Software Design Document
for
PDGo

Version 1.0 approved

Prepared by Alberto Barboza, Mina Mekhaiel, Kevin Ornelas, Fernando Perez, Elizabeth Silvestre, Elias Schablowski, Qian Wang, Yin Win, Tommy Youn, Andrew Zou

Santa Barbara Public Defender / Deepak Budwani, Brent Modell, Mark Perez

Dec 8, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Initial draft	Dec 5, 2023	Initial draft of documentation	1.0
Update doc	Dec 8, 2023	Update Design, UI, Diagrams, System Design, Glossary	1.1

1. Introduction

1.1 Purpose

The purpose of this document is to define the software design of PDGo.

1.2 Document Conventions

The entire documentation will be written in Times New Roman font at size 12 for descriptions, size 14 and bolded for minor headings, and size 20 and bolded for major headings.

1.3 Intended Audience and Reading Suggestions

This document is primarily intended for the following audiences:

- PDGo Developers
- PDGo Project Liaisons
- PDGo Documentation writers

The document could also be intended for the users of the app, who are interested in how the app is designed as well as a better understanding of how to navigate the app.

1.4 System Overview

The system is a management application that is built using Microsoft's PowerApps in order to automate many filing systems within the Santa Barbara Public Defender's office. The software is primarily designed for the mobile environment and will be utilized on small devices for quick and easy access to department-wide information. This information includes, but is not limited to, time off request submissions and statuses, user profiles, attorney training videos, Microsoft 365 Calendars, and facility requests. The intended use of the application is for users to manage their internal resources on a one-in-all application and receive relevant notifications and updates on internal forms. The basic design involves combining Microsoft PowerApps, SharePoint, and Automate in order to create a standard frontend and backend interface for the application.

2. Design Considerations

2.1 Assumptions and Dependencies

Assumptions:

- PDGo will run on standard office computers and mobile devices used within the Santa Barbara Public Defender's Information Technology department.
- Users have access to the internet while using the application to be able to use all of its features.

Dependencies:

- Power Apps
- Microsoft SharePoint
- Power Automate
- Internet Browser

2.2 General Constraints

- Stable Internet connection
- Each user would need access to a valid SBPD account with the appropriate PowerApps licensing

2.3 Goals and Guidelines

Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system's software. For each such goal or guideline, unless it is implicitly obvious, describe the reason for its desirability. Feel free to state and describe each goal in its own subsubsection if you wish. Such goals might be:

- The KISS principle ("Keep it simple stupid!")
- The Software has a mandatory delivery date that must be met (end of the cd3337 class)
- Emphasis on speed versus memory use
- The product should work, look, or "feel" like an existing product

The goal of the product is to create a fully functional, accessible, responsive, and robust administrative system that allows SPBD employees to manage their Time Off Requests, Trainings, and Facility Requests with ease in a mobile environment. The product should work, look, or "feel" like Workday.

2.4 Development Methods

Briefly describe the method or approach used for this software design. If one or more formal/published methods were adopted or adapted, then include a reference to a more detailed description of these methods. If several methods were seriously considered, then each such method should be mentioned, along with a brief explanation of why all or part of it was used or not used.

These would be things such as the ‘Water Fall Development’ methods, ‘Agile Development’, ‘Unplanned Mad Scramble Development’, or other development models and variations. Describe how these were applied in the case of your project.

3. Architectural Strategies

Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Such decisions might concern (but are not limited to) things like the following:

- Use of a particular type of product (programming language, database, library, etc. ...)
- Reuse of existing software components to implement various parts/features of the system
- Future plans for extending or enhancing the software
- User interface paradigms (or system input and output models)
- Hardware and/or software interface paradigms
- Error detection and recovery
- Memory management policies
- External databases and/or data storage management and persistence
- Distributed data or control over a network
- Generalized approaches to control
- Concurrency and synchronization
- Communication mechanisms
- Management of other resources

Each significant strategy employed should probably be discussed in its own subsection. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose).

PowerApps Canvas was used to create PowerApps. Canvas applications follow a UI-first approach that means that many actions are embedded alongside their UI components which trigger them. This approach was used as it allows greater flexibility in adding actions and data displays than a model-driven application.

PowerApps was chosen due to the existing infrastructure at SBPD to support PowerApps and related functions. Furthermore, as all SBPD employees have their own Microsoft 365 account, access can easily be controlled.

PDGo follows a mobile-first design. This allows SBPD employees to access PDGo on their county issued devices, along with greater access management, and the ability to scan information.

