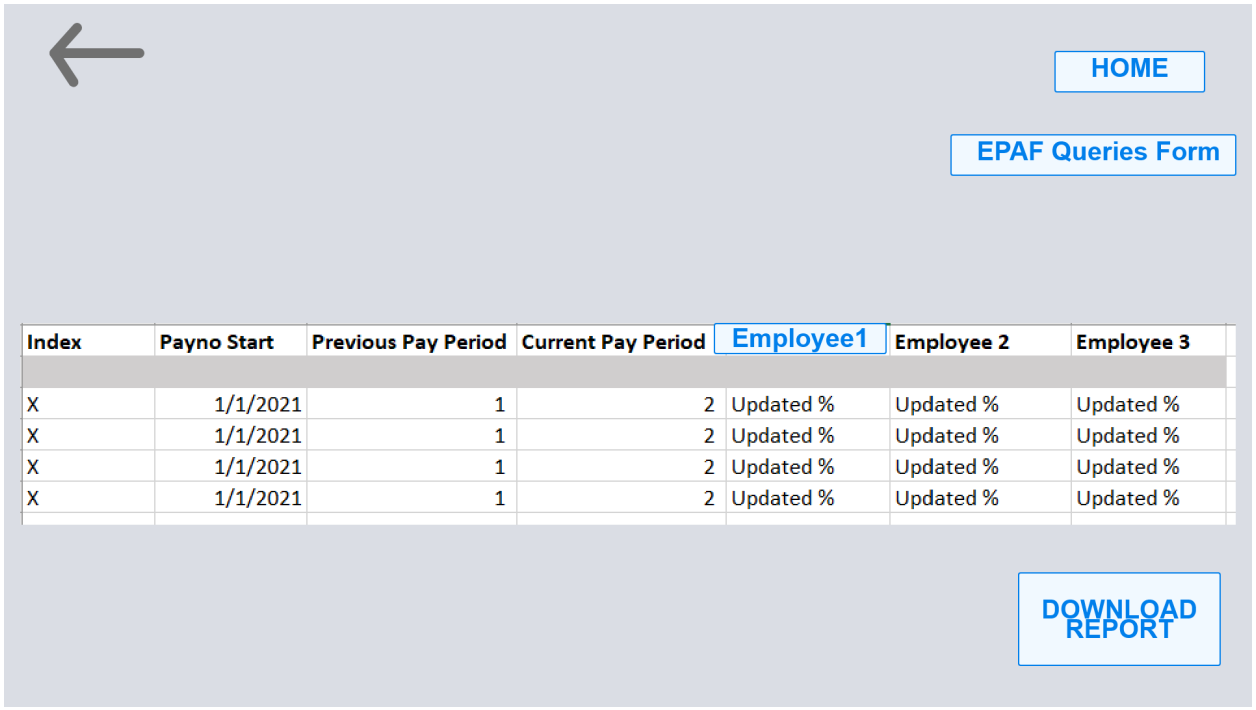
This is an example of a visualization created by the above screen. This table is static and doesn't get updated manually by the user. A low level access user would only be able to look at this table, but an Operations Manager or Finance Admin would be able to make edits to the encumbrances if they desired to do so

Epaf Queries:



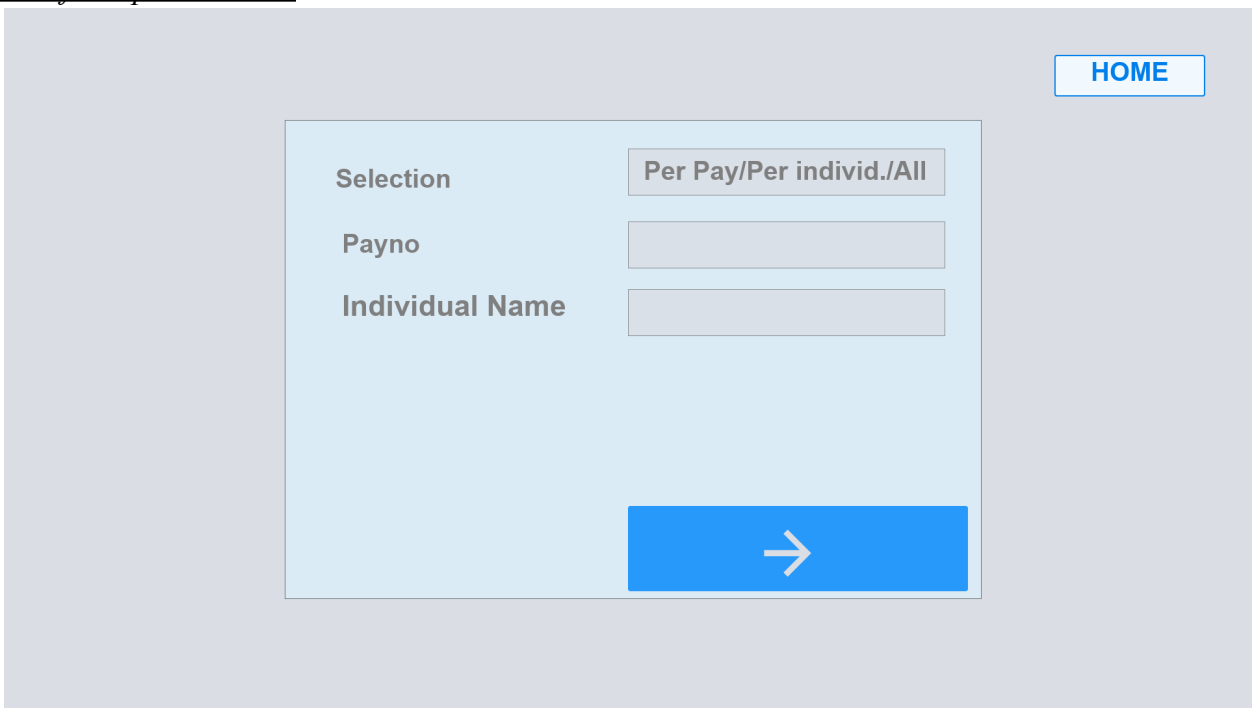
This screen allows the user to create a visualization of updated pay percentages since the previous pay number for the indexes from a certain team or from just one index of choice.

Alternatively, the user can choose to create a visualization of updated pay percentages since the previous pay number for all indices across all teams.

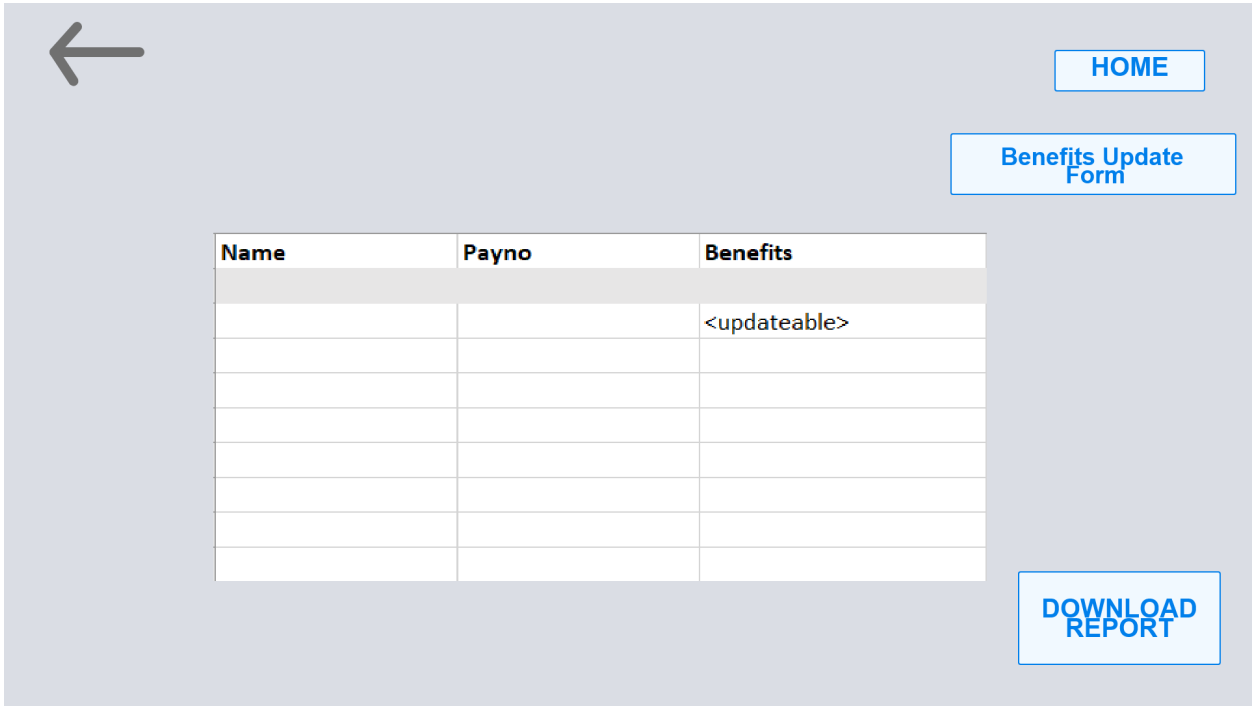


This is an example of a visualization created by the above screen. This table is static and doesn't get updated manually by the user.

Benefits Update Form:

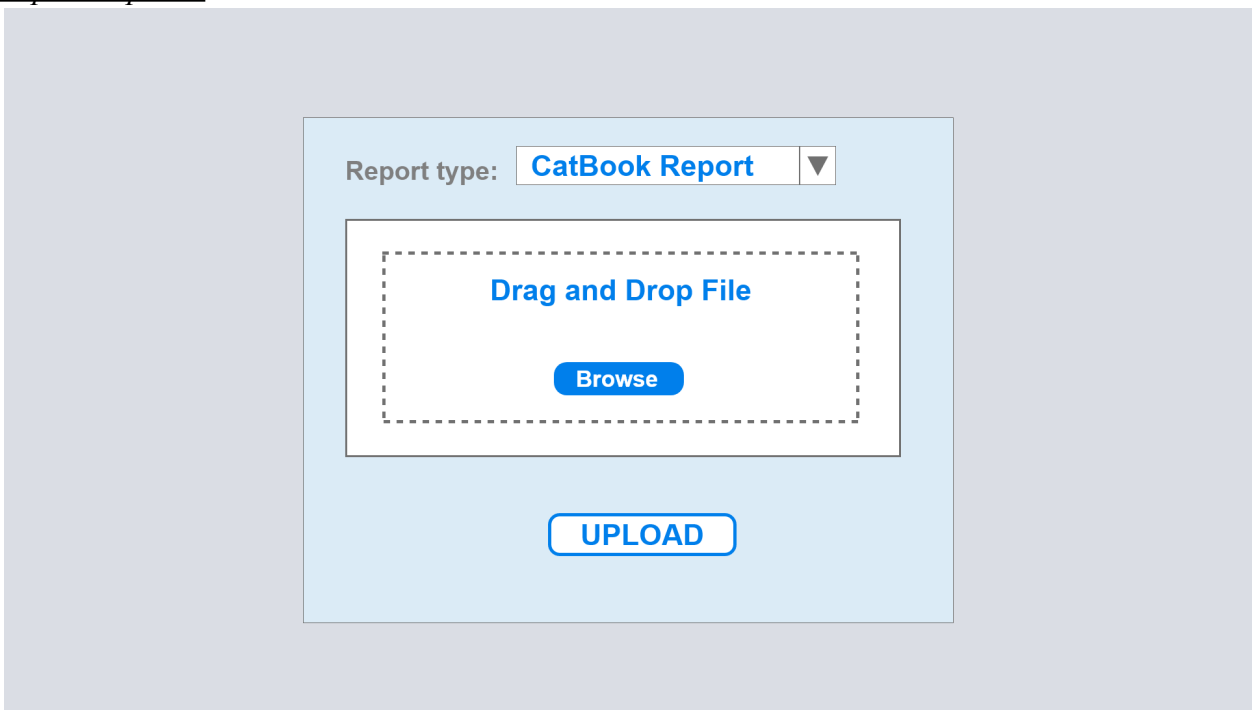


This screen allows the user to create a visualization of the employee benefits for a particular employee, or the employee benefits for all employees given a particular pay number.

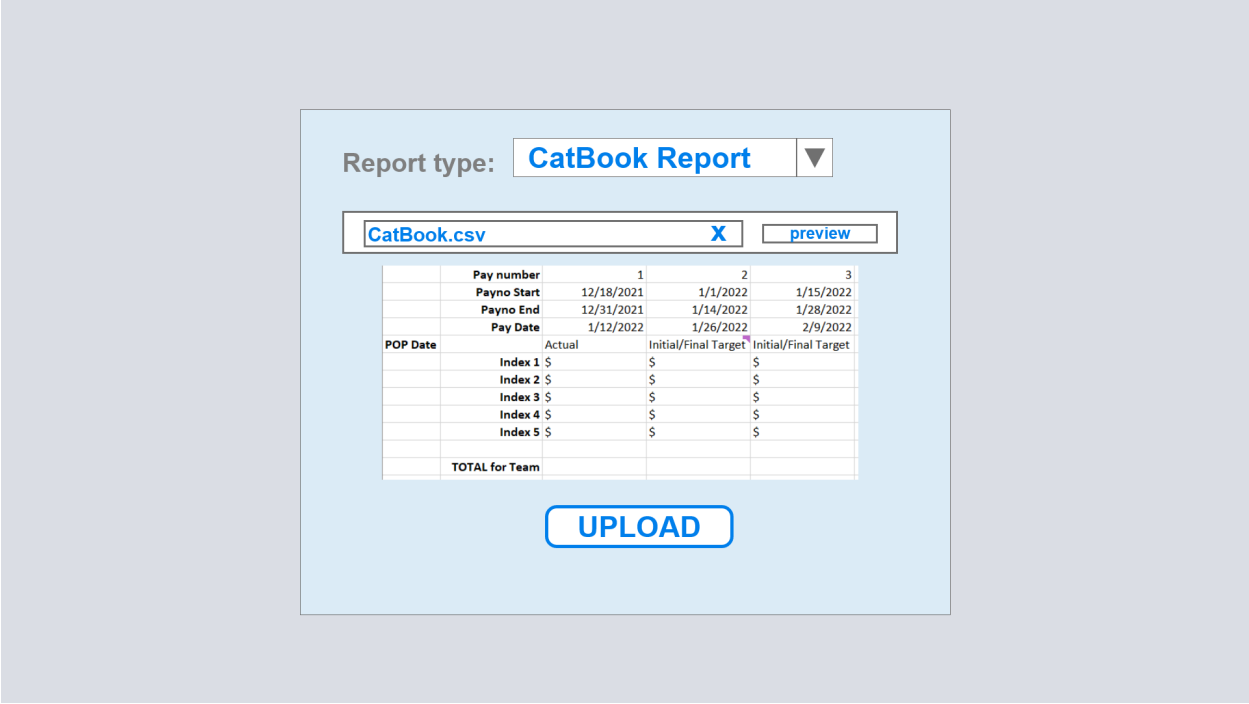


This is an example of a visualization created by the above screen. A low level access user or an Operations Manager would only be able to look at this table, but a Finance Admin would be able to make edits to the benefits if they desired to do so.

Import Reports:



This screen allows the user to import various reports into the database. A low level user would not be able to access this feature.

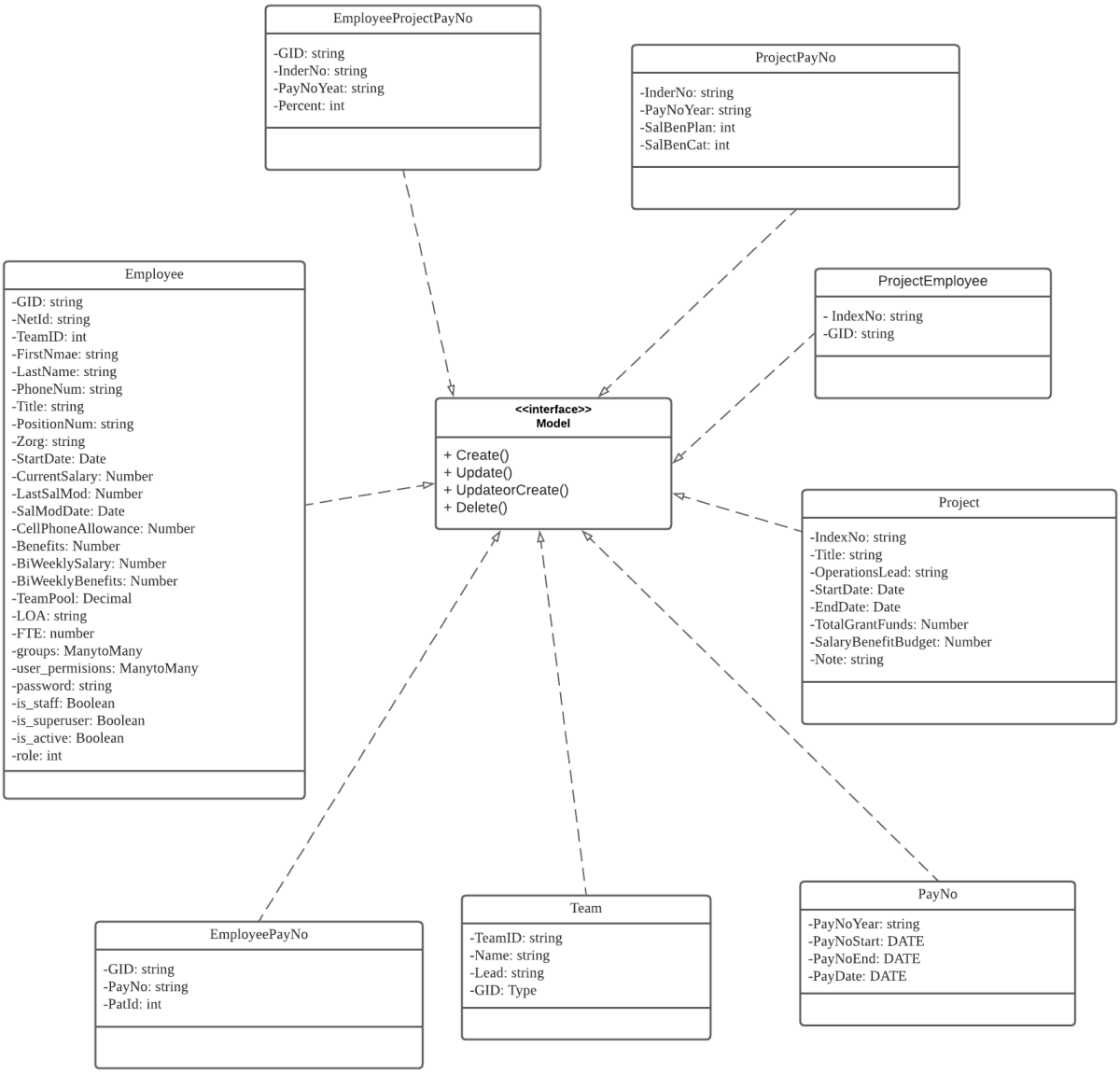


This is an extra confirmation screen after selecting a file to import. A window pops up showing a preview of the data in the selected file, giving the user a second chance to make sure that the data they selected to import is correct.

