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

Version 1.0 approved

Prepared by Nshan Kazaryan, Marie Karibyan, Kevin Trochez, Mark Perez, Brandon Estrada, Christian Armendariz, Haoi Nam Cao, Gilbert Hopkins, Geovany Huerta

Santa Barbara Public Defenders Office / Deepak Budwani, Brent Modell, Luis Ramirez

September 8, 2022

Revision History

Name	Date	Reason For Changes	Version
Initial Draft	9/08	Initial draft of document	1.0
Update Document	11/30	Filling in info for certain sections	1.1

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 Azure SQL: fully managed platform as a service database engine that handles most of the database management functions such as upgrading, patching, backups, and monitoring without any user involvement. Relational management system.
- 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.

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.

2.3 Goals and Guidelines

The application should be fully completed and deployed by the end of spring semester 2023 (May 2023). 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

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).

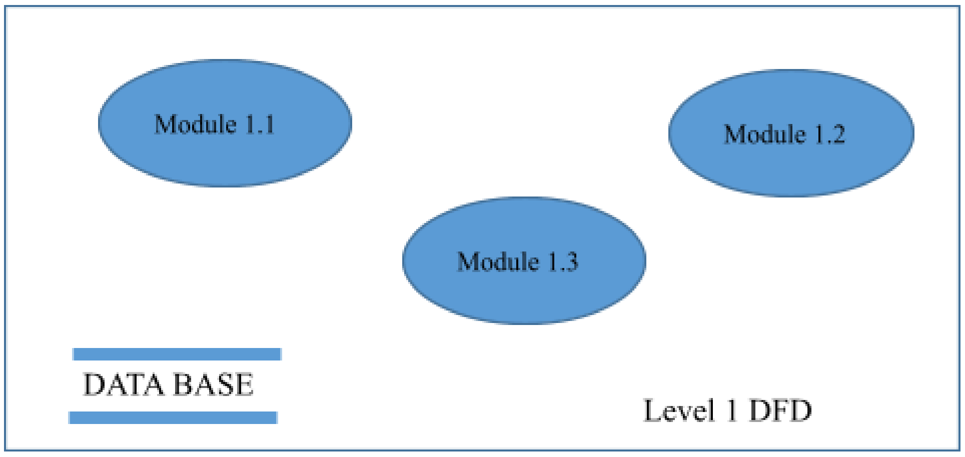
4. System Architecture

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

5.1.1 Microsoft Azure SQL

5.1.2 Microsoft Power Apps

5.1.3 Github

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

5.2 Plans for ensuring requirements traceability

These

…Describe…

5.3 Plans for testing the software

For th

…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).

