Software Design Document
for
Digitization and Modernization of PD HelpDesk Ticketing System

Version 1.2 approved

Prepared by Nshan Kazaryan, Marie Karibyan, Kevin Trochez, Mark Perez, Brandon Estrada, Hoai Nam Cao, Gilbert Hopkins, Geovany Huerta, Biruk Mengeste

Santa Barbara Public Defender's Office / Deepak Budwani, Brent Modell, Luis Ramirez

May 5, 2023

Table of Contents..... pg 2
Revision History.....pg 3
1. Introduction.....pg 4
1.1. Purpose.....pg 4
1.2. Document Conventions……..... pg 4
1.3. Intended Audience and Reading Suggestions.....pg 4
1.4. System Overview..... pg 4
2. Design Considerations..... pg 5
2.1. Assumptions and dependencies.....pg 5
2.2. General Constraints.....pg 5
2.3. Goals and Guidelines.....pg 5
2.4. Development Methods..... pg 6
3. Architectural Strategies.....pg 7
4. System Architecture.....pg 8
5. Policies and Tactics.....pg 12
5.1. Specific Products Used..... pg 12
5.2. Requirements traceability.....pg 12
5.3. Testing the software.....pg 12
5.4. Engineering trade-offs.....pg 12
5.5. Guidelines and conventions..... pg 12
5.6. Protocols..... pg 13
5.7. Maintaining the software..... pg 13
5.8. Interfaces.....pg 13
5.9. System's deliverables..... pg 13
5.10. Abstraction.....pg 13
6. Detailed System Design.....pg 14
7. Detailed Lower level Component Design……..pg 24
7.1 SharePoint Database.....pg 24
7.2 Power Apps Controls……pg 24
8. Database Design……pg 26
8.1. SharePoint LIst…….....pg 26
8.2. Tables and Entities Overview.....pg 26
9. User Interface……pg 29
9.1. Overview of User Interface…….....pg 29
9.2. Screen Frameworks or Images..……pg 29
9.3. User Interface Flow Model……pg 32
10. Requirements Validation and Verification.....pg 35
11. Glossary.....pg 40
12. References.....pg 40

Revision History

Name	Date	Reason For Changes	Version
Initial Draft	9/08/22	Initial draft of document	1.0
Update Document	11/30/22	Filling in info for certain sections	1.1
Final Draft	4/30/23	Updating document and adding information	1.2

1. Introduction

1.1 Purpose

The purpose of this document is to describe in full detail the way we designed our project, PDHelpdesk. This document can act as a guide on how our application was implemented.

1.2 Document Conventions

All content is written in size 12 Times New Roman font. Important sections and words are in bold. Every requirement statement has their own priority.

1.3 Intended Audience and Reading Suggestions

The intended audience is the staff members of Santa Barbara Public Defender's office. The admin/tech role will be given to IT technicians of the department. The rest of the staff members will be given the user role. If any user is interested in how this application was designed or is having trouble navigating, you can read through this document for better understanding.

1.4 System Overview

Our system is a typical IT ticketing system that has many different features and capabilities built-in using Microsoft’s PowerApps framework. It is a software intended for users to be able to submit tickets for IT related issues, view their current tickets, view the knowledge base, and view system based notifications such as outages. A technician on the other hand has all these capabilities in addition to responding to tickets, assigning tags to tickets, closing out tickets, sending system wide alerts, and assigning tickets to themselves or other technicians.The basic design approach is similar to a lot of websites that have a frontend and backend database connection. The API acts as the middle man between the frontend and backend. Essentially, the API grabs the user requests/actions that happen in the frontend and updates the database with the information the user provided. For example, if a user creates a ticket, then the tickets table is updated with the respectable attributes.

2. Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

2.1 Assumptions and Dependencies

Technologies used for PD HelpDesk.

- PowerApps: This is a Microsoft framework part of the Power platform that allows developers with little technical experience to build powerful web/mobile applications. It also provides a rapid development environment to build custom apps for any business needs.
- Microsoft SharePoint: web-based collaboration and content management platform developed by Microsoft. It allows users to create and manage websites for sharing and collaborating on documents, lists, and calendars. SharePoint can also be used for business intelligence, workflow automation, and integration with other Microsoft applications such as Office and Teams. SharePoint acts as the database for PD HelpDesk
- Power Automate: Cloud-based service provided by Microsoft that allows users to create automated workflows between different applications and services without needing to write code.
- Internet Browsers: Any browser such as Microsoft Edge, Google Chrome, Firefox, etc.. will work.

The app depends on the MS PowerApps environment and users must use the application through PowerApps “play” feature. It is assumed that users have valid Santa Barbara credentials to play the application along with having Office 365 PowerApps Standard Licenses.

2.2 General Constraints

- Users need a valid SBPD account with appropriate PowerApps licensing.
- Stable internet connection.
- Any browser can be used but PowerApps is needed to launch the application. PowerApps mobile is needed for users that are using the app in a mobile environment. However, Edge is the preferred browser for the Public Defender’s environment.

2.3 Goals and Guidelines

The end product should be a fully functional ticketing system. The website also must be intuitive for the sole use by SBPD technicians and attorneys/staff members. This product should fully be able to function in a mobile environment. The product should work, look, or “feel” like ServiceNow, ZenDesk, etc.

2.4 Development Methods

We are using the Agile Development method. Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Every week, new tasks and assignments are given to team members and the team has quick every few week deadlines. This iterative approach allows us to determine where we are at on the project and allows us to handle issues as they arise. The components/features of the web application have been divided up among the group members to optimize time and efficiency.

3. Architectural Strategies

PowerApps Canvas was used to create the PD HelpDesk application. PowerApps follows a model-driven app architecture strategy. The model-driven app architecture separates the user interface and logic from the data model, allowing for more flexibility and scalability.

The app's architecture consists of the following components:

- Data source: Defines where the app's data will be stored, in this case SharePoint. The data source was originally Azure SQL. Design and migration to SharePoint was needed because of PowerApps licensing issues. (SQL connector is a premium feature)
- Model-driven app framework: The framework provides the UI components, data handling, and security features. It allows developers to design the user interface and define the app's business logic.
- User interface: The user interface includes the screens, forms, and controls that users interact with to access and manipulate data. Canvas uses prebuilt UI elements that are available through Microsoft's low code solution.The user interface also provides relevant information about a ticket's status and other relevant information.
- Security: PowerApps Canvas provides a range of security options such as data loss prevention, conditional access, and multi-factor authentication. Depending on the sensitivity of the data being stored and processed in the app, additional security measures are applied to protect data.
- Power Automate: With Power Automate, users can create workflows that perform a variety of tasks, such as sending notifications, collecting data, and synchronizing files. The workflows can be triggered by various events, such as a new email arriving in the inbox, a new record being added to a database, or a new file being uploaded to a cloud storage service. PD HelpDesk uses Power Automate to pull tickets from the database, send email notifications when there are updates on tickets, and attach files to tickets if need be. By automating repetitive tasks, Power Automate can save time and reduce errors associated with manual data entry or other tedious tasks.
- Communication mechanism: PD HelpDesk supports communication between users and system administrators/technicians. The system provides email notifications and instant messaging to ensure that users can communicate effectively with each other.
- Extensibility: PD HelpDesk is designed to accommodate future enhancements and changes. The system should be modular and scalable to enable the addition of new features and functionalities without affecting existing functionality.
- Error Detection and Recovery: PowerApps provides detailed error messages that can help developers quickly identify and resolve errors. Error messages can appear in various places within the PowerApps interface, such as the formula bar, data source view, and app screen. For example, the formula bar provides syntax highlighting, autocomplete suggestions, and error highlighting to help identify and resolve issues with formulas.