PowerAutomate was chosen for some complex and scheduled tasks, as it runs independently of any client devices, and can therefore have higher availability along with higher responsiveness for heavy workloads.

PDGo primarily uses push notifications as this allows easier deep linking into the specific screen to see where the notification originated from.

PDGo was designed to have all companion information, i.e. information linked to restricted fields in other applications, to sharepoint lists with dropdowns to ensure that data entry is valid.

PDGo was designed with tracing to allow easier debugging both during development, monitoring, and maintenance as errors can be re-engineered more easily.

The user interface of PDGo was designed to be themable to allow easier adjustments when SBPD updates their colors.

PDGo uses SharePoint lists to store information due to high availability and easy integration into PowerApps. Furthermore, SharePoint is one of the few connectors allowed for government entities.

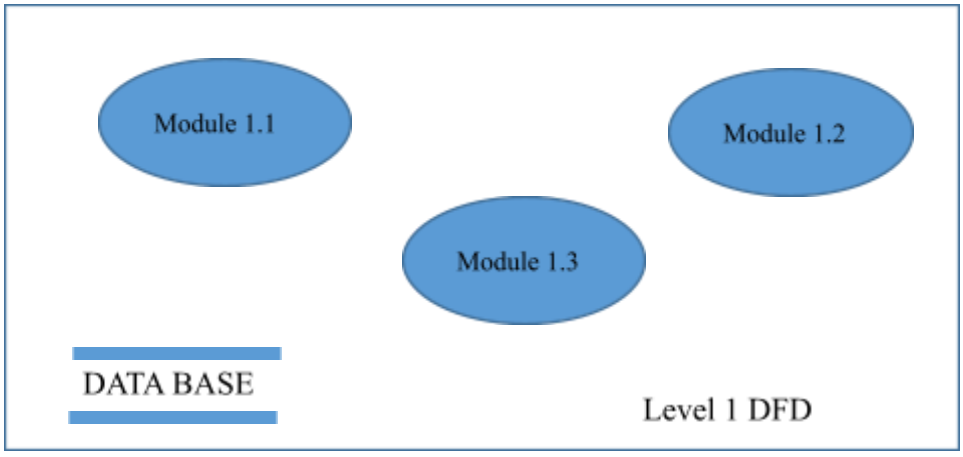
4. System Architecture

This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.

This is where the level 0 DFD will probably work best.

At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its modules/components/subsystems (identifying each top-level modules/component/subsystem and the roles/responsibilities assigned to it).

Each subsection (i.e. “4.1.3 The ABC Module”) of this section will refer to or contain a detailed description of a system software component.



Level 1 Data Flow Diagrams (DFD) and Control Flow Diagrams (CFD) should probably go here.

Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Feel free to make use of design patterns, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them. Diagrams that describe a particular component or subsystem in detail should be included within the particular subsection that describes that component or subsystem.

5. Policies and Tactics

Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose). Such decisions might concern (but are not limited to) things like the following (Must include 5.1, 5.2, and 5.3. The rest of these categories or custom ones can be added as needed.):

5.1 Choice of which specific products used

(IDE, compiler, interpreter, database, library, etc. ...)

5.2 Plans for ensuring requirements traceability

…Describe…

5.3 Plans for testing the software

…Describe…

5.# Engineering trade-offs

…Describe…

5.# Coding guidelines and conventions

…Describe…

5.# The protocol of one or more subsystems, modules, or subroutines

…Describe…

5.# The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality

…Describe…

5.# Plans for maintaining the software

…Describe…

5.# Interfaces for end-users, software, hardware, and communications

…Describe…

5.# Hierarchical organization of the source code into its physical components (files and directories).

…Describe…

5.# How to build and/or generate the system's deliverables (how to compile, link, load, etc.)

…Describe…

5.# Describe tactics such as abstracting out a generic DatabaseInterface class, so that changing the database from MySQL to Oracle or PostGreSQL is simply a matter of rewriting the DatabaseInterface class.

For this particular section, it may become difficult to decide whether a particular policy or set of tactics should be discussed in this section, or in the System Architecture section, or in the Detailed System Design section for the appropriate component. You will have to use your own "best" judgement to decide this. There will usually be some global policies and tactics that should be discussed here, but decisions about interfaces, algorithms, and/or data structures might be more appropriately discussed in the same (sub) section as its corresponding software component in one of these other sections.

6. Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

This is where Level 2 (or lower) DFD's will go. If there are any additional detailed component diagrams, models, user flow diagrams or flowcharts they may be included here.