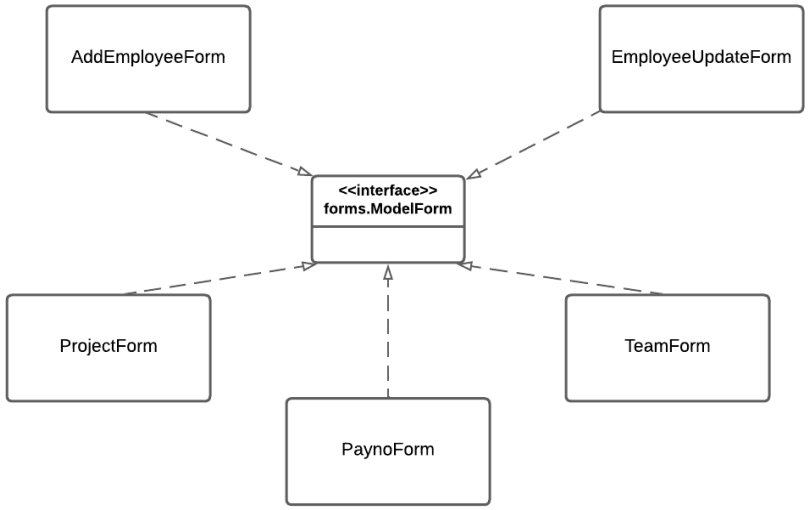
Security:

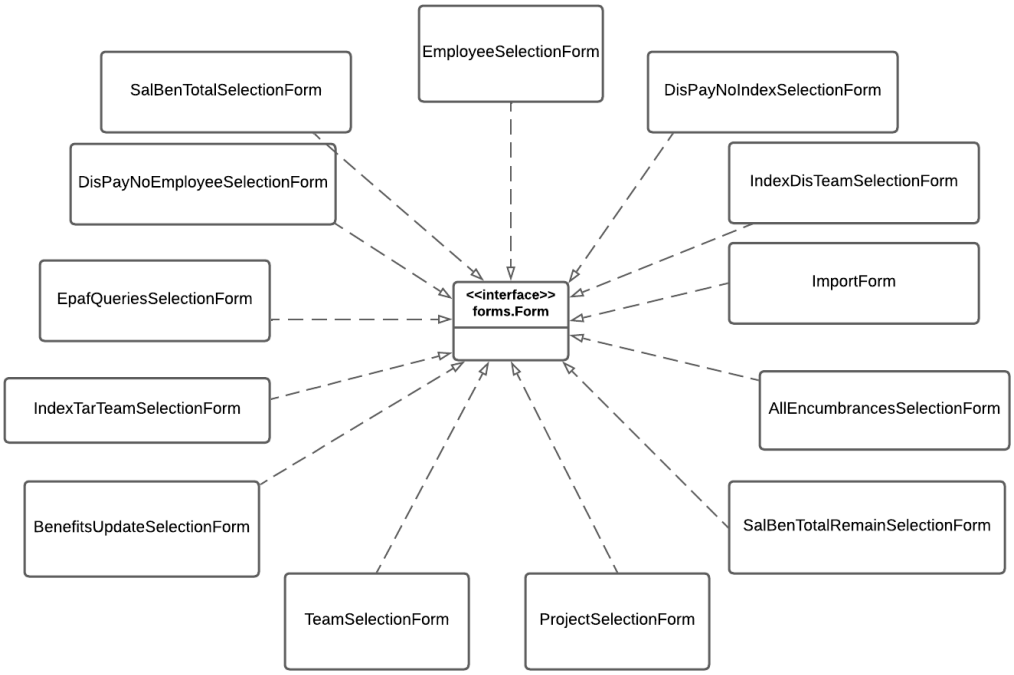
Because our team is dealing with sensitive information, security is an important part of our web-app. We are using the SHA3-256 hashing algorithm to store the user's password to keep them secure. This is one of the current recommended hash algorithms by the US government. (CSD) We are also using a 16 byte salt appended to the beginning of the user's plaintext password to maintain uniqueness between passwords when stored as hashes. The salt is stored in the database along with the hash of both the salt and the plaintext password but the plaintext password by itself is not stored.

Class Diagrams:

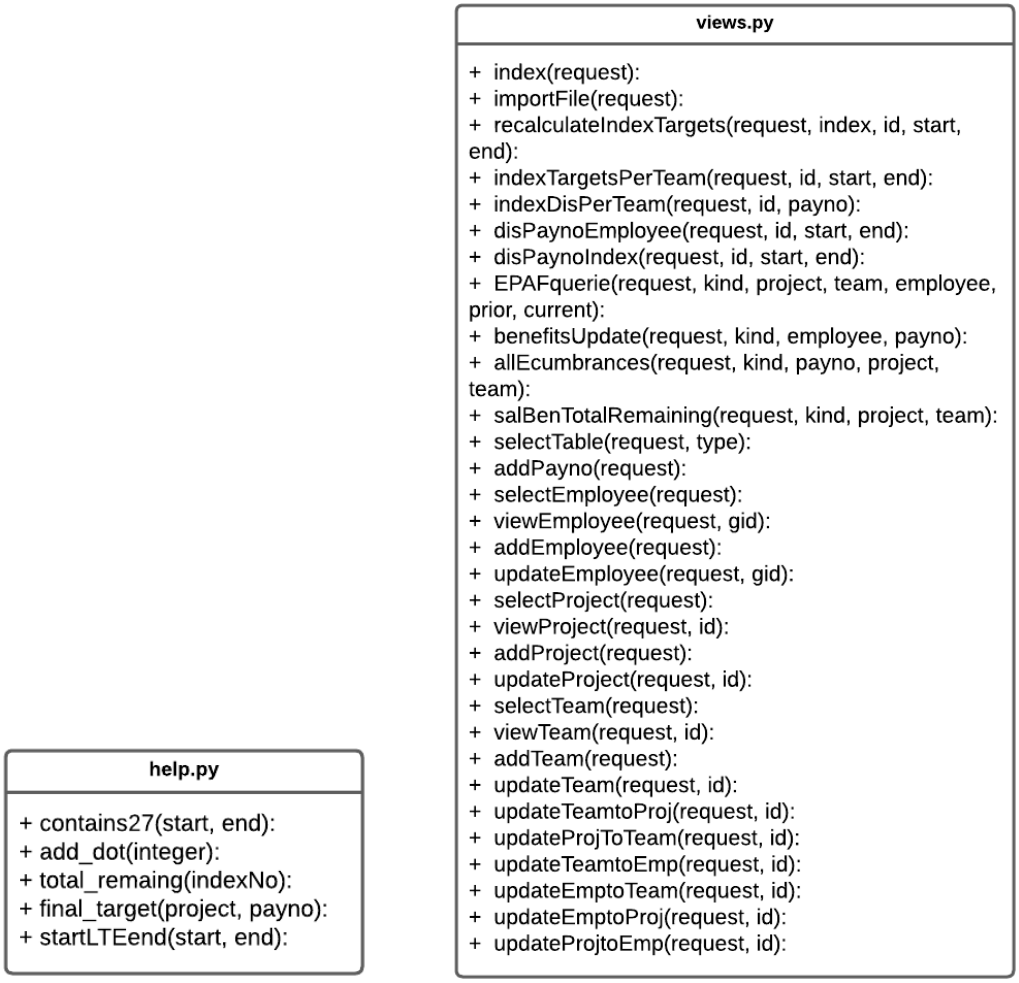


These classes implement the django model interface provided by python. They will all have CRUD functionality that will be used by the front end to update the database as necessary when users input changes.

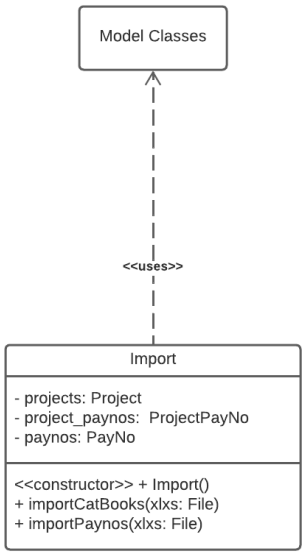




The forms classes inherit from either the Django models.Form interface or the Django forms.Form interface. These are used to create the forms that are displayed on the front end allowing us to update model objects or select what to view on the pages.



Help and Views are collections of functions. The functions in help are used within the functions in views. The functions in views help to create the pages that are seen on the front end. These functions allow us to get the data to fill in the cells on the front end using the information filled in by the user in the initial form field before they access a specific view. The functions in views also check the permissions of the user before they are allowed to view or change anything on the front end.



This class uses the different model classes from the first class diagram. A more detailed explanation as to why is provided in the section labeled “Design Pattern”.

Design Pattern:

Our team decided to make use of the Decorator Pattern when designing the import class for our back-end. This class is used to import different reports from xlsx files and to update the database accordingly. The Import class was the main reason we decided to use this design pattern. The Import class has to use the functions from the different model classes to be able to update the database with information from the xlsx file provided but import functions cannot be added to the individual model classes because the xlsx file might contain information needed in multiple tables in the database. So, we had to create an import class that uses the model class objects rather than the other way around. We decided to have the Import class contain objects of all the model classes that were initialized in the constructor for Import and used by the different functions as needed. This simplifies the implementation of the Import functionalities in the front-end because instead of having to initialize multiple of the different model classes explicitly we just initialize Import and then call a singular function to initialize the different model classes

for us and then we can use them within our different functions making the front-end implementation cleaner.

Design Tradeoffs:

Initially our team was planning on using PHP to build the back-end of our web-app. We considered PHP because of our LAMP stack design. One of our team members already had experience building multiple LAMP stack web-apps using PHP as the back-end. However, the other members of the team had not used PHP before. We still wanted to use a LAMP stack so our team decided to switch our design to be using Python as our back-end using the Django framework. All of our team members have previous experience using Python. As well as Python, and specifically Django, would provide us with more security moving forward. (DjangoProject) We needed our web-app to be as secure as possible because we are dealing with sensitive financial and personal information from the government. Python was the better choice in this case.

Our choice of using Python in our web stack also led to another design tradeoff. While considering PHP for our final design we were planning on building our own ORM tool within PHP since PHP does not have one already baked in. This would have given us more direct access to the SQL and by definition more direct access to the underlying database. However when we switched to considering Python we saw that Django had a built in ORM tool in the form of the models library from Python. Models is a lightweight framework for mapping Python objects to tables in a database. We decided to make the switch out of convenience. Even though it will give us less direct access to the database, this will cut down on our development time allowing us to do more with our web-app.

Another design tradeoff we made was when we were designing the Import and Export classes. There were three choices we could have made: having the export and importing functions be a part of the model classes and not separated, having one class for both import and export, and having two classes one for import and one for export. We decided to go with the third option. The first one wouldn't have worked because both import and export would have required the use of multiple model classes and therefore it wouldn't have made sense to contain the different functions to a singular model class. For instance, when importing a report from CatBooks, the information contained pertains to both EmployeeProjectPayNo, and EmployeePayNo so it would have been difficult to implement that in a single class without it getting lost. The second option would have violated the Single Responsibility Principle in OOP because it would have had two responsibilities: importing and exporting. The third option made the most sense to our team. It allowed us to import and export efficiently without having to worry about what report belonged to which model object and it would separate the responsibilities cleanly between the two classes.

Expected Results:

Our expected results for this project was the successful production of a LAMP stack web app capable of enabling a measurable decrease in bi-weekly budgeting time for the Miltech finance management team. We did have two stretch goals for this project. The first stretch goal was to implement an entry logging feature. Whenever a user would make a change to the database, the entry logger would track this change. A history of recent changes made to the database could be viewed by accessing a particular screen from the dashboard. Ideally, the user would be able to revert the database to a previous version by accessing this history. The idea is that if a mistake is made, a user can undo the mistake by reverting to an older version of the

database before the erroneous edit was made. Additionally, this feature would help Miltech understand who was making changes to the database and when. The second stretch goal was to implement simultaneous editing functionality. This would allow multiple users to make edits to the same table in the database at the same time without causing issues in the database. This feature could be very useful, as the different Operations Managers and Finance Admins often work together on budgeting issues and are accessing and updating things at the same or similar times.

```
 1 from django.contrib import admin
 2 from .models import *
 3 from django.contrib import admin
 4 from django.contrib.auth.admin import UserAdmin
 5 from .forms import EmployeeCreationForm, EmployeeChangeForm
 6 from .models import Employee
 7
 8
 9 class EmployeeAdmin(UserAdmin):
10     add_form = EmployeeCreationForm
11     form = EmployeeChangeForm
12     model = Employee
13     list_display = ("GID", "role",)
14     list_filter = ("GID", "role",)
15     fieldsets = (
16         (None, {"fields": ("GID", "password")}),
17         ("Permissions", {"fields": ("role", "groups", "
   user_permissions")}),
18     )
19     add_fieldsets = (
20         (None, {
21             "classes": ("wide",),
22             "fields": (
23                 "GID", "password1", "password2", "role",
24                 "groups", "user_permissions"
25             )}
26         ),
27     )
28     search_fields = ("GID",)
29     ordering = ("GID",)
30
31
32 admin.site.register(Employee, EmployeeAdmin)
33 admin.site.register(Project)
34 admin.site.register(Project_Employee)
35 admin.site.register(Team)
36 admin.site.register(Team_Employee)
37 admin.site.register(Project_Team)
38 admin.site.register(PayNo)
39 admin.site.register(Employee_PayNo)
40 admin.site.register(Project_PayNo)
41 admin.site.register(Employee_Project_PayNo)
42
```

```
 1 from django import forms
 2 from django.contrib.auth.forms import UserCreationForm,
   UserChangeForm
 3 from django.core.validators import FileExtensionValidator
 4 from .models import *
 5
 6
 7 class EmployeeCreationForm(UserCreationForm):
 8     NetId = forms.CharField(max_length=50, required=True)
 9     FirstName = forms.CharField(max_length=50, required=True)
10     LastName = forms.CharField(max_length=50, required=True)
11     PhoneNum = forms.CharField(max_length=50, required=True)
12     Email = forms.EmailField(required=True)
13     Title = forms.CharField(max_length=50, required=True)
14     PositionNum = forms.CharField(max_length=50, required=True)
15     Zorg = forms.CharField(max_length=50, required=True)
16     StartDate = forms.DateField(required=True,)
17     CurrentSalary = forms.IntegerField(required=True)
18     LastSalMod = forms.IntegerField(required=True)
19     SalModDate = forms.DateField(required=True)
20     CellphoneAllowance = forms.IntegerField(required=True)
21     Benefits = forms.IntegerField(required=True)
22     BiweeklySalary = forms.IntegerField(required=True)
23     BiweeklyBenefits = forms.IntegerField(required=True)
24     TeamPool = forms.CharField(max_length=50, required=True)
25     LOA = forms.CharField(max_length=50, required=True)
26     FTE = forms.CharField(max_length=50, required=True)
27
28     class Meta:
29         model = Employee
30         fields = ('GID', 'NetId', 'FirstName', 'LastName', '
   PhoneNum', 'Email',
31                   'Title', 'PositionNum', 'Zorg', 'StartDate',
   'CurrentSalary',
32                   'LastSalMod', 'SalModDate', '
   CellphoneAllowance', 'Benefits',
33                   'BiweeklySalary', 'BiweeklyBenefits', '
   TeamPool', 'LOA', 'FTE',
34                   'password1', 'password2')
35
36
37 class EmployeeChangeForm(UserChangeForm):
38     class Meta:
39         model = Employee
40         fields = ('GID', 'NetId', 'FirstName', 'LastName', '
   PhoneNum', 'Email',
41                   'Title', 'PositionNum', 'Zorg', 'StartDate',
   'CurrentSalary',
```

```
42                  'LastSalMod', 'SalModDate', '
   CellphoneAllowance', 'Benefits',
43                  'BiweeklySalary', 'BiweeklyBenefits', '
   TeamPool', 'LOA', 'FTE')
44
45 class ProjectForm(forms.ModelForm):
46     IndexNo = models.CharField(max_length=50)
47     Title = models.CharField(max_length=50)
48     OperationsLead = models.CharField(max_length=50)
49     StartDate = models.DateField()
50     EndDate = models.DateField()
51     TotalGrantFunds = models.IntegerField()
52     SalaryBenefitBudget = models.IntegerField()
53     Note = models.CharField(max_length=200)
54     class Meta:
55         model = Project
56         fields = ('IndexNo','Title','OperationsLead','StartDate
   ','EndDate',
57                   'TotalGrantFunds','SalaryBenefitBudget','Note
   ')
58
59 class TeamForm(forms.ModelForm):
60     TeamId = forms.CharField(max_length=50)
61     Name = forms.CharField(max_length=50)
62     Lead = forms.CharField(max_length=50)
63     class Meta:
64         model = Team
65         fields = ('TeamId','Name','Lead')
66
67 class PaynoForm(forms.ModelForm):
68     Year = forms.CharField(max_length=4)
69     Start = forms.DateField(required=True)
70     End = forms.DateField(required=True)
71     Date = forms.DateField(required=True)
72     class Meta:
73         model = PayNo
74         fields = ('Year', 'Start', 'End', 'Date')
75 class TeamSelectionForm(forms.Form):
76     all_teams = Team.objects.all().order_by('TeamId')
77     team_choices = []
78     for t in all_teams:
79         team_choices.append((t.TeamId,t.Name))
80     team = forms.ChoiceField(choices=team_choices, required=
   True)
81 class ProjectSelectionForm(forms.Form):
82     all_projects = Project.objects.all().order_by('IndexNo').
   exclude(Note="Subaccount")
83     project_choices = []
```

```
 84     for p in all_projects:
 85         project_choices.append((p.IndexNo, p.IndexNo))
 86     project = forms.ChoiceField(choices=project_choices,
    required=True)
 87
 88
 89 class EmployeeSelectionForm(forms.Form):
 90     all_employees = Employee.objects.all().order_by('FirstName
    ', 'LastName')
 91     employee_choices = []
 92     for e in all_employees:
 93         employee_choices.append((e.GID, e.FirstName + " " + e.
    LastName))
 94     employee = forms.ChoiceField(choices=employee_choices,
    required=True)
 95
 96 class IndexTarTeamSelectionForm(forms.Form):
 97     all_teams = Team.objects.all().order_by('Name')
 98     team_choices = [('all', 'Get All Indexes')]
 99     for t in all_teams:
100         team_choices.append((t.TeamId, t.Name))
101     team = forms.ChoiceField(choices=team_choices, required=
    True)
102     all_PayNo = PayNo.objects.all().order_by('Start')
103     payno_choices = []
104     for p in all_PayNo:
105         payno_choices.append((p.Year, p.Year + "(" + str(p.
    Start) + ", " + str(p.End) + ")"))
106     start = forms.ChoiceField(choices=payno_choices, required=
    True)
107     end = forms.ChoiceField(choices=payno_choices, required=
    True)
108
109 class IndexDisTeamSelectionForm(forms.Form):
110     all_teams = Team.objects.all().order_by('Name')
111     team_choices = [('all', 'Get All Indexes')]
112     for t in all_teams:
113         team_choices.append((t.TeamId, t.Name))
114     team = forms.ChoiceField(choices=team_choices, required=
    True)
115     all_PayNo = PayNo.objects.all().order_by('Start')
116     payno_choices = []
117     for p in all_PayNo:
118         if p.Year[:2] != "27":
119             payno_choices.append((p.Year, p.Year + "(" + str(p
    .Start) + ", " + str(p.End) + ")"))
120     payno = forms.ChoiceField(choices=payno_choices, required=
    True)
```

```
121
122 class SalBenTotalSelectionForm(forms.Form):
123     selection = forms.ChoiceField(choices=[('i','index'),('t',
    'team'),('a','all')], required=True)
124     all_teams = Team.objects.all().order_by('Name')
125     team_choices = [('none','No Team Selected')]
126     for t in all_teams:
127         team_choices.append((t.TeamId, t.Name))
128     team = forms.ChoiceField(choices=team_choices)
129     all_projects = Project.objects.all().order_by('IndexNo')
130     project_choices = [('none','No Project Selected')]
131     for p in all_projects:
132         project_choices.append((p.IndexNo, p.IndexNo))
133     project = forms.ChoiceField(choices=project_choices)
134
135 class AllEncumSelectionForm(forms.Form):
136     selection = forms.ChoiceField(choices=[('i','index'),('t',
    'team'),('a','all')], required=True)
137     all_teams = Team.objects.all().order_by('Name')
138     team_choices = [('none','No Team Selected')]
139     for t in all_teams:
140         team_choices.append((t.TeamId, t.Name))
141     team = forms.ChoiceField(choices=team_choices)
142     all_projects = Project.objects.all().order_by('IndexNo')
143     project_choices = [('none','No Project Selected')]
144     for p in all_projects:
145         project_choices.append((p.IndexNo, p.IndexNo))
146     project = forms.ChoiceField(choices=project_choices)
147     all_PayNo = PayNo.objects.all().order_by('Start')
148     payno_choices = []
149     for p in all_PayNo:
150         if p.Year[:2] != "27":
151             payno_choices.append((p.Year, p.Year + "(" + str(p
    .Start) + ", " + str(p.End) + ")"))
152     payno = forms.ChoiceField(choices=payno_choices, required=
    True)
153
154 class EPAFQueriesSelectionForm(forms.Form):
155     selection = forms.ChoiceField(choices=[('i', 'index'), ('t
    ', 'team'), ('e','employee'),('a', 'all')], required=True)
156     all_teams = Team.objects.all().order_by('Name')
157     team_choices = [('none','No Team Selected')]
158     for t in all_teams:
159         team_choices.append((t.TeamId, t.Name))
160     team = forms.ChoiceField(choices=team_choices)
161     all_projects = Project.objects.all().order_by('IndexNo')
162     project_choices = [('none','No Project Selected')]
163     for p in all_projects:
```

```
164         project_choices.append((p.IndexNo, p.IndexNo))
165     project = forms.ChoiceField(choices=project_choices)
166     all_employees = Employee.objects.all().order_by('GID')
167     employee_choices = [('none', 'No Employee Selected')]
168     for e in all_employees:
169         employee_choices.append((e.GID, e.__str__()))
170     employee = forms.ChoiceField(choices=employee_choices)
171     all_PayNo = PayNo.objects.all().order_by('Start')
172     payno_choices = []
173     for p in all_PayNo:
174         if p.Year[:2] != "27":
175             payno_choices.append((p.Year, p.Year + "(" + str(p
    .Start) + ", " + str(p.End) + ")"))
176     prior = forms.ChoiceField(choices=payno_choices, required=
    True)  # maybe have this be automatic based on current date/
    selected 'current' payno
177     current = forms.ChoiceField(choices=payno_choices,
    required=True)
178 class BenefitsUpdateSelectionForm(forms.Form):
179     selection = forms.ChoiceField(choices=[('p', 'Per Payno'
    ), ('i', 'Per Individual'), ('a', 'all')], required=True)
180     all_PayNo = PayNo.objects.all().order_by('Start')
181     payno_choices = [('none','No Payno Selected')]
182     for p in all_PayNo:
183         if p.Year[:2] != "27":
184             payno_choices.append((p.Year, p.Year + "(" + str(p
    .Start) + ", " + str(p.End) + ")"))
185     payno = forms.ChoiceField(choices=payno_choices, required=
    True)
186     all_employees = Employee.objects.all().order_by('FirstName
    ', 'LastName')
187     employee_choices = [('none','No Employee Selected')]
188     for e in all_employees:
189         employee_choices.append((e.GID, e.FirstName + " " + e.
    LastName))
190     employee = forms.ChoiceField(choices=employee_choices,
    required=True)
191
192 class DisPayNoEmployeeSelectionForm(forms.Form):
193     all_employees = Employee.objects.all().order_by('FirstName
    ', 'LastName')
194     employee_choices = []
195     for e in all_employees:
196         employee_choices.append((e.GID, e.FirstName + " " + e.
    LastName))
197     employee = forms.ChoiceField(choices=employee_choices,
    required=True)
198     all_PayNo = PayNo.objects.all().order_by('Start')
```

```
199     payno_choices = []
200     for p in all_PayNo:
201         if p.Year[:2] != "27":
202             payno_choices.append((p.Year, p.Year + "(" + str(p
    .Start) + ", " + str(p.End) + ")"))
203     start = forms.ChoiceField(choices=payno_choices, required=
    True)
204     end = forms.ChoiceField(choices=payno_choices, required=
    True)
205
206 class DisPayNoIndexSelectionForm(forms.Form):
207     all_indexes = Project.objects.all().order_by('Title')
208     index_choices = []
209     for i in all_indexes:
210         index_choices.append((i.IndexNo, i.Title))
211     index = forms.ChoiceField(choices=index_choices, required=
    True)
212     all_PayNo = PayNo.objects.all().order_by('Start')
213     payno_choices = []
214     for p in all_PayNo:
215         if p.Year[:2] != "27":
216             payno_choices.append((p.Year, p.Year + "(" + str(p
    .Start) + ", " + str(p.End) + ")"))
217     start = forms.ChoiceField(choices=payno_choices, required=
    True)
218     end = forms.ChoiceField(choices=payno_choices, required=
    True)
219
220
221
222 class AddEmployeeForm(forms.ModelForm):  # not used?
223     GID = forms.CharField(max_length=50, required=True)
224     NetId = forms.CharField(max_length=50, required=True)
225     FirstName = forms.CharField(max_length=50, required=True)
226     LastName = forms.CharField(max_length=50, required=True)
227     PhoneNum = forms.CharField(max_length=50, required=True)
228     Email = forms.EmailField(required=True)
229     Title = forms.CharField(max_length=50, required=True)
230     PositionNum = forms.CharField(max_length=50, required=True
    )
231     Zorg = forms.CharField(max_length=50, required=True)
232     StartDate = forms.DateField(required=True)
233     CurrentSalary = forms.IntegerField(required=True)
234     LastSalMod = forms.IntegerField(required=True)
235     SalModDate = forms.DateField(required=True)
236     CellphoneAllowance = forms.IntegerField(required=True)
237     Benefits = forms.IntegerField(required=True)
238     BiweeklySalary = forms.IntegerField(required=True)
```

```
239     BiweeklyBenefits = forms.IntegerField(required=True)
240     TeamPool = forms.CharField(max_length=50, required=True)
241     LOA = forms.CharField(max_length=50, required=True)
242     FTE = forms.CharField(max_length=50, required=True)
243 
244     class Meta:
245         model = Employee
246         fields = ['GID', 'NetId', 'FirstName', 'LastName', '
    PhoneNum', 'Email',
247                   'Title', 'PositionNum', 'Zorg', 'StartDate'
    , 'CurrentSalary',
248                   'LastSalMod', 'SalModDate', '
    CellphoneAllowance', 'Benefits',
249                   'BiweeklySalary', 'BiweeklyBenefits', '
    TeamPool', 'LOA', 'FTE']
250 
251 
252 class ImportForm(forms.Form):
253     fileType = forms.ChoiceField(choices=((1, 'CatBooks'), (2
    , 'Paynos')),
254                                  required=True)  # placeholder
     for actual import functions
255     file = forms.FileField(required=True,
256                            validators=[FileExtensionValidator(
    allowed_extensions=['xml', 'csv', 'xlsx'])])
257 
258 class EmployeeUpdateForm(forms.ModelForm):
259     GID = forms.CharField(max_length=50, required=True)
260     NetId = forms.CharField(max_length=50, required=True)
261     FirstName = forms.CharField(max_length=50, required=True)
262     LastName = forms.CharField(max_length=50, required=True)
263     PhoneNum = forms.CharField(max_length=50, required=True)
264     Email = forms.EmailField(required=True)
265     Title = forms.CharField(max_length=50, required=True)
266     PositionNum = forms.CharField(max_length=50, required=True
    )
267     Zorg = forms.CharField(max_length=50, required=True)
268     StartDate = forms.DateField(required=True)
269     CurrentSalary = forms.IntegerField(required=True)
270     LastSalMod = forms.IntegerField(required=True)
271     SalModDate = forms.DateField(required=True)
272     CellphoneAllowance = forms.IntegerField(required=True)
273     Benefits = forms.IntegerField(required=True)
274     BiweeklySalary = forms.IntegerField(required=True)
275     BiweeklyBenefits = forms.IntegerField(required=True)
276     TeamPool = forms.CharField(max_length=50, required=True)
277     LOA = forms.CharField(max_length=50, required=True)
278     FTE = forms.CharField(max_length=50, required=True)
```

```
279
280     class Meta:
281         model = Employee
282         fields = ['GID', 'NetId', 'FirstName', 'LastName', '
    PhoneNum', 'Email',
283                   'Title', 'PositionNum', 'Zorg', 'StartDate'
    , 'CurrentSalary',
284                   'LastSalMod', 'SalModDate', '
    CellphoneAllowance', 'Benefits',
285                   'BiweeklySalary', 'BiweeklyBenefits', '
    TeamPool', 'LOA', 'FTE']
286
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\models.py