4. System Architecture

We have separated the system into 3 sections, User Base, Devices, and Power Apps. Our application was developed using Power Apps as well as Power Automate which are a part of the Power Platform, Sharepoint was also used as a database technology to store data.

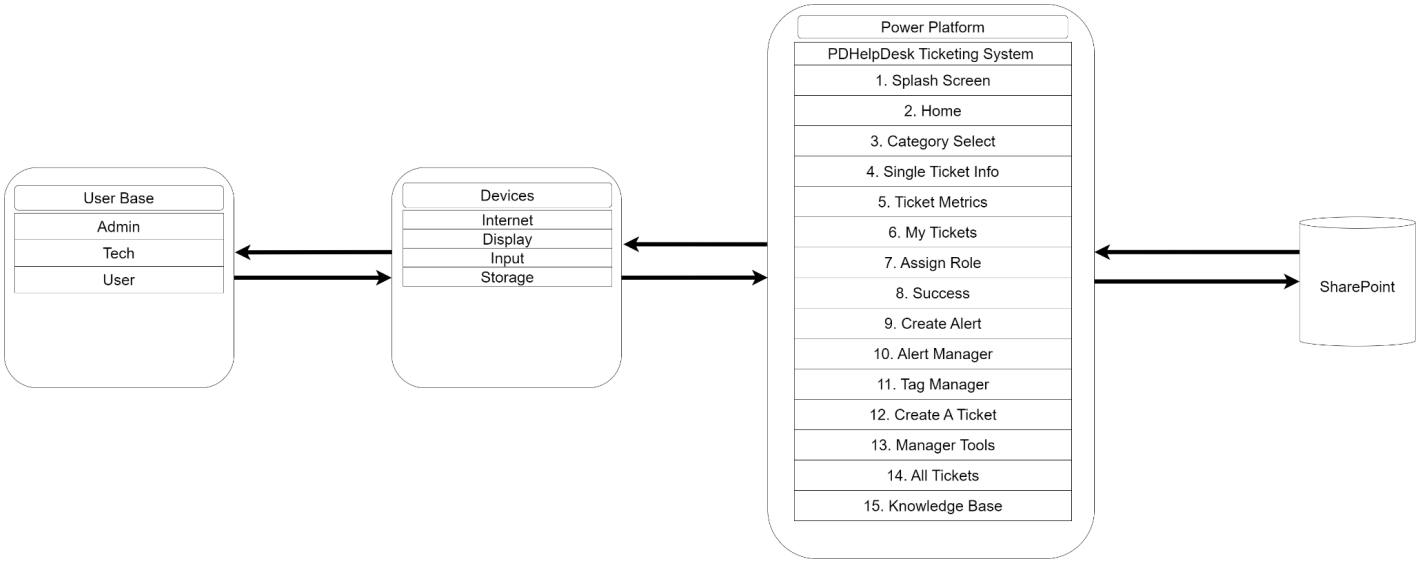


Figure 1: DFD level 0, shows the overview of the entire system.

User Base: Everyone who will use this application, and fall into one of three roles. They are to provide data to the application and will receive data based on their privilege levels.

Devices: These act as the middle men between those using the application and the application itself. Can be any device that can connect to the internet, display data, accept input in the form of touch or tactile keystrokes, and have at least a storage component to both access and store information. The application sends and receives requests from these devices in order to function.

Power Platform: This is an assortment of technologies, one of them being Power Apps, that was used to create the ticketing system. Listed are the modules that are responsible for particular tasks, such as ticket creation, dashboards, ticket management, as well as analytics. It relies on requests from devices usually in the form of data to send or receive. it interacts with Sharepoint, where it is used as a database, to store, retrieve, or update data. Examples of data interacted with are tickets, comments, and images.

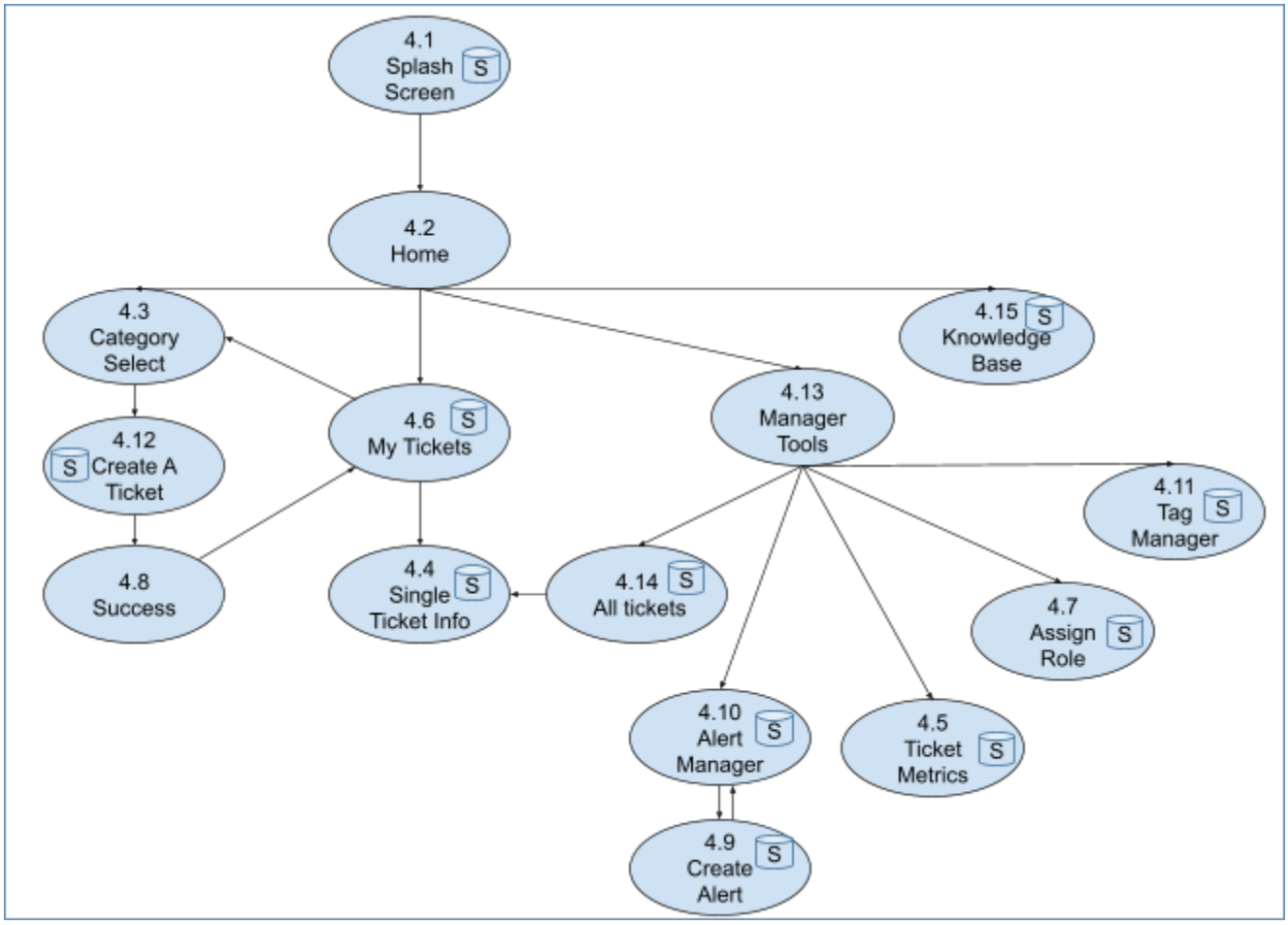


Figure 2: DFD level 1 of our system, when a module accesses Sharepoint it will contain a small database icon.