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

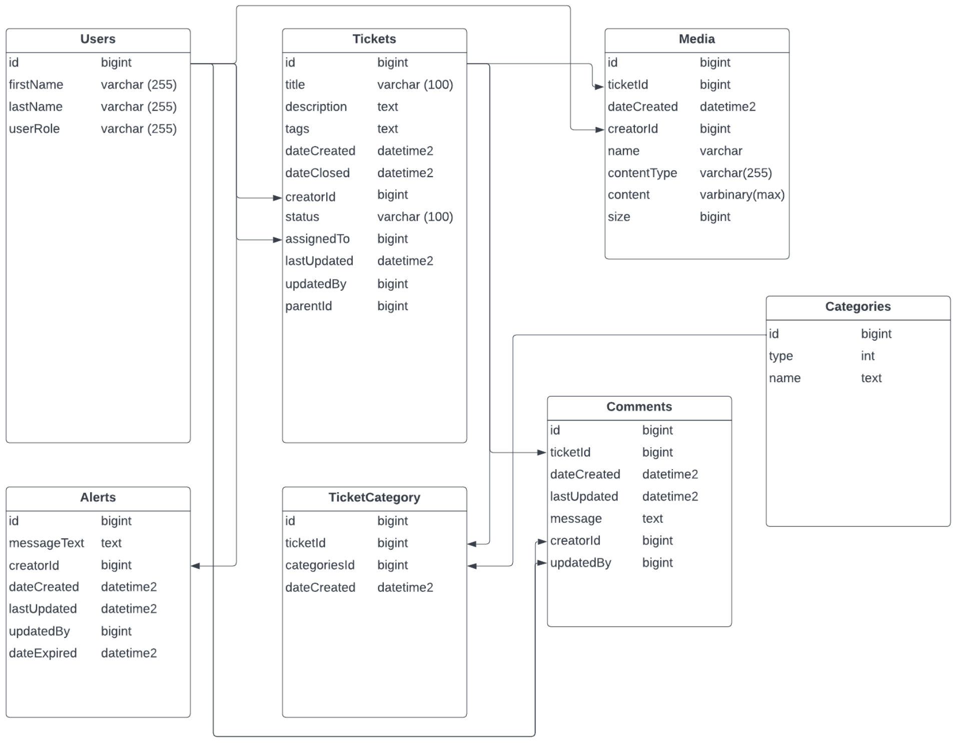
7.x.4.3 Performance Issues

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.



This is our database schema. We started with the Tickets table since this is the most important information we need to store in our database. We listed out all the details that a tickets record should contain. By doing this we realized which entities need to be its own table to have a one to many relationship with the Tickets table. There can be many Media, TicketCategory, and Comments records for one Ticket. All tables have a unique identifier (id) and the date that record was created. The Alerts table is to send mass notifications out to all users. It also has a dateExpired to prevent the alert from always showing.

More details will be here next semester.

9. User Interface

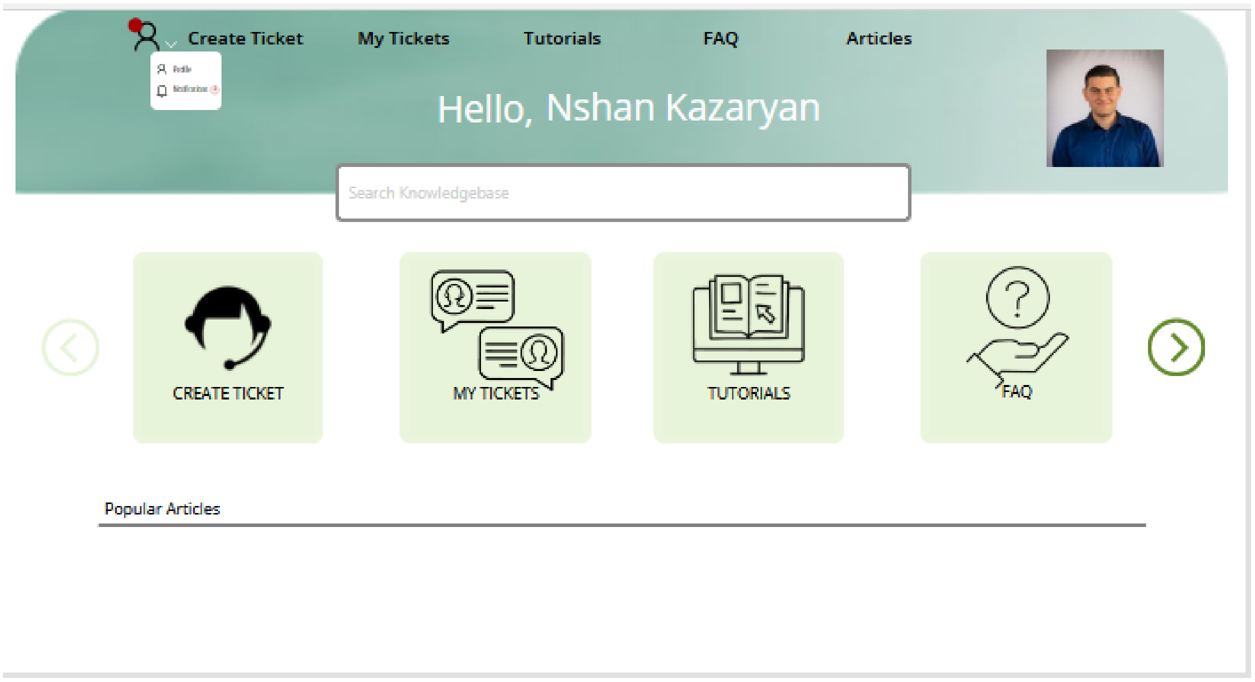
The user interface is the application, from the point of view of the users. Do your classes and their interactions (the logical and process views) impose restrictions on the user interface? Would removing some of these restrictions improve the user interface? Use some form of user interface flow model to provide an overview of the UI steps and flows. Don't go into too much refinement. You should include screen shots or wireframe layouts of significant pages or dialog elements. Make sure to indicate which of the system level modules or components that each of these user interface elements is interacting with.