```
from django.db import models
from django.contrib.auth.models import AbstractBaseUser,
PermissionsMixin
from .managers import CustomUserManager

class Employee(AbstractBaseUser, PermissionsMixin):
    GID = models.CharField(max_length=50, unique=True)
    NetId = models.CharField(max_length=50, unique=True)
    #AccessLevel = models.BooleanField(default=False)
    TeamId = models.CharField(max_length=50)
    FirstName = models.CharField(max_length=50)
    LastName = models.CharField(max_length=50)
    PhoneNum = models.CharField(max_length=50)
    Email = models.EmailField(max_length=50)
    Title = models.CharField(max_length=50)
    PositionNum = models.CharField(max_length=50)
    Zorg = models.CharField(max_length=50)
    StartDate = models.DateField()
    CurrentSalary = models.IntegerField()
    LastSalMod = models.IntegerField()
    SalModDate = models.DateField()
    CellphoneAllowance = models.IntegerField()
    Benefits = models.IntegerField()
    BiweeklySalary = models.IntegerField()
    BiweeklyBenefits = models.IntegerField()
    TeamPool = models.CharField(max_length=50)
    LOA = models.CharField(max_length=50)
    FTE = models.CharField(max_length=50)

    groups = models.ManyToManyField(to='auth.Group')
    user_permissions = models.ManyToManyField(to='auth.
Permission')
    password = models.CharField(max_length=100)
    is_staff = models.BooleanField(default=True)
    is_superuser = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)

    FINANCE_LEAD = 1
    OPERATIONS_LEAD = 2
    OPERATIONS_SUPPORT = 3
    HR = 4
    EPAF_SUPPORT = 5

    ROLE_CHOICES = (
        (FINANCE_LEAD, 'Finance Lead'),
        (OPERATIONS_LEAD, 'Operations Lead'),
        (OPERATIONS_SUPPORT, 'Operations Support'),
        (HR, "HR"),
```

```
47          (EPAF_SUPPORT, "EPAF Support")
48      )
49      role = models.PositiveSmallIntegerField(choices=
    ROLE_CHOICES, blank=True, null=True)
50
51      USERNAME_FIELD = 'NetId'
52      REQUIRED_FIELDS = ['GID', 'role', 'TeamId', 'FirstName', '
    LastName', 'PhoneNum', 'Email', 'Title', 'PositionNum', 'Zorg'
    , 'StartDate', 'CurrentSalary', 'LastSalMod', 'SalModDate', '
    CellphoneAllowance', 'Benefits', 'BiweeklySalary', '
    BiweeklyBenefits', 'TeamPool', 'LOA', 'FTE']
53
54      objects = CustomUserManager()
55      def __str__(self):
56          return self.FirstName + ' ' + self.LastName
57
58  class Project(models.Model):
59      IndexNo = models.CharField(max_length=50)
60      Title = models.CharField(max_length=50)
61      OperationsLead = models.CharField(max_length=50)
62      StartDate = models.DateField()
63      EndDate = models.DateField()
64      TotalGrantFunds = models.IntegerField()
65      # CurrentGrantFunds = models.IntegerField(default=0)
66      SalaryBenefitBudget = models.IntegerField()
67      # SalBenBudgetSpentCat = models.IntegerField(default=0)
68      # SalBenBudgetSpentPlan = models.IntegerField(default=0)
69      # SalBenEnCat = models.IntegerField(default=0)
70      # SalBenEnPlan = models.IntegerField(default=0)
71      # SalBenCatPlan = models.IntegerField(default=0)
72      # SalBenRemainPlan = models.IntegerField(default=0)
73      Note = models.CharField(max_length=200)
74
75      def __str__(self):
76          return self.IndexNo + ' ' + self.Title
77
78  class Project_Employee(models.Model):
79      IndexNo = models.CharField(max_length=50)
80      GID = models.CharField(max_length=50)
81
82  class Team(models.Model):
83      TeamId = models.CharField(max_length=50)
84      Name = models.CharField(max_length=50)
85      Lead = models.CharField(max_length=50)
86
87  class Team_Employee(models.Model):
88      TeamId = models.CharField(max_length=50)
89      GID = models.CharField(max_length=50)
```

```
 90
 91 class Project_Team(models.Model):
 92     IndexNo = models.CharField(max_length=50)
 93     TeamId = models.CharField(max_length=50)
 94
 95 class PayNo(models.Model):
 96     Year = models.CharField(max_length=50)
 97     Start = models.DateField()
 98     End = models.DateField()
 99     Date = models.DateField()
100     def __str__(self):
101         return self.Year
102
103 class Employee_PayNo(models.Model):
104     GID = models.CharField(max_length=50)
105     Year = models.CharField(max_length=50)
106     Paid = models.IntegerField(default=0)
107     Benefits = models.IntegerField(default=0)
108
109 class Project_PayNo(models.Model):
110     IndexNo = models.CharField(max_length=50)
111     Year = models.CharField(max_length=50)
112     SalBenPlan = models.IntegerField(default=0)
113     SalCat = models.IntegerField(default=0)
114     BenCat = models.IntegerField(default=0)
115     def __str__(self):
116         return self.IndexNo + ' ' + self.Year
117
118 class Employee_Project_PayNo(models.Model):
119     GID = models.CharField(max_length=50)
120     IndexNo = models.CharField(max_length=50)
121     Year = models.CharField(max_length=50)
122     Percent = models.IntegerField(default=0)
123
124
```

```python
from django.contrib.auth.base_user import BaseUserManager
from django.utils.translation import gettext_lazy as _


class CustomUserManager(BaseUserManager):
    """
    Custom user model manager where email is the unique identifiers
    for authentication instead of usernames.
    """
    def create_user(self, GID, password, **extra_fields):
        """
        Create and save a user with the given email and password.
        """
        if not GID:
            raise ValueError(_("The GID must be set"))
        user = self.model(GID=GID, **extra_fields)
        user.set_password(password)
        user.save()
        return user

    def create_superuser(self, GID, password, **extra_fields):
        """
        Create and save a SuperUser with the given email and password.
        """
        extra_fields.setdefault("is_staff", True)
        extra_fields.setdefault("is_superuser", True)
        extra_fields.setdefault("is_active", True)

        if extra_fields.get("is_staff") is not True:
            raise ValueError(_("Superuser must have is_staff=True."))
        if extra_fields.get("is_superuser") is not True:
            raise ValueError(_("Superuser must have is_superuser=True."))
        return self.create_user(GID, password, **extra_fields)

```

```python
from django.http import HttpResponseRedirect
from django.shortcuts import render, redirect
from django.contrib import messages
# from lxml import objectify
import xml.etree.ElementTree as et
import pandas as pd
from datetime import datetime

from main.models import *
from main.forms import *

class Import():
    project_paynos = Project_PayNo.objects
    projects = Project.objects
    paynos = PayNo.objects

    def importPaynos(self, xlxs):
        dataframe = pd.read_excel(xlxs)

        dataframe = dataframe.reset_index()
        for index, row in dataframe.iterrows():
            begin = row['Begin']
            end = row['End']
            pay = row['Pay']
            num = str(row['Paynum'])

            if len(num) == 1:
                num = "0" + num

            year = datetime.now().strftime("%Y")
            num = num + year[2:]

            payno = self.paynos.create(Year=num, Start=begin, End=end, Date=pay)
            payno.save()

    def importCatbooks(self, xml):
        dataframe = pd.read_excel(xml)
        test_list = ["61123", "61124", "61123N", "61125", "61165", "61224", "61225", "61308", "61311"]

        dataframe = dataframe.reset_index()
        for index, row in dataframe.iterrows():
            account_code = str(row['Account'])
            description = row['Description']
            amount = int(float(row['Amount'])*100)
            encumbered = int(float(row['Enc']) * 100)
            Index = row['Index']
```

```
47              clear = int(row['Clear'])
48              sub = row['SubAccount']
49
50              if clear in range(0, 12) or clear == 20:
51                  description = description.lower()
52                  if account_code == "61499":
53                      # Benefits
    ------------------------------------------------------------------
    ---------------------------------
54
55                      # Finding PayNo
56                      payno_check = "payno "
57                      cut = description.find(payno_check) + len(
    payno_check)
58                      payno_id = description[cut:]
59                      space_position = payno_id.find(" ")
60                      if space_position != -1:
61                          payno_id = payno_id.partition(' ')[0]
62                      if len(payno_id) == 1:
63                          payno_id = "0" + payno_id
64                      year = datetime.now().strftime("%Y")
65                      payno_id = payno_id + year[2:]
66                      payno = self.project_paynos.get_or_create(
    Year=payno_id, IndexNo=Index)
67                      payno = payno[0]
68
69                      new_benifts = payno.BenCat + amount
70                      payno.BenCat = new_benifts
71                      payno.save()
72
73                  elif account_code in test_list:
74                      # Salary
    ------------------------------------------------------------------
    ---------------------------------
75
76                      payno_id = description.split()
77                      payno_id = payno_id[len(payno_id)-2]
78                      if len(payno_id) == 1:
79                          payno_id = "0" + payno_id
80                      year = datetime.now().strftime("%Y")
81                      payno_id = payno_id + year[2:]
82                      payno = self.project_paynos.get_or_create(
    Year=payno_id, IndexNo=Index)
83                      payno = payno[0]
84
85                      new_benifts = payno.SalCat + amount
86                      payno.SalCat = new_benifts
87                      payno.save()
```

```
88
89                  elif account_code == "61100":
90                      # Encumbrance
     ------------------------------------------------------------
    -------------------------------
91
92                      # Finding PayNo
93                      payno_check = "payno "
94                      cut = description.find(payno_check) + len(
    payno_check)
95                      payno_id = description[cut:]
96                      space_position = payno_id.find(" ")
97                      if space_position != -1:
98                          payno_id = payno_id.partition(' ')[0]
99                      if len(payno_id) == 1:
100                         payno_id = "0" + payno_id
101                     year = datetime.now().strftime("%Y")
102                     payno_id = payno_id + year[2:]
103
104                     if type(sub) is str:
105                         project_id = sub.split()
106                         project_id = project_id[1]
107                         parent_project = self.projects.get(
    IndexNo=project_id)
108                         project = self.projects.get_or_create(
    IndexNo=sub, Title=sub,
109
    OperationsLead=parent_project.OperationsLead,
110
    StartDate=parent_project.StartDate,
111
    EndDate=parent_project.EndDate,
112
    TotalGrantFunds=parent_project.TotalGrantFunds,
113
    SalaryBenefitBudget=parent_project.SalaryBenefitBudget,
114
    Note="Subaccount")
115                         project = project[0]
116                         project.save()
117                         payno = self.project_paynos.
    get_or_create(Year=payno_id, IndexNo=sub)
118                         payno = payno[0]
119                         new_encumbrance = payno.SalBenPlan +
    encumbered
120                         payno.SalBenPlan = new_encumbrance
121                         payno.save()
122                     else:
```

```
123                             payno = self.project_paynos.
    get_or_create(Year=payno_id, IndexNo=Index)
124                             payno = payno[0]
125                             new_encumbrance = payno.SalBenPlan +
    encumbered
126                             payno.SalBenPlan = new_encumbrance
127                             payno.save()
128
129
130
131
132
133
134
135
```

```css
.Team_table{
    width: 50%;
    margin-left: auto;
    margin-right: auto;
    border-collapse: collapse;
}
.Team_tr,.Team_td{
    padding: 20px;
    background-color: white;
    text-align: center;
}

.Team_link_td{
    padding: 10px;
    text-align: center;
}
```

```
 1
 2 /*public css*/
 3 h1 {
 4     text-align: center;
 5     color: #707070;
 6     font-size: 40px;
 7     text-decoration: underline 1px #0C24F6;
 8 }
 9 .link,a{
10      text-decoration: none;
11  }
12
13
14
15
16
17
```

```css
.project_table{
    width: 80%;
    margin-left: auto;
    margin-right: auto;
    border: 1px solid;
    border-collapse: collapse;
}
.project_tr,.project_td{
    padding: 10px;
    background-color: white;
    text-align: center;
}
.project_body{
    background-color: #DADDE4;
}
.project_Note{
    padding: 50px;
    width: 100%;
    text-align: center;
    background-color: white;
}
.project_link_td{
    padding: 10px;
    text-align: center;
}
```

```css
/*employee table*/
.employee_table{
    width: 70%;
    margin-left: auto;
    margin-right: auto;
    border: 1px solid white;
    border-collapse: collapse;
}
.employee_tr,.employee_td{
    background-color: #F1F9FF;
    text-align: left;
    font-size: 20px;
}
.employee_td_title{
    font-size: 20px;
}
.employee_td_Name{
    font-size: 30px;
}
a{
    text-decoration: none;
}

```

File - C:\Users\yywwc\PycharmProjects\miltech\main\static\functions.js

```
yellow = 'gold'
green = 'lightgreen'
red = 'lightsalmon'

function reTotal(id) {
    col = id.split(' ')[0]
    row = 0
    total = 0
    while (document.getElementById(col+' '+String(row)) != null){
        total += Number(document.getElementById(col+' '+String(row)).value)
        row += 1
    }
    if (document.getElementById(col+"base") != null){
        total += Number(document.getElementById(col+"base").value)
    }
    document.getElementById(col).innerHTML = String(total) + "%"
    if (total > 100){
        document.getElementById(col).style.backgroundColor = red
    } else if(total < 100) {
        document.getElementById(col).style.backgroundColor = yellow
    } else {
        document.getElementById(col).style.backgroundColor = green
    }
}

function baseTotals(id){ // add color changes to element
    ids = document.getElementById(id).name.split(" ")
    base = Number(document.getElementById(ids[1]+" "+ids[0]+"base").value)
    percent = Number(document.getElementById(id).value)
    document.getElementById(ids[1]+" "+ids[0]).innerHTML = String(base+percent) + "%"
    if (base+percent > 100){
        document.getElementById(ids[1]+" "+ids[0]).style.backgroundColor = red
    } else if(base+percent < 100) {
        document.getElementById(ids[1]+" "+ids[0]).style.backgroundColor = yellow
    } else {
        document.getElementById(ids[1]+" "+ids[0]).style.backgroundColor = green
```

```
37      }
38  }
39
40  function dispaynoChange(id){
41      baseTotals(id);
42      reFinal(id);
43  }
44
45  function reFinal(id) {
46      col = id.split(' ')[0]
47      row = 0
48      final = 0
49      while (document.getElementById(col+' '+String(row)) != null
    ){
50          employee = document.getElementById(col+' '+String(row))
51          wage = Number(document.getElementById(employee.name.
    split(' ')[1]).value)
52          final += Math.round(((Number(employee.value)/100) *
    wage / 100), 2)
53          row += 1
54      }
55      document.getElementById(col+"fin").innerHTML = String(final
    )
56      document.getElementById(col+"dif").innerHTML = String(Math.
    round(Number(document.getElementById(col+"init").innerHTML)-
    final))
57  }
```