4.1 Splash Screen

This is the entry point into the application; it accesses Sharepoint to retrieve data, and is accessible once, when the application is run for the first time.

4.2 Home

Acts as a hub module, and involves itself with 4 other modules, those being: Create A Ticket, My Tickets, Manager Tools (admin/tech only), and Knowledge Base.

4.3 Category Select

This module offers a selection based on categories retrieved from Sharepoint, and is connected to Create A Ticket as it will pass data into the module.

4.4 Single Ticket Info

This module is accessed after a selection is made from the My Tickets module or All Tickets module, it accesses Sharepoint to retrieve ticket data to display, and accepts user data to store comment data.

4.5 Ticket Metrics

This module can be accessed from the Manager Tools module, and uses Sharepoint to retrieve ticket data to display simple analytics.

4.6 My Tickets

This module is accessible after the Success Module, accepts input from users to search through tickets associated with the user, Sharepoint retrieves ticket data. Can interact with the Category Select module to perform the ticket creation process anew.

4.7 Assign Role

This module is accessible from the Manager Tools module, it accepts users input to update data stored in Sharepoint.

.4.8 Success

This module is only accessible after a ticket has been created, it is associated with the Create A Ticket module, and the My Tickets Module

4.9 Create Alert

Module accessible from the Alert Manager module, it has two functions, one where data is pulled from Sharepoint to update, and another to store data when creating a new alert. Also redirects to the Alert Manager module upon function completion.

4.10 Alert Manager

Accessible from the Manager Tools module, only retrieves data from sharepoint to display, and accepts minimal user input. Can access the Create Alert module to update an existing alert, or to create a new alert.

4.11 Tag Manager

Accessible from the Manager Tools module, accepts user input to create, remove, or update tags and accesses Sharepoint to perform these actions accordingly. Displays tag data retrieved from Sharepoint as well.

4.12 Create A Ticket

Accessible after the Category Select module, receives data to prefill a template, accepts additional user input to create ticket data, and sends it to be stored in Sharepoint. It also immediately interacts with the Success module.

4.13 Manager Tools

Another module that acts as a hub, input allows for interacting with other modules , those being: Tag Manager, All Tickets, Alert Manager, Ticket Metrics, and Assign Role.

4.14 All Tickets

Accessible from the Manager Tools module, uses Sharepoint to display a table of tickets from retrieved data. A selection here will redirect to the Single Ticket Info module.

4.15 Knowledge Base

Accessible from the Home module, retrieves data from Sharepoint to display results, as well as populate popular search terms. Input used to search and sort through data stored on Sharepoint.

5. Policies and Tactics

Building a computer software can be a difficult process with many phases and decision points along the way. In this section, we'll go over several policies and strategies that can assist guide the development process and assure the success of your project.

5.1 Choice of which specific products used

5.1.1 Microsoft SharePoint

5.1.2 Microsoft Power Apps

5.1.3 Github

5.2 Plans for ensuring requirements traceability

Traceability of requirements refers to the ability to track and verify the relationships between project requirements and project deliverables. It is an important part of project management and software development since it guarantees that the final product fits all of the requirements outlined in the project plan. Here are some strategies for maintaining requirements traceability:

5.2.1 Develop a Requirements Traceability Matrix (RTM): An RTM is a document that connects project requirements to system design and implementation. We can track and verify the relationship between project requirements and project deliverables by constructing an RTM.

5.2.2 Conduct Regular Reviews: Another technique to maintain requirements traceability is to conduct regular reviews of the project plan and requirements. Reviews aid in identifying any discrepancies or inconsistencies in the project plan and ensuring that all requirements are completed.

5.2.3 Document modifications: documenting any modifications made to the project plan or requirements can help us ensure the traceability of requirements. You can trace the evolution of the project and verify that all changes are appropriately accounted for and implemented by documenting modifications.

5.3 Plans for testing the software

Begin by defining the test objectives in detail based on the software's requirements and expectations of users. Identifying the precise functionalities, features, and performance requirements that must be tested is part of this process. We used mock data to test our application’s requirements and ensured that the mock data reached the database.

5.4 Engineering trade-offs

- N/A

5.5 Coding guidelines and conventions

We used consistent naming conventions to make the code easier to read and understand. Variables, functions, and classes are named in a way that appropriately reflects their role. Consistent indentation and formatting make the code easier to read and understand.

5.6 Protocols

- N/A

5.7 Plans for maintaining the software

- Performance Optimization: As the software is used, system functionality can deteriorate over time. Performance testing and optimizing on a regular basis can assist keep the system efficient and responsive.
- Security patches: Software security flaws can result in data breaches, system outages, and other issues. By resolving vulnerabilities as they develop, regular security upgrades can help prevent these concerns.

Plans to maintain the software will be left to the Public Defender's IT department.

5.8 Interfaces for end-users, software, hardware, and communications

Staff members, IT technicians, and administrators work with the PDHelpDesk interface. The user interface is intuitive, user-friendly, and visually appealing. The interface should also provide feedback to the user and assist them through the system's many operations.

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

The app is launched through the PowerApps environment or through the app's "web link" provided to users when they click on the app's details.

- Understand the Requirements: Understanding the requirements is the initial step in building and generating system deliverables. This includes determining the system's purpose, the users and their demands, as well as the system's functional and non-functional requirements.
- Design the System: Once the requirements have been determined, the system must be designed. This entails developing a thorough architecture, defining the components and modules, and defining the interfaces between them.
- Develop the System: The development phase begins once the system design is finalized. This includes coding the system's many components and modules, integrating the components, and testing the system to ensure that it fits the requirements.
- Deploy the system: After the system and its deliverables have been tested and certified, the system is ready to be deployed. This includes installing software, configuring the system, and training users on how to utilize the system successfully.

5.10 Abstraction

- N/A

6. Detailed System Design

6.1 Splash Screen

6.1.1 Responsibilities

Splash Screen is responsible for authenticating the user and initializing any global variables. Once the user is authenticated, they are redirected to the Home Screen.

6.1.2 Constraints

All users are assumed to be part of SBPD and have the proper credentials and licensing to use this application. If the user does not have a valid user role, they are redirected to User Not Valid Screen.

6.1.3 Composition

N/A

6.1.4 Uses/Interactions

Initializes the user information, collections, and global variables used on the other screens within the application.

6.1.5 Resources

Splash screen needs access to the SharePoint database to determine the user's role and initialize the user's information and global variables.

6.1.6 Interface/Exports

N/A

6.2 Home Screen

6.2.1 Responsibilities

The Home Screen serves as the landing page of PD HelpDesk and is responsible for navigating to other screens. Home displays two different views based on whether the user has a staff or an admin/tech role.

6.2.2 Constraints

All users will need to be authenticated beforehand, in the ‘Splash’ screen, in order to have access to this screen.

6.2.3 Composition

N/A

6.2.4 Uses/Interactions

Home allows all users to navigate to other screens within the application. Certain options will be omitted based on the user's role.

6.2.5 Resources

N/A

6.2.6 Interface/Exports

N/A

6.3 Category Select

6.3.1 Responsibilities

Category Select is responsible for selecting a main ticket category. This screen displays two different views based on whether the user has a staff or an admin/tech role.

6.3.2 Constraints

N/A

6.3.3 Composition

N/A

6.3.4 Uses/Interactions

This screen provides all users with an easy to use interface that allows them to select a main category before proceeding to the Create A Ticket screen.

6.3.5 Resources

Category Select needs access to the SharePoint database to display the proper main ticket category options.

6.3.6 Interface/Exports

N/A

6.4 Single Ticket Info

6.4.1 Responsibilities

Single Ticket Info displays the relevant details for a specific ticket and supports communication between the staff and the admin/tech through the use of a comment section. This screen displays two different views based on whether the user has a staff or an admin/tech role.The admin/tech view has access to closing a ticket, assigning assignees, and adding tags. Whereas a normal staff user doesn't.