9.1 Overview of User Interface

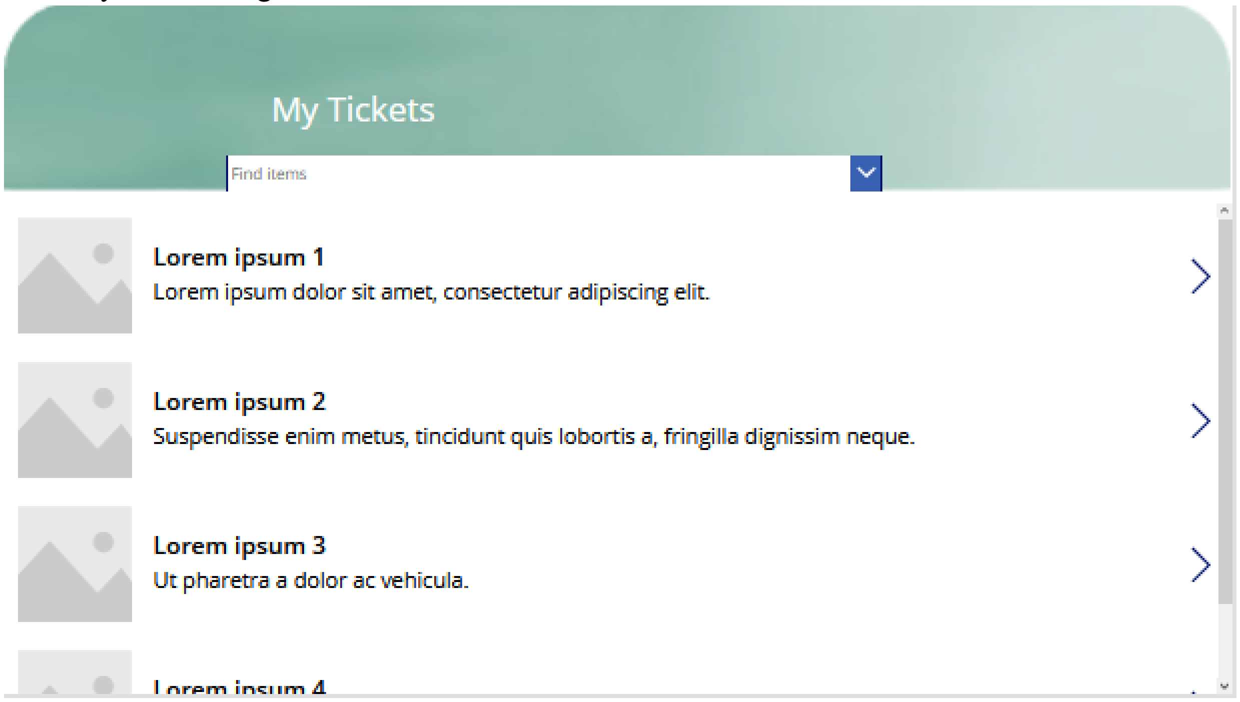
Before the user gets to PD Helpdesk, they must be logged in using their Microsoft 365 Office credentials. Once they have their credentials, the application will authenticate them using Microsoft's Security Groups and they will have the role of either user, technician, or admin. The first thing any user will see is the home page. The home page consists of a navbar along with big buttons for less tech savvy users. The options the user has include the following: Create Ticket, My Tickets, Tutorials, FAQ, Articles. Users can also use a keyword search in the knowledge base if they are having trouble finding something. The admin and technician role will have more functionalities/buttons such as Assign Ticket, Create Alert, Assign Category to Ticket. When a user clicks on Create a Ticket, they are redirected to a page and prompted with a form. Here, users can fill out the form and upload screenshots of their problems. When a user visits the My Tickets screen, they can see the history of all their tickets along with date and time stamps. Once a user clicks on one of those tickets, they can see the details of that particular ticket including the status. In addition, users can go to the tutorials or articles pages to see media or text based guides on popular issues.