```css
.dashboard_body{
   background-color: #DADDE4;
}
.dashboard_table{
    width: 100%;
    border-spacing: 50px;
}
.dashboard_td{
    padding-top: 30px;
    color: #007FEB;
    font-size: 20px;
    text-align: center;
    border: 3px solid #007FEB;
    background-color: #F1F9FF;
}
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\static\tableshow.css

```css
table {
    border-collapse: collapse;
}
td, th {
    border: black 2px solid;
    padding: 10px;
}
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\static\addEmployee.css

```css
p{
    font-size: 18px;
}
table{
    width: 100%;

}
form{

}
```

```css
table {
    border-collapse: collapse;
}

td, th {
    border: black 2px solid;
    padding: 10px;
}
```

```css
form{
    text-align: center;
}
label{
    font-size: 20px;
}
select{
    border: thin solid black;
    padding: 5px 12px;
    font-size: 15px;
}
option{
    font-size: 15px;
}
.container{
    position: relative;
    height: 200px;
    border: 3px solid white;
}
.button_center{
    margin: 0;
    position: absolute;
    top: 10%;
    left: 45%;
}
.newEmployee_button{
    text-align: center;
}
a{
    text-decoration: none;
}
```

```css
form{
    text-align: center;
}
label{
    font-size: 20px;
}
select{
    border: thin solid black;
    padding: 5px 12px;
    font-size: 15px;
}
option{
    font-size: 15px;
}
.container{
    position: relative;
    height: 200px;
    border: 3px solid white;
}
.button_center{
    margin: 0;
    position: absolute;
    top: 10%;
    left: 45%;
}
.select_button{
    text-align: center;
}
a{
    text-decoration: none;
}
```

```css
form{
    text-align: center;
}
label{
    font-size: 20px;
}
select{
    border: thin solid black;
    padding: 5px 12px;
    font-size: 15px;
}
.container{
    position: relative;
    height: 200px;
    border: 3px solid white;
}
.button_center{
    margin: 0;
    position: absolute;
    top: 10%;
    left: 45%;
}
.newEmployee_button{
    text-align: center;
}
a{
    text-decoration: none;
}
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\static\indexdisperTeam.css

```css
table {
    border-collapse: collapse;
}
td, th {
    border: black 2px solid;
    padding: 10px;
}
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\static\indextargsperTeam.css

```css
table {
    border-collapse: collapse;
}
td, th {
    border: black 2px solid;
    padding: 10px;
}
tbody{
    margin-left: 20%;
}
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\templates\403.html

```
 1 <!DOCTYPE html>
 2 <html lang="en">
 3 <head>
 4     <meta charset="UTF-8">
 5     <title>ERROR: FORBIDDEN</title>
 6 </head>
 7 <body>
 8 <p>You do not have permission to access this page!</p>
 9 <a href="/dash">Back to the home page!</a>
10 </body>
11 </html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/public.css">
<link rel="stylesheet" href="/static/Team.css">
<head>
    <meta charset="UTF-8">
    <title>Team Information</title>
</head>
<body>
<button type="button" STYLE="float:right;"><a href="/dash">Home</a></button>
<h1>Team Information</h1>
<table class="Team_table">
    <tr class="Team_tr">
        <td class="Team_td">Team ID: {{ team.0 }}</td>
        <td class="Team_td">Name: {{ team.1 }}</td>
        <td class="Team_td">Lead: {{ team.2 }}</td>
    </tr>

</table>
<table class="Team_table">
    <tr class="Team_tr">
        <td class="Team_link_td"><button><a href="updateTeam/{{ team.0 }}">Update Team</a></button></td>
        <td class="Team_link_td"><button><a href="/projectTeam/{{ team.0 }}">Update Team's Projects</a></button></td>
        <td class="Team_link_td"><button><a href="/employeeTeam/{{ team.0 }}">Update Team's Employees</a></button></td>
    </tr>
</table>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Error</title>
</head>
<body>
    <a class="homebutton" href="/dash" style="float: right">Home</a>

    <p>{{ message }}</p>
    <a href="{{ url }}">Try Again</a>
</body>
</html>
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\templates\index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login Page</title>
</head>
<body>
<form method="POST" action="{% url 'login' %}">
    {% csrf_token %}
  <div>
    <div>
      {{ form.username }}
    </div>
  </div>
  <div>
    <div>
      {{ form.password }}
    </div>
  </div>
  <button type="submit">Login</button>
</form>
</body>
</html>
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\templates\import.html

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/public.css">
<head>
    <meta charset="UTF-8">
    <title>File upload and display testing</title>
</head>
<body>
<button type="button" STYLE="float:right;"><a href="/dash">
Home</a></button>
<h1>File upload and display testing</h1>
<form method="POST" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Import File</button>
</form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/public.css">
<head>
    <meta charset="UTF-8">
    <title>Update Employee</title>
</head>

<body class="update_employee_body">
<button type="button" STYLE="float:right;"><a href="/dash">
Home</a></button>
<h1>Update Employee</h1>
    <table class="update_employee_table">
    {% if updateUser.errors %}
        <ol>
        {% for key, value in updateUser.errors.items %}
            <li>{{ value }}</li>
        {% endfor %}
        </ol>>
    {% endif %}
    <form method="POST" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <tr>
            <td><button type="submit">Update Employee</button
></td>
            <td><button type="reset">Reload Information</button
></td>
        </tr>

    </form>
    </table>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="static/public.css">

<head>
    <meta charset="UTF-8">
    <title>Add New Team</title>
</head>
<body>
    <button type="button" STYLE="float:right;"><a href="/dash">
    Home</a></button>
    <h1>Add New Team</h1>
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">SUBMIT</button>
    </form>
</body>
</html>
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\templates\Project.html

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" href="/static/public.css">
<link rel="stylesheet" href="/static/project.css">
<head>
    <meta charset="UTF-8">
    <title>Project Information</title>
</head>


<body class="project_body">
<button type="button" STYLE="float:right;"><a href="/dash">Home</a></button>
<h1>Project Information</h1>
    <table class="project_table">
        <tr class="project_tr">
            <td class="project_td">Title: {{ project.1 }}</td>
            <td class="project_td">Index Number: {{ project.0 }}</td>
        </tr>
    <tr class="project_tr">
        <td class="project_td">Operations Lead: {{ project.2 }}</td>
        <td class="project_td">Starts: {{ project.3 }}</td>
    </tr>
    <tr class="project_tr">
        <td class="project_td">Ends: {{ project.4 }}</td>
        <td class="project_td">Total Grant Funds: {{ project.5 }}</td>
    </tr>
        </table>
    <table class="project_table">
            <tr class="project_tr">
            <td class="project_Note">Note: {{ project.14 }}</td>
            </tr>
    </table>

<table class="project_table">
    <tr class="project_tr">
        <td class="project_link_td"><button><a href="updateProject/{{ project.0 }}">Update Project</a></button></td>
        <td class="project_link_td"><button><a href="/teamProject/{{ project.0 }}">Update Project's Teams</a></button></td>
        <td class="project_link_td"><button><a href="/employeeProject/{{ project.0 }}">Update Project's Employees</a></button></td>
```

```
39      </tr>
40 </table>
41
42 </body>
43 </html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="static/public.css">

<head>
    <meta charset="UTF-8">
    <title>Add New Payno</title>
</head>
<body>
    <button type="button" STYLE="float:right;"><a href="/dash">
   Home</a></button>
    <h1>Add New Payno</h1>
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        <table>
            {{ form.as_p }}
        </table>
        <button type="submit">SUBMIT</button>
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/public.css">
<link rel="stylesheet" href="/static/tableshow.css">
<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
</head>
<body>
    <button type="button"><a href="/table/AEPN">Back</a> </button>
    <button type="button" STYLE="float:right;"><a href="/dash"> Home</a></button>
    <h1>{{ title }}</h1>
    <table>
        <tr>
            {% for element in header %}
                <th>{{ element }}</th>
            {% endfor %}
        </tr>
        {% for indexNo, indexEnd, paynoEnd, encumbrance, color in body %}
            <tr style="background-color: {{ color }}">
                <th>{{ indexNo }}</th>
                <td>{{ indexEnd }}</td>
                <td>{{ paynoEnd }}</td>
                <td>{{ encumbrance }}</td>
            </tr>
        {% endfor %}
    </table>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/public.css">
<link rel="stylesheet" href="/static/tableshow.css">
<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
</head>
<body>
    <button><a href="/table/BUF">Back</a> </button>
    <button type="button" STYLE="float:right;"><a href="/dash">
 Home</a></button>
    <h1>{{ title }}</h1>
    <form method="POST" enctype="multipart/form-data" action="
{{ url }}">
        {% csrf_token %}
        <table>
            {% for row in context %}
                {% if forloop.counter == 1 %}
                    <tr>
                        <th></th>
                        <th>{{ row }}</th>
                    </tr>
                {% elif forloop.counter >= 2 %}
                <tr>
                    <td>{{ row.1 }}</td>
                    <td><input type="text" pattern="[0-9]*\.[0-
9][0-9]" placeholder="{{ row.2 }}" value="{{ row.2 }}" name="
{{ row.0 }}" id="{{ row.0 }}"></td>
                </tr>
                {% endif %}
            {% endfor %}
        </table>
        <button type="submit">Submit Changes</button>
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/Employee.css">
<link rel="stylesheet" href="/static/public.css">

<head>
    <meta charset="UTF-8">
    <title>Employee Information</title>
</head>
    <button type="button" STYLE="float:right;"><a href="/dash">
 Home</a></button>
    <h1>Employee information</h1>

<body class="employee_table_body">
    <table class="employee_table">
        <tr class="employee_tr">
            <td class="employee_td_Name">Name: {{ Employee.2
 }} {{ Employee.3 }}</td>
            <td class="employee_td">Position Number: {{
Employee.7 }}</td>
        </tr>
        <tr class="employee_tr">
            <td class="employee_td_title">< {{ Employee.6 }} >
</td>
            <td class="employee_td">Current Salary: {{ Employee
.10 }}</td>
        </tr>
        <tr class="employee_tr">
            <td class="employee_td">Telephone Number: {{
Employee.4 }}</td>
            <td class="employee_td">Last Salary Mod: {{
Employee.11 }}</td>
        </tr>
        <tr class="employee_tr">
            <td class="employee_td">Email: {{ Employee.5 }}</td
>
            <td class="employee_td">Salary Mod Date: {{
Employee.12 }}</td>

        </tr>
        <tr class="employee_tr">
            <td class="employee_td">Start Date: {{ Employee.9
 }}</td>
            <td class="employee_td">Cellphone Allowance: {{
Employee.13 }}</td>
        </tr>
        <tr class="employee_tr">
```

```
38              <td class="employee_td">Zorg: {{ Employee.8 }}</td>
39              <td class="employee_td">Benefits: {{ Employee.14 }}
   </td>
40          </tr>
41          <tr class="employee_tr">
42              <td class="employee_td">NetID: {{ Employee.1 }}</td
   >
43              <td class="employee_td">Team Pool: {{ Employee.17
    }}</td>
44          </tr>
45          <tr class="employee_tr">
46              <td></td>
47              <td class="employee_td">LOA: {{ Employee.18 }}</td>
48          </tr>
49          <tr class="employee_tr">
50              <td></td>
51              <td class="employee_td">FTE: {{ Employee.19 }}</td>
52          </tr>
53          <tr class="employee_tr">
54              <td></td>
55              <td class="employee_td">BiWeekly Salary: {{
   Employee.15 }}</td>
56          </tr>
57          <tr class="employee_tr">
58              <td></td>
59              <td class="employee_td">BiWeekly Benefits: {{
   Employee.16 }}</td>
60
61          </tr>
62          <tr class="employee_tr">
63              <td></td>
64              <td class="employee_td">GID: {{ Employee.0 }}</td>
65          </tr>
66      </table>
67      <table class="employee_table">
68          <tr class="employee_tr">
69              <td><button><a href="updateEmployee/{{ Employee.0
    }}">Update Employee</a></button></td>
70              <td><button><a href="/teamEmployee/{{ Employee.0 }}
   ">Update Employee's Teams</a></button></td>
71              <td><button><a href="/projectEmployee/{{ Employee.0
    }}">Update Employee's Projects</a></button></td>
72          </tr>
73      </table>
74 </body>
75 </html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="static/public.css">
<link rel="stylesheet" href="static/dashboard.css">
<head>
    <meta charset="UTF-8">
    <title>Capacity Plan Dashboard</title>
</head>
<body class="dashboard_body">
<h4>Welcome back, {{ user }}</h4>
<button type="submit" style="float:right"><a href="/logout">logout</a></button>
<h1>Capacity Plan Dashboard</h1>
<table class="dashboard_table">
    <tr>
        <td class="dashboard_td"><a class="link" href="selectEmployee">Employee Pay Information</a></td>
        <td class="dashboard_td"><a class="link" href="selectProject">Project Information</a></td>
    </tr>
    <tr>
        <td class="dashboard_td"><a class="link" href="table/ITT">Index Targets Per Team</a></td>
        <td class="dashboard_td"><a class="link" href="table/IDT">Index Distributions Per Team</a></td>
    </tr>
    <tr>
        <td class="dashboard_td"><a class="link" href="table/SBT">Sal/Ben Budget Remaining</a></td>
        <td class="dashboard_td"><a class="link" href="import">Import Reports</a></td>
    </tr>
    <tr>
        <td class="dashboard_td"><a class="link" href="table/DPNE">Distribution Across Paynos -1 Employee</a></td>
        <td class="dashboard_td"><a class="link" href="table/DPNI">Distribution Across Paynos -1 Index</a></td>
    </tr>
    <tr>
        <td class="dashboard_td"><a class="link" href="table/AEPN">All Encumbrances -1 Payno</a></td>
        <td class="dashboard_td"><a class="link" href="table/EQ">EPAF Queries</a></td>
    </tr>
    <tr>
        <td class="dashboard_td"><a class="link" href="table/BUF">Benefits Update Form</a></td>
```

```
37          <td class="dashboard_td"><a class="link" href="
   selectTeam">Team Information</a></td>
38      </tr>
39 </table>
40 </body>
41 </html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/public.css">
<head>
    <meta charset="UTF-8">
    <title>Add {{ form.1 }} to {{ form.2 }}</title>
</head>
<body>
<button type="button" STYLE="float:right;"><a href="/dash">
Home</a></button>
<h1>Add {{ form.1 }} to {{ form.2 }}</h1>
<h2>{{ form.1 }}:</h2>
<form enctype="multipart/form-data" action="{{ form.0 }}"
method="POST">
    {% csrf_token %}
    {% for row in form %}
        {% if forloop.counter > 3 %}
            <input type="checkbox" id="{{ row.0 }}" name="{{
row.0 }}" value="{{ row.1 }}" {% if row.2 == 1 %} checked {%
endif %}>
            <label for="{{ row.0 }}">{{ row.0 }}: {{ row.1 }}</
label><br>
        {% endif %}
    {% endfor %}
    <button type="submit">Submit Changes</button>
</form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="static/public.css">

<head>
    <meta charset="UTF-8">
    <title>Add New Project</title>
</head>
<body>
    <button type="button" STYLE="float:right;"><a href="/dash">
   Home</a></button>
    <h1>Add New Project</h1>
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">SUBMIT</button>
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="static/public.css">
<link rel="stylesheet" href="static/selectproject.css">
<head>
    <meta charset="UTF-8">
    <title>Team Selection</title>
</head>
<body>

<form method="POST">
    <button type="button" STYLE="float:right;"><a href="/dash">
 Home</a></button>
    <h1>Select Team to View Information</h1>
    {% csrf_token %}
    {{ form }}
    <button type="submit">Select</button>
</form>
<div class="container">
    <div class="button_center">
    <button class="select_button"><a href="/addTeam">Add New
Team</a>
    </div>
</div>
</body>
</html>
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\templates\updateTeam.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Update Team</title>
</head>
<body>
<button type="button" STYLE="float:right;"><a href="/dash">
Home</a></button>
    <form method="POST" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Update Team</button>
        <button type="reset">Reload Information</button>
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" href="/static/addEmployee.css">
<link rel="stylesheet" href="/static/public.css">

<head>
    <meta charset="UTF-8">
    <title>Add New Employee</title>
</head>
<body>
    <button type="button" STYLE="float:right;"><a href="/dash">
   Home</a></button>
    <h1>Add New Employee</h1>

    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        <table>
            {{ form.as_p }}
        </table>

        <button type="submit">SUBMIT</button>
    </form>

</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/public.css">
<link rel="stylesheet" href="/static/tableshow.css">
<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
    <script src="{% static 'functions.js' %}"></script>
</head>
<body>
    <button type="button"><a href="/table/DPNE">Back</a> </button>
    <button type="button" STYLE="float:right;"><a href="/dash"> Home</a></button>
    <h1>{{ title }}</h1>
    <form enctype="multipart/form-data" method="POST" action="{{ url }}">
        {% csrf_token %}
        <table>
            {% for row in header %}
                <tr>
                {% for element in row %}
                    <th>{{ element }}</th>
                {% endfor %}
                </tr>
            {%  endfor %}
            {% for row in body %}
                <tr>
                    {% for element in row %}
                        {% if forloop.counter < 3 %}
                            <td>{{ element }}</td>
                        {% elif element == 'x' %}
                            <td></td>
                            <td></td>
                        {% else %}
                            <td><input oninput="reTotal(this.id)" type="text" pattern="[0-9]?[0-9]" placeholder="{{ element.0.0 }}" value="{{ element.0.0 }}" name="{{ element.0.1 }}" id="{{ element.0.2 }}">%</td>
                            <td style="background-color: {{ element.1.1 }}">{{ element.1.0 }}</td>
                        {% endif %}
                    {% endfor %}
                </tr>
            {% endfor %}
            <tr>
                <td></td>
```

```
                {% for element in tail %}
                    {% if forloop.counter == 1 %}
                        <th>{{ element }}</th>
                    {% else %}
                        <td id="{{ element.1 }}" style="
background-color: {{ element.2 }}">{{ element.0 }}%</td>
                        <td></td>
                    {% endif %}
                {% endfor %}
            </tr>
        </table>
        <button type="submit" onclick="return confirm('Are you
sure you want to make changes?\nPress \'Ok\' to confirm.')">
Submit Changes</button>
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/public.css">
<link rel="stylesheet" href="/static/tableshow.css">
<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
</head>
<body>
    <a href="/table/EQ">Back</a>
    <button type="button" STYLE="float:right;"><a href="/dash">
 Home</a></button>
    <h1>{{ title }}</h1>
    <table>
        {% for row in context %}
            {% if forloop.counter == 1 %}
                <tr>
                    {% for element in row %}
                        <th>{{ element }}</th>
                    {% endfor %}
                </tr>
            {% else %}
            <tr>
                {% for element in row %}
                    {% if forloop.counter <= 4 %}
                        <th>{{ element }}</th>
                    {% else %}
                        {% if element.2 == 'c' %}
                            <td style="background-color: yellow
">{{ element.0 }}%</td>
                            <td style="background-color: yellow
">{{ element.1 }}%</td>
                        {% else %}
                            <td>{{ element.0 }}%</td>
                            <td>{{ element.1 }}%</td>
                        {% endif %}
                    {% endif %}
                {% endfor %}
            </tr>
            {% endif %}
        {% endfor %}
    </table>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" href="/static/employeeForm.css">
<link rel="stylesheet" href="/static/public.css">
<head>
    <meta charset="UTF-8">
    <title>Employee Selection</title>
</head>
<body>
<button type="button" STYLE="float:right;"><a href="/dash">Home</a></button>
<h1>Select Employee to View Information</h1>
<form method="POST">
    {% csrf_token %}
    {{ form }}
    <button type="submit">Select</button>
</form>
<div class="container">
    <div class="button_center">
        <button class="newEmployee_button"><a href="/addEmployee">Add New Employee</a></button>
    </div>
</div>


</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/public.css">
<link rel="stylesheet" href="/static/tableshow.css">
<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
    <script src="{% static 'functions.js' %}"></script>
</head>
<body>
    <button> <a href="/table/DPNI">Back</a></button>
    <button type="button" STYLE="float:right;"><a href="/dash" style="float: right"> Home</a></button>
    <h1>{{ title }}</h1>
    <form enctype="multipart/form-data" method="POST" action="{{ url }}">
        {% csrf_token %}
        {% for employee, wage in wages %}
            <input type="hidden" id="{{ employee }}" value="{{ wage }}">
        {% endfor %}
        {% for base, id in base_totals %}
            <input type="hidden" id="{{ id }}" value="{{ base }}">
        {% endfor %}
        <table>
            {% for row in header %}
                <tr>
                    {% for element, color in row %}
                    <th style="background-color: {{ color }}">{{ element }}</th>
                    {% endfor %}
                </tr>
            {% endfor %}
            {% for row in body %}
                <tr>
                    {% for element in row %}
                        {% if forloop.counter == 1 %}
                            <th>{{ element }}</th>
                        {% elif forloop.counter > 1 and not forloop.counter|divisibleby:2 %}
                            <td id="{{ element.1 }}" style="background-color: {{ element.2 }}">{{ element.0 }}%</td>
                        {% else %}
                            <td><input oninput="dispaynoChange(this.id)" type="text" pattern="[0-9]?[0-9]" placeholder="{{ element.0 }}" value="{{ element.0 }}" name="{{ element.1 }}" id
```

```
39 ="{{ element.2 }}">%</td>
40                             {% endif %}
41                         {% endfor %}
42                     </tr>
43                 {% endfor %}
44                 {% for row in tail %}
45                     <tr>
46                     {% for element in row %}
47                         {% if forloop.counter == 1 %}
48                             <th>{{ element }}</th>
49                         {% else %}
50                             <td id="{{ element.1 }}">{{ element.0
    }}</td>
51                         {% endif %}
52                     {% endfor %}
53                     </tr>
54                 {% endfor %}
55             </table>
56             <button type="submit" onclick="return confirm('Are you
   sure you want to make changes?\nPress \'Ok\' to confirm.')">
   Submit Changes</button>
57         </form>
58 </body>
59 </html>
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\templates\selectionForm.html

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="static/public.css">
<head>
    <meta charset="UTF-8">
    <title>Select</title>
</head>
<body>
<button type="button" STYLE="float:right;"><a href="/dash">Home</a></button>
<h1>Placeholder for the index target/EPAF/Project selection</h1>
<form method="POST" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form }}
    <button type="submit">Select</button>
</form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="static/public.css">
<link rel="stylesheet" href="static/selectproject.css">
<head>
    <meta charset="UTF-8">
    <title>Project Selection</title>
</head>


<body class="selectproject_page">
<form method="POST">
    <button type="button" STYLE="float:right;"><a href="/dash">Home</a></button>
    <h1>Select Project to View Information</h1>
    {% csrf_token %}
    {{ form }}
    <button type="submit">Select</button>
</form>

<div class="container">
    <div class="button_center">
        <button class="select_button"><a href="/addProject">Add New Project</a></button>
    </div>
</div>

</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" href="/static/public.css">
<head>
    <meta charset="UTF-8">
    <title>Update Project</title>
</head>
<body>
<button type="button" STYLE="float:right;"><a href="/dash">Home</a></button>
    <form method="POST" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Update Project</button>
        <button type="reset">Reload Information</button>
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/public.css">
<link rel="stylesheet" href="/static/EpafQueries.css">
<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
</head>
<body>
<button type="button" STYLE="float:right;"><a href="/dash">
Home</a></button>
<h1>{{ title }}</h1>
<form method="POST" enctype="multipart/form-data">
    {% csrf_token %}
    <table>
        {{ form }}
    </table>

<div class="container">
    <div class="button_center">
        <button type="submit">Select</button>
    </div>
</div>
</form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/tableshow.css">
<link rel="stylesheet" href="/static/public.css">

<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
    <script src="{% static 'functions.js' %}"></script>
</head>
<body>
    <button type="button"><a href="/table/IDT">Back</a></button>
    <button type="button" STYLE="float:right;"><a href="/dash">Home</a></button>
    <h1>{{ title }}</h1>
    <form enctype="multipart/form-data" method="POST" action="{{ url }}">
        {% csrf_token %}
        {% for id, base in base_totals %}
            <input type="hidden" id="{{ id }}" value="{{ base }}">
        {% endfor %}
        <table>
            {% for row in context %}
                {% if forloop.counter == 1 %}
                    <tr>
                        {% for element in row %}
                            <th>{{ element }}</th>
                        {% endfor %}
                    </tr>
                {% elif row.0 != 'Total' and forloop.counter > 1 %}
                    <tr>
                        {% for element in row %}
                            {% if element.0 == 't' %}
                                <th style="background-color: {{ element.2 }}">{{ element.1 }}</th>
                            {% elif element.0 == 'p' %}
                                <td><input oninput="reTotal(this.id)" type="text" pattern="[0-9]?[0-9]" placeholder="{{ element.1 }}" value="{{ element.1 }}" name="{{ element.2 }}" id="{{ element.3 }}">%</td>
                            {% else %}
                                <th>{{ element }}</th>
                            {% endif %}
                        {% endfor %}
```

```
                    </tr>
                {% elif forloop.counter > 1 %}
                    <tr>
                    {% for element in row %}
                        {% if forloop.counter < 3 %}
                            <th>{{ element }}</th>
                        {% else %}
                            <td id="{{ element.1 }}" style="
background-color: {{ element.2 }}">{{ element.0}}%</td>
                        {% endif %}
                    {% endfor %}
                    <td></td>
                    <td></td>
                    <td></td>
                </tr>
                {% endif %}
            {% endfor %}
        </table>
        <button type="submit" onclick="return confirm('Are you
sure you want to make changes?\nPress \'Ok\' to confirm.')">
Submit Changes</button>
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/tableshow.css">
<link rel="stylesheet" href="/static/public.css">
<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
</head>
<body>
    <button type="button"><a href="/table/SBT">Back</a> </button>
    <button type="button" STYLE="float:right;"><a href="/dash"> Home</a></button>
    <h1>{{ title }}</h1>
    <table>
        <tr>
            {% for element in header %}
                <th>{{ element }}</th>
            {% endfor %}
        </tr>
        {% for index, indexEnd, remaining, color in body %}
            <tr style="background-color: {{ color }}">
                <th>{{ index }}</th>
                <td>{{ indexEnd }}</td>
                <td>{{ remaining }}</td>
            </tr>
        {% endfor %}
    </table>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<link rel="stylesheet" href="/static/tableshow.css">
<link rel="stylesheet" href="/static/public.css">

<head>
    <meta charset="UTF-8">
    <title>{{ title }}</title>
</head>
<body>
    <button type="button"><a href="/table/ITT">Back</a></button
>
    <button type="button" STYLE="float:right;"><a href="/dash"
style="float: right"> Home</a></button>
    <h1>{{ title }}</h1>
    {% for f in form %}
        <form method="POST" action="{{ f.0 }}" id="{{ f.1 }}">
{% csrf_token %}</form>
    {% endfor %}
    <form enctype="multipart/form-data" method="POST" action="
{{ url }}">
        {% csrf_token %}
        <table>
            {% for row in header %}
                <tr>
                    {% if row.0 == 'POP Date'%}
                        {% for element in row %}
                        <th>{{ element }}</th>
                        {% endfor %}
                    {% else %}
                        <th></th>
                        {% for element in row %}
                            <th>{{ element }}</th>
                        {% endfor %}
                    {% endif %}
                </tr>
            {% endfor %}
            {% for row in body%}
                <tr>
                    {% for element in row %}
                        {% if forloop.counter < 3 %}
                            <th>{{ element }}</th>
                        {% else %}
                            {% if element.1 == 'im' %}
                                <td style="background-color
: {{ element.2 }}">{{ element.0 }}</td>
                            {% elif element == 'x' %}
```

```
                                    <td></td>
                                {% elif element.1 == 'f' %}
                                    <td>
                                        <input type="submit" value
="Recalculate Initial Targets" form="{{ element.0 }}"
                                                onclick="return
confirm('Please save all other changes to this form before ' +
                                                 'recalculating. \n
ALL OTHER CHANGES WILL BE LOST IF NOT ' +
                                                  'SAVED FIRST.\n
Press \'Ok\' to recalculate.')">
                                    </td>
                                {% else %}
                                    <td style="background-color
: {{ element.2 }}"><input type="text"  pattern="-?[0-9]*\.[0-9
][0-9]" placeholder="{{ element.0 }}" value="{{ element.0 }}"
name="{{ element.1 }}" id="{{ element.1 }}"></td>
                                {% endif %}
                            {% endif %}
                        {% endfor %}
                    </tr>
                {% endfor %}
                <tr>
                    {% for element in tail.0 %}
                        {% if forloop.counter < 3 %}
                            <th>{{ element }}</th>
                        {% else %}
                            <td>{{ element }}</td>
                        {% endif %}
                    {% endfor %}
                </tr>
            </table>
            <button type="submit" onclick="return confirm('Are you
sure you want to make changes?\nPress \'Ok\' to confirm.')">
Submit Changes</button>
        </form>
</body>
</html>
```