6.4.2 Constraints

Only admin/tech roles will have the ability to make changes on individual tickets on this screen.

6.4.3 Composition

N/A

6.4.4 Uses/Interactions

Single Ticket Info provides a clean user interface for admin/tech to manage individual tickets once they are selected from either the All Tickets or My Tickets screen. But, normal staff will only be able to view single tickets.

6.4.5 Resources

Single Ticket Info requires access to the SharePoint database to retrieve and update ticket data.

6.4.6 Interface/Exports

N/A

6.5 Ticket Metrics

6.5.1 Responsibilities

Ticket Metrics is responsible for displaying the ticket received, resolved, and assigned to the user for the current month.

6.5.2 Constraints

This screen is only accessible to users with admin/tech roles.

6.5.3 Composition

N/A

6.5.4 Uses/Interactions

N/A

6.5.5 Resources

Ticket Metrics requires access to the SharePoint database to retrieve the number tickets received, resolved, and assigned to a user.

6.5.6 Interface/Exports

N/A

6.6 My Tickets

6.6.1 Responsibilities

My Tickets is responsible for displaying the user's tickets. This screen also provides users the option to search and filter their tickets.

6.6.2 Constraints

Certain Filters are only accessible to admin/tech roles.

6.6.3 Composition

My Tickets consists of the following subcomponents:

- **Search Bar** - Text input box that allows all users to search by ticket title and exact ticket id.
- **Filter Panel** - Displays a list of filter options based on the user role
    - My Tickets: Filters the tickets that are created by the user. This option is set as default for staff users.
    - My Assigned: Filters the tickets that are assigned to the user. Only visible to admin/tech roles and set as their default option.
    - Status: Filters tickets by ticket status.
    - Tag: Filters tickets by tag. Only visible to admin/tech roles.

6.6.4 Uses/Interactions

This screen will interact with the Home screen. This screen is accessible from the Home Screen.My Tickets redirects to the Single Ticket Info once a ticket is selected. It also redirects to Category Select when the new ticket button is pressed.

6.6.5 Resources

My Tickets requires access to the SharePoint database to retrieve the user's tickets and the list of tags.

6.6.6 Interface/Exports

N/A

6.7 Assign Role

6.7.1 Responsibilities

Assign Role is responsible for allowing admin/techn to change a user's role.

6.7.2 Constraints

This screen is only accessible to users with admin/tech roles.

6.7.3 Composition

N/A

6.7.4 Uses/Interactions

This screen will interact with the Manager Tools. This will only be accessible from the Manager Tools screen.

6.7.5 Resources

This screen requires access to the SharePoint database to retrieve all users from the SharePoint database.

6.7.6 Interface/Exports

N/A

6.8 Success

6.8.1 Responsibilities

Displays the 'Success' screen when a ticket is created successfully.

6.8.2 Constraints

A ticket has to be created successfully.

6.8.3 Composition

N/A

6.8.4 Uses/Interactions

This screen interacts with the 'Create A Ticket' screen. Once a ticket is created successfully, it will relay to the 'My Tickets' screen afterwards.

6.8.5 Resources

N/A

6.8.6 Interface/Exports

N/A

6.9 Manager Tools

6.9.1 Responsibilities

Displays all the tools that are accessible to admin/tech users.

6.9.2 Constraints

This screen is only accessible to users with admin/tech roles.

6.9.3 Composition

N/A

6.9.4 Uses/Interactions

This screen interacts with All Tickets, Tag Manager, Alert Manager, Ticket Metrics, and Assign Role screens. Users can only access the screens above through Manager Tools.

6.9.5 Resources

Manager Tools will require access to the SharePoint database, in order to determine if the user's role is either ‘admin’ or ‘tech’.

6.9.6 Interface/Exports

N/A

6.10 All Tickets

6.10.1 Responsibilities

Displays all the tickets from the SharePoint database. The tickets displayed are from the last 30 days, and are sorted in ascending order (this means that the most recently created tickets are always displayed first) with the assistance of pagination. Also, this screen provides admin/tech users the ability to search and filter the tickets.

6.10.2 Constraints

This screen is only accessible to users with admin/tech roles.

6.10.3 Composition

This screen is composed of the following components:

- **Pagination -** Provides the user with the ability to get the next or previous batches of tickets.

- **Search Bar** - Text input box that allows admin/tech to search by ticket title and exact ticket id.
- **Filter Panel** - Displays a list of filter options exclusive to this screen.
    - Unassigned/Assigned: Filter tickets by ticket assignment.
    - Status: Filters tickets by ticket status.

6.10.4 Uses/Interactions

This screen will interact with the Manager Tools and Single Ticket Info screen. This will only be accessible from the Manager Tools screen. Admin/Tech users will be able to select any ticket in order to view more information, which will redirect them to the Single Ticket Info screen. Additonally, Filters and the Search Bar will be at the admin/tech users disposal in order to view tickets past the 30 days.

6.10.5 Resources

All Tickets will require access to the SharePoint database, in order to determine if the user's role is either ‘admin’ or ‘tech’, as well as grabbing all the tickets from the database.

6.10.6 Interface/Exports

N/A

6.11 Create A Ticket

6.11.1 Responsibilities

Provides an easy to use UI for users to enter and submit their ticket requests.

6.11.2 Constraints

All users are required to enter a ticket title, description, category, sub category (if applicable), and location in order to submit a ticket request.

6.11.3 Composition

N/A

6.11.4 Uses/Interactions

This screen interacts with the Category Select and Success Screens. At the start of creating a ticket, the users will have to choose a category from the Category Select screen. Afterwards, it will redirect them to the Create A Ticket screen and prefill the ticket form with the data gathered from the Category Select screen. Once the user submits their ticket successfully, the user will be redirected to the Success Screen.

6.11.5 Resources

Create A Ticket will require access to the SharePoint database to retrieve category data, and to store the ticket request in the SharePoint database.

6.11.6 Interface/Exports

N/A

6.12 Alert Manager

6.12.1 Responsibilities

Displays the alerts that have not expired and provides an easy to use UI for managing the alerts.

6.12.2 Constraints

This screen is only accessible to users with admin/tech roles.

6.12.3 Composition

N/A

6.12.4 Uses/Interactions

This screen interacts with the Manager Tools and Create Alert Screens. Alert Manager is only accessible from the Manager Tools Screen. Admin/Tech users will be able to create alerts at the click of a button, or edit existing alerts by clicking them. Both options will redirect them to the Create Alert screen.

6.12.5 Resources

Alert Manager requires access to the SharePoint database to retrieve alert data.

6.12.6 Interface/Exports

N/A

6.13 Create Alert

6.13.1 Responsibilities

This screen provides an easy to use UI for admin/tech to edit or create their alerts.

6.13.2 Constraints

This screen is only accessible to users with admin/tech roles.

6.13.3 Composition

N/A

6.13.4 Uses/Interactions

This screen will interact with the Alert Manager screen. Admin/Tech will be using a form to create alerts, and once successful they will be redirected to the Alert Manager screen.

6.13.5 Resources

Create Alert requires access to the SharePoint database to store alerts data.

6.13.6 Interface/Exports

N/A

6.14 Tag Manager

6.14.1 Responsibilities

This screen provides an easy to use UI for managing Tags.

6.14.2 Constraints

This screen is only accessible to users with admin/tech roles.

6.14.3 Composition

This screen is composed of the following components:

- **Search Bar** - Text input box that allows admin/tech to type the name of their tickets.