9.2 Screen Frameworks or Images

9.2.1 Home page



9.2.2 My Tickets Page



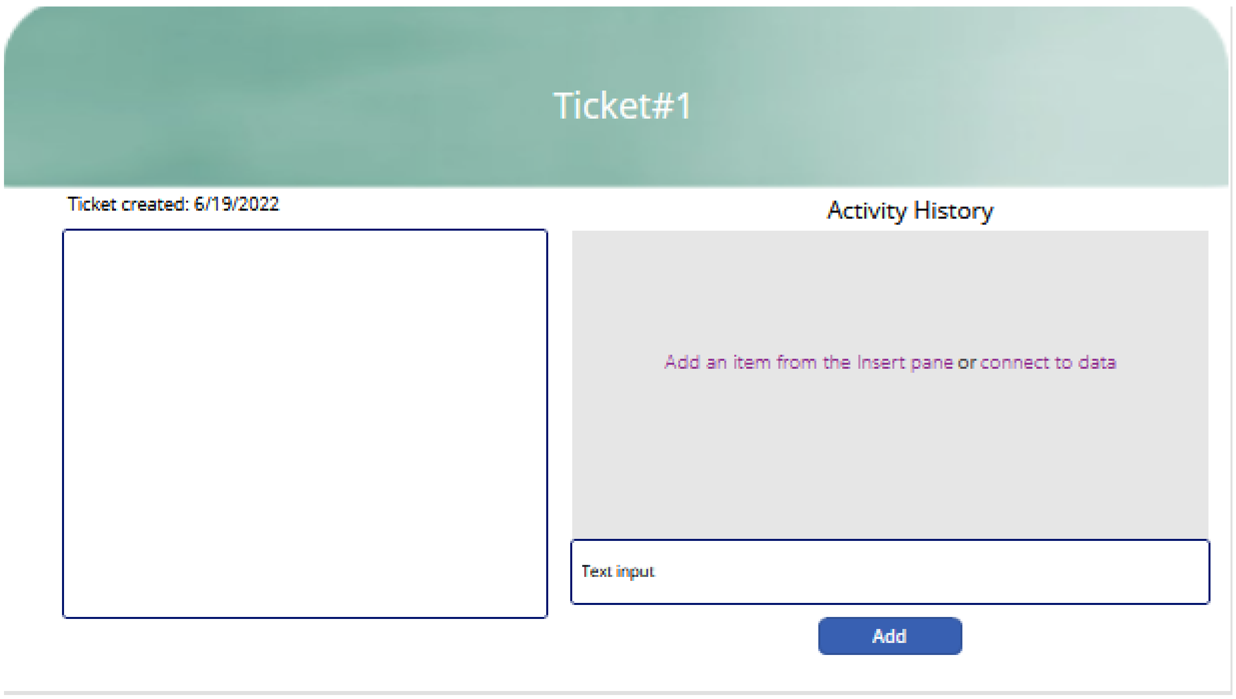
Lorem ipsum 1
Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum 2
Suspendisse enim metus, tincidunt quis lobortis a, fringilla dignissim neque.

