Senior Design Final Report

Digitization and Modernization of PD HelpDesk Ticketing System

Version 1.0 - 05/07/2023

Team Members:

Team: Nshan Kazaryan, Gilbert Hopkins, Hoai Nam Cao, Mark Perez, Brandon Estrada, Geovanny Huerta, Marie Karibyan, Biruk Mengeste, Kevin Trochez Grajeda

Faculty Advisor: Dr. Chengyu Sun

Liaisons: Deepak Budwani, Brent Modell, Luis Ramirez, Angella Stoke

Table of Contents

1. Introduction

1.1 Background:

Santa Barbara Public Defender provides legal services across Santa Maria, Santa Barbara, and Lompoc. The Santa Barbara Public Defender's (SBPD) Office currently utilizes an IT ticketing system constructed internally and operates through Outlook's email platform. This system allows SBDP staff to submit and store IT support requests but has limited functionality and capabilities. The current ticketing system, which operates through email communication, has presented challenges in maintaining a clear overview and tracking customer support requests.

What is a ticketing system? A ticketing system is a software application for managing customer support and service requests. It enables organizations to centralize, prioritize, and track incoming customer inquiries or issues. Businesses can improve their customer service efficiency and effectiveness by using a ticketing system and increasing customer satisfaction. Standard features of a ticketing system include a user portal for submitting tickets, automated notifications, ticket assignment/tracking, and reporting/analytics. There are different ticketing systems, such as IT help desk ticketing systems, customer service ticketing systems, and event management ticketing systems. We will be developing a help desk ticketing system.

Utilizing a digitized and modernized ticketing system using Microsoft Power Apps will effectively address the shortcomings of the current email-based system by providing a centralized hub and an easy-to-use application that improves the speed and efficiency of support ticket requests. Therefore, it will enhance customer experience and increase productivity for the Santa Barbara Public Defender's Office.

1.2 Design Principles:

One of the major problems is that the current system needs more visibility. This makes it difficult for the IT department to quickly and efficiently identify and resolve customer issues. Not having the proper visibility can lead to occasional delays in addressing requests and difficulties in ensuring that all inquiries are attended to promptly. We designed the application with the end-users in mind. We kept it simple, intuitive, mobile-friendly, and easy to use. PD HelpDesk is designed to manage and store data efficiently using a robust database. This includes managing ticket submissions, updates, and tracking. PD HelpDesk is designed with security in mind. We implemented security measures such as role-based access control, authentication, and authorization to ensure data privacy. In addition, the application is designed to integrate with other systems and tools, such as email and comment system, which allows communication between users and technicians. PD HelpDesk is still in development, so the app's design must

remain simple. This will help ensure that future maintenance and expansions can be easily managed without unnecessary complexity.

1.3 Design Benefits:

With this project, our proposed solution is to develop a streamlined application that will offer a wide range of solutions to the SBPD office. Having a centralized hub for support tickets will gain visibility on the status of the tickets, enabling them to know which tickets are being worked on and which still need to be addressed. The new system will also be helpful for their staff by providing an interface that displays all their current support tickets and real-time status updates. The application can be easily accessed through various mobile and desktop devices, offering flexibility and convenience for users. This proposed system should improve the staff's experience and lead to faster response times. Additionally, it should increase IT productivity and enable them to use this data for analytical insight. Overall, the new digitized and modernized ticketing system addresses the current system's problems, gaps, and challenges by introducing a more efficient, organized, and user-friendly solution to effectively manage and track customer support requests. By consistently monitoring the application's performance and making necessary adjustments, we can continuously enhance the user experience

1.4 Achievements

Our project has achieved significant milestones that have greatly enhanced the functionality and security of the system. One of the notable achievements is the implementation of data protection through the use of security permissions. By establishing granular access controls, we have ensured the ticketing system's data's confidentiality, integrity, and availability. This feature is crucial in protecting sensitive information and preventing unauthorized access.

Another key achievement is the development of a robust database management system. This has revolutionized the way ticket information is stored and retrieved. The database architecture has been designed to handle large volumes of data while maintaining data integrity efficiently. We have improved performance and reliability in storing and retrieving ticket information through effective data structuring and optimization.