6.14.4 Uses/Interactions

This screen will interact with the Manager Tools. This screen is only accessible from the Manager Tools screen. Admin/Tech will be able to create and delete tags.

6.14.5 Resources

Tag Manager requires access to the SharePoint database to store and update tags.

6.14.6 Interface/Exports

N/A

6.15 Knowledge Base

6.15.1 Responsibilities

This screen provides an easy to use UI for all users to look up additional information that the users may want to learn more about.

6.15.2 Constraints

N/A

6.15.3 Composition

This screen is composed of the following components:

- **Search Bar** - Text input box that allows users to lookup information based on keywords and titles.

6.15.4 Uses/Interactions

This screen will interact with the Home screen. This screen is accessible from the Home Screen. All users will be able to use the Search Bar to retrieve information from the Knowledge Base stored in SharePoint.

6.15.5 Resources

Knowledge base requires access to the SharePoint database to retrieve data.

6.15.6 Interface/Exports

N/A

7. Detailed Lower level Component Design

7.1 SharePoint Database

7.1.1 Classification

SharePoint is classified as a ***database system***.

7.1.2 Processing Narrative (PSPEC)

N/A

7.1.3 Interface Description

Developers use a GUI interface to interact with the SharePoint Database, in our case we used Powerapps to directly interact with the SharePoint Database.

7.1.4 Processing Detail

7.1.4.1 Design Class Hierarchy

N/A

7.1.4.2 Restrictions/Limitations

Limited when creating relationships due to not having Foreign Keys accessible. Must have a Microsoft 365 account to access.

7.1.4.3 Performance Issues

When submitting the individual data entries, there are not many performance issues. If there is a high volume of data being entered into the database, then there might be some delays.

7.1.4.4 Design Constraints

N/A

7.1.4.5 Processing Detail For Each Operation

The program's backend creates and updates data on the SharePoint Database using Powerapps Patch API that provides ways to interact with data and perform CRUD operations.

7.2 Power Apps Controls

7.2.1 Classification

The Power Apps Control component provides a visual allowing users to interact with tickets inside the PD HelpDesk Ticketing System. The Power Apps controls can be

classified into data input controls, data presentation controls, layout and navigation controls, data integration controls and custom controls.

7.2.2 Processing Narrative (PSPEC)

N/A

7.2.3 Interface Description

The Power Apps Control interacts with other components like data sources and APIs to get or update tickets using input fields, buttons, and other user interactions in the ticketing system.

7.2.4 Processing Detail

7.2.4.1 Design Class Hierarchy

Provides relationships toward components and gives an overview of key functionalities within the user interface.

7.2.4.2 Restrictions/Limitations

The functionality may be subject to the specific licensing options.

7.2.4.3 Performance Issues

N/A

7.2.4.3 Designs Constraints

The Power Apps Controls were designed for the user to give as much information about the issue they have on their ticket and to upload files to give a visual as well.

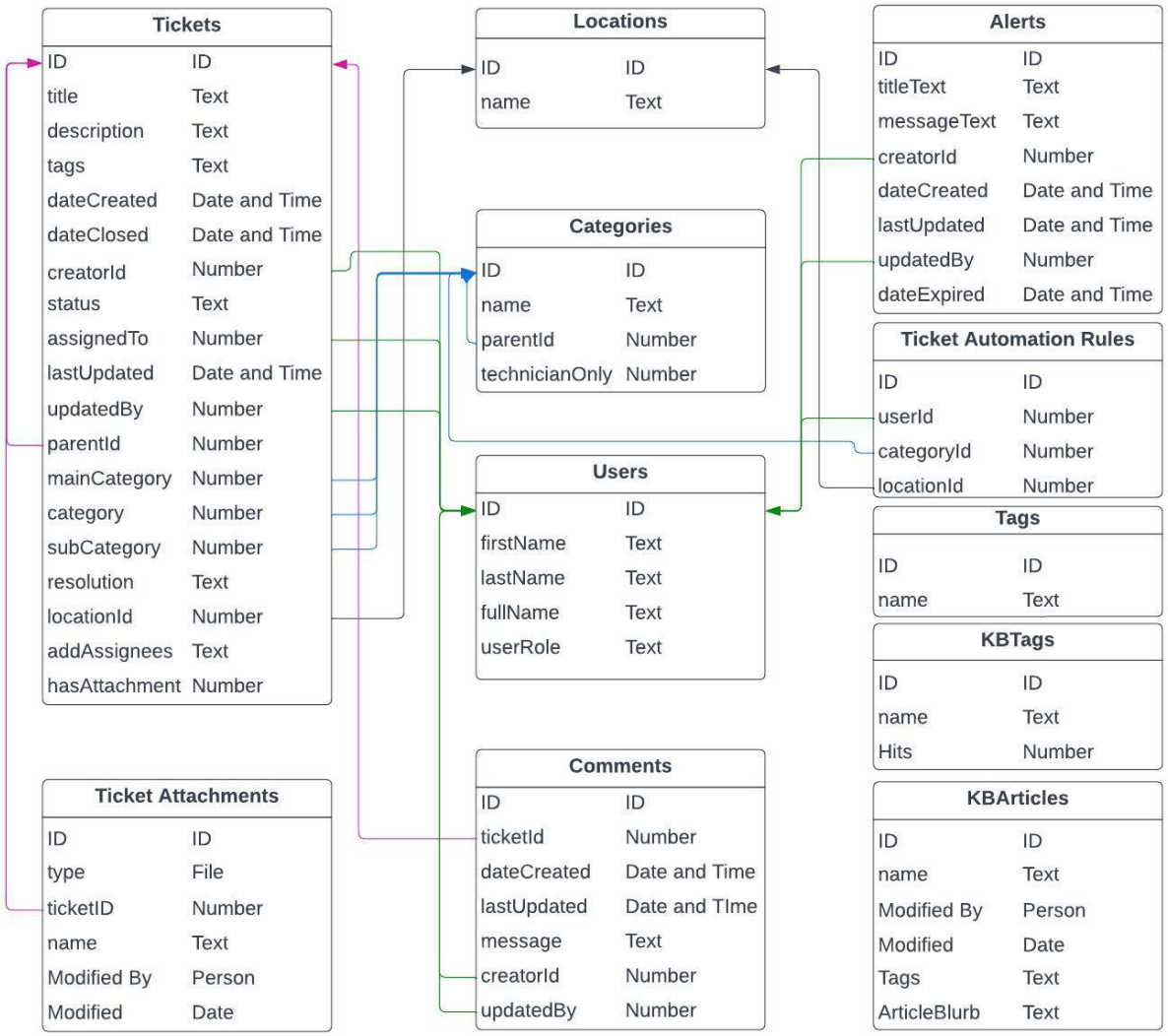
7.2.4.5 Processing Detail For Each Operation

When the user creates a ticket, it asks to input a title, category, sub category, specific issue, description, location and attachments if there are any.
The user, technician, staff, or admin are able to close tickets. However, if the user clicks on close ticket, a small pop up will show up and ask for the user to give a description on how the ticket was resolved.

8. Database Design

8.1 Sharepoint List

The database used is Microsoft SharePoint List. The database schema comprises 11 tables, each with a specific purpose in the ticketing system. The tables are **Users, Category, Tags, Locations, Alerts, Tickets, Comments, Ticket**



8.2 Tables and Entities Overview

All tables will have a unique ID column. This ensures each record will have a unique ID with no duplications, so when querying the database for a record we will not get multiple results with the same ID.

The **Users table** is the first table we created, and it stores the details of every user in the system. The table contains fields for the user's first name, last name, full name, user role, and unique email address. The user role field has three options: staff, tech, and admin, which determine the level of access the user has in the system.