6.x Name of Component (Module)

6.x.1 Responsibilities

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

6.x.2 Constraints

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

6.x.3 Composition

A description of the use and meaning of the subcomponents that are a part of this component.

6.x.4 Uses/Interactions

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclass's, and metaclasses.

6.x.5 Resources

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

6.x.6 Interface/Exports

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Time-Off Request Notification System

Responsibilities

1. Notify Supervisors of new Time Off Requests
2. Monitor Stale Time Off requests and notify Supervisors periodically

Constraints

Performance Constraints:

The performance of the Time-Off Request Notification System has several constraints:

- Internet connection speed
- Microsoft PowerApps' service latency.

Security Constraints:

In terms of security, the Time-Off Request Notification System faces several constraints:

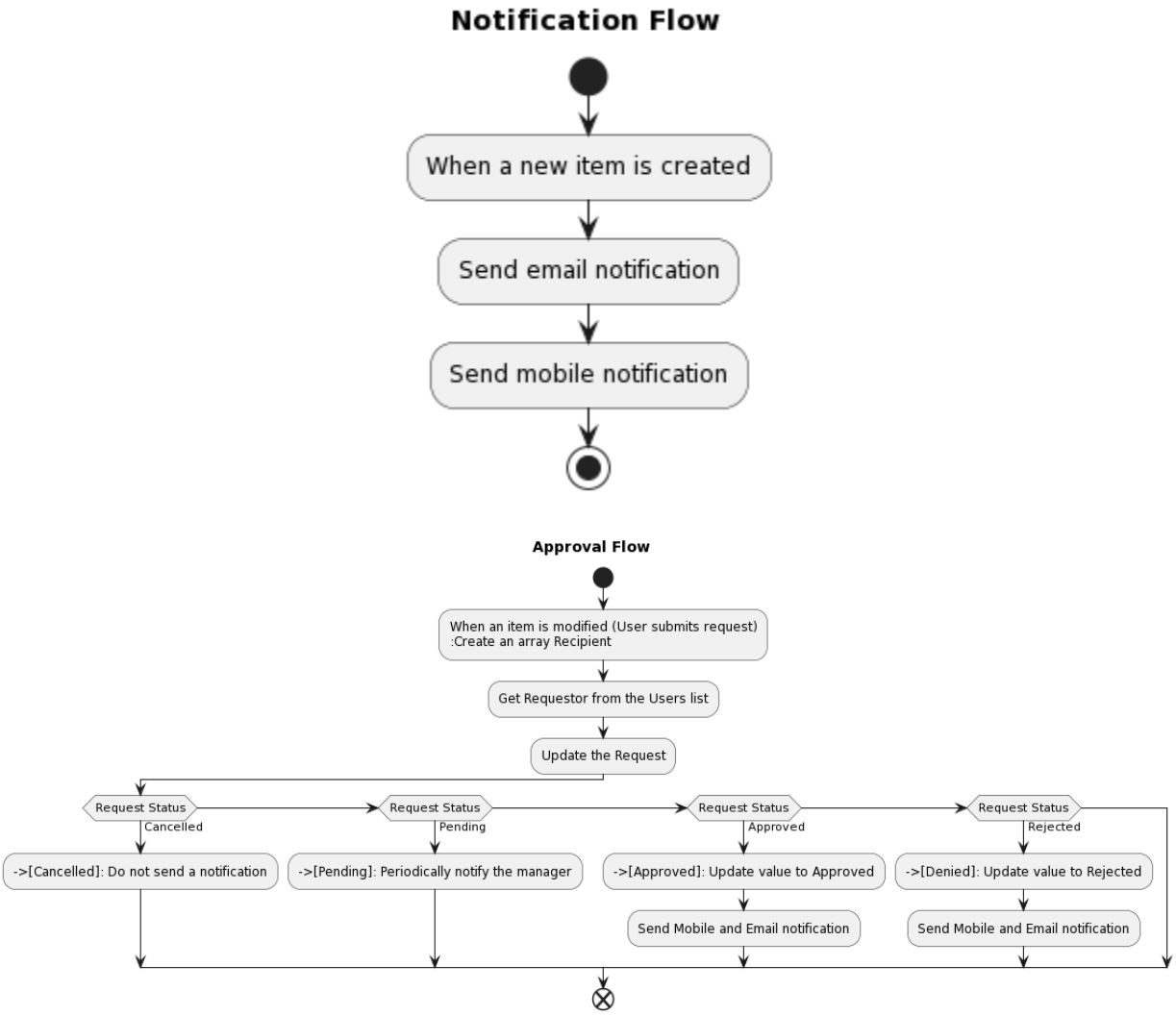
- Reliability of security audits and vulnerability assessments.
- Restrictions of user and data access to authorized personnel.