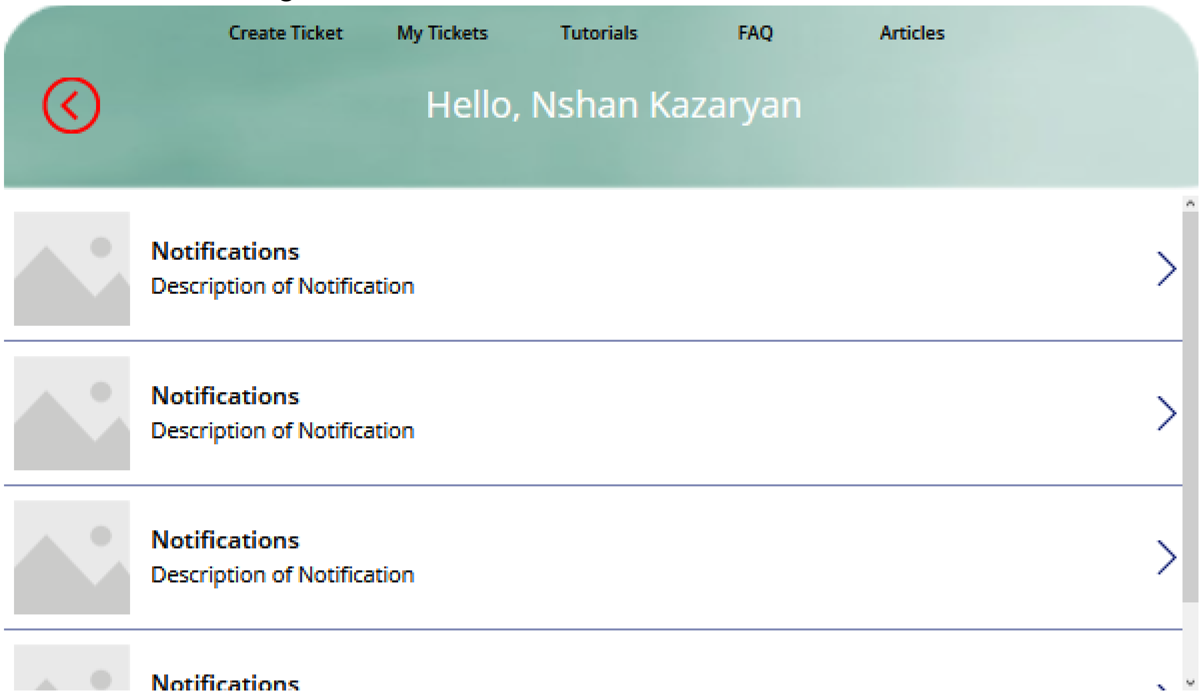
Lorem ipsum 3
Ut pharetra a dolor ac vehicula.