The **Category table** columns are category's name, parentId, and technicianOnly. The parentId serves as a foreign key reference to the same "Category" table and identifies the ID of the category that is a subcategory of the current record. The technicianOnly column is a binary field that determines accessibility, with a value of 0 indicating availability for all users and a value of 1 allowing access only to tech and admin users.

The **Tags table** has only two columns: a unique ID and a name column. Since this table will only be accessed by tech and admin, there is no need for additional fields.

The **Locations table** follows a similar structure as the Tags table, but with the key difference that it will be accessible to all users of the ticketing system. Therefore, it will have a name column, and no additional fields will be needed.

The **Alerts table** contains a column for titleText, messageText, creatorId, dateCreated, lastUpdated, and dateExpired. The titleText column stores a text of 255 characters or less, while the messageText column is a text field type that can store longer messages to convey alerts within the app. The creatorId column is a foreign key that stores the ID from the Users table of the person who created the alert. The dateCreated and lastUpdated columns hold date values, while the expiredDate column can be either null or a date.

The **Ticket table** is a crucial component of our ticketing system and contains several foreign keys. It includes a title, description, mainCategory, category, subCategory, dateCreated, dateClosed, metricDate, creatorId, assignTo, status, lastUpdated, updatedBy, resolution, locationId, tags, addAssignees, and hasAttachment columns. The mainCategory, category, and subCategory columns store the unique ID from the Category table, while creatorId, assignTo, and updatedBy columns store the ID from the Users table. Lastly, for the foreign keys, the locationId column stores the ID from the Locations table. Tags and addAssignees columns, although related to Tags and Users tables, are text fields that store record names in CSV format for multiple tags or assignees. The title, description, and resolution columns are text fields. The dateCreated, dateClosed, metricDate, and lastUpdated columns are date and time fields, except for the metricDate column, which is in month and year format. The status column is a text field that contains only "open" or "closed." Lastly, the hasAttachment column is a binary field with a value of 0 for no attachments and 1 for the presence of an attachment.

The **Comments table** is a key component of our ticketing system, allowing users to leave notes and updates on specific tickets. It contains a foreign key reference to the associated ticket in the ticketId column. The creatorId column also stores a foreign key reference to the user who created the comment. The dateCreated and lastUpdated columns store the date and time when the comment was created or last updated. Finally, the message field is a text field that can hold a substantial amount of text. Overall, the Comments table is closely linked to the Tickets and Users tables, allowing seamless collaboration and communication between users and technical staff.

The **Ticket Attachment table** stores any files or images associated with a ticket. It contains a foreign key reference to the Tickets table called ticketId. In addition, the table includes fields for the file name (stored as a text field called "name"), the file itself (stored as a binary field called "file"), the date the file was last modified (stored as a date field called "modified"), and the user who last modified the file (stored as a text field called "modifiedBy" containing the user's full name).

The **KB Tags** table will function in the same way as the Tags table; however, it will also include an additional column called Hits, which will be a number type used to track how many times a tagged article has been requested by a user.

The **KB Articles** table will have the same structure as the Ticket Attachment table but will also include additional fields for Tags, which will be a comma-separated values (CSV) text field, and an article blurb, which will be a text field.

The **Ticket Automation Rules table** is essential for automating the ticket-assignment process and ensuring that the appropriate issues are routed to the appropriate technicians for handling. This table includes a name field for the technician's name, and foreign keys to the Users, Category, and Locations tables (stored as userId, categoryId, and locationId columns, respectively) for storing their corresponding IDs.

9. User Interface

9.1 Overview of User Interface

Before the user gets to PD Helpdesk, they must be logged in using their Microsoft 365 Office credentials. Once they have their credentials, the application will authenticate them using Microsoft's Security Groups and they will have the role of either user (staff), technician or admin. The first thing any user will see is the home page. The home page consists of a hamburger menu for navigation along with big buttons for less tech savvy users. The options the user has include the following: Create a Ticket, My Tickets, Knowledge Base, and, for admin, Manager Tool. Users can also use a keyword search in the knowledge base if they are having trouble finding something. The technician and admin roles will have more functionalities/buttons which they can navigate to in the Manager Tool such as All Tickets, Tag Manager, Alerts Manager, Tickets Metrics, and Assign Role.

One of the most important features of the application is Create a Ticket. Once navigated to this screen from the navigation menu, if staff need to create a new ticket, they simply click on the "Create a Ticket" button, which takes them to a category page where they can select the appropriate issue category. From there, the user will be asked to fill out a series of drop-down menus and text boxes to describe their support request. Once the ticket is submitted, the user can track its progress and receive live updates on their request from the "My Tickets" page.

When a technician logs into the app, they're presented with the home screen, where they can navigate to "My Tickets". This feature allows them to view all the tickets that are currently open and assigned to them, filter tickets by status, and perform keyword searches. Once they select a ticket, they can view additional details about the issue, add comments, assign additional technicians if necessary, and add relevant tags. After the technician completes the support ticket, they can click the "Close Ticket" button and provide a resolution. This resolution information is sent to the staff via email and is also recorded in the app for future reference.

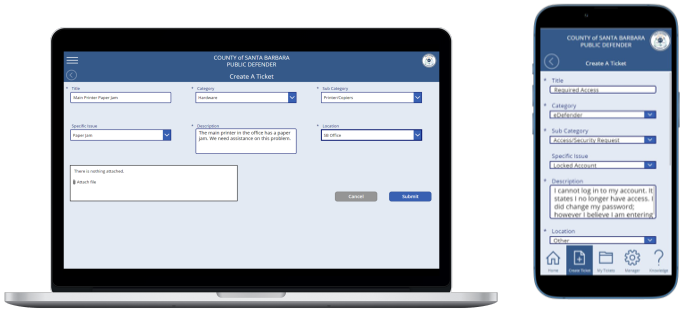
In addition to these important features, users can also use the knowledge base to see media or text based guides on popular issues.