Reliability Constraints

The reliability of the Time-Off Request Notification System is dependent on several factors:

- Uptime of the Microsoft PowerApps service
- An active carrier device is available
- Automated backups

Composition



Uses/Interactions

The Time-Off Request Notification System is divided into two flows: **Notification Flow** and **Approval Flow**.
The Notification Flow operates as follows:

1. The addition of a Time-Off request triggers subsequent actions.
2. The email notification is triggered
3. The mobile notification is triggered.

The Approval Flow operates within the following procedure:

1. The modification of a request will trigger the search for a Requestor from the user.
2. The request will be updated according to its status as decided by a switch, options are:
    1. Cancelled: A notification is not sent
    2. Pending: Notify the manager in a periodic manner

3. Approved: Update the value to Approved and trigger Notification Flow.
4. Rejected: Update the value to 'Rejected' and trigger Notification Flow.

Interface/Exports

The application is aimed at providing specific interfaces and exports for internal use by the Santa Barbara Public Defender's Office. These interfaces are only to be used within the organization and are not authorized for external use.

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code.

7. Detailed Lower level Component Design

Other lower-level Classes, components, subcomponents, and assorted support files are to be described here. You should cover the reason that each class exists (i.e. its role in its package; for complex cases, refer to a detailed component view.) Use numbered subsections below (i.e. “7.1.3 The ABC Package”.) Note that there isn't necessarily a one-to-one correspondence between packages and components.

7.x Name of Class or File

7.x.1 Classification

The kind of component, such as a subsystem, class, package, function, file, etc.

7.x.2 Processing Narrative (PSPEC)

A process specification (PSPEC) can be used to specify the processing details

7.x.3 Interface Description

7.x.4 Processing Detail

7.x.4.1 Design Class Hierarchy

Class inheritance: parent or child classes.