Additionally, our project has successfully delivered a responsive application that provides users with a customized experience for both mobile and desktop views. The responsive design ensures that users can seamlessly access and interact with the ticketing system across different devices. This adaptability enhances user convenience and accessibility, making it easier for staff, technicians, and administrators to manage and resolve tickets effectively.

2. Related Works and Technologies

2.1 Existing Solutions:

We have looked into some existing frameworks for creating ticketing system applications. These include popular web development frameworks such as Ruby on Rails, Django, or Laravel, which provide pre-built components for handling user authentication, data storage, and web-based user interfaces. Most ticketing systems require some form of data storage, such as MySQL, PostgreSQL, or Oracle. These databases commonly store customer information, ticket data, and other relevant information. The front end of a ticketing system is typically built using HTML, CSS, and JavaScript. Popular front-end frameworks like React, Angular, or Vue can be used to build rich, interactive user interfaces. Many ticketing systems must integrate with other systems, such as email clients, customer relationship management (CRM) tools, or other third-party services. Integration technologies such as RESTful APIs or webhooks can be used to facilitate these integrations.

The existing ticketing systems that we referred to were ServiceNow, ZenDesk, and Jira. However, all of these ticketing systems require a subscription or license fee. The subscription fee typically includes access to the software, updates, and customer support services. The cost of these subscriptions can vary depending on the size of the organization and the specific features and services needed. All these ticketing systems are cloud-based. We took inspiration from Jira's ticket detail layout and reporting style when developing our solution.

Since SBPD already uses the Microsoft 365 environment, they have access to Power Apps, which can be integrated with other Microsoft services such as SharePoint and Dynamics 365 and third-party applications, making it easy to connect to data sources and extend the functionality of the app. In addition, Power Apps provides an intuitive drag-and-drop interface that allows users to create apps without extensive programming knowledge. Power Apps allows users to build apps quickly, reducing the time required to develop custom business solutions. Furthermore, Power Apps allows users to customize the app's user interface, data sources, and workflows, making it possible to create highly tailored solutions that meet specific business needs. Since SBPD needed a mobile-friendly application, we leveraged Power Apps to optimize mobile device apps, enabling users to access business data and processes from anywhere.

2.2 Reused Products

Power Apps was decided upon by the student team and liaison team as the technology for developing their ticketing system. Power Apps has a programming language similar to Excel.

Considering that SBPD already had access to Microsoft's Power Platform licensing, building the application in this environment was not only cost-efficient. Still, it would allow them to utilize their current licenses fully.

3. System Architecture

3.1 Overview

Our system is a typical IT ticketing system with many different features and capabilities built using Microsoft's Power Apps framework. This software is intended for users to submit tickets for IT-related issues, view their current tickets, view the knowledge base, and view system-based notifications such as outages. On the other hand, a technician 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 many websites with a frontend and backend database connection. The API acts as the middleman between the front and back ends. Essentially, the API grabs the user requests/actions in the front end and updates the database with the information the user provided. For example, if a user creates a ticket, the ticket table is updated with respectable attributes.

Figure 1 provides an overview of the technologies used to create our application.