9.2 Screen Frameworks or Images

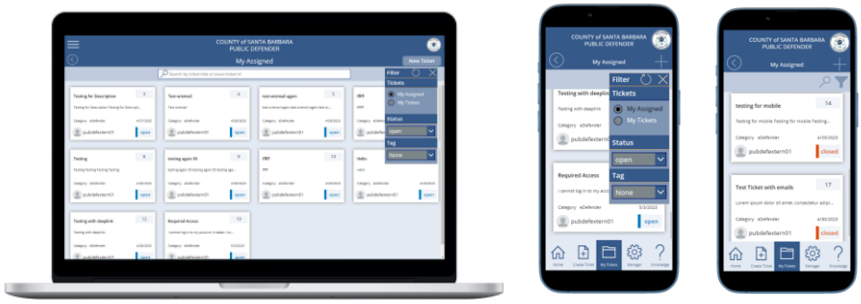
9.2.1. Splash Screen

9.2.2. Home

9.2.3. Create a Ticket

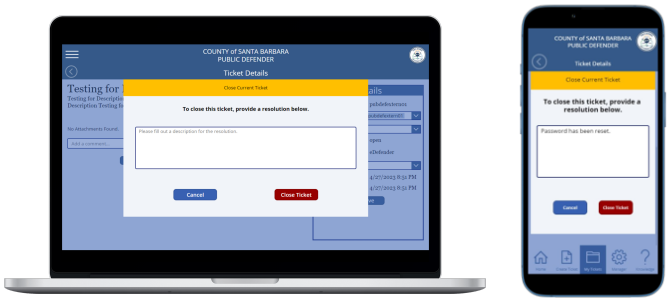


9.2.4. My Tickets



9.2.5. Individual Ticket

9.2.6. Resolve a Ticket

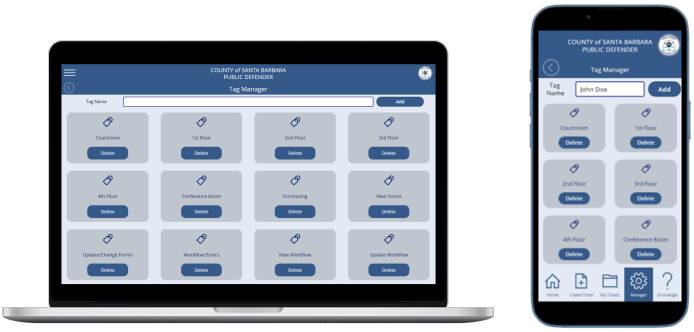


9.2.7. Knowledge Base

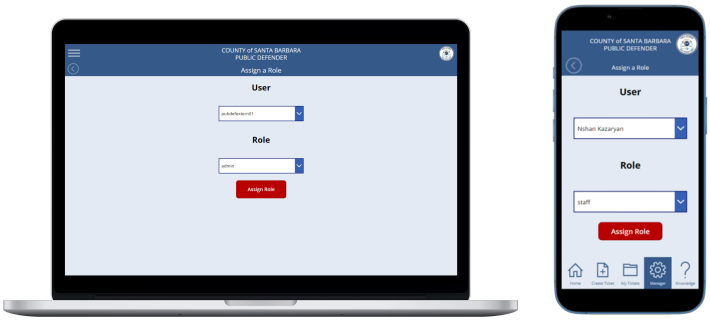
9.2.8. Manager Tools

9.2.9. All Tickets

9.2.10. Tag Manager

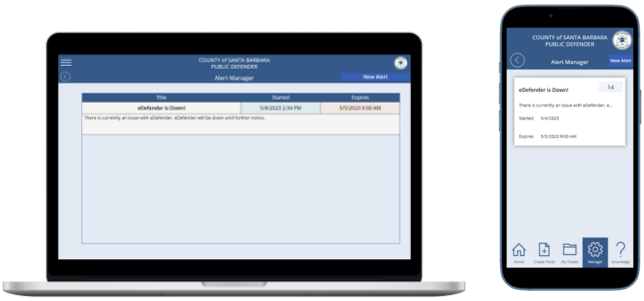


9.2.11. Assign Role



9.2.12. Create an Alert

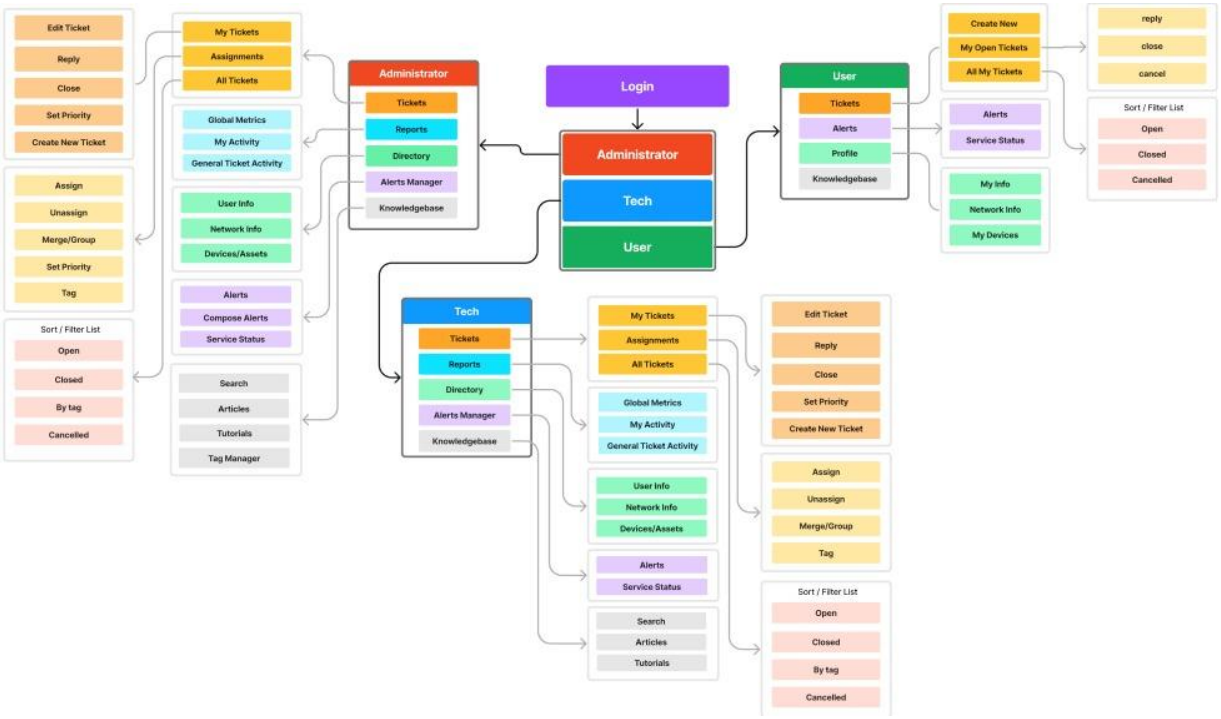
9.2.7. Alert Manager



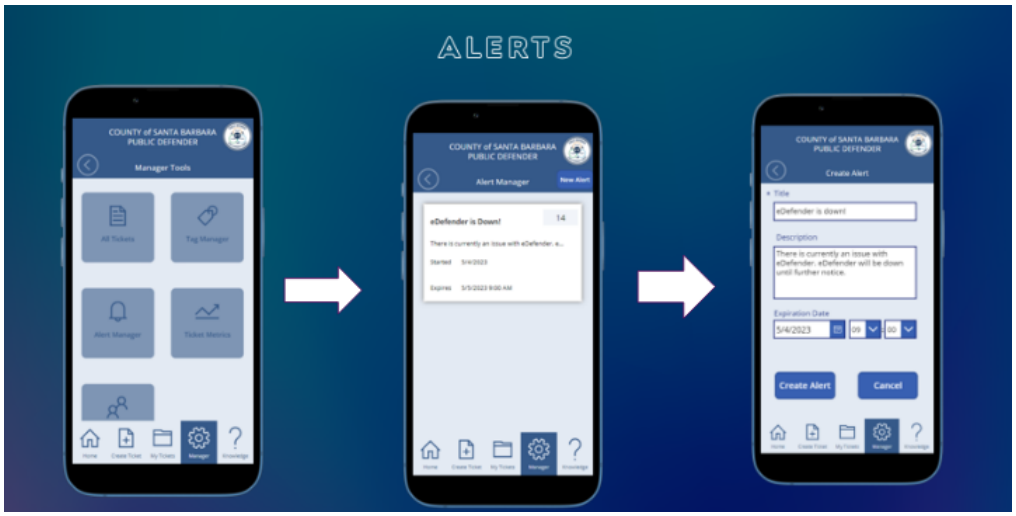
9.3 User Interface Flow Model

During our requirements-gathering process, SBPD provided us with a flowchart outlining their desired approach. The flowchart begins with user login, with three types of user roles specified. In this discussion, we will focus specifically on user requirements. After logging in, users should have access to Tickets, Alerts and Messages, their profile,

and the knowledge base. Two key requirements regarding tickets are creating new tickets and viewing "My Tickets". For creating a new ticket, users should be able to add comments and attach files to explain their issue. In "My Tickets", users should be able to view and sort/filter all their created tickets. The ability to create, update, and monitor their own tickets is a crucial aspect of our project requirements. The Tech and Admin roles have similar requirements as the User role, but with additional access and sections such as the Admin Manager.