File - C:\Users\yywwc\PycharmProjects\miltech\main\templates\registration\login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login Page</title>
</head>
<body>
    <h4>Log in to your account</h4>
    <form method="post">
      {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

```
from main.models import *


def contains27(start, end): # checks to see is pn27 is in the range of pns
    starting = PayNo.objects.get(Year=start).Start
    ending = PayNo.objects.get(Year=end).End
    range = PayNo.objects.all().filter(End__range=[starting, ending]) # get paynos from start to end
    if end[:2] == "26": # removes 27 bc it is grabbed automatically with 26
        range = list(range)
        range.pop()
    contains = False  # false until proven true
    for pn in range:
        if pn.Year[:2] == "27":  # checks paynumber
            contains = True  # range contains pn 27
            break
    return contains

def add_dot(integer): # Add the monetary dot to cent value
    string = str(integer)
    if(len(string) == 1):
        return '0.0' + string
    elif(len(string) == 2): # either single cent value with minus or double cent value
        if('-' in string):
            string = string.replace('-', '') # get rid of minus
            return '-0.0'+string # add minus back
        else:
            return '0.'+string
    else:
        return string[:len(string) - 2] + '.' + string[len(string) - 2:]

def total_remaing(indexNo):
    project = Project.objects.get(IndexNo=indexNo)
    paynos_from_filter = list(PayNo.objects.filter(End__gte=project.StartDate, Start__lte=project.EndDate).order_by('Start'))
    paynos = paynos_from_filter
    count = 0
    for pn in paynos_from_filter: # removes all payno 27 since those dont contribute to spending
        if pn.Year[:2] == "27":
            paynos.pop(count)
        count += 1
    total = project.SalaryBenefitBudget # get the initial
```

```
40 budget
41     for pn in paynos:
42         proj_pn = Project_PayNo.objects.get_or_create(IndexNo=
   indexNo, Year=pn.Year)[0]
43         total -= (proj_pn.BenCat + proj_pn.SalCat) # remove the
    actuals from the initial budget
44     return round(total)  # round to whole cent value
45 
46 def final_target(project, payno):
47     f_target = 0
48     for e in Project_Employee.objects.filter(IndexNo=project):
   # for each emp on the proj
49         employee = Employee.objects.get(GID=e.GID)
50         percent = Employee_Project_PayNo.objects.get_or_create(
   IndexNo=project, Year=payno, GID=e.GID)[0].Percent
51         f_target += (employee.BiweeklySalary + employee.
   BiweeklyBenefits) * (percent / 100)  # % * monthly sal and ben
52     return round(f_target) # round to whole cent value
53 
54 def startLTEend(start, end): # checks that the starting pn is
   b4 the ending pn
55     s = PayNo.objects.get(Year=start)
56     e = PayNo.objects.get(Year=end)
57     return s.Start <= e.Start
58 
```

```python
"""miltech URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
from django.contrib.auth import views as auth_views

from miltech import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', auth_views.LoginView.as_view(next_page='/dash'), name="login"),
    path('logout/', auth_views.LogoutView.as_view(next_page='/')),
    path('import', views.importFile),
    path('dash', views.index),
    path('addPayno', views.addPayno),
    path('selectEmployee', views.selectEmployee),
    path('addEmployee', views.addEmployee),
    path('employee/updateEmployee/<str:gid>', views.updateEmployee),
    path('employee/<str:gid>', views.viewEmployee),
    path('selectProject', views.selectProject),
    path('addProject', views.addProject),
    path('project/updateProject/<str:id>', views.updateProject),
    path('project/<str:id>', views.viewProject),
    path('selectTeam', views.selectTeam),
    path('addTeam', views.addTeam),
    path('team/updateTeam/<str:id>', views.updateTeam),
```

```
40      path('team/<str:id>', views.viewTeam),
41      path('table/<str:type>', views.selectTable),
42      path('teamProject/<str:id>', views.updateTeamtoProj),
43      path('projectTeam/<str:id>', views.updateProjToTeam),
44      path('teamEmployee/<str:id>', views.updateTeamtoEmp),
45      path('employeeTeam/<str:id>', views.updateEmptoTeam),
46      path('employeeProject/<str:id>', views.updateEmptoProj),
47      path('projectEmployee/<str:id>', views.updateProjtoEmp),
48      path('benefitUpdate/<str:kind>/<str:employee>/<str:payno>'
    , views.benefitsUpdate),
49      path('EPAFquerie/<str:kind>/<str:project>/<str:team>/<str:
    employee>/<str:prior>/<str:current>', views.EPAFquerie),
50      path('indexTargetsPerTeam/<str:id>/<str:start>/<str:end>',
    views.indexTargetsPerTeam),
51      path('indexDisPerTeam/<str:id>/<str:payno>', views.
    indexDisPerTeam),
52      path('disPayNoEmployee/<str:id>/<str:start>/<str:end>',
    views.disPaynoEmployee),
53      path('disPayNoIndex/<str:id>/<str:start>/<str:end>', views.
    disPaynoIndex),
54      path('salBenTotal/<str:kind>/<str:project>/<str:team>',
    views.salBenTotalRemaining),
55      path('allEncumbrances/<str:kind>/<str:payno>/<str:project
    >/<str:team>', views.allEcumbrances),
56      path('recalcIndexTargets/<str:index>/<str:id>/<str:start>/<
    str:end>', views.recalculateIndexTargets),
57      #path(r'^keycloak/', include('django_keycloak.urls'))
58  ]
59
```

```
 1 import datetime
 2
 3 from django.contrib.auth.decorators import permission_required
   , login_required
 4 from django.http import HttpResponseRedirect
 5 from django.shortcuts import render, redirect
 6 from django.contrib import messages
 7 from datetime import date
 8 from datetime import timedelta
 9
10 from main.models import *
11 from main.forms import *
12
13
14 from main.file_import import *
15 from miltech.help import *
16
17 yellow = 'gold'
18 green = 'lightgreen'
19 red = 'lightsalmon'
20
21 login_url = "/"
22 dashboard_url = "/dash"
23
24 @login_required(login_url=login_url)
25 def index(request):
26     return render(request, 'dashboard.html')
27
28 @login_required(login_url=login_url)
29 @permission_required('Employee.FINANCE_LEAD', 'Employee.
   OPERATIONS_LEAD', 'Employee.OPERATIONS_SUPPORT')
30 def importFile(request):
31     if request.method == 'POST':
32         form = ImportForm(request.POST)
33         choice = form['fileType'].value()
34         i = Import()
35         if choice == '1':
36             i.importCatbooks(request.FILES['file'])
37         elif choice == '2':
38             i.importPaynos(request.FILES['file'])
39         return redirect(dashboard_url)
40     else:
41         form = ImportForm()
42         return render(request, 'import.html', {'form': form})
43
44 @login_required(login_url=login_url)
45 @permission_required('Employee.FINANCE_LEAD', 'Employee.
   OPERATIONS_LEAD', 'Employee.OPERATIONS_SUPPORT')
```

```
46 def recalculateIndexTargets(request, index, id, start, end):
47     project = Project.objects.get(IndexNo=index) # gets all
   paynos from today to pop date
48     payno_range = list(PayNo.objects.filter(End__gte=date.today
   (), Start__lte=project.EndDate).order_by('Start'))
49     paynos = payno_range
50     count = 0
51     for pn in payno_range:  # removes all payno 27 since those
   dont contribute to spending
52         if pn.Year[:2] == "27":
53             paynos.pop(count)
54         count += 1
55     balanced_index_targ = round(total_remaing(index)/len(paynos
   )) # averages out spending
56     for p in paynos: # input new index target
57         object = Project_PayNo.objects.get_or_create(IndexNo=
   index, Year=p.Year)[0]
58         object.SalBenPlan = balanced_index_targ
59         object.save()
60     return redirect('/indexTargetsPerTeam/' + id + '/' + start
    + '/' + end)
61
62 @login_required(login_url=login_url)
63 @permission_required('Employee.FINANCE_LEAD', 'Employee.
   OPERATIONS_LEAD', 'Employee.OPERATIONS_SUPPORT')
64 def indexTargetsPerTeam(request, id, start, end):  # TODO: add
   displaying subaccounts, encumbrances only
65     if request.method == 'POST':
66         values = list(request.POST.items())
67         values.pop(0)  # get rid of csrf token
68         for input in values:  # (id, value)
69             ids = input[0].split(" ")
70             if len(ids) > 2: # deal with subaccount ids having
   spaces
71                 subaccount = ""
72                 for s in ids[1:-1]:
73                     subaccount += s + " "
74                 subaccount += ids[-1]
75                 ids[1] = subaccount
76             print(ids)
77             proj_payno = Project_PayNo.objects.get(IndexNo=ids[
   1], Year=ids[0])
78             proj_payno.SalBenPlan = int(input[1].replace('.','')
   )
79             proj_payno.save()
80         return redirect('/indexTargetsPerTeam/' + id + '/' +
   start + '/' + end)
81     else:
```

```
 82         url = '/indexTargetsPerTeam/' + id + '/' + start + '/'
    + end
 83         title = "Index Targets for " + id + " from " + start
    + " to " + end
 84         header, body, form, tail = [], [], [], []
 85         form_count = 0
 86         starting = PayNo.objects.get(Year=start).Start
 87         ending = PayNo.objects.get(Year=end).End
 88         if id == 'all': # get all projects in range of pns
 89             if not contains27(start, end): # get rid of all
    projects out of range of pns
 90                 projects = Project.objects.all().order_by('
    IndexNo').exclude(Note="Subaccount").filter(EndDate__gte=
    starting, StartDate__lte=ending)
 91             else: # get all regardless of whether or not they'
    re in range
 92                 projects = Project.objects.all().order_by('
    IndexNo').exclude(Note="Subaccount").filter(StartDate__lte=
    ending)
 93             projs = list(projects)
 94             count = 1
 95             for p in projects: # add in subaccounts after main
     project
 96                 subaccounts = Project.objects.filter(
    IndexNo__contains=p.IndexNo, Note="Subaccount")
 97                 for subaccount in subaccounts:
 98                     projs.insert(count, subaccount)
 99                     count += 1
100                 count += 1
101             projects = projs
102         else: # get projects for team
103             projects = Project_Team.objects.filter(TeamId=id).
    order_by('IndexNo')
104             projs = list(projects)
105             count = 1
106             for p in projects:
107                 project = Project.objects.get(IndexNo=p.
    IndexNo)
108                 if ending < project.StartDate and starting >
    project.EndDate and not contains27(start, end):
109                     projs.pop(count-1)
110                 else: # add in subaccounts after main project
111                     subaccounts = Project.objects.filter(
    IndexNo__contains=p.IndexNo, Note="Subaccount")
112                     for subaccount in subaccounts:
113                         projs.insert(count, subaccount)
114                         count += 1
115                     count += 1
```

```
116             projects = projs
117         if len(projects) == 0: # no projects to display values
     for
118             return render(request, 'error.html',
119                           {'message': "There are no projects
    to display for the pay numbers you requested",
120                            'url': "/table/ITT"})
121         paynos = PayNo.objects.filter(End__range=[starting,
    ending]).order_by('Start')
122         if end[:2] == "26": # removes 27 bc it is grabbed
    automatically with 26
123             range = list(paynos)
124             range.pop()
125             paynos = range
126         row = ['Pay Number'] # headers
127         row2 = ['Payno Start']
128         row3 = ['Payno End']
129         row4 = ['Pay Date']
130         row5 = ['POP Date', 'Index']
131         total = ['','Total for Team']
132         for p in paynos: # creates header and tail rows
133             row.append(p.Year)
134             row2.append(p.Start)
135             row3.append(p.End)
136             row4.append(p.Date)
137             if p.Year[:2] == "27": # bulk encumbrances
138                 row5.append('Bulk Encumbrances')
139                 total.append('0.00')
140             elif p.End < date.today(): # paynos ending before
    today
141                 row5.append('Actual')
142                 tot = 0
143                 for t in projects:
144                     proj_pn = Project_PayNo.objects.
    get_or_create(Year=p.Year, IndexNo=t.IndexNo)[0]
145                     tot += proj_pn.SalCat + proj_pn.BenCat
146                 total.append(add_dot(tot))
147             elif p.Start <= date.today() <= p.End: # current
    payno
148                 row.append(p.Year) # needed to add a second
    col of paynos to accomidate index tars
149                 row2.append(p.Start)
150                 row3.append(p.End)
151                 row4.append(p.Date)
152                 row5.append('Initial Target')
153                 tot = 0
154                 for t in projects:
155                     proj_pn = Project_PayNo.objects.
```

```
155 get_or_create(Year=p.Year, IndexNo=t.IndexNo)[0]
156                     tot += proj_pn.SalBenPlan
157                 total.append(add_dot(tot))
158                 row5.append('Final Target')
159                 tot = 0
160                 for t in projects:
161                     tot += final_target(t.IndexNo, p.Year)
162                 total.append(add_dot(tot))
163             else: # future paynos and pn27
164                 row5.append('Initial Target')
165                 tot = 0
166                 for t in projects:
167                     proj_pn = Project_PayNo.objects.
    get_or_create(Year=p.Year, IndexNo=t.IndexNo)[0]
168                     tot += proj_pn.SalBenPlan
169                 total.append(add_dot(tot))
170         tail.append(total)
171         header.append(row); header.append(row2); header.append
    (row3); header.append(row4); header.append(row5)
172         for t in projects: # construct body of table
173             pj = Project.objects.get(IndexNo=t.IndexNo)
174             new = [pj.EndDate, pj.IndexNo] # start of row
175             for p in paynos:
176                 if (p.Start > pj.EndDate or p.End < pj.
    StartDate) and p.Year[:2] != "27": # ends before or starts
    after range of pns
177                     new.append('x') # display nothing
178                 else:
179                     if p.Year[:2] == "27": # bulk encumbrances
180                         proj_pn = Project_PayNo.objects.
    get_or_create(Year=p.Year, IndexNo=pj.IndexNo)[0]
181                         new.append([add_dot(proj_pn.SalBenPlan
    ), 'im', 'white'])
182                     elif p.End < date.today(): # past pns
183                         proj_pn = Project_PayNo.objects.
    get_or_create(Year=p.Year, IndexNo=pj.IndexNo)[0]
184                         if p.Start <= pj.EndDate <= p.End: #
    last pn for proj
185                             new.append([add_dot(proj_pn.SalCat
     + proj_pn.BenCat), 'im', red])
186                         elif p.Start <= pj.StartDate <= p.End
    : # first pn for proj
187                             new.append([add_dot(proj_pn.SalCat
     + proj_pn.BenCat), 'im', green])
188                         else:
189                             new.append([add_dot(proj_pn.SalCat
     + proj_pn.BenCat), 'im', 'white'])
190                     elif p.Start <= date.today() <= p.End and
```

```
p.Year[:2] != "27": # current pn
                        if p.Start <= pj.EndDate <= p.End:
                            new.append([add_dot(Project_PayNo.objects.get_or_create(Year=p.Year, IndexNo=pj.IndexNo)[0].SalBenPlan), p.Year+" "+pj.IndexNo, red])
                            new.append([add_dot(final_target(t.IndexNo, p.Year)), 'im', red])
                        elif p.Start <= pj.StartDate <= p.End:
                            new.append([add_dot(Project_PayNo.objects.get_or_create(Year=p.Year, IndexNo=pj.IndexNo)[0].SalBenPlan),p.Year + " " + pj.IndexNo, green])
                            new.append([add_dot(final_target(t.IndexNo, p.Year)), 'im', green])
                        else:
                            new.append([add_dot(Project_PayNo.objects.get_or_create(Year=p.Year, IndexNo=pj.IndexNo)[0].SalBenPlan), p.Year+" "+pj.IndexNo, 'white'])
                            new.append([add_dot(final_target(t.IndexNo, p.Year)), 'im', 'white'])
                    else: # future pns and pn27
                        if p.Start <= pj.EndDate <= p.End:
                            new.append([add_dot(Project_PayNo.objects.get_or_create(Year=p.Year, IndexNo=pj.IndexNo)[0].SalBenPlan), p.Year+" "+pj.IndexNo, red])
                        elif p.Start <= pj.StartDate <= p.End:
                            new.append([add_dot(Project_PayNo.objects.get_or_create(Year=p.Year, IndexNo=pj.IndexNo)[0].SalBenPlan),p.Year + " " + pj.IndexNo, green])
                        else:
                            new.append([add_dot(Project_PayNo.objects.get_or_create(Year=p.Year, IndexNo=pj.IndexNo)[0].SalBenPlan), p.Year+" "+pj.IndexNo, 'white'])
            form_count += 1
            new.append(["form"+str(form_count), 'f']) # recalculating index targs form
            form.append(["/recalcIndexTargets/"+t.IndexNo+"/"+id+"/"+start+"/"+end, "form"+str(form_count)])
            body.append(new)
        return render(request, 'indextargsperTeam.html', {'url': url, 'title': title, 'header': header, 'body': body, 'form': form, 'tail': tail})

@login_required(login_url=login_url)
@permission_required('Employee.FINANCE_LEAD', 'Employee.OPERATIONS_LEAD')
def indexDisPerTeam(request, id, payno):
    if request.method == 'POST':
        values = list(request.POST.items())
```

```
218         values.pop(0)  # get rid of csrf token
219         for input in values:  # (id, value)
220             ids = input[0].split(" ")
221             percent = Employee_Project_PayNo.objects.get(Year=
    payno, GID=ids[0], IndexNo=ids[1])
222             percent.Percent = int(input[1])
223             percent.save()
224         return redirect('/indexDisPerTeam/'+id+'/'+payno)
225     else:
226         url = '/indexDisPerTeam/'+id+'/'+payno
227         title = "Index Distributions for " + id + " for " +
    payno
228         context, base_totals, team_projects = [], [], []
229         pn = PayNo.objects.get(Year=payno)
230         if (id == 'all'): # get all projs except subaccounts
    and all employees
231             team_projects = Project.objects.all().order_by('
    IndexNo').exclude(Note="Subaccount").filter(EndDate__gte=pn.
    Start, StartDate__lte=pn.End)
232             team_members = Employee.objects.all().order_by('
    FirstName', 'LastName')
233         else: # get projects and employees for team
234             pt = Project_Team.objects.filter(TeamId=id)
235             for p in pt:
236                 proj = Project.objects.get(IndexNo=p.IndexNo)
237                 if pn.End >= proj.StartDate and pn.Start <=
    proj.EndDate:
238                     team_projects.append(Project.objects.get(
    IndexNo=p.IndexNo))
239             team_members = Team_Employee.objects.filter(TeamId
    =id)
240         if len(team_projects) == 0: # no projects throw error
    page
241             return render(request, 'error.html',
242                           {'message': "There are no projects
    to display for the pay number you requested",
243                            'url': "/table/IDT"})
244         row = ['Index', 'POP Date'] # header
245         employees = [] # also
246         total = ['Total', ''] # bottom row
247         for t in team_members: # calculate total %s for each
    employee
248             employees.append(Employee.objects.get(GID=t.GID))
249             row.append(Employee.objects.get(GID=t.GID).__str__
    ())
250             tot, base = 0, 0
251             employee_projects = Project_Employee.objects.
    filter(GID=t.GID)
```

```
252             for p in employee_projects: # now doing all
    projects for that employee not just the team
253                 proj = Project.objects.get(IndexNo=p.IndexNo)
254                 if pn.End >= proj.StartDate and pn.Start <=
    proj.EndDate:
255                     if id != 'all':
256                         if (Project_Team.objects.filter(TeamId
    =id, IndexNo=p.IndexNo).exists() == False): # get the % not
    included in the team projects
257                             base += Employee_Project_PayNo.
    objects.get_or_create(Year=payno, GID=t.GID, IndexNo=p.IndexNo
    )[0].Percent
258                     tot += Employee_Project_PayNo.objects.
    get_or_create(Year=payno, GID=t.GID, IndexNo=p.IndexNo)[0].
    Percent
259             base_totals.append([t.GID+"base", base])
260             if tot > 100: # set initial highlight colors
261                 total.append([tot, t.GID, red]) # over
262             elif tot < 100:
263                 total.append([tot, t.GID, yellow]) # under
264             else:
265                 total.append([tot, t.GID, green]) # 100
266         row.append('Final Target'); row.append('Initial Target
    '); row.append('Difference')
267         context.append(row)
268         row_count = [0] * len(employees)
269         pn = PayNo.objects.get(Year=payno)
270         for i in team_projects: # construct body of table
271             proj = Project.objects.get(IndexNo=i.IndexNo)
272             if pn.End <= proj.EndDate or pn.Start >= proj.
    StartDate: # only include index in table if it's still being
    paid out
273                 if pn.Start <= proj.EndDate <= pn.End: # tests
     if in last payno for index
274                     row = [['t' ,proj.IndexNo, red], ['t',
    proj.EndDate, red]]
275                 elif pn.Start <= proj.StartDate <= pn.End: #
    first pn for index
276                     row = [['t', proj.IndexNo, green], ['t',
    proj.EndDate, green]]
277                 else:
278                     row = [['t', proj.IndexNo, 'white'], ['t'
    , proj.EndDate, 'white']]
279                 employee_count = 0
280                 for e in employees: # for each employee get
    the percent for index for that payno
281                     if Project_Employee.objects.filter(GID=e.
    GID, IndexNo=i.IndexNo).exists(): # if that employee works on
```

```
281 that project
282                         percent = Employee_Project_PayNo.
    objects.get_or_create(Year=payno, GID=e.GID, IndexNo=proj.
    IndexNo)[0]
283                         row.append(['p', percent.Percent, e.
    GID+" "+i.IndexNo, e.GID +" "+str(row_count[employee_count])])
284                         row_count[employee_count] += 1
285                     else:
286                         row.append('')
287                     employee_count += 1
288                 proj_payno = Project_PayNo.objects.
    get_or_create(Year=payno, IndexNo=proj.IndexNo)[0]
289                 row.append(add_dot(final_target(i.IndexNo,
    payno))) # final target
290                 row.append(add_dot(proj_payno.SalBenPlan)) #
    initial target
291                 row.append(add_dot(proj_payno.SalBenPlan -
    final_target(i.IndexNo, payno))) # difference
292                 context.append(row)
293         context.append(total)
294         return render(request, 'indexdisperTeam.html', {'url'
    : url, 'base_totals': base_totals, 'title': title, 'context':
    context})
295
296 @login_required(login_url=login_url)
297 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD')
298 def disPaynoEmployee(request, id, start, end):
299     if request.method == 'POST':
300         values = list(request.POST.items())
301         values.pop(0)  # get rid of csrf token
302         for input in values:  # (id, value)
303             ids = input[0].split(" ")
304             percent = Employee_Project_PayNo.objects.get(Year=
    ids[0], GID=id, IndexNo=ids[1])
305             percent.Percent = input[1]
306             percent.save()
307         return redirect('/disPayNoEmployee/'+id+'/'+start+'/'+
    end)
308     else:
309         url = '/disPayNoEmployee/'+id+'/'+start+'/'+end
310         title = "Distributions for " + id + " from " + start
     + " to " + end
311         header, body, tail, projects = [], [], [], []
312         starting = PayNo.objects.get(Year=start).Start
313         ending = PayNo.objects.get(Year=end).End
314         ep = Project_Employee.objects.filter(GID=id) # all
    projects for employee
```

```
315         for p in ep:
316             proj = Project.objects.get(IndexNo=p.IndexNo)
317             if ending >= proj.StartDate and starting <= proj.
    EndDate: # only add if project within range
318                 projects.append(Project.objects.get(IndexNo=p.
    IndexNo))
319         if len(projects) == 0: # no projects in range throw
    error
320             return render(request, 'error.html',
321                           {'message': "There are no projects
    to display for the pay numbers you requested",
322                            'url': "/table/DPNE"})
323         paynos = PayNo.objects.filter(End__range=[starting,
    ending]).order_by('Start')
324         payno = ['', 'Pay Number'] # header rows
325         pn_start = ['', 'Payno Start']
326         pn_end = ['', 'Payno End']
327         POP = ['POP Date', 'Pay Date']
328         for p in paynos: # build header rows
329             payno.append(p.Year); payno.append('')
330             pn_start.append(p.Start); pn_start.append('')
331             pn_end.append(p.End); pn_end.append('')
332             POP.append(p.Date)