Lorem ipsum 4

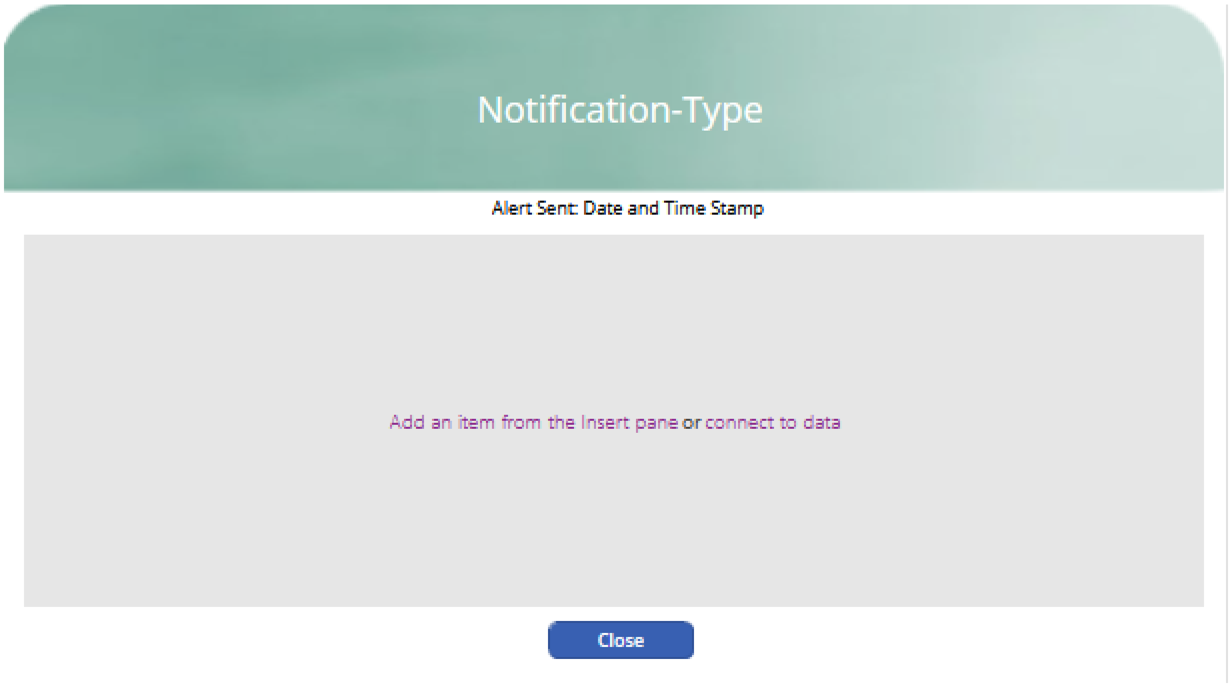
9.2.3 Individual Ticket Page


Ticket#1
Ticket created: 6/19/2022
Activity History
Add an item from the Insert pane or connect to data
Text input
Add