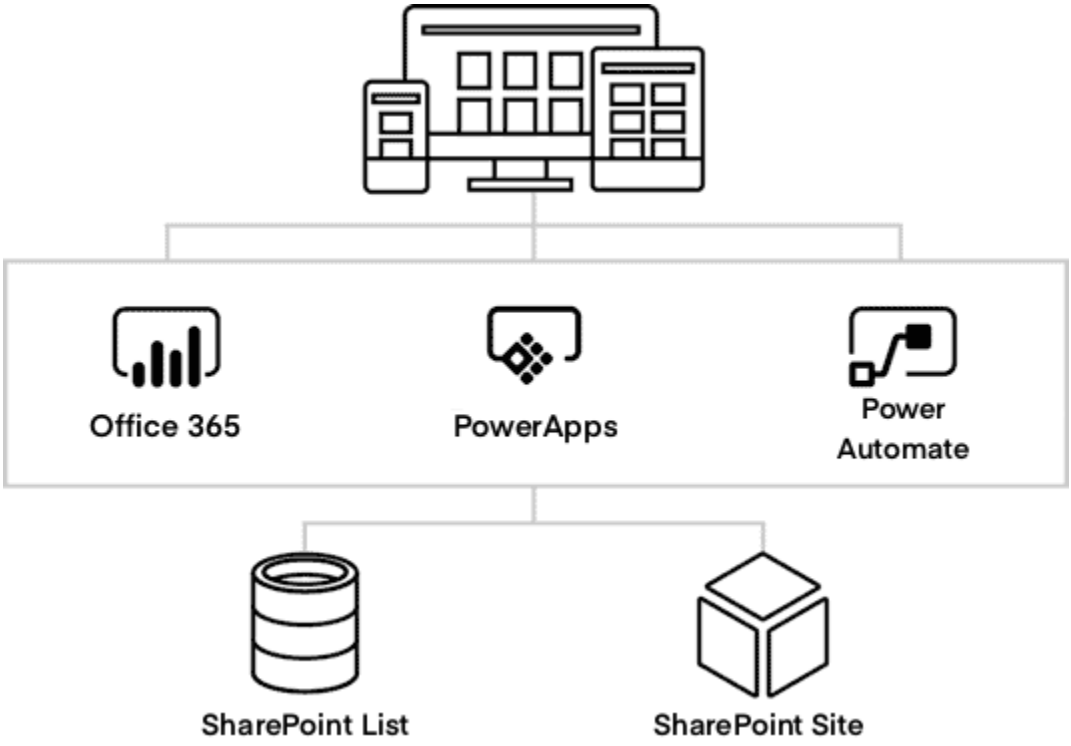

Figure 1: DFD0

- **Office 365:** Office 365 is often used for emails in Power Apps, providing a reliable and scalable platform for sending and receiving emails. Power Apps can be configured to send emails automatically based on certain triggers or events, such as when a form is

submitted. Office 365 provides various tools for gathering user information in Power Apps, such as the user's name and profile picture. The data collected can then be stored in SharePoint or other Office 365 data sources and used for reporting and analysis.

- **Power Apps:** It is one of the other two services on Microsoft's Power Platform. Microsoft Power Apps is a low-code development platform that enables users to create custom apps for business purposes. It simplifies the app development process by offering pre-built templates, drag-and-drop interface design, and integration with various data sources. This means those who are not technically inclined can create applications. Power Apps offers pre-built functions that eliminate the need for users to create them from scratch. Think of the functions in Power Apps as similar to those in Excel. Just as you can use functions in Excel to manipulate data, you can use functions in Power Apps to create powerful custom applications without extensive programming knowledge.
- **Power Automate**: With Power Automate, we created workflows that perform various tasks, such as sending notifications, collecting data, and adding attachments. 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 ticket updates, and attach files to tickets if needed. By automating repetitive tasks, Power Automate can save time and reduce errors associated with manual data entry or other tedious tasks.
- **SharePoint List**: Defines where the app's data will be stored. The data source was originally Azure SQL. Design and migration to SharePoint were needed because of Power Apps licensing issues.
- **SharePoint Site**: The server that hosts our database. SharePoint lists must be stored in a SharePoint Site for application use. This is our centralized repository for storing and organizing various types of content, such as document libraries and lists.