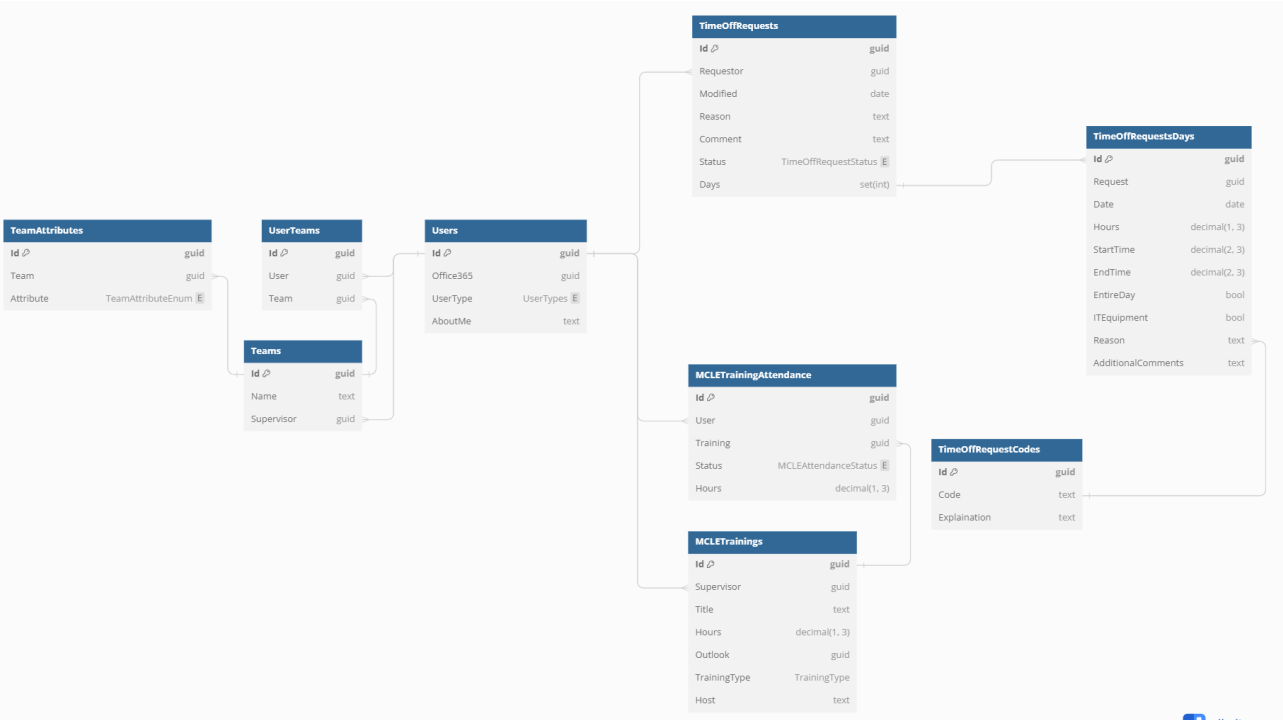
7.x.4.2 Restrictions/Limitations

7.x.4.3 Performance Issues

7.x.4.4 Design Constraints

7.x.4.5 Processing Detail For Each Operation

8. Database Design



Include details about any databases used by the software. Include tables and descriptions.

PDGo uses various Sharepoint lists as connectors to store and retrieve datasets and acts like a simple database management system. Its paid time off request, MCLE trainings, and userbase uses Sharepoint in order to access values associated to the user information associated to the page.

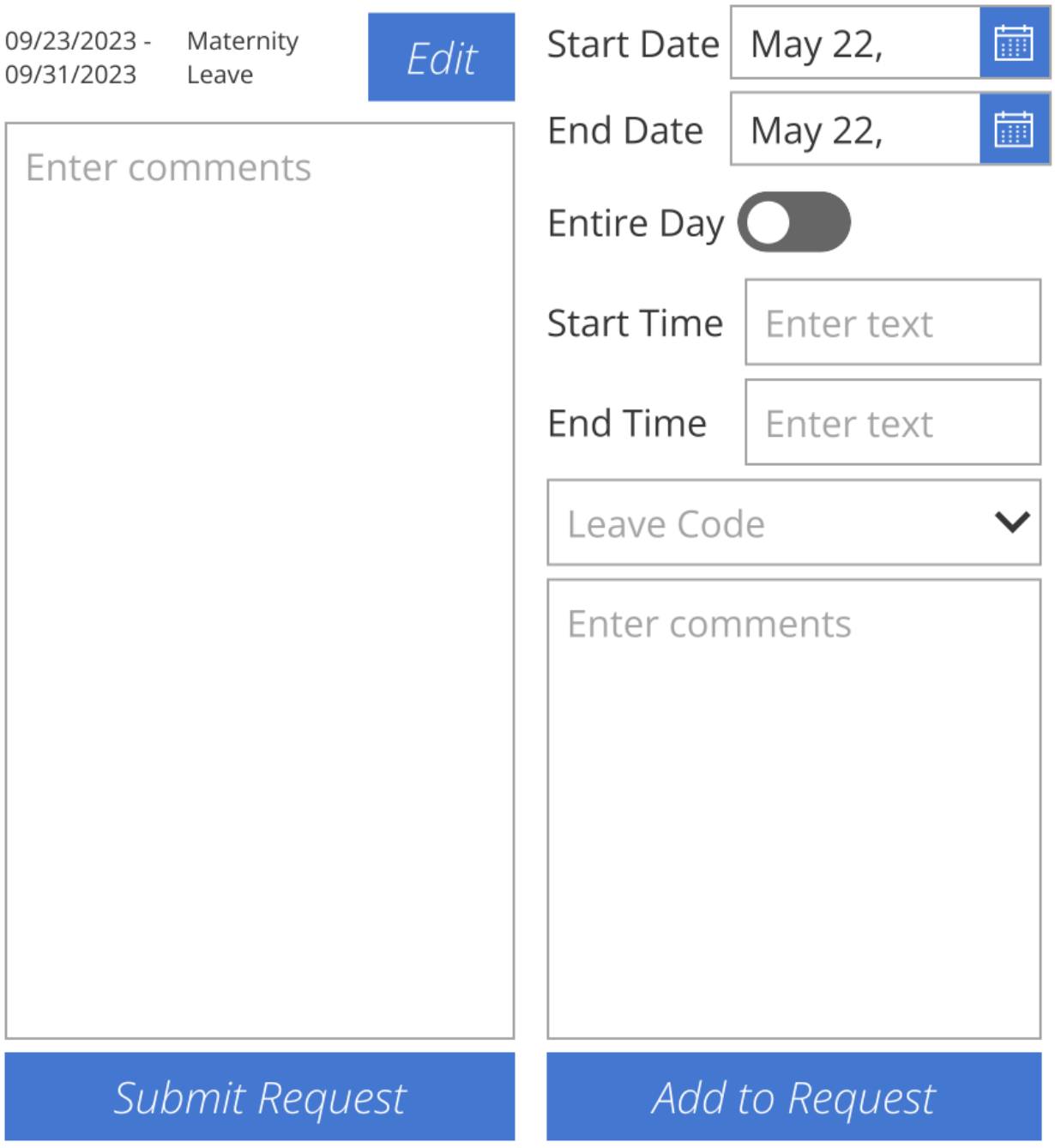
- Starting with Users, they carry personnel information regarding their position and role. This organizes how PDGo's interface will be presented as certain features can be shown or hidden. Having a user list will also filter information from the MCLE trainings and PTO request to present data that is tailored to the user based on previous entrees.
- PTO request feature stores user requests and for supervisors to decide whether to approved or deny the request. sharepoint stores all the data that users from PDGo have submitted and is changed based on the supervisors choice.
- The MCLE trainings will hold data for the user to view what trainings are available based on the positions requirements. Supervisors will be able to view their employees progress and completions of these trainings as a way to document employee experiences.