RESOLVE A TICKET


ALERTS

10. Requirements Validation and Verification

Requirement	Satisfied By	Testing Method
The system shall authenticate users and assign respective roles.	Splash Screen + Users SharePoint List	Attempt to login with valid/invalid credentials. Assign various roles to users and verify they have appropriate access
The system shall protect data for all users.	SharePoint Security Groups + List Item Permissions	Attempt to access protected data without authorization or through unauthorized means, and verify that the system prevents such access.
The system shall provide an automated workflow for ticket assignment.	Automation Rules SharePoint List	Creating new tickets with various criteria and verifying that they are assigned to the appropriate user or group based on the defined rules. Test various scenarios, such as assigning tickets based on category and geographic location.
The system shall provide a navigation menu for users to navigate to different screens.	Navigation menu component (All screens)	Verifying that it is present on all screens and allows users to navigate to different areas of the application. Test that the menu displays the correct options based on user role and permissions.
The system shall have responsive design.	PowerApps Use of Containers + Formulas	Test the responsiveness of the system on different devices and screen sizes, including desktops, laptops, tablets, and smartphones. Verify that the user interface adjusts dynamically to fit the available screen size, and that all functionality and content is accessible and usable regardless of screen size.

The system shall provide all users with the ability to create a ticket.	Create A Ticket Screen + Create A Ticket Form	Create a new ticket and verify it appears in the system/database.
The system shall allow all users to attach files when they are creating a ticket.	Attach Files Component + Create A Ticket Screen	Test the attachment functionality by verifying that all users are able to attach files when creating a ticket. Ensure that the system accepts a variety of file types, and that there are no file size limits or other restrictions that would prevent users from attaching necessary files.
The system shall provide all users with the ability to view all their tickets.	My Tickets Screen	Test the ticket viewing functionality by verifying that all users are able to view a list of their own tickets, and that the list contains all necessary information such as ticket number, status, and category.
The system shall provide technicians and admins with the ability to resolve a ticket.	Resolve Ticket Screen +Close Ticket Button	Verifying that technicians are able to access a ticket detail view, which displays all relevant information about the ticket, including the ability to close the ticket. Ensure that the ticket is closed when the Close button is clicked, and that the system prevents the ticket from being closed if required fields are not completed or if the ticket is already closed.
The system shall allow all users to use filters on my tickets screen.	Filter Tickets Power Automate	Test the filter functionality by verifying that all users are able to access My Tickets screen and apply various filters to the displayed tickets, such as by status and tag. Ensure that the filter component is easy to use and intuitive,

The system shall allow all users to keyword search on my tickets screen.	Search Tickets Power Automate	Test the search functionality by verifying that all users are able to access My Tickets screen and use a search bar component to search for tickets by title and/or ticket number. Ensure that the search bar is prominently displayed and easy to use, and that it returns relevant results in a timely manner.
The system shall display relevant ticket information when users select a ticket to view.	Single Ticket Info Screen	Test the ticket details functionality by verifying that all users are able to select a ticket from the My Tickets screen and view all relevant information related to that ticket, such as the status, description, assigned technician, and any comments,etc. Ensure that the ticket details component is easy to navigate and provides all necessary information in a clear and organized manner.
The system shall allow all users to add comments on a ticket.	Single Ticket Info Screen + Add Comment Button	Test the comment functionality by verifying that all users are able to access a ticket and add comments to it. Ensure that the comment section is easy to use and clearly displays all existing comments in a threaded or nested format.
The system shall allow technicians and admins the ability to change/add certain details on a ticket.	Single Ticket Info Screen, Assignee + Other Assignees Combo box, Status, Tags Combo box	Test the ability to change/add certain details on a ticket by verifying that technicians and admins can access the Edit Ticket Details component and modify specific fields such as the ticket status, Tags, or assigned technician(s). Ensure that the component is

		easy to use and clearly displays all editable fields in a logical and organized manner.
The system shall provide all users with a knowledge base where they can search for helpful documents.	Knowledge Base Screen + KB Articles Document Library	Test the ability to search for documents in the Knowledge Base by verifying that the Search Bar is functioning properly and returns relevant search results when users enter search terms. Ensure that the Search Bar is easy to use and clearly displays suggested search terms or results as the user types.
The system shall provide technicians and admins with manager tools.	Manager Tools Screen, All Tickets, Tag Manager, Alert Manager, Ticket Metrics, Assign Role	Test the functionality of the Manager Dashboard and Manager Tools UI Components by verifying that technicians and admins can access and use the tools as intended.
The system shall provide technicians and admins the ability to view all tickets in the database with search and filtering capabilities.	All Tickets Screen, Search Tickets Power Automate, Filter Tickets Power Automate	Test the functionality of the Ticket List UI Component and Search/Filter UI Components by verifying that technicians and admins can access and view all tickets in the database, as well as search and filter tickets based on specific criteria.
The system shall allow technicians and admins to add/remove tags from the database.	Tag Manager Screen, Add Tag/Delete Tag Button	Test the functionality of the Tags UI Component and Database API by verifying that technicians and admins can add or remove tags from the database. This includes testing the ability to create new tags and delete existing tags.
The system shall allow technicians and admins to	Assign Role Screen	Test the functionality of the User Role UI Component and

change user roles.		Database API by verifying that technicians and admins can change the roles of users in the system. This includes testing the ability to assign or remove roles to/from users and view the roles assigned to each user.
The system shall allow technicians and admins to create and manage alerts.	Alert Manager Screen, Create Alert Screen, Edit Alert Screen	Test the functionality of the Alerts UI Component and Database API by verifying that technicians and admins can create and manage alerts in the system. This includes testing the ability to create new alerts, edit existing alerts, and view alerts in a user-friendly and intuitive interface.
The system shall send email notifications to users when a ticket is created, an alert is created, ticket has been updated, or ticket has been assigned to a technician.	Send Email Power Automate, Create Ticket Button, Add Comment Button, Save Button, Create Alert Button	Test the Email Service and Notification API to ensure that email notifications are properly sent to users when a ticket is created, an alert is created, a ticket has been updated, or a ticket has been assigned to a technician. The Ticket Module, and Alert Module should be designed to generate the appropriate notifications based on user preferences and system rules. The Email Service should be able to deliver emails reliably and securely,

11. Glossary

- Application Programming Interface API: Method for when two or more programs communicate with each other
- ChatBot: Program where a A.I can make automated responses to the user
- Hypertext Transfer Protocol Secure (HTTPS): A type of request made by a web browser in order to load a webpage. HTTPS is a secure version of Hypertext Transfer Protocol (HTTP) and is commonly used when transferring private data like logging into an email or bank account.
- Microsoft Azure: Cloud computing platform that interacts with other microsoft products/software like PowerApps.
- MySQL: An open source relational database management system (RDMS) used to store and access data.
- OffBoarding: Process of removing an employee/member from a system. Also involves revoking/freezing employees access to system database
- Onboarding: Process of integrating a new employee/member into a system
- Open source: Software where the creator allows other users direct access to the source code so the user can alter and distribute the software for their own purpose
- Power Apps: Program used to develop ticketing system app for mobile and desktop platforms
- SBPD: Santa Barbara Public Defenders
- Secure Mail Transfer Protocol (SMTP):A type of request that is made when on an email from one account to another
- Ticketing system: Software program used by a support team for the purpose of keeping track of problems/requests submitted by users/customers

12. References

https://csns.cysun.org/department/cs/project/view?id=7913647

SBPD Website: https://www.countyofsb.org/187/Public-Defender

Software Requirements Document: Software Requirements Document

ZenDesk: https://www.zendesk.com

ServiceNow: https://www.servicenow.com/

Shane Young Youtube Channel: https://www.youtube.com/@ShanesCows

Reza Dorrani Youtube Channel: https://www.youtube.com/@RezaDorrani