3.2 Data Flow

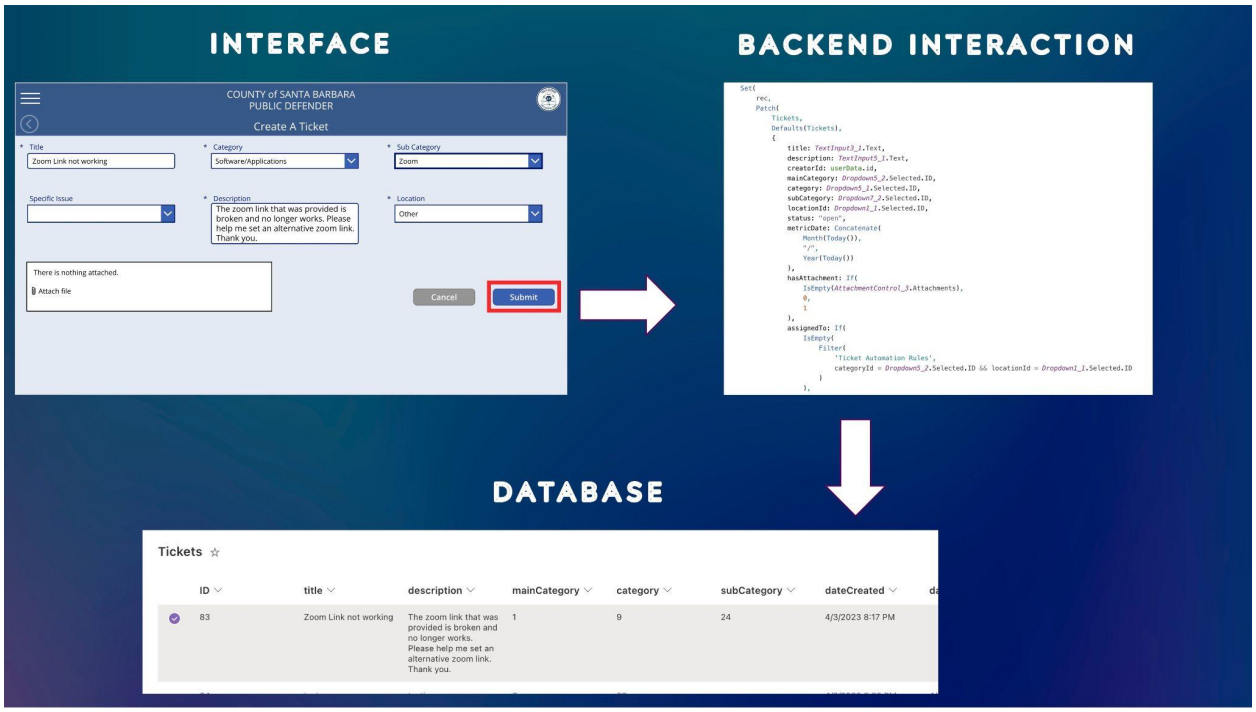

Figure 2: Interface to Database Interaction

When using the ticketing system, users will interact with the user interface to input data, including buttons, drop-down lists, combo boxes, and text boxes within a form. The user will then submit the form once all necessary data has been entered. After submission, the system moves into the data processing phase. Data processing involves taking the raw data input and transforming it into a format that can be stored and analyzed by the data storage system.

Backend Interaction: To ensure data integrity, we will validate each input to ensure it matches the correct data type for its corresponding column in the SharePoint list. Once we have confirmed the correct schema, we will use the built-in "Patch" function, which utilizes a key-value object, with the key being the column name and the value being the raw input data. When a user submits the form, this function will execute, and the next stage will be to store the data.

After executing the Patch function, a new record will be created in the specified List with a unique ID. Data retrieval is crucial to this stage as it enables users to access their data easily. Data retrieval involves querying the data storage system to retrieve specific data needed for the user interface or other system parts. Proper security measures were implemented to ensure that sensitive data was not compromised during the flow. When a user creates a new record in any list, a column is included to keep track of their User ID for later reference. This feature ensures that staff members have access only to their created tickets, based on the User ID, while techs and admins have full control access to retrieve all records.

3.3 Implementation:

This project was split into three sections to optimize our time for development**.**Ticket data, data management, and user experience. Each section plays a crucial role in presenting the progressions of the project as a whole.

3.3.1. Ticket Data

Ticket data is gathered through a form users fill out when creating a ticket. The key information of a ticket includes the title, description of the issue, issue category and subcategory, location of the incident, and any attachments the user would like to upload. In addition, tickets will be associated with assignee(s), tags that let technicians categorize tickets, and time stamps of when the ticket was created and updated. Once users submit their form, the ticket data is sent to the SharePoint list, creating a new item.