9. User Interface
Overview of User Interface

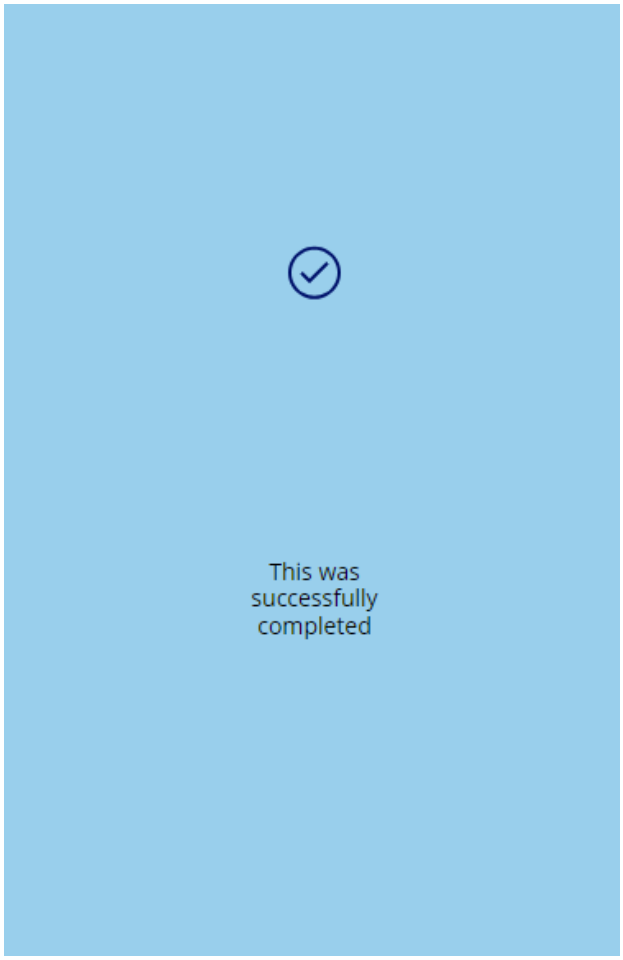
Screen Wireframes or Images

Time Off Request Screens

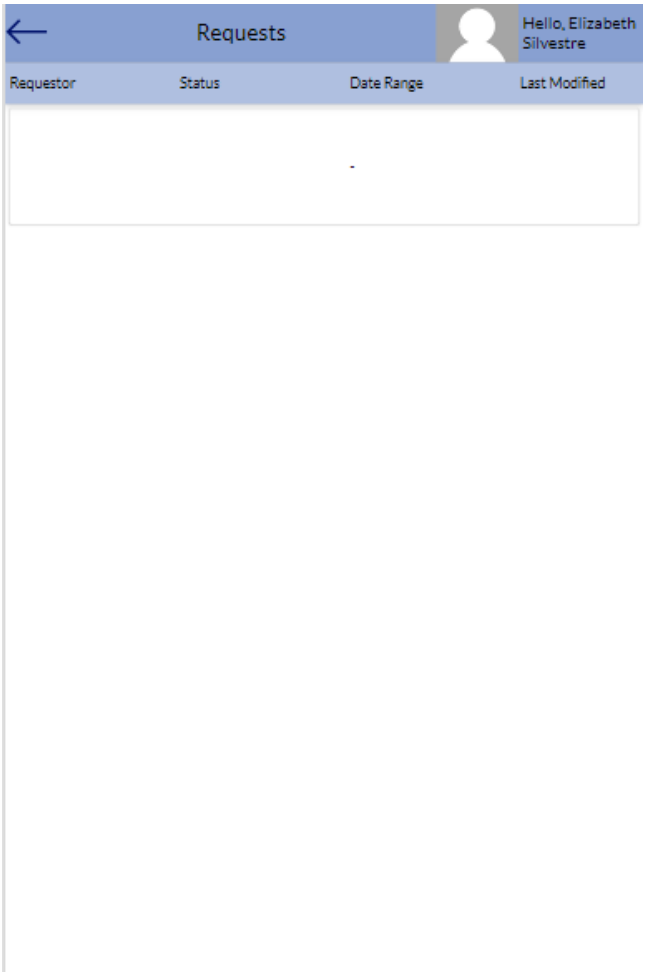
When the user selects the time-off request option from the menu, they are redirected to the screen on the left. In this screen, the user would be able to see all of the requests that they started but haven't submitted yet. For each request that appears on the screen, the user would be able to edit that request or delete the request. We made the color choice blue for "edit" and red for "delete" so that the user could tell them apart. The user can also write in the text box if they have additional information they wish to add. The second screen is what the user would see when they make a new request or if they're editing an existing