9.2.4 Notifications Page

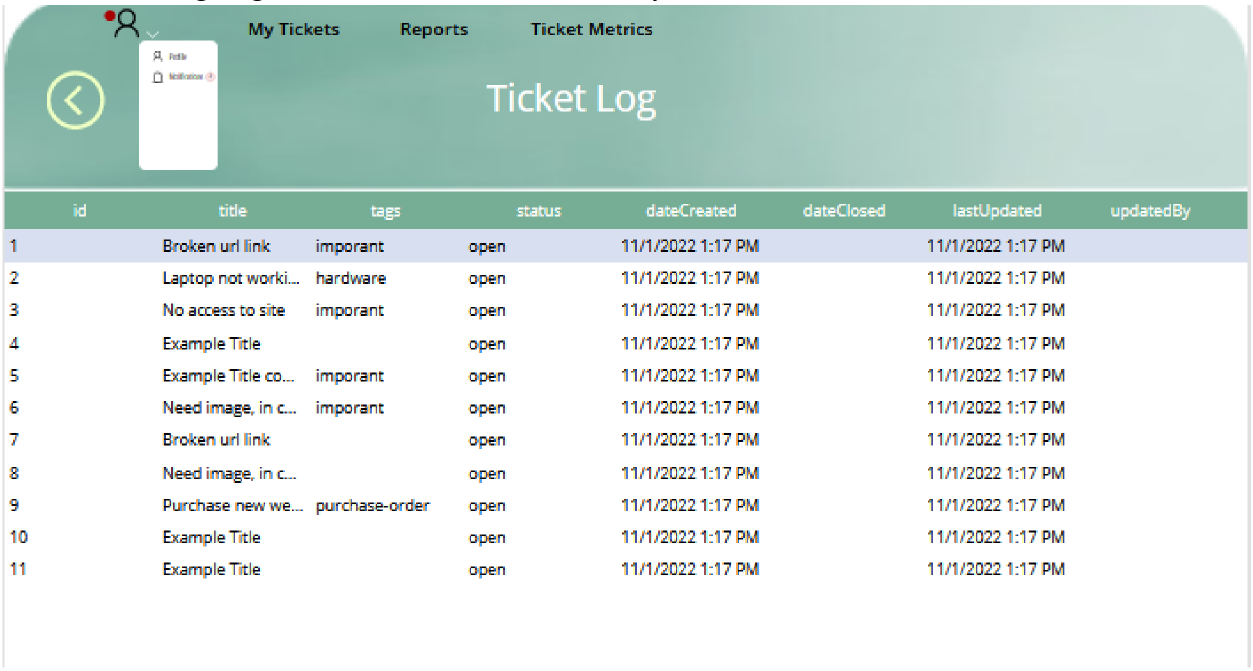


9.2.5 Notification Details Page


Notification-Type

Alert Sent: Date and Time Stamp

9.2.6 Ticket Log Page - Technician and Admin Only

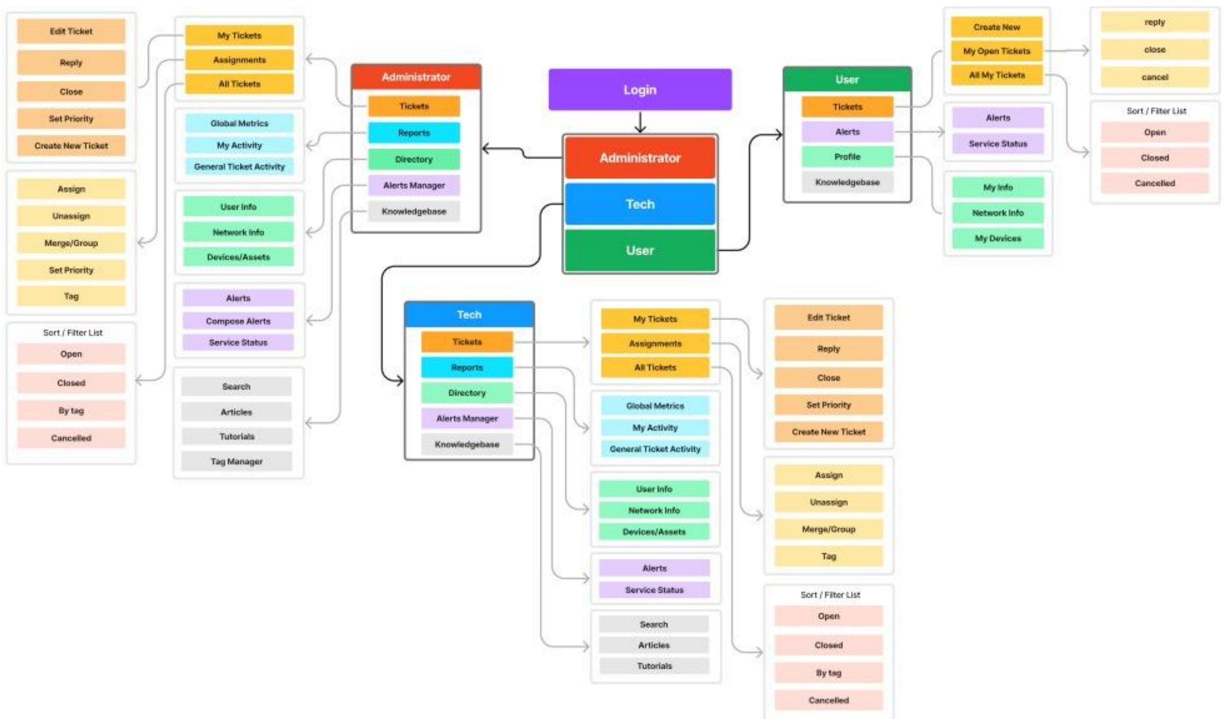


Ticket Log

id	title	tags	status	dateCreated	dateClosed	lastUpdated	updatedBy
1	Broken url link	imporant	open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	
2	Laptop not worki...	hardware	open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	
3	No access to site	imporant	open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	
4	Example Title		open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	
5	Example Title co...	imporant	open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	
6	Need image, in c...	imporant	open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	
7	Broken url link		open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	
8	Need image, in c...		open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	
9	Purchase new we...	purchase-order	open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	
10	Example Title		open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	
11	Example Title		open	11/1/2022 1:17 PM		11/1/2022 1:17 PM	

9.3 User Interface Flow Model

A discussion of screen objects and actions associated with those objects. This should include a flow diagram of the navigation between different pages.



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.

- 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

<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.>

Brad Appleton <brad@bradapp.net> http://www.bradapp.net

https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate_Fall08.doc

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/