3.3.2. Data Management

Using the SharePoint Site, raw data from the SBPD Power Apps can be efficiently stored and processed in a SharePoint List.The process involves connecting to Power Apps PD HelpDesk and retrieving the raw data. This data can then be parsed and formatted into a structured database using Power Apps Patch Function. The structured data can then be pushed into the SharePoint Site, which gets routed to the correct SharePoint List to be securely stored and easily accessed.The SharePoint List can be designed to support the necessary data fields and relationships to ensure efficient and effective data management. This will enable the data to be easily updated and overwritten.

3.3.3. User Experience

An intuitive user interface was purposefully crafted to showcase the ticketing system visually engagingly. This UI empowers users to seamlessly interact with the framework's features without requiring prior knowledge or access to its underlying components. In addition, the application was to be made responsive to accommodate different devices and modern browsers. The design philosophy of the UI focused on the end users, specifically both older and younger users.

4. Conclusions

4.1 Results

We aimed to achieve four goals initially, and I'm proud to say that we have completed three of them. We designed and developed a highly functional ticketing system with an optimized user interface for mobile devices. Additionally, one of our sub-goals was to automate the ticketing process by implementing rules for assigning technicians to specific tickets based on their type, and we were successful in achieving this goal. We have constructed a robust database utilizing Microsoft SharePoint List, which is scalable and can accommodate increasing traffic demands as SBPD grows. Although we created a robust database using Microsoft SharePoint list, the application currently provides only basic data analysis. However, SBPD can expand the development and add more analytics to gain insights into trends.

The product solution produced by our project is a custom-built help desk ticketing system for the SBPD office that allows staff members to submit tickets for IT issues and allows technicians to resolve these issues efficiently. The system is built using Microsoft Power Apps and SharePoint, providing an intuitive user interface for staff members and advanced data management capabilities for technicians and administrators. The system includes features such as automated ticket assignment and a knowledge base for IT articles. Overall, the system provides a robust and efficient helpdesk ticketing system for the SBPD office.

4.2 Future

It's worth considering the potential impact that this project could have on the computer science field. In today's world, low-code platforms have become more accessible than ever before. As a result, building custom applications has become significantly easier, especially for people without a programming background. Our project is an excellent example of how these low-code platforms can be leveraged to create powerful applications without requiring a huge budget or extensive development time. By contributing to this growing list of successful projects, we hope to encourage others to explore the possibilities of low-code platforms and discover what they can achieve.

There are a few things to remember for future development using Power Apps. Firstly, it's important to ensure that everyone working on the project understands the Microsoft accounts they have, as only premium accounts can use premium connectors, such as an SQL database. This will save time and prevent unnecessary issues down the line. We had to switch from our SQL database to a SharePoint List since we did not have premium accounts. Luckily we could port over the same schema; however, all connections in Power Apps had to be updated to use the share point list connection. Another consideration is the potential issues with the GitHub beta program. While it's a useful tool, it's still in beta, and we encounter some merging conflicts when merging our code from the dev branch to the master branch. In these situations, we needed to force merge to the master branch to work around the issue. Also, fixing merge conflicts can be

challenging since the code is often hashed and unreadable. Finally, it's important to ensure that the SharePoint List permissions are set up correctly. This will prevent unauthorized access to the data, as people can go to the SharePoint site itself and see all the data if permissions are not set up. These points will help future projects avoid facing the same issues as ours.

5. References:

Microsoft Power Apps documentation: https://learn.microsoft.com/en-us/power-apps/

ZenDesk: https://www.zendesk.com

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

Power Apps Community: https://powerusers.microsoft.com/t5/Power-Apps-Community/ct-p/PACommunity

Microsoft Learn: https://docs.microsoft.com/learn/browse/?products=power-apps

Shane Young's YouTube channel: https://www.youtube.com/user/shaney1220

Power Platform YouTube Channel: https://www.youtube.com/c/PowerPlatform

Reza Dorrani's blog: https://rezadorrani.com/