333             if p.End < date.today(): # past pns
334                 POP.append('Actual')
335             else: # current and future pns
336                 POP.append('Initial Target')
337         header.append(payno); header.append(pn_start); header.
    append(pn_end); header.append(POP)
338         tail = ['SUM'] # bottom row
339         for p in projects:
340             proj = Project.objects.get(IndexNo=p.IndexNo)
341             body.append([proj.EndDate, proj.IndexNo])
342         for pn in paynos:
343             context_counter = 0
344             total = 0
345             row_count = 0
346             for p in projects:
347                 proj = Project.objects.get(IndexNo=p.IndexNo)
348                 if pn.Start > proj.EndDate or pn.End < proj.
    StartDate: # project not on pn
349                     element = 'x' # display nothing
350                 else:
351                     element = []
352                     percent = Employee_Project_PayNo.objects.
    get_or_create(Year=pn.Year, GID=id, IndexNo=proj.IndexNo)[0].
    Percent
353                     total += percent # build total % for
```

```
353 employee for pn
354                     element.append([percent, pn.Year+" "+p.
    IndexNo, pn.Year+" "+str(row_count)])
355                     row_count +=1
356                     if pn.End < date.today(): # past pn
    display actuals
357                         actual = Project_PayNo.objects.
    get_or_create(Year=pn.Year, IndexNo=proj.IndexNo)[0]
358                         element.append([add_dot(actual.SalCat+
    actual.BenCat)])
359                     else: # current and future pns display
    index targ
360                         element.append([add_dot(Project_PayNo.
    objects.get_or_create(Year=pn.Year, IndexNo=proj.IndexNo)[0].
    SalBenPlan)])
361                     if pn.Start <= proj.EndDate <= pn.End: #
    last pn for proj
362                         element[1].append(red)
363                     elif pn.Start <= proj.StartDate <= pn.End
    : # first pn for proj
364                         element[1].append(green)
365                     else:
366                         element[1].append('white')
367                 body[context_counter].append(element)
368                 context_counter += 1
369             if total > 100: # total % over 100
370                 tail.append([str(total), pn.Year, red])
371             elif total < 100: # total % under 100
372                 tail.append([str(total), pn.Year, yellow])
373             else: # total % is 100
374                 tail.append([str(total), pn.Year, green])
375         return render(request, 'dispaynoEmp.html', {'url': url
    , 'tile': title, 'header': header, 'body': body, 'tail': tail
    })
376
377 @login_required(login_url=login_url)
378 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD')
379 def disPaynoIndex(request, id, start, end):
380     if request.method == 'POST':
381         values = list(request.POST.items())
382         values.pop(0)  # get rid of csrf token
383         for input in values:  # (id, value)
384             ids = input[0].split(" ")
385             percent = Employee_Project_PayNo.objects.get(Year=
    ids[0], GID=ids[1], IndexNo=id)
386             percent.Percent = input[1]
387             percent.save()
```

```
388         return redirect('/disPayNoIndex/'+id+'/'+start+'/'+end
    )
389     else:
390         url = '/disPayNoIndex/'+id+'/'+start+'/'+end
391         wages, base_totals, header, body, tail
     = [], [], [], [], []
392         starting = PayNo.objects.get(Year=start).Start
393         ending = PayNo.objects.get(Year=end).End
394         project = Project.objects.get(IndexNo=id)
395         title = "Index Distributions for " + id + " from " +
    start + " to " + end
396         employees = Project_Employee.objects.filter(IndexNo=id
    )
397         paynos = PayNo.objects.filter(End__range=[starting,
    ending]).order_by('Start')  # table selection cuts off pn
    after proj end
398         payno = ['Pay Number'] # header rows
399         pn_start = ['Payno Start']
400         pn_end = ['Payno End']
401         pn_date = ['Pay Date']
402         color = ['white']
403         for p in paynos: # build header
404             payno.append(p.Year); payno.append('')
405             pn_start.append(p.Start); pn_start.append('')
406             pn_end.append(p.End); pn_end.append('')
407             pn_date.append(p.Date); pn_date.append('Total')
408             if p.Start <= project.EndDate <= p.End: # test if
    in last payno for project
409                 color.append(red); color.append(red)
410             elif p.Start <= project.StartDate <= p.End: # test
     if first payno for project
411                 color.append(green); color.append(green)
412             else:
413                 color.append('white'); color.append('white')
414         payno = list(zip(payno, color)); pn_start = list(zip(
    pn_start, color)) # add colors to data
415         pn_end = list(zip(pn_end, color)); pn_date = list(zip(
    pn_date, color))
416         header.append(payno); header.append(pn_start); header.
    append(pn_end); header.append(pn_date)
417         row_count = 0
418         for e in employees:
419             employee = Employee.objects.get(GID=e.GID)
420             wages.append([e.GID, employee.BiweeklyBenefits+
    employee.BiweeklySalary]) # wages for js final targ calcs
421             row = [employee.GID]
422             employee_projects = Project_Employee.objects.
    filter(GID=e.GID).exclude(IndexNo=id)
```

```
423             base = 0
424             for pn in paynos: # build body of table
425                 for ep in employee_projects:
426                     base += Employee_Project_PayNo.objects.
    get_or_create(Year=pn.Year,GID=employee.GID,IndexNo=ep.IndexNo
    )[0].Percent
427                 percent = Employee_Project_PayNo.objects.
    get_or_create(Year=pn.Year,GID=employee.GID,IndexNo=id)[0].
    Percent
428                 row.append([percent, pn.Year+" "+e.GID, pn.
    Year+" "+str(row_count)])
429                 if (base+percent > 100): # total % less than
    100
430                     row.append([base + percent, e.GID + " " +
    pn.Year, red])
431                 elif (base+percent < 100): # total % greater
    than 100
432                     row.append([base + percent, e.GID + " " +
    pn.Year, yellow])
433                 else: # total % is 100
434                     row.append([base + percent, e.GID + " " +
    pn.Year, green])
435                 base_totals.append([base, e.GID+" "+pn.Year+"
    base"])
436             row_count += 1
437             body.append(row)
438         iit = ['Initial Index Target'] # tail rows
439         fit = ['Final Index Target']
440         dit = ['Difference']
441         ipd = ['Index POP Date']
442         for pn in paynos: # add gaps for totals, build tails
443             project_payno = Project_PayNo.objects.
    get_or_create(Year=pn.Year, IndexNo=id)[0]
444             iit.append([add_dot(project_payno.SalBenPlan), pn.
    Year+"init"]); iit.append('')
445             fit.append([add_dot(final_target(id, pn.Year)), pn
    .Year+"fin"]); fit.append('')
446             dit.append([add_dot(project_payno.SalBenPlan-
    final_target(id, pn.Year)), pn.Year+"dif"]); dit.append('')
447             ipd.append([project.EndDate, pn.Year+"end"]); ipd.
    append('')
448         tail.append(iit); tail.append(fit); tail.append(dit);
    tail.append(ipd)
449         return render(request, 'dispaynoIndex.html', { 'url':
    url, 'base_totals': base_totals, 'wages': wages,
450                                                     'title'
    : title, 'header': header, 'body': body,
451                                                     'tail'
```

```
451 : tail, 'color': color})
452 @login_required(login_url=login_url)
453 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD', 'Employee.EPAF_SUPPORT')
454 def EPAFquerie(request, kind, project, team, employee, prior,
    current):
455     start = PayNo.objects.get(Year=prior)
456     end = PayNo.objects.get(Year=current)
457     employees, projects = [], []
458     if kind == 'a': # both all projects and all employees
459         title = "EPAF Queries for All Employees and Indexes
    between " + prior + " and " + current
460         projects = Project.objects.all().exclude(Note="
    Subaccount").filter(EndDate__gte=start.Start, StartDate__lte=
    end.End)
461         employees = Employee.objects.all()
462     elif kind == 'i': # for specific project
463         title = "EPAF Queries for " + project + " between " +
    prior + " and " + current
464         project_employees = Project_Employee.objects.filter(
    IndexNo=project)
465         for e in project_employees:
466             employees.append(Employee.objects.get(GID=e.GID))
467             proj = Project.objects.get(IndexNo=project)
468             if end.End >= proj.StartDate and start.Start <=
    proj.EndDate:
469                 projects.append(proj)
470     elif kind == 't': # for specific team
471         title = "EPAF Queries for " + team + " between " +
    prior + " and " + current
472         project_teams = Project_Team.objects.filter(TeamId=
    team).filter(EndDate__gte=start.Start, StartDate__lte=end.End)
473         for p in project_teams:
474             proj = Project.objects.get(IndexNo=p.IndexNo)
475             if end.End >= proj.StartDate and start.Start <=
    proj.EndDate:
476                 projects.append(proj)
477         team_employees = Team_Employee.objects.filter(TeamId=
    team)
478         for e in team_employees:
479             employees.append(Employee.objects.get(GID=e.GID))
480     elif kind == 'e': # for specific employee
481         emp = Employee.objects.get(GID=employee)
482         title = "EPAF Queries for " + emp.__str__() + "
    between " + prior + " and " + current
483         employees.append(emp)
484         project_employees = Project_Employee.objects.filter(
    GID=employee)
```

```
485         for p in project_employees:
486             proj = Project.objects.get(IndexNo=p.IndexNo)
487             if end.End >= proj.StartDate and start.Start <=
    proj.EndDate:
488                 projects.append(proj)
489     if len(projects) == 0: # if no project throw error page
490         return render(request, 'error.html',
491                       {'message': "There are no projects to
    display for the pay numbers you requested",
492                        'url': "/table/EQ"})
493     header = ['Projects', 'Payno Start', 'Previous Payno', '
    Current Payno']
494     context = []
495     for e in employees: # build header
496         header.append(e.__str__() + " Past PN")
497         header.append(e.__str__() + " Current PN")
498     context.append(header)
499     for p in projects: # build body of table
500         row = [p.IndexNo, start, prior, current]
501         for e in employees:
502             if Project_Employee.objects.filter(GID=e.GID,
    IndexNo=p.IndexNo).exists(): # check emp works on proj
503                 past = Employee_Project_PayNo.objects.
    get_or_create(Year=prior, IndexNo=p.IndexNo, GID=e.GID)[0].
    Percent
504                 present = Employee_Project_PayNo.objects.
    get_or_create(Year=current, IndexNo=p.IndexNo, GID=e.GID)[0].
    Percent
505                 if (past != present): # % changed from pn1 to
    pn2
506                     row.append([past, present, 'c'])
507                 else: # no change
508                     row.append([past, present, 'nc'])
509             else: # employee not on project
510                 row.append(['x','x'])  # display nothing
511         context.append(row)
512     return render(request, 'EpafQueries.html', {'title': title
    , 'context': context})
513
514 @login_required(login_url=login_url)
515 def benefitsUpdate(request, kind, employee, payno):
516     if request.method == 'POST':
517         if kind == 'a': # one employee for one pn
518             benefits = Employee_PayNo.objects.get(Year=payno,
    GID=employee)
519             benefits.Benefits = int(request.POST[employee].
    replace('.',''))
520             benefits.save()
```

```
521             return redirect("/benefitUpdate/" + kind + '/' +
    employee + '/' + payno)
522         elif kind == 'p': # all employees for one pn
523             pn = PayNo.objects.get(Year=payno)
524             employees = Employee.objects.all().order_by('
    FirstName', 'LastName')
525             for e in employees:
526                 benefits = Employee_PayNo.objects.
    get_or_create(Year=payno, GID=e.GID)[0]
527                 benefits.Benefits = int(request.POST[e.GID].
    replace('.',''))
528                 benefits.save()
529             return redirect("/benefitUpdate/" + kind + '/' +
    employee + '/' + payno)
530         elif kind == 'i': # one employee for current and
    future pns
531             emp = Employee.objects.get(GID=employee)
532             paynos = PayNo.objects.all().order_by('Start').
    exclude(Start__lt=date.today())
533             for p in paynos:
534                 benefits = Employee_PayNo.objects.
    get_or_create(Year=p.Year, GID=employee)[0]
535                 benefits.Benefits = int(request.POST[p.Year].
    replace('.',''))
536                 benefits.save()
537             return redirect("/benefitUpdate/" + kind + '/' +
    employee + '/' + payno)
538         else:
539             return redirect(dashboard_url)
540     else:
541         url = "/benefitUpdate/" + kind + '/' + employee + '/'
     + payno
542         context = []
543         if kind =='a': # one employee for one pn
544             title = "Update Benefits for " + employee + " for
     " + payno
545             emp = Employee.objects.get(GID=employee)
546             pn = PayNo.objects.get(Year=payno)
547             context.append(pn.Year + "(" + str(pn.Start) +
    ", " + str(pn.End) + ")")
548             benefits = add_dot(Employee_PayNo.objects.
    get_or_create(Year=payno,GID=employee)[0].Benefits)
549
550             context.append((emp.GID, emp.FirstName+' '+emp.
    LastName, benefits))
551             return render(request, 'BUFtable.html', {'title':
    title,'url': url, 'context': context})
552         elif kind == 'p': # all employees for one pn
```

```
553             title = "Update Benefits for All Employees for "
     + payno
554             pn = PayNo.objects.get(Year=payno)
555             employees = Employee.objects.all().order_by('
    FirstName','LastName')
556             context.append(pn.Year + "(" + str(pn.Start) +
    ", " + str(pn.End) + ")")
557             for e in employees:
558                 benefits = add_dot(Employee_PayNo.objects.
    get_or_create(Year=payno, GID=e.GID)[0].Benefits)
559                 context.append((e.GID, e.FirstName+' '+e.
    LastName, benefits))
560             return render(request, 'BUFtable.html', {'title':
    title,'url': url, 'context': context})
561         elif kind == 'i': # one employee for current and
    future pns
562             title = "Update Benefits for " + employee + " for
    All Upcoming Pay Numbers"
563             emp = Employee.objects.get(GID=employee)
564             paynos = PayNo.objects.all().order_by('Start').
    exclude(Start__lt=date.today())
565             context.append(emp.FirstName+' '+emp.LastName)
566             for p in paynos:
567                 benefits = add_dot(Employee_PayNo.objects.
    get_or_create(Year=p.Year, GID=employee)[0].Benefits)
568                 context.append((p.Year, p.Year + "(" + str(p.
    Start) + ", " + str(p.End) + ")", benefits))
569             return render(request, 'BUFtable.html', {'title':
    title,'url': url, 'context': context})
570         else:
571             redirect('/table/BUF')
572
573 @login_required(login_url=login_url)
574 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD', 'Employee.OPERATIONS_SUPPORT')
575 def salBenTotalRemaining(request, kind, project, team):
576     header = ['Index', 'POP Date', 'Salary & Benefit Budget
    Remaining']
577     if kind == 'a': # salben remaining for all projects
578         title = "Salary and Benefits Budget Remaining for All
    Indexes"
579         projects = Project.objects.all().order_by('IndexNo').
    exclude(Note="Subaccount")
580     elif kind == 'i': # for one project
581         title = "Salary and Benefits Budget Remaining for " +
    project
582         projects = [Project.objects.get(IndexNo=project)]
583     elif kind == 't': # for all projects on team
```

```
584         title = "Salary and Benefits Budget Remaining for All
    Projects for " + team
585         projects_for_team = Project_Team.objects.filter(TeamId
    =team)
586         projects = []
587         for p in projects_for_team:
588             projects.append(Project.objects.get(IndexNo=p.
    IndexNo))
589     else: # nothing
590         title = ''
591         projects = []
592     body = []
593     for p in projects:
594         if date.today() + timedelta(days=-14) <= p.EndDate <=
    date.today() + timedelta(days=14): # test if near end
595             body.append([p.IndexNo, p.EndDate, add_dot(
    total_remaing(p.IndexNo)), red])
596         elif date.today() + timedelta(days=-14) <= p.StartDate
     <= date.today() + timedelta(days=14): # test if near start
597             body.append([p.IndexNo, p.EndDate, add_dot(
    total_remaing(p.IndexNo)), green])
598         else:
599             body.append([p.IndexNo, p.EndDate, add_dot(
    total_remaing(p.IndexNo)), 'white'])
600     return render(request, 'salbentotremain.html', {'title':
    title, 'header': header, 'body': body})
601
602 @login_required(login_url=login_url)
603 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD', 'Employee.OPERATIONS_SUPPORT')
604 def allEcumbrances(request, kind, payno, project, team):
605     pn = PayNo.objects.get(Year=payno)
606     header = ['Index', 'POP Date', 'Pay Period End Date', '
    Total Amount to Encumber']
607     body, projects = [], []
608     if kind == 'a': # all projects for pn
609         title = "All Encumbrances for All Indexes for " +
    payno
610         projects = Project.objects.all().exclude(Note="
    Subaccount").filter(EndDate__gte=pn.Start, StartDate__lte=pn.
    End).order_by('IndexNo')
611     elif kind == 'i': # one index for pn
612         title = "All Ecumbrances for " + project + " for " +
    payno
613         proj = Project.objects.get(IndexNo=project)
614         if pn.End >= proj.StartDate and pn.Start <= proj.
    EndDate:
615             projects.append(proj)
```

```
616     elif kind == 't': # indexes on team for pn
617         title = "All Ecumbrances for " + team + " for " +
    payno
618         projects_for_team = Project_Team.objects.filter(TeamId
    =team)
619         for p in projects_for_team:
620             proj = Project.objects.get(IndexNo=p.IndexNo)
621             if pn.End >= proj.StartDate and pn.Start <= proj.
    EndDate:
622                 projects.append(Project.objects.get(IndexNo=p.
    IndexNo))
623     else:
624         title = ""
625     if len(projects) == 0:
626         return render(request, 'error.html',
627                {'message': "There are no encumbrances to
    display for the pay number you requested", 'url': "/table/AEPN
    "})
628     for p in projects:
629         if pn.Start <= p.EndDate <= pn.End: # last pn for proj
630             body.append([p.IndexNo, p.EndDate, pn.End, add_dot
    (final_target(p.IndexNo, payno)), red])
631         elif pn.Start <= p.StartDate <= pn.End: # first pn for
     proj
632             body.append([p.IndexNo, p.EndDate, pn.End, add_dot
    (final_target(p.IndexNo, payno)), green])
633         else:
634             body.append([p.IndexNo, p.EndDate, pn.End, add_dot
    (final_target(p.IndexNo, payno)), 'white'])
635     return render(request, 'allEncum.html', {'title': title, '
    header': header, 'body': body})
636
637 @login_required(login_url=login_url)
638 def selectTable(request, type):
639     if request.method == 'POST':
640         if type == "ITT":
641             form = IndexTarTeamSelectionForm(request.POST or
    None)
642             id = form['team'].value()
643             start = form['start'].value()
644             end = form['end'].value()
645             if (not startLTEend(start,end)):
646                 return render(request, 'error.html', {'message
    ': "Starting pay number was greater than ending pay number.",
647                                                       'url':
    "/table/ITT"})
648             return redirect('/indexTargetsPerTeam/' + id + '/'
     + start + '/' + end)
```

```
649         elif type == "IDT":
650             form = IndexDisTeamSelectionForm(request.POST or
    None)
651             id = form['team'].value()
652             payno = form['payno'].value()
653             return redirect('/indexDisPerTeam/' + id + '/' +
    payno)
654         elif type == "SBT":
655             form = SalBenTotalSelectionForm(request.POST or
    None)
656             selection = form['selection'].value()
657             project = form['project'].value()
658             team = form['team'].value()
659             return redirect('/salBenTotal/'+selection+'/'+
    project+'/'+team)
660         elif type == "DPNE":
661             form = DisPayNoEmployeeSelectionForm(request.POST
    or None)
662             id = form['employee'].value()
663             start = form['start'].value()
664             end = form['end'].value()
665             if (not startLTEend(start,end)): # checks valid
    range
666                 return render(request, 'error.html', {'message
    ': "Starting pay number was greater than ending pay number.",
667                                                       'url':
    "/table/DPNE"})
668             return redirect('/disPayNoEmployee/' + id + '/' +
    start + '/' + end)
669         elif type == "DPNI":
670             form = DisPayNoIndexSelectionForm(request.POST or
    None)
671             id = form['index'].value()
672             start = form['start'].value()
673             end = form['end'].value()
674             if (not startLTEend(start,end)): # checks valid
    range
675                 return render(request, 'error.html', {'message
    ': "Starting pay number was greater than ending pay number.",
676                                                       'url':
    "/table/DPNI"})
677             project = Project.objects.get(IndexNo=id)
678             s = PayNo.objects.get(Year=start)
679             e = PayNo.objects.get(Year=end)
680             if s.Start > project.EndDate or e.End < project.
    StartDate: # checks project within range of pns
681                 return render(request, 'error.html', {'message
    ': "Pay Numbers selected out of range for Index",
```

```
682                                                     'url':
    "/table/DPNI"})
683             elif e.Start > project.EndDate:  # cuts off all pn
     after project end date
684                 end = PayNo.objects.filter(Start__lte=project.
    EndDate).exclude(End__lte=project.EndDate)[0].Year
685             elif s.End < project.StartDate:  # cuts off all pn
     before project start date
686                 start = PayNo.objects.filter(Start__lte=
    project.StartDate).exclude(End__lte=project.StartDate)[0].Year
687             return redirect('/disPayNoIndex/' + id + '/' +
    start + '/' + end)
688         elif type == "AEPN":  # All Encumbrances -1 Paynos
689             form = AllEncumSelectionForm(request.POST or None)
690             selection = form['selection'].value()
691             payno = form['payno'].value()
692             project = form['project'].value()
693             team = form['team'].value()
694             return redirect('/allEncumbrances/'+selection+'/'+
    payno+'/'+project+'/'+team)
695         elif type == "EQ":  # EPAF Queries
696             form = EPAFQueriesSelectionForm(request.POST or
    None)
697             selection = form['selection'].value()
698             prior = form['prior'].value()
699             current = form['current'].value()
700             if (not startLTEend(prior, current)): # checks
    valid range
701                 return render(request, 'error.html', {'message
    ': "Prior pay number was greater than current pay number.",
702                                                       'url':
    "/table/EQ"})
703             elif(prior == current): # cant select same pns
704                 return render(request, 'error.html',
705                               {'message': "Prior pay number
    cannot be equal to current pay number.",
706                                'url': "/table/EQ"})
707             project = form['project'].value()
708             team = form['team'].value()
709             employee = form['employee'].value()
710             return redirect('/EPAFquerie/'+selection+'/'+
    project+'/'+team+'/'+employee+'/'+prior+'/'+current)
711         elif type == "BUF":  # Benefits Update Form
712             form = BenefitsUpdateSelectionForm(request.POST or
     None)
713             selection = form['selection'].value()
714             payno = form['payno'].value()
715             employee = form['employee'].value()
```

```
716             return redirect('/benefitUpdate/'+selection+'/'+
    employee+'/'+payno)
717         else:
718             return redirect(dashboard_url)
719     else:
720         if type == "ITT":  # Index Targets Per Team
721             if request.user.has_perm('Employee.FINANCE_LEAD')
    or request.user.has_perm('Employee.OPERATIONS_LEAD') or