request. The start and the end date are set two weeks in advance from the current date by default. If the user selects the entire day toggle, the start and end times will be deleted. The start and end times as well as the reason code are chosen by drop boxes. If the user selects "other" for the reason code, they would have to write something in the text box before adding the request. Once the user selects the "add to requests" button, the request will be added to the request screen on the left.



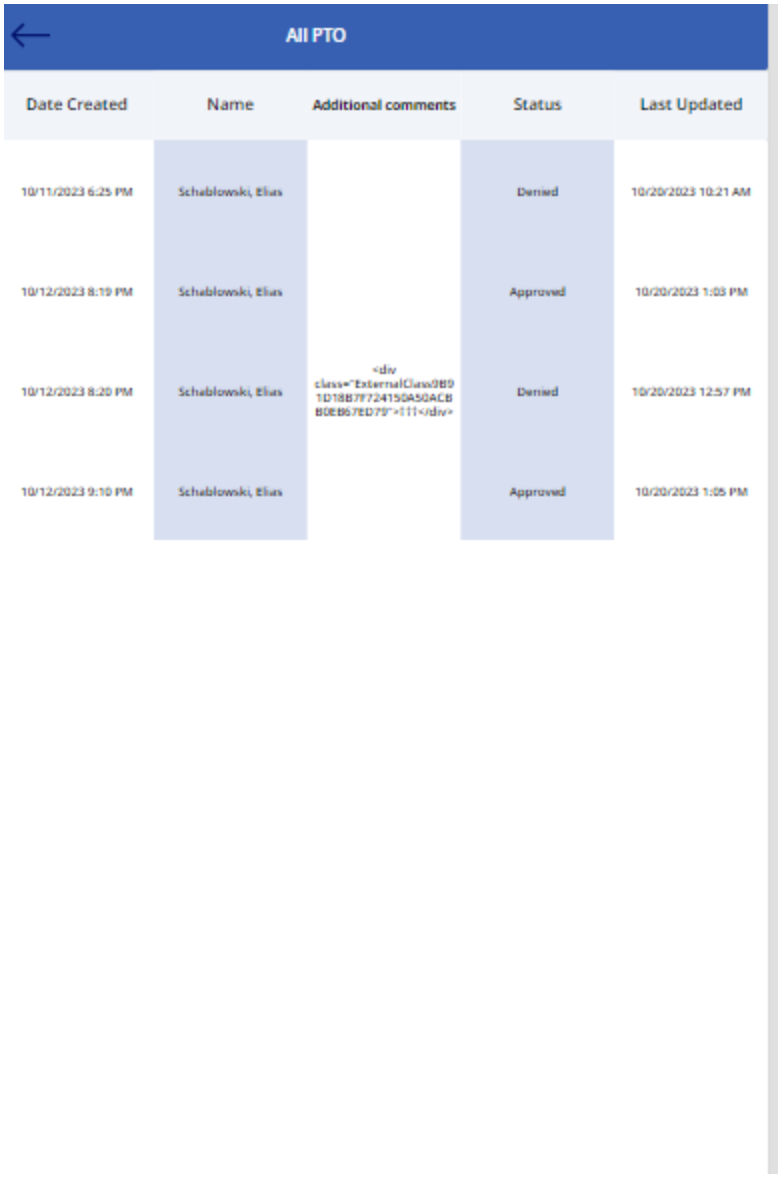
Once the user submits their request, they will be redirected to the success screen. This would let the user know that their requests have been submitted successfully. The success screen would be on the user's screen for around 3 seconds and then transition to the main menu of the app.



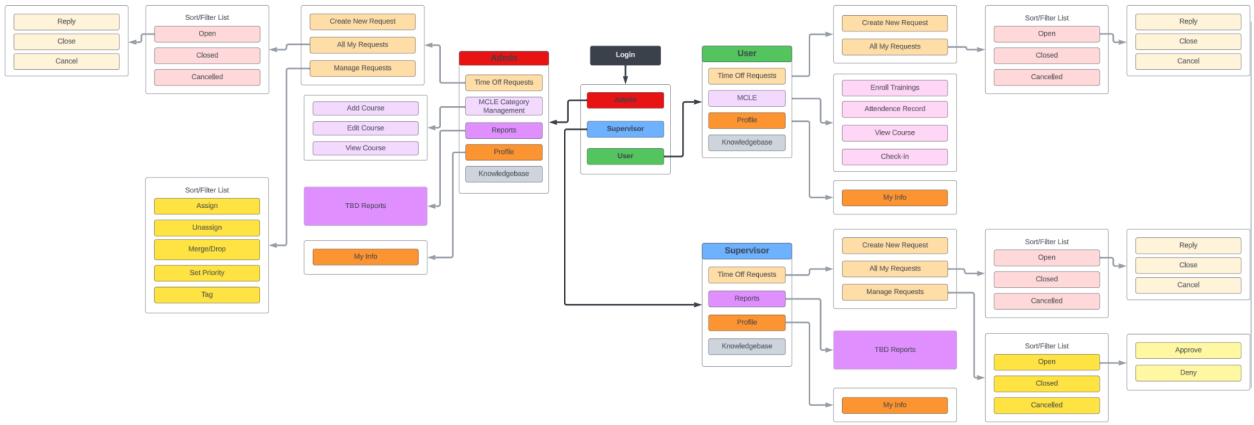
The user is also able to view all of the requests that they submitted as well as the status of those requests.



Those who are supervisors would be able to see all of the time-off requests from all employers as well as their status.


All PTO
Date Created
Name
Additional comments
Status
Last Updated
10/11/2023 6:25 PM
Schablowski, Elias
Denied
10/20/2023 10:21 AM
10/12/2023 8:19 PM
Schablowski, Elias
Approved
10/20/2023 1:03 PM
10/12/2023 8:20 PM
Schablowski, Elias
111
Denied
10/20/2023 12:57 PM
10/12/2023 9:10 PM
Schablowski, Elias
Approved
10/20/2023 1:05 PM

User Interface Flow Model



Users are categorized by three types, user, supervisor and admin. All users will see the same interface menu with a similar set of categories varied by role. Users will have access to most functions for regular use such as scheduling, receiving notifications, completing trainings and viewing their own personnel profile. Supervisors and Admins on the other hand, can be given administrative privileges that allows them to respond to user requests, along with the same user features accessible in the menu.

10. Requirements Validation and Verification

Create a table that lists each of the requirements that were specified in the SRS document for this software.
For each entry in the table list which of the Component Modules and if appropriate which UI elements and/or low level components satisfies that requirement.
For each entry describe the method for testing that the requirement has been met.

11. Glossary

An ordered list of defined terms and concepts used throughout the document. Provide definitions for any relevant terms, acronyms, and abbreviations that are necessary to understand the SDD document. This information may be listed here or in a completely separate document. If the information is not directly listed in this section provide a note that specifies where the information can be found.

Term	Definition
PDGo	The application being developed
SBPD	Santa Barbara Public Defender's Office
MCLE	Minimum Continuing Legal Education
TO	Time Off
PTO	Paid Time Off
SRS	Software Requirements Specification (this Document)
Section 508	A legal requirement to make applications accessible to people with impairments using technology
Microsoft Power Apps	A low-code environment to create apps for business applications
Microsoft Sharepoint Lists	A component of Microsoft Sharepoint

12. References

<List any other documents or Web addresses to which this SDD refers. These may include other SDD or SRS documents, user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

PDGo Software Requirements Specification: https://ascent.cysun.org/project/project/view/203, under the "References" section

Workday: https://www.workday.com/