    request.user.has_perm('Employee.OPERATIONS_SUPPORT'):
722                 form = IndexTarTeamSelectionForm
723                 return render(request, 'tableSelection.html'
    , {'form': form, 'title': "Index Targets Per Team"})
724             else:
725                 return render(request, '403.html')
726         elif type == "IDT":  # Index Distributions Per Team
727             if request.user.has_perm('Employee.FINANCE_LEAD')
    or request.user.has_perm('Employee.OPERATIONS_LEAD'):
728                 form = IndexDisTeamSelectionForm
729                 return render(request, 'tableSelection.html'
    , {'form': form, 'title': "Index Distributions Per Team"})
730             else:
731                 return render(request, '403.html')
732         elif type == "SBT":  # Sal/Ben Remaining
733             if request.user.has_perm('Employee.FINANCE_LEAD')
    or request.user.has_perm('Employee.OPERATIONS_LEAD') or
    request.user.has_perm('Employee.OPERATIONS_SUPPORT'):
734                 form = SalBenTotalSelectionForm
735                 return render(request, 'tableSelection.html'
    , {'form': form, 'title': "Salary and Benefits Remaining"})
736             else:
737                 return render(request, '403.html')
738         elif type == "DPNE":  # Distribution across Paynos -1
    Employee
739             if request.user.has_perm('Employee.FINANCE_LEAD')
    or request.user.has_perm('Employee.OPERATIONS_LEAD'):
740                 form = DisPayNoEmployeeSelectionForm
741                 return render(request, 'tableSelection.html'
    , {'form': form, 'title': "Distribution across Paynos -1
    Employee"})
742             else:
743                 return render(request, '403.html')
744         elif type == "DPNI":  # Distribution Across Paynos -1
    index
745             if request.user.has_perm('Employee.FINANCE_LEAD')
    or request.user.has_perm('Employee.OPERATIONS_LEAD'):
746                 form = DisPayNoIndexSelectionForm
747                 return render(request, 'tableSelection.html'
    , {'form': form, 'title': "Distribution Across Paynos -1 index
```

```
747 "})
748             else:
749                 return render(request, '403.html')
750         elif type == "AEPN":  # All Encumbrances -1 Paynos
751             if request.user.has_perm('Employee.FINANCE_LEAD')
    or request.user.has_perm(
752                     'Employee.OPERATIONS_LEAD') or request.
    user.has_perm('Employee.OPERATIONS_SUPPORT'):
753                 form = AllEncumSelectionForm
754                 return render(request, 'tableSelection.html'
    , {'form': form, 'title': "All Encumbrances -1 Paynos"})
755             else:
756                 return render(request, '403.html')
757         elif type == "EQ":  # EPAF Queries
758             if request.user.has_perm('Employee.FINANCE_LEAD')
    or request.user.has_perm(
759                     'Employee.OPERATIONS_LEAD') or request.
    user.has_perm('Employee.EPAF_SUPPORT'):
760                 form = EPAFQueriesSelectionForm
761                 return render(request, 'tableSelection.html'
    , {'form': form, 'title': "EPAF Queries"})
762             else:
763                 return render(request, '403.html')
764         elif type == "BUF":  # Benefits Update Form
765             if request.user.has_perm('Employee.FINANCE_LEAD')
    or request.user.has_perm('Employee.HR'):
766                 form = BenefitsUpdateSelectionForm
767                 return render(request, 'tableSelection.html'
    , {'form': form, 'title': "Benefits Update Form"})
768             else:
769                 return render(request, '403.html')
770         else:
771             return redirect(dashboard_url)
772
773 @login_required(login_url=login_url)
774 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD')
775 def addPayno(request):
776     if request.method == 'POST':
777         form = PaynoForm(request.POST)
778         if form.is_valid():
779             form.save()
780             messages.success(request, 'Payno Added
    Successfully')
781             return redirect(dashboard_url)
782     else:
783         form = PaynoForm()
784         return render(request, 'addPayno.html', {'form': form
```

```
784 })
785
786 @login_required(login_url=login_url)
787 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD', 'Employee.HR')
788 def selectEmployee(request):
789     if request.method == 'POST':
790         selection = EmployeeSelectionForm(request.POST or None
    )
791         choice = selection['employee'].value()
792         return redirect('employee/' + choice)
793     else:
794         selectionForm = EmployeeSelectionForm
795         return render(request, 'employeeForm.html', {'form':
    selectionForm})
796
797
798 @login_required(login_url=login_url)
799 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD','Employee.HR')
800 def viewEmployee(request, gid):
801     employee_object = Employee.objects.get(GID=gid)
802     employee = [employee_object.GID, employee_object.NetId,
    employee_object.FirstName,
803                 employee_object.LastName, employee_object.
    PhoneNum, employee_object.Email,
804                 employee_object.Title, employee_object.
    PositionNum, employee_object.Zorg,
805                 employee_object.StartDate, add_dot(
    employee_object.CurrentSalary), employee_object.LastSalMod,
806                 employee_object.SalModDate, add_dot(
    employee_object.CellphoneAllowance), add_dot(employee_object.
    Benefits),
807                 add_dot(employee_object.BiweeklySalary),
    add_dot(employee_object.BiweeklyBenefits), employee_object.
    TeamPool,
808                 employee_object.LOA, employee_object.FTE]
809     return render(request, 'Employee.html', {'Employee':
    employee})
810
811
812 @login_required(login_url=login_url)
813 @permission_required('Employee.FINANCE_LEAD', 'Employee.HR')
814 def addEmployee(request):
815     if request.method == 'POST':
816         form = EmployeeCreationForm(request.POST)
817         if form.is_valid():
818             gid = form['GID'].value()
```

```
819             employee = form.save(commit=False)
820             employee.CurrentSalary = int(float(form['
    CurrentSalary'].value()) * 100)
821             employee.LastSalMod = int(float(form['LastSalMod'
    ].value()) * 100)
822             employee.CellphoneAllowance = int(float(form['
    CellphoneAllowance'].value()) * 100)
823             employee.Benefits = int(float(form['Benefits'].
    value()) * 100)
824             employee.TeamPool = int(float(form['TeamPool'].
    value()) * 100)
825             employee.BiweeklyBenefits = int(float(form['
    BiweeklyBenefits'].value()) * 100)
826             employee.BiweeklySalary = int(float(form['
    BiweeklySalary'].value()) * 100)
827             employee.save()
828             messages.success(request, 'Employee Added
    Successfully')
829             return redirect("/employee/" + gid)
830     else:
831         form = EmployeeCreationForm()
832         return render(request, 'addEmployee.html', {'form':
    form})
833
834
835 @login_required(login_url=login_url)
836 @permission_required('Employee.FINANCE_LEAD', 'Employee.HR')
837 def updateEmployee(request, gid):
838     if request.method == 'POST':
839         form = EmployeeChangeForm(request.POST, instance=
    Employee.objects.get(GID=gid))
840         if form.is_valid():
841             employee = form.save(commit=False)
842             employee.CurrentSalary = int(float(form['
    CurrentSalary'].value()) * 100)
843             employee.LastSalMod = int(float(form['LastSalMod'
    ].value()) * 100)
844             employee.CellphoneAllowance = int(float(form['
    CellphoneAllowance'].value()) * 100)
845             employee.Benefits = int(float(form['Benefits'].
    value()) * 100)
846             employee.BiweeklyBenefits = int(float(form['
    BiweeklyBenefits'].value()) * 100)
847             employee.BiweeklySalary = int(float(form['
    BiweeklySalary'].value()) * 100)
848             employee.save()
849             messages.success(request, 'Profile updated
    successfully!')
```

```
850             return redirect("/employee/" + gid)
851     else:
852         employee = Employee.objects.get(GID=gid)
853         cursal = round(employee.CurrentSalary / 100)
854         lsm = round(employee.LastSalMod / 100)
855         cpa = round(employee.CellphoneAllowance / 100)
856         ben = round(employee.Benefits / 100)
857         bwb = round(employee.BiweeklyBenefits / 100)
858         bws = round(employee.BiweeklySalary / 100)
859         form = EmployeeUpdateForm(instance=Employee.objects.
    get(GID=gid),
860                                   initial={'CurrentSalary':
    cursal, 'LastSalMod':lsm, 'CellphoneAllowance': cpa,
861                                            'Benefits': ben,'
    BiweeklyBenefits': bwb, 'BiweeklySalary': bws})
862         return render(request, 'update.html', {'form': form})
863
864 @login_required(login_url=login_url)
865 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD')
866 def selectProject(request):
867     if request.method == 'POST':
868         selection = ProjectSelectionForm(request.POST or None)
869         choice = selection['project'].value()
870         return redirect('project/' + choice)
871     else:
872         selectionForm = ProjectSelectionForm
873         return render(request, 'selectproject.html', {'form':
    selectionForm})
874
875 @login_required(login_url=login_url)
876 def viewProject(request, id):
877     project_object = Project.objects.get(IndexNo=id)
878     project = [project_object.IndexNo, project_object.Title,
    project_object.OperationsLead,
879                project_object.StartDate, project_object.
    EndDate, add_dot(project_object.TotalGrantFunds),
880                 add_dot(project_object.SalaryBenefitBudget),
    project_object.Note]
881     return render(request, 'Project.html', {'project': project
    })
882
883 @login_required(login_url=login_url)
884 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD')
885 def addProject(request):
886     if request.method == 'POST':
887         form = ProjectForm(request.POST)
```

```
888         if form.is_valid():
889             indexNo = form['IndexNo'].value()
890             budget = int(float(form['SalaryBenefitBudget'].
    value())*100) # initializes index targets for index
891             total_grant = int(float(form['TotalGrantFunds'].
    value()) * 100)
892             paynos = PayNo.objects.filter(End__gte=datetime.
    strptime(form['StartDate'].value(),"%m/%d/%Y"),
893                                           Start__lte=
    datetime.strptime(form['EndDate'].value(), "%m/%d/%Y")).
    order_by('Start')
894             balanced_index_targ = round(budget / len(paynos))
895             for p in paynos:
896                 object = Project_PayNo.objects.get_or_create(
    IndexNo=indexNo, Year=p.Year)[0]
897                 object.SalBenPlan = balanced_index_targ
898                 object.save()
899             new_project = form.save(commit=False)
900             new_project.SalaryBenefitBudget = budget
901             new_project.TotalGrantFunds = total_grant
902             new_project.save()
903             messages.success(request, 'Project Added
    Successfully')
904             return redirect("/project/" + indexNo)
905     else:
906         form = ProjectForm()
907         return render(request, 'addProject.html', {'form':
    form})
908
909 @login_required(login_url=login_url)
910 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD')
911 def updateProject(request, id):
912     if request.method == 'POST':
913         form = ProjectForm(request.POST, instance=Project.
    objects.get(IndexNo=id))
914         if form.is_valid():
915             budget = int(float(form['SalaryBenefitBudget'].
    value()) * 100)
916             total_grant = int(float(form['TotalGrantFunds'].
    value()) * 100)
917             project = form.save(commit=False)
918             if project.SalaryBenefitBudget != budget:
919                 paynos = PayNo.objects.filter(End__gte=
    datetime.strptime(form['StartDate'].value(), "%m/%d/%Y"),
920                                               Start__lte=datetime.strptime(
    form['EndDate'].value(), "%m/%d/%Y")).order_by('Start')
921                 balanced_index_targ = round(budget / len(
```

```
paynos))
                for p in paynos:
                    object = Project_PayNo.objects.
get_or_create(IndexNo=project.IndexNo, Year=p.Year)[0]
                    object.SalBenPlan = balanced_index_targ
                    object.save()
            project.SalaryBenefitBudget = budget
            project.TotalGrantFunds = total_grant
            project.save()
            messages.success(request, 'Project updated
successfully!')
            return redirect('/project/' + id)
    else:
        project = Project.objects.get(IndexNo=id)
        budget = round((project.SalaryBenefitBudget) / 100)
        total_grant = round((project.TotalGrantFunds) / 100)
        start = datetime.strftime(project.StartDate, "%m/%d/%Y
")
        end = datetime.strftime(project.EndDate, "%m/%d/%Y")
        form = ProjectForm(instance=Project.objects.get(
IndexNo=id),
                           initial={'SalaryBenefitBudget':
budget, 'TotalGrantFunds': total_grant,
                                    'StartDate': start, '
EndDate': end})
        return render(request, 'updateProject.html', {'form':
form})

@login_required(login_url=login_url)
@permission_required('Employee.FINANCE_LEAD', 'Employee.
OPERATIONS_LEAD')
def selectTeam(request):
    if request.method == 'POST':
        selection = TeamSelectionForm(request.POST or None)
        choice = selection['team'].value()
        return redirect('team/' + choice)
    else:
        selectionForm = TeamSelectionForm
        return render(request, 'selectteam.html', {'form':
selectionForm})

@login_required(login_url=login_url)
@permission_required('Employee.FINANCE_LEAD', 'Employee.
OPERATIONS_LEAD')
def viewTeam(request, id):
    team_object = Team.objects.get(TeamId=id)
    team = [team_object.TeamId, team_object.Name, team_object.
Lead]
```

```
958     return render(request, 'Team.html', {'team': team})
959
960 @login_required(login_url=login_url)
961 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD')
962 def addTeam(request):
963     if request.method == 'POST':
964         form = TeamForm(request.POST)
965         if form.is_valid():
966             id = form['TeamId'].value()
967             form.save()
968             messages.success(request, 'Team Added
    Successfully')
969             return redirect("/team/" + id)
970     else:
971         form = TeamForm()
972         return render(request, 'addTeam.html', {'form': form
    })
973
974 @login_required(login_url=login_url)
975 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD')
976 def updateTeam(request, id):
977     if request.method == 'POST':
978         form = TeamForm(request.POST, instance=Team.objects.
    get(TeamId=id))
979         if form.is_valid():
980             form.save()
981             messages.success(request, 'Team updated
    successfully!')
982             return redirect('/team/' + id)
983     else:
984         form = TeamForm(instance=Team.objects.get(TeamId=id))
985         return render(request, 'updateTeam.html', {'form':
    form})
986
987 @login_required(login_url=login_url)
988 @permission_required('Employee.FINANCE_LEAD', 'Employee.
    OPERATIONS_LEAD')
989 def updateTeamtoProj(request, id):
990     teams = Team.objects.all().order_by('TeamId')
991     if request.method == 'POST':
992         for t in teams:
993             employees = Team_Employee.objects.filter(TeamId=t
    .TeamId)
994             # add team and team's employees to project
995             if request.POST.get(t.TeamId, False) and
    Project_Team.objects.filter(TeamId=t.TeamId, IndexNo=id).
```

```
 995 exists() == False:
 996                 Project_Team.objects.create(IndexNo=id,TeamId
     =t.TeamId)
 997                 for e in employees:
 998                     Project_Employee.objects.get_or_create(
     IndexNo=id,GID=e.GID)
 999             # remove team and team's employees from project
1000             elif request.POST.get(t.TeamId, False) == False
     and Project_Team.objects.filter(TeamId=t.TeamId,IndexNo=id).
     exists():
1001                 Project_Team.objects.get(TeamId=t.TeamId,
     IndexNo=id).delete()
1002                 for e in employees:
1003                     if Project_Employee.objects.filter(
     IndexNo=id,GID=e.GID).exists():
1004                         Project_Employee.objects.filter(
     IndexNo=id, GID=e.GID).delete()
1005         return redirect("/project/" + id)
1006     else:
1007         form = ['/teamProject/' + id, 'Teams', 'Project']
1008         for t in teams: # create list of all teams
1009             if Project_Team.objects.filter(TeamId=t.TeamId,
     IndexNo=id).exists():
1010                 form.append([t.TeamId, t.Name, 1])
1011             else:
1012                 form.append([t.TeamId, t.Name, 0])
1013         return render(request, 'eptChange.html', {'form':
     form})
1014
1015 @login_required(login_url=login_url)
1016 @permission_required('Employee.FINANCE_LEAD', 'Employee.
     OPERATIONS_LEAD')
1017 def updateProjToTeam(request, id):
1018     projects = Project.objects.all().order_by('IndexNo').
     exclude(Note="Subaccount")
1019     if request.method == 'POST':
1020         employees = Team_Employee.objects.filter(TeamId=id)
1021         for p in projects:
1022             if request.POST.get(p.IndexNo, False) and
     Project_Team.objects.filter(TeamId=id, IndexNo=p.IndexNo).
     exists() == False:
1023                 Project_Team.objects.create(TeamId=id,
     IndexNo=p.IndexNo)
1024                 for e in employees:
1025                     Project_Employee.objects.get_or_create(
     IndexNo=p.IndexNo, GID=e.GID)
1026             elif request.POST.get(p.IndexNo, False) == False
     and Project_Team.objects.filter(TeamId=id, IndexNo=p.IndexNo
```

```
1026 ).exists():
1027                 Project_Team.objects.get(TeamId=id, IndexNo=p
     .IndexNo).delete()
1028                 for e in employees:
1029                     if Project_Employee.objects.filter(
     IndexNo=p.IndexNo, GID=e.GID).exists():
1030                         Project_Employee.objects.filter(
     IndexNo=p.IndexNo, GID=e.GID).delete()
1031         return redirect("/team/" + id)
1032     else:
1033         form = ['/projectTeam/' + id, 'Projects', 'Team']
1034         for p in projects:
1035             if Project_Team.objects.filter(TeamId=id, IndexNo
     =p.IndexNo).exists():
1036                 form.append([p.IndexNo, p.Title, 1])
1037             else:
1038                 form.append([p.IndexNo, p.Title, 0])
1039         return render(request, 'eptChange.html', {'form':
     form})
1040
1041 @login_required(login_url=login_url)
1042 @permission_required('Employee.FINANCE_LEAD', 'Employee.
     OPERATIONS_LEAD')
1043 def updateTeamtoEmp(request, id):
1044     teams = Team.objects.all().order_by('TeamId')
1045     if request.method == 'POST':
1046         for t in teams:
1047             projects = Project_Team.objects.filter(TeamId=t.
     TeamId)
1048             # add employee to team
1049             if request.POST.get(t.TeamId, False) and
     Team_Employee.objects.filter(TeamId=t.TeamId, GID=id).exists
     () == False:
1050                 Team_Employee.objects.create(GID=id, TeamId=t
     .TeamId)
1051                 for p in projects:
1052                     Project_Employee.objects.get_or_create(
     IndexNo=p.IndexNo, GID=id)
1053             # remove employee from team
1054             elif request.POST.get(t.TeamId, False) == False
     and Team_Employee.objects.filter(TeamId=t.TeamId, GID=id).
     exists():
1055                 Team_Employee.objects.get(TeamId=t.TeamId,
     GID=id).delete()
1056                 for p in projects:
1057                     Project_Employee.objects.filter(IndexNo=p
     .IndexNo, GID=id).delete()
1058         return redirect("/employee/" + id)
```

```
1059     else:
1060         form = ['/teamEmployee/' + id, 'Teams', 'Employee']
1061         for t in teams: # create list of all teams
1062             if Team_Employee.objects.filter(TeamId=t.TeamId,
     GID=id).exists():
1063                 form.append([t.TeamId, t.Name, 1])
1064             else:
1065                 form.append([t.TeamId, t.Name, 0])
1066         return render(request, 'eptChange.html', {'form':
     form})
1067
1068
1069 @login_required(login_url=login_url)
1070 @permission_required('Employee.FINANCE_LEAD', 'Employee.
     OPERATIONS_LEAD')
1071 def updateEmptoTeam(request, id):
1072     employees = Employee.objects.all().order_by('FirstName','
     LastName')
1073     if request.method == 'POST':
1074         projects = Project_Team.objects.filter(TeamId=id)
1075         for e in employees:
1076             # add employee to team and team's projects
1077             if request.POST.get(e.GID, False) and
     Team_Employee.objects.filter(TeamId=id,GID=e.GID).exists
     () == False:
1078                 Team_Employee.objects.create(TeamId=id,GID=e.
     GID)
1079                 for p in projects:
1080                     Project_Employee.objects.get_or_create(
     IndexNo=p.IndexNo, GID=id)
1081             # add employee from team and team's projects
1082             elif request.POST.get(e.GID, False) == False and
     Team_Employee.objects.filter(TeamId=id,GID=e.GID).exists():
1083                 Team_Employee.objects.get(TeamId=id,GID=e.GID
     ).delete()
1084                 for p in projects:
1085                     Project_Employee.objects.filter(IndexNo=p
     .IndexNo, GID=id).delete()
1086         return redirect("/team/" + id)
1087     else:
1088         form = ['/employeeTeam/' + id, 'Employees', 'Team']
1089         for e in employees: # create list of all employees
1090             if Team_Employee.objects.filter(TeamId=id,GID=e.
     GID).exists():
1091                 form.append([e.GID, e.FirstName+" "+e.
     LastName, 1])
1092             else:
1093                 form.append([e.GID, e.FirstName+" "+e.
```

```
1093 LastName, 0])
1094         return render(request, 'eptChange.html', {'form':
     form})
1095
1096
1097 @login_required(login_url=login_url)
1098 @permission_required('Employee.FINANCE_LEAD', 'Employee.
     OPERATIONS_LEAD')
1099 def updateEmptoProj(request, id):
1100     employes = Employee.objects.all().order_by('FirstName', '
     LastName')
1101     if request.method == 'POST':
1102         for e in employes:
1103             # add employee to project
1104             if request.POST.get(e.GID, False) and
     Project_Employee.objects.filter(IndexNo=id, GID=e.GID).exists
     () == False:
1105                 Project_Employee.objects.create(IndexNo=id,
     GID=e.GID)
1106             # remove employee from project
1107             elif request.POST.get(e.GID, False) == False and
     Project_Employee.objects.filter(IndexNo=id,GID=e.GID).exists
     ():
1108                 Project_Employee.objects.get(IndexNo=id, GID=
     e.GID).delete()
1109         return redirect("/project/" + id)
1110     else:
1111         form = ['/employeeProject/' + id, 'Employees', '
     Project']
1112         for e in employes: # create list of all employees
1113             if Project_Employee.objects.filter(IndexNo=id,
     GID=e.GID).exists():
1114                 form.append([e.GID, e.FirstName + " " + e.
     LastName, 1])
1115             else:
1116                 form.append([e.GID, e.FirstName + " " + e.
     LastName, 0])
1117         return render(request, 'eptChange.html', {'form':
     form})
1118
1119 @login_required(login_url=login_url)
1120 @permission_required('Employee.FINANCE_LEAD', 'Employee.
     OPERATIONS_LEAD')
1121 def updateProjtoEmp(request, id):
1122     projects = Project.objects.all().order_by('IndexNo').
     exclude(Note="Subaccount")
1123     if request.method == 'POST':
1124         for p in projects:
```

```
1125             # add project to employee
1126             if request.POST.get(p.IndexNo, False) and
     Project_Employee.objects.filter(IndexNo=p.IndexNo,GID=id).
     exists() == False:
1127                 Project_Employee.objects.create(IndexNo=p.
     IndexNo,GID=id)
1128             # remove project from employee
1129             elif request.POST.get(p.IndexNo, False) == False
     and Project_Employee.objects.filter(IndexNo=p.IndexNo,GID=id
     ).exists():
1130                 Project_Employee.objects.get(IndexNo=p.
     IndexNo,GID=id).delete()
1131         return redirect("/employee/" + id)
1132     else:
1133         form = ['/projectEmployee/' + id, 'Projects', '
     Employee']
1134         for p in projects: # create list of al projects
1135             if Project_Employee.objects.filter(IndexNo=p.
     IndexNo,GID=id).exists():
1136                 form.append([p.IndexNo, p.Title, 1])
1137             else:
1138                 form.append([p.IndexNo, p.Title, 0])
1139         return render(request, 'eptChange.html', {'form':
     form})
1140
1141
```

```
"""
Django settings for miltech project.

Generated by 'django-admin startproject' using Django 4.1.5.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.1/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-10n&9=g)vequl5m%u+bouu(rx@=puw*k38q&-65^ymc=5+poue'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    #'django_keycloak.apps.KeycloakAppConfig',
    'main',
]

```

```
44 MIDDLEWARE = [
45     'django.middleware.security.SecurityMiddleware',
46     'django.contrib.sessions.middleware.SessionMiddleware',
47     'django.middleware.common.CommonMiddleware',
48     'django.middleware.csrf.CsrfViewMiddleware',
49     'django.contrib.auth.middleware.AuthenticationMiddleware',
50     'django.contrib.messages.middleware.MessageMiddleware',
51     'django.middleware.clickjacking.XFrameOptionsMiddleware',
52     #'django_keycloak.middleware.BaseKeycloakMiddleware',
53 ]
54
55 ROOT_URLCONF = 'miltech.urls'
56
57 TEMPLATES = [
58     {
59         'BACKEND': 'django.template.backends.django.
   DjangoTemplates',
60         'DIRS': [BASE_DIR / 'main/migrations/../main/templates'
   ]
61         ,
62         'APP_DIRS': True,
63         'OPTIONS': {
64             'context_processors': [
65                 'django.template.context_processors.debug',
66                 'django.template.context_processors.request',
67                 'django.contrib.auth.context_processors.auth',
68                 'django.contrib.messages.context_processors.
   messages',
69             ],
70         },
71     },
72 ]
73
74 WSGI_APPLICATION = 'miltech.wsgi.application'
75
76
77 # Database
78 # https://docs.djangoproject.com/en/4.1/ref/settings/#databases
79
80 DATABASES = {
81     'default': {
82         'ENGINE': 'django.db.backends.sqlite3',
83         'NAME': BASE_DIR / 'db.sqlite3',
84     }
85 }
86
87
88 # Password validation
```

```
 89 # https://docs.djangoproject.com/en/4.1/ref/settings/#auth-
    password-validators
 90
 91 AUTH_USER_MODEL = "main.Employee"
 92
 93 AUTH_PASSWORD_VALIDATORS = [
 94     {
 95         'NAME': 'django.contrib.auth.password_validation.
    UserAttributeSimilarityValidator',
 96     },
 97     {
 98         'NAME': 'django.contrib.auth.password_validation.
    MinimumLengthValidator',
 99     },
100     {
101         'NAME': 'django.contrib.auth.password_validation.
    CommonPasswordValidator',
102     },
103     {
104         'NAME': 'django.contrib.auth.password_validation.
    NumericPasswordValidator',
105     },
106 ]
107
108 # AUTHENTICATION_BACKENDS = [
109 #     'django_keycloak.auth.backends.
    KeycloakAuthorizationCodeBackend',
110 # ]
111 # LOGIN_URL = 'keycloak_login'
112 # KEYCLOAK_OIDC_PROFILE_MODEL = 'django_keycloak.
    OpenIdConnectProfile'
113
114 # Internationalization
115 # https://docs.djangoproject.com/en/4.1/topics/i18n/
116
117 LANGUAGE_CODE = 'en-us'
118
119 TIME_ZONE = 'UTC'
120
121 USE_I18N = True
122
123 USE_TZ = True
124
125
126 # Static files (CSS, JavaScript, Images)
127 # https://docs.djangoproject.com/en/4.1/howto/static-files/
128
129 STATIC_URL = 'static/'
```

```
130
131 # Default primary key field type
132 # https://docs.djangoproject.com/en/4.1/ref/settings/#default-
    auto-field
133
134 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
135
```

References:

"SDLC - Iterative Model." *Tutorials Point*, www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm.

"Django." *Django Project*, DjangoProject, www.djangoproject.com/.

Computer Security Division, Information Technology Laboratory. "NIST Policy on Hash Functions - Hash Functions: CSRC." *CSRC*, NIST, 22 June 2020, https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions.