Omniscient File Assistant

by

Cameron Bergman, Benjamin Evans, John Manny

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Cameron Bergman	04/12/2021
Cameron Bergman	Date
Benjamin Evans	04/12/2021
Benjamin Evans	Date
John Manny	04/12/2021
John Manny	Date
	04/12/2021
Ryan Moore, Faculty Advisor	Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2021

Table of Contents

List of Illustrations

Tables

Figures

Abstract

Omniscient File Assistant is a file management platform created for easy monitoring of file systems of remote end-user workstations. With the number of employees working from home now eclipsing 66%, the likelihood of insider misuse is at an all-time high as employees may struggle to balance their new work life (Herhold). Through its monitoring and analysis features, OFA allows an organization to mitigate improper device usage and data storage before serious security incidents arise. Capabilities include regular metadata collection, centralized data storage and reporting, file matching via hashing and other criteria, file content scanning, server backups, and a robust searching tool. By utilizing Omniscient File Assistant, an organization can lessen their chances of having end-user issues by preventing long-term storage of personal files, storage of restricted materials, and unauthorized modifications. Omniscient File Assistant provides a one stop interface for administrators to log, flag, track and remove files at their discretion.

Introduction

Problem Statement

"We found that 66% of employees currently work from home at least part of the week as a result of the coronavirus outbreak" (Herhold). With more employers implementing work-from-home policies, insider misuse of company equipment/time poses a threat to these businesses now more than ever. Reported by Computer Economics Avasant Research, "... adapted from the executive summary of our full report, Insider Misuse of Computing Resources, which analyzes 14 forms of insider misuse in detail," these vary everywhere from portable storage misuse, remote-access programs, and business email misuse. While administrators have some control of the end-user's computers, there is no concrete way of knowing exactly what is being put onto company equipment in an easy to navigate fashion. This could result in countless amounts of security breaches, lost data, and insider misuse situations. It is important to not only teach end-users what kind of behavior is acceptable while on their work machines, but also to monitor the data they store on their machines to mitigate any potential misuse.

Our Solution

Omniscient File Assistant is a management platform that allows IT administrators to monitor the files on end-user workstations with confidence. At its core, Omniscient File Assistant is a file monitor installed on end-user machines that allows for data collection of file name, location, size, computer name, owner and more to be sent to a centralized database for reporting. When administrators feel it is necessary to check how his or her end-users are utilizing

the company's equipment, Omniscient File Assistant will provide an interface to query a database that keeps a running list of a items located and their associated metadata.

To capitalize on the monitoring capabilities of the end-user workstations, Omniscient File Assistant's management portal offers file auditing and data loss prevention services. Through backend programs, configured in the management portal, additional analysis, collection, and reporting can be enabled on a per file, computer, or location basis. Additional features include scanning file content for patterns, such as social security or credit card numbers, alerts for specific files located on machines, and auditing functionality. To facilitate a more robust management experience, Omniscient File Assistant offers server backup of files, for easier hands-on inspection, multiple ways to verify files, via hashing or other data checks, and a comprehensive search functionality.

Project Goals

Develop a management application that will provide IT administrators the ability to configure and view reports of end-user files.

Data collection is handled by a Windows service, written in C# and compatible with 64-bit versions of Windows 10. Data storage is provided by a Microsoft SQL database. And the webservice utilizes NodeJS for the backend and ExpressJS for the frontend, running on a Windows Server virtual machine that is hosted in Azure. Together, these components work cohesively to provide administrators with the following features:

- Reports and alerts, giving administrators an overview of categories and parameters deemed important.

- Configuration of data to be collected, giving administrators control over when, what, and how much data is needed for their organization.

- The collection of and direct viewing of indicated files, giving administrators the ability to inspect the content of files.

- Time and logic-based toggles and options for scanning an end-user machine, allowing administrators to manually disable, and run file scans.

Overview

The remainder of this report outlines, in detail, how the project was completed. The report includes the following sections: design objectives, methodology, budget, timeline, problems encountered, and future recommendations.

Discussion

Project Concept

Omniscient File Assistant was inspired by John Manny's time while serving as end-user support. Often, visitors of the helpdesk would complain of downloads no longer working, and constant alerts for failed updates. Although mundane for a technology professional, end-users frequently feared and questioned what the problems could be, as they believed they had done nothing wrong. After further inspection by IT support, it was commonly found these users stored a plethora of non-work-related files on their computers. Typical culprit files included family photos, restaurant menus, and seemingly random pictures from the Internet. Although it is commonplace for IT personnel to have the right to confiscate and remove data from company

technology, as a clause in the organization's Acceptable Use Policy, doing so would typically upset end-users and cause distrust to form between departments.

The team began initial communication via Microsoft Teams, sending instant messages to discuss any potential topics that might have piqued any interest. Remembering his time at the helpdesk, John asked if the group had any odd scenarios involving file storage on user devices. Cameron stated how he often felt uncomfortable when his tasks involved looking through the images stored on people's computer, and Ben voiced the frustration he felt when having to explain directory structures while helping end-users find lost files. The group then agreed the project should revolve around resolving the potential stresses of end-user storage.

The group came to the consensus that the purpose of IT is to facilitate business, not disrupt it. Because of this, the team believed a method of mitigation or prevention would be best, to ensure IT's responsibility of data and while maintaining a positive organizational culture. The team decided that intervening with a user before he or she accumulated too many files would be the best way to prevent abrupt interactions with IT that would require deleting files without a backup.

Design Objectives

In this section, we will discuss the design of our project, with special focus on important characteristics. The main design objective for the project did not change during the design process. From the beginning, the goal was to always provide IT professionals with a tool to effectively monitor the usage of company devices without causing disruption by physically watching over the users. The tool was designed to be a one stop shop where administrators can

scan, view, flag and remove files on client devices with ease. Deployed as a Windows service on client devices, the tool scans file metadata and sends any data collected to a database. Admins can also determine criteria for what files will be flagged, making the process as simple as possible.

The best way to carry out the design was to host it on the web. Using Azure, the team created a web server to host the admin interface, and Microsoft SQL Express to act as the SQL database. This design provided the team with the most functionality and flexibility. The team could scale the database up or down if needed, add additional web servers to distribute the load for the application and even implement any additional functions or features at the team's discretion. This design allowed the team to manage the project easily with very low cost and very high potential.

Methodology and Technical Approach

The following information is a breakdown of the technical elements of Omniscient File Assistant and how they work together.

Windows Service

Performing all the data collection is a Windows service, programmed in C#, that is installed on workstation PCs. As the service runs in the background, there are no graphical or command line interfaces for the end-user to directly interact with. By default, the service is always running and will actively scan the computer at intervals set in the administrative web

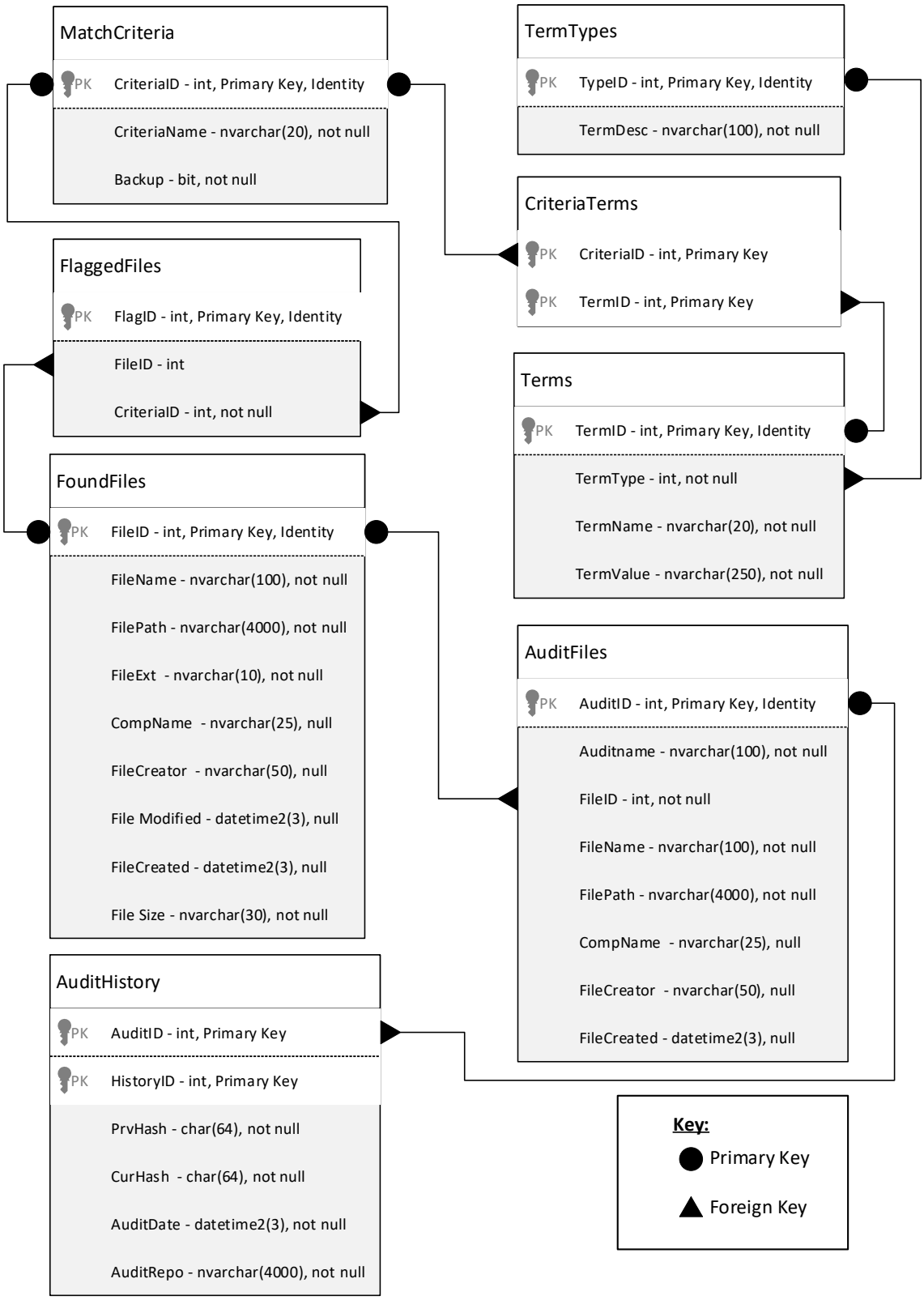
portal. Rather than only running when a scan is scheduled, the service stays on to allow for impromptu configuration changes or scans initiated by the administrative web portal.

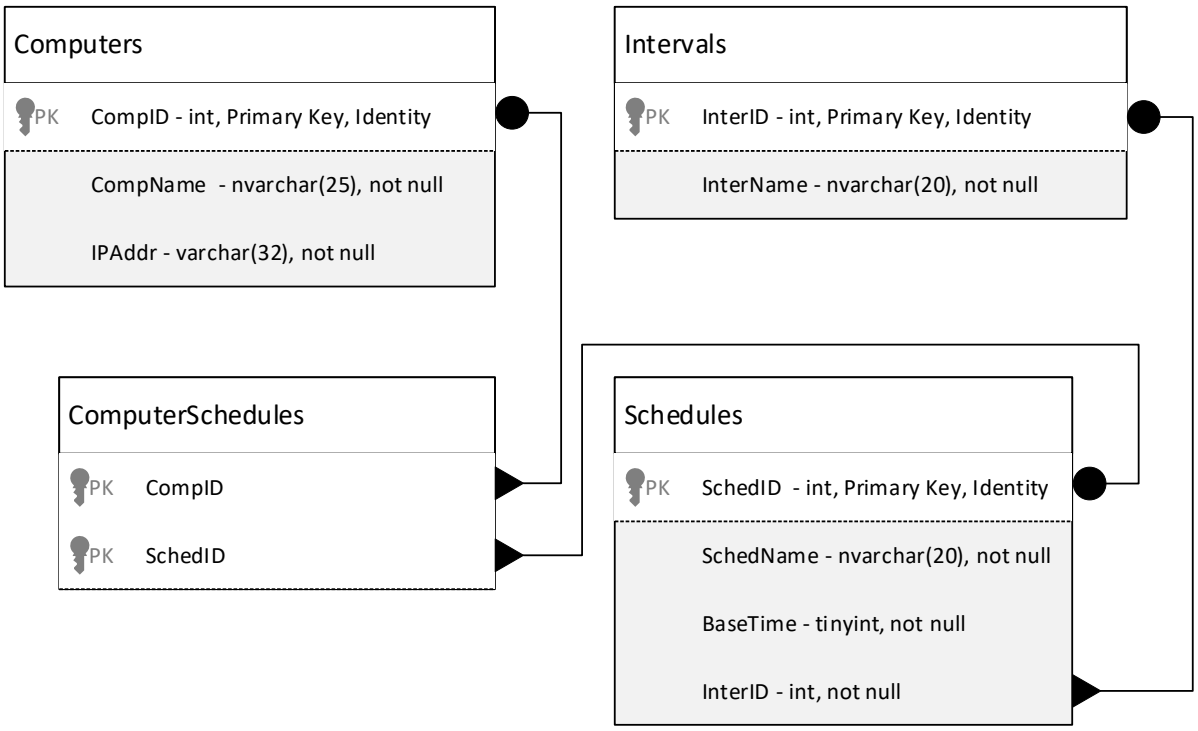
The scanning feature collects the follow items: file name, location, extension, size, computer of origin, creator account, date of creation, date of last modification, and content of file. In combination with configurations from the administrative web portal, the service can search for patterns in file contents, report file modification for auditing, and send files to a central repository for future inspection.

Database

To house the data collected is a SQL Express database, located in an "on-premises" server hosted in an Azure environment. Microsoft Azure was chosen as the platform as it allowed for easy scalability, an essential feature to allow the Windows service to be deployed across multiple campuses. The database schema for Omniscient File Assistant can be seen in Figure 1.

Figure 1: Database Diagram





Administrative Web Portal

As the web portal is the central location for configuration, data queries, and live updates, it was built upon a NodeJS environment utilizing ExpressJS, for the web framework, and EJS, for building dynamic pages. The web portal consists of four main sections, each selectable from a vertical navigation bar: Dashboard, Scanning, Patterns, and Auditing. The Dashboard serves as the portal's homepage with customizable widgets to allow for quick information updates when first visiting the site. Scanning is the dedicated page to querying all the files found. In the center of Scanning is the large table that will populate with data that matches the filters applied by the query search feature. Above the table is the previously mentioned search feature that will allow users to filter items based on the different metadata collected from files.

At the root of the Patterns page is where users can select to view currently configured pattern searches, create new pattern searches, create new terms, modify current pattern searches, and modify current terms. Patterns consist of one or more rules, defined as terms, that must be matched for a file to be flagged. Terms typically consist of specific strings or patterns in the document and can be added to as many patterns as need, allowing for special configuration for files with multiple violations or refinement of items flagged. Lastly, the Auditing page facilitates a user to configure and view the audit history of files. Once a file is requested to be tracked, a SHA2 hash of the file is created and saved in the database. On every successive scan the hashes are compared to check whether or not the file has been modified. If the file has been included in an audit or pattern search, a copy request can be toggled to have the windows service send a copy of the file to a network share for viewing by an IT employee.

User Profiles

The following tables breakdown a different type of end-user of Omniscient File Assistant, specifying considerations made and expected interactions.

Table 1: User Profile – System Administrator

User Profile – System Administrator
Application: Microsoft Azure, GitHub, Express.JS, Node.JS, C#, SQL, Linux command line, HTML, CSS.
Potential Users: System Administrators

Software and Interface Experience: As OFA utilizes a web portal with GUI elements, it is best for administrators to be familiar with administrative pages and forms commonly found in other software. If the administrators wish to customize OFA, familiarity with JavaScript and SQL is needed.
Experience with Similar Applications: Experience with SQL databases used and maintained by software. Interfering/manipulating databases that are regularly being used by software can cause issues. It is best for the user to understand how to manage a SQL database that is primarily managed by software. Experience with inventory systems would be beneficial as OFA operates as an inventory manager of files.
Task Experience: Comfortable with applying search filters, creating task schedules, monitoring alerts, and utilizing audits. Basic web page interactions: mouse to click on specific areas and keyboards to type text strings.
Frequency of Use: Administrator would use OFA at least once a day to gauge the software's performance. Additional use would be based on conditional needs, but semi-hourly viewing intervals would be considered normal.
Key Interface Design Requirements that the Profile Suggests: The user will need some knowledge of C#, JavaScript, and SQL to properly use/maintain Omniscient File Assistant. Large areas dedicated to viewing results, while also having space for options and site menus.

Table 2: User Profile – Help Desk Technician

User Profile – Help Desk Technician
Application: Omniscient File Assistant
Application: Microsoft Azure, GitHub, Express.JS, Node.JS, C#, SQL, Linux command line, HTML, CSS.
Potential Users: Helpdesk Technicians
Software and Interface Experience: Familiarity with SQL select statements and filterable queries would allow for optimal use of the reporting features.

Experience with Similar Applications: Software that Utilizes Web Based Administrative Portals • Office 365 • TeamViewer • Jenkins
Task Experience: Comfortable with applying search filters, creating task schedules, monitoring alerts, and utilizing audits. Basic web page interactions: mouse to click on specific areas and keyboards to type text strings.
Frequency of Use: Technicians will use OFA at the same rate as they work with clients, as they will check the software for any alerts that may be correlated to the client visiting. As helpdesk technicians can see anywhere from 0 to 20 people in an hour, it is expected a technician would use OFA 0 to 20 times an hour.
Key Interface Design Requirements that the Profile Suggests: • Quick loading search results. • Large viewing space for reports • Easily identifiable filters that can be toggled and created.

Table 3: User Profile – End User

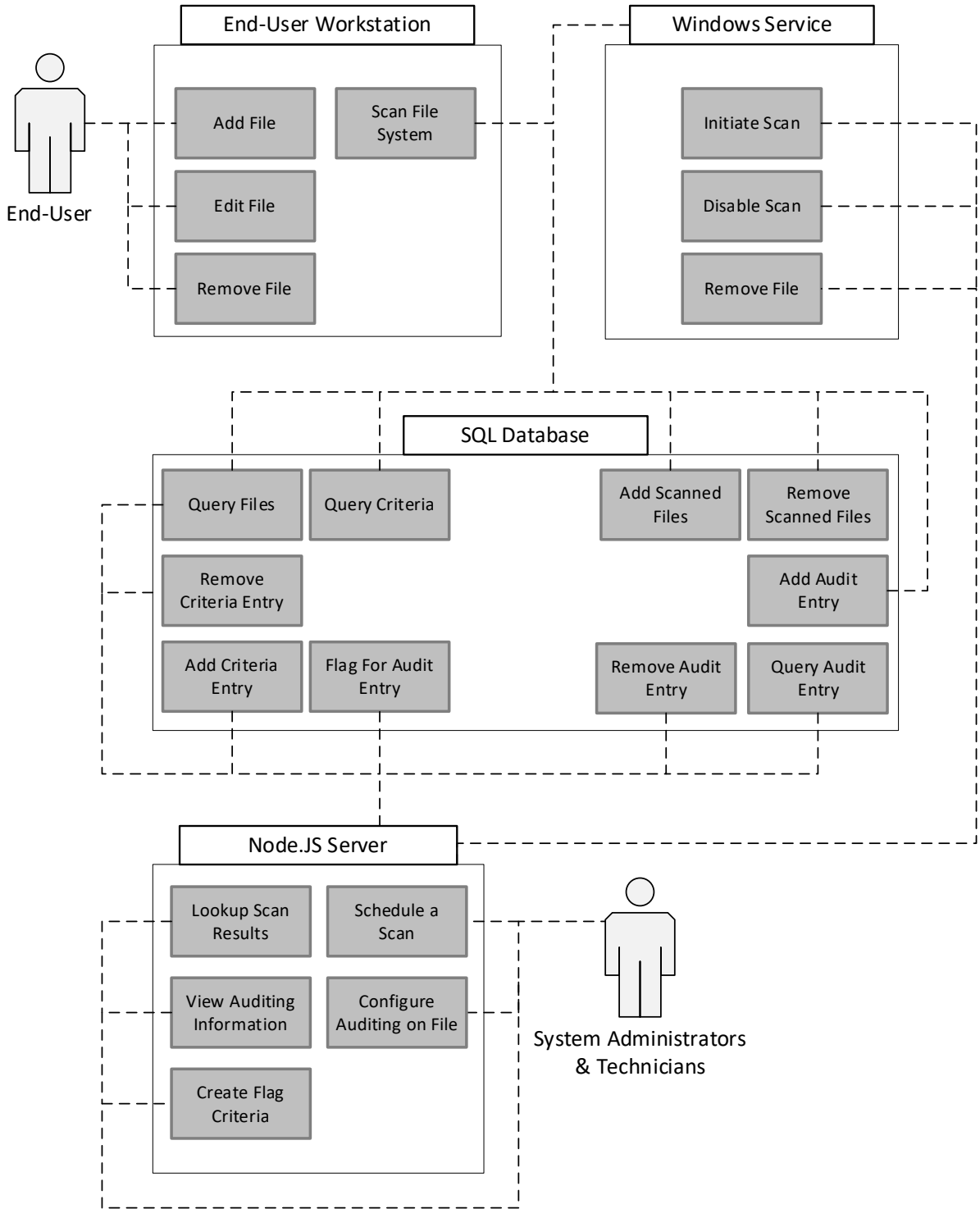
User Profile – End User
Application: Omniscient File Assistant Windows Service
Potential Users: End-users
Software and Interface Experience: Proper file creation, modification, and saving operations.
Experience with Similar Applications: Applications that Run in the Background • TeamViewer (Host Module) • DyKnow

Task Experience: Common File Operations • Create a new file • Modify a file • Delete a file • Open a file • Save a file
Frequency of Use: The frequency at which the Windows service scans the machine is at the rate the administrators set in the web portal. The end-user will be using the workstation during his or her work hours, but the scans occur at pre-determined intervals.
Key Interface Design Requirements that the Profile Suggests: No errors or pop-up should appear on end-user environments. Any information produced by the software should either go to Event Viewer or the Administrative Web Page.

Use Case Diagram

The following diagram (Figure 2) is a visual representation of the interactions between the end-user, workstation, Windows service, database, administrative portal, and IT staff.

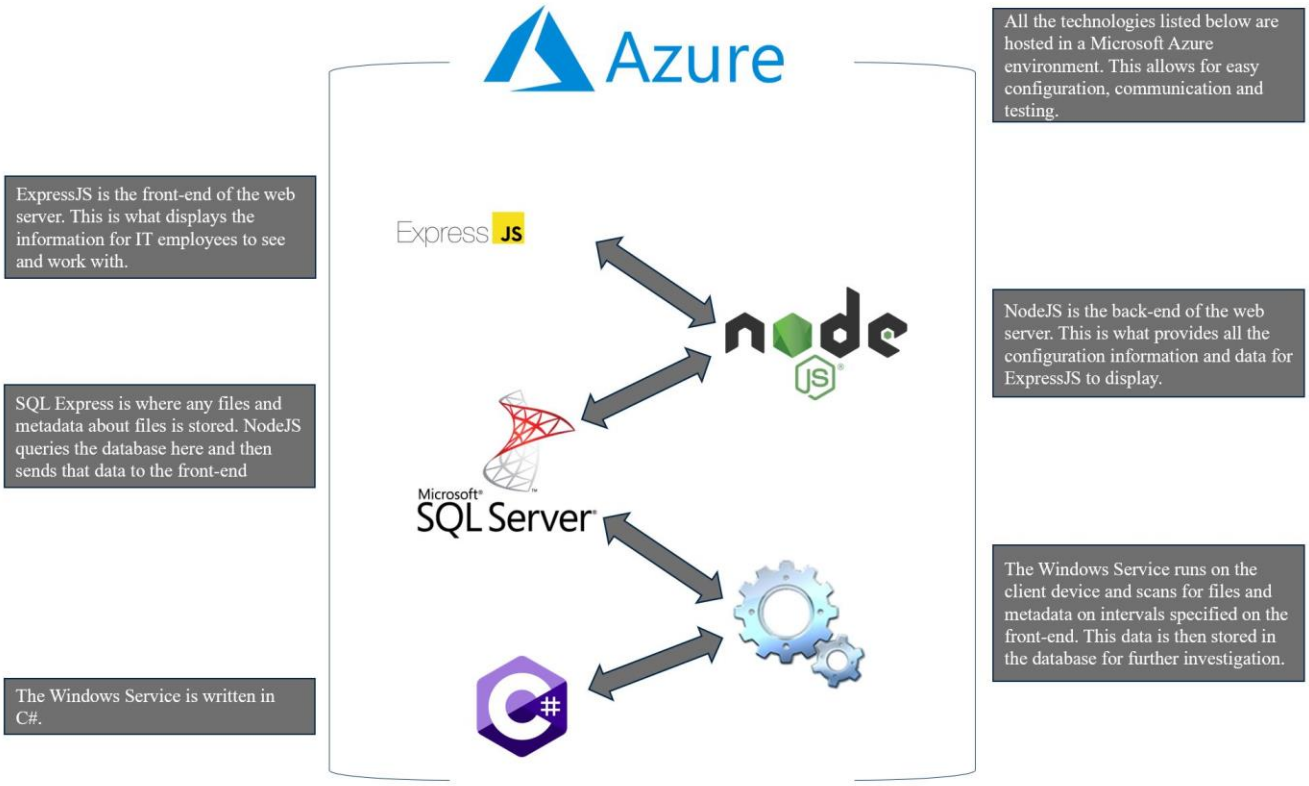
Figure 2: Use Case Diagram



Technical Architecture

The following diagram (Figure 3) shows the relationship between each technology used in Omniscient File Assistant, along with the individual functionalities they provide. Besides each technology is a brief description that encapsulates this information.

Figure 3: Technical Architecture Diagram



Testing

The following section has been created to detail how Omniscient File Assistant was tested and the results of those tests. The first section, Testing Methodology, defines what aspects

of the tests were monitored, how data was recorded, and how interactions with test users would occur. The Scope of Testing section outlines what areas of the software were tested. The Objectives define what specific actions were desired to test and how these helped the project. Test Results and Procedures is where the official observations and results of each test were recorded. Lastly, the Review section summarizes what was learned from the testing and how the team utilized what they learned.

Testing Methodology

As a software designed to assist the burden of tasks asked of help desk technicians, tests for Omniscient File Assistant were designed for speed and ease of use. Test users were given a list of ten tasks to be completed from the web portal. As Omniscient File Assistant is a new service for the users, small hints were present at the bottom of the sheet, acting as pseudo documentation which would be available to users in a production environment. Administrators of the tests also functioned as observers, recording necessary data. Observers timed the intervals between the completion of each task while observing mental or emotion reactions during the test. Test users were encouraged to complete all task but were only required to attempt 1 minute and 30 seconds for a task before skipping to the next. Observers also recorded what they believed the test user's certainty when completing each task. A user's certainty was gaged by a question given after each task. These questions consisted of the test user providing a single piece of data that they retrieved during the task. A correct answer was not required to move onto the next question, its purpose was to ensure the user understand what the goal of a task could be.

Scope of Testing

As Omniscient File Assistant only requires interaction from a help desk technician position, the scope of the test was to evaluate the administrative side of the software. The tests revolved around the following features: Using different fields to query metadata about found files, creation of Match Criteria to search for new items in files, the inspection of flagged files, and viewing of audit history. These would be the go-to actions for a user of Omniscient File Assistant, making them ideal for testing.

Objectives

- Test a multitude of search queries, searches must include different combinations of fields and values supplied. Allows the team to test stability and usability of software when user error occurs.
- Identify all bugs that need to be resolved before the IT Expo.
- Test other administrative functionalities, such as creating patterns and audits. Allows the team to test stability and usability of software when user error occurs.

Test Procedures

1. Search for all files with an extension of .rdp

   Question: What date was the file with a size of 85 bytes created on?
2. Find all the files created on by NT AUTHORITY\SYSTEM

   Question: How many files retrieved are images?
3. Look for empty files. (File Size = 0 Bytes)

   Question: What is odd about the files found?

4. Look for empty .log files.

   Question: What is the name of this/these file(s)?

5. Create a new pattern to search for named Possible Password that searches for all indicators of a password. Indicate the files should be backed up upon matching the pattern.

6. Create a new pattern, named Credit Cards & PIN to search for credit card numbers with PIN numbers. Include all possible abbreviations of credit card. Ensure the files are NOT backed up upon matching the pattern.

7. Find out how many computers are on the Johns PC v2 Scanning Schedule.

8. View all files that have been flagged by the PasswordsOnly pattern. Download the first file retrieved.

   Question: What was the password in the file?

9. Look at the most recently scanned files.

   Question: What day where the 3 most recently scanned files modified?

10. Add an auditing requirement for the NewFiles.txt file and name the audit New File Audits.

Hint: Search by FileName.

Test Results

Table 4: Test User A – Administrator: John Manny

Task #	Time to Complete	Answer Correct?	Additional Notes
1	46 Seconds	Yes	Did not know to go to the Scanning page.
2	1 Minute 25 Seconds	Yes	
3	1 Minute 5 Seconds	Yes	

4	24 Seconds	Yes – but did not adjust the query to filter results.	
5	1 Minute 19 Seconds		
6	43 Seconds		
7	21 Seconds	Yes	
8	1 Minute 12 Seconds	Yes	
9	44 Seconds	Yes	
10	1 Minute 59 Seconds		Did not click the button to confirm the file option. (Had to redo.) Typed the AuditName before searching for the file which reset the AuditName when he searched for the file.

Table 5: Test User B – Administrator: Cameron Bergman

Task #	Time to Complete	Answer Correct?	Additional Notes
1	1 Minute 45 Seconds	Yes	Had to tell him where to start but was able to figure it out once I showed him where to go.
2	4 minutes 30 seconds	No	He was not sure how to make the syntax to get the search to work.
3	1 minute 35 seconds	Yes	
4	1 minute 10 seconds	Yes	
5	55 seconds		
6	43 seconds		
7	56 seconds	Yes	
8	1 minute 15 seconds	Yes	

9	38 seconds	Yes	
10	2 minutes 3 seconds		Took a couple of tries, but eventually figured out the syntax.

Table 6: Test User C – Administrator: Benjamin Evans

Task #	Time to Complete	Answer Correct?	Additional Notes
1	69.44 Seconds	Yes	Pressed 'Search' before entering info
2	126 Seconds	Yes	Searched by FilePath at first
3	87.77 Seconds	Yes	Searched 0 KB first
4	71.24 Seconds	Yes	No Issues
5	107.73 Seconds		
6	43.05 Seconds		
7	180.27 Seconds	Yes	Tried to search on Scanning page
8	60.51 Seconds	Yes	
9	125.63 Seconds	No	Did not go to Dashboard first
10	300 Seconds		Gave up. Went through process but could not find New Audits that were created

Review

The tests revealed the project's problems with user experience. Often, test users attempted tasks multiple times incorrectly. These incorrect attempts were typically the result of

poor labelling and layout of the web portals interface. Common issues include not knowing of a necessary button or option, poorly titled pages, and not understanding the purpose of a feature.

In the future, the team will be changing the naming scheme of our core features in Omniscient File Assistant. To some, scanning, patterns, and auditing are almost synonyms if the user is unfamiliar with the tool. The team will also focus on developing a more robust set of documentation. Without definitions for key terms of the tool or examples for each possible action, users felt lost when given a task. As a configurable tool, clearly delineating what is and what is not possible will allow users to become more comfortable.

Budget

The following tables contain the estimated and final budgets of the Omniscient File Assistant project.

Table 7: Estimated Budget

Budget: Omniscient File Assistant				
No.	**Item**	**Unit/Hours**	**Unit Price**	**Line Item Total**
Hosting				
1	Monthly Azure Hosting	5	$50.00	$250.00
	Subtotal			**$250.00**
Labor				
6	Website Development	300	$25.00	$7,500.00
7	Database Development	200	$25.00	$5,000.00
8	Windows Service Development	500	$25.00	$12,500.00
	Subtotal			**$25,000.00**
	Total			**$25,250.00**

Table 8: Final Budget

Budget: Omniscient File Assistant				
No.	**Item**	**Unit/Hours**	**Unit Price**	**Line Item Total**
Hosting				
1	Azure Hosting (November)	1	$42.22	$42.22
2	Azure Hosting (December)	1	$51.12	$51.12
3	Azure Hosting (January)	1	$66.25	$66.25
4	Azure Hosting (Feburary)	1	$86.90	$86.90
5	Azure Hosting (March)	1	$143.47	$143.47
	Subtotal			**$389.96**
Labor				
6	Website Development	750	$25.00	$18,750.00
7	Database Development	400	$25.00	$10,000.00
8	Windows Service Development	600	$25.00	$15,000.00
	Subtotal			**$43,750.00**
	Total			**$44,139.96**

Project Timeline

The following table outlines the team's schedule when developing Omniscient File Assistant. The data presented has been modified to represent any deviations from the original timelines that were created during early stages of the project.

Table 9: Project Timeline

Task #	Task Name	Duration	Start Date	End Date
Task 1	Formally Declare Team and Project	1 Day	08/24/2020	08/24/2020
Task 2	Team Contract 1st Draft	7 Days	08/24/2020	08/31/2020
Task 3	Project Abstract for Tech Expo	42 Days	08/31/2020	10/12/2020
Task 4	Team Contract 2nd Draft	42 Days	08/31/2020	10/12/2020
Task 5	3-Minute Elevator Speech	7 Days	10/12/2020	10/19/2020
Task 6	User Profile	7 Days	10/12/2020	10/19/2020
Task 7	Use Case Diagram	7 Days	10/12/2020	10/19/2020

Task 8	Draft Report	21 Days	10/19/2020	11/09/2020
Task 9	Oral Presentations	28 Days	10/19/2020	11/16/2020
Task 10	Final Fall Semester Report	28 Days	11/09/2020	12/07/2020
Task 11	Design Database	36 Days	9/14/2020	10/19/2020
Task 12	Design Node.JS Environment	36 Days	9/14/2020	10/19/2020
Task 13	Design Express.JS	36 Days	9/14/2020	10/19/2020
Task 14	Design Windows Service	36 Days	9/14/2020	10/19/2020
Task 15	Azure Setup & Configuration	36 Days	9/14/2020	10/19/2020
Task 16	Develop Node.JS Environment	75 Days	9/17/2020	12/01/2020
Task 17	Develop Express.JS	75 Days	9/17/2020	12/01/2020
Task 18	Setup Database	75 Days	9/17/2020	12/01/2020
Task 19	Develop Stored Procedures	75 Days	9/17/2020	12/01/2020
Task 20	Develop Windows Service	75 Days	9/17/2020	12/01/2020
Task 21	Web Server Testing	2 Days	01/11/2021	01/13/2021
Task 22	Database Testing	4 Days	01/14/2021	01/18/2021
Task 23	Windows Service Testing	2 Days	01/19/2021	01/21/2021
Task 24	Interconnectivity Testing	2 Days	01/25/2021	01/27/2021
Task 25	Penetration Testing	11 Days	01/28/2021	02/08/2021
Task 26	Bug Fixing	41 Days	01/11/2021	02/21/2021
Task 27	Create Diagrams	14 Days	02/22/2021	03/08/2021
Task 28	Create Expo Presentation	7 Days	03/08/2021	03/15/2021

Problems Encountered and Analysis of Problem Solving

This section looks at the challenges the team faced throughout the development of the project. The biggest issues encountered can be categorized by a lack of experience, a lack of foresight or a lack of resources.

The first challenge the team faced, which continued to be a hurdle throughout, was a simple lack of experience. Unfortunately, none of the group members had ever been a true developer. Each member had experience scripting and some fundamentals for each language used in the project, but no one had ever built a dynamic web site or Windows service from

scratch. To overcome this obstacle each member put aside 4 – 5 hours a week during the early weeks of brainstorming to educating themselves with the items required. Resources used include LinkedIn Learning, W3Schools, Tutorials Point, YouTube, and Stack Overflow. As with any educational process, mistakes were made. To compensate, team members were encouraged to openly disclose when they had come across an obstacle.

The next challenge the team faced was a lack of foresight. As with any project, everything was planned a certain way, but adjustments and adaptations had to be made. To begin with, these adaptations were not accounted for, and this made it difficult to change the design on the fly. One specific example of this is when the database server had to be changed. One thing the team decided to implement was a single-sign-on feature on the database, that way users could log into the Windows VM(s) and use those credentials to connect to the database. What the team did not know then was that the original database server, which was an Azure SQL Database, did not support this feature. The only way to get around this was to create an on-premises database on the Domain Controller VM in Azure. This took a good amount of time but was successful in the end. In the future, these kinds of details should be determined before setting up any infrastructure.

One last challenge that was encountered was an unfortunate lack of resources. The team had a lack of experience building a project of this scale, so to make things easier, it was decided that everything would be built through Azure. Unfortunately, everything Azure offers is not free and there was limited amount of money to spend. The team was only able to use the bare-minimum resources so that costs could be as low as possible. This included steps like only having one server for everything and one test VM. Of course, this is not ideal, but for testing, this

was enough. Ideally, there would have been multiple servers and many VMs to thoroughly test with.

Recommendations for Improvement

In later version of Omniscient File Assistant, the team would like to implement a more robust inspection system for the files on a computer. Currently, files must contain all the terms of a pattern in order to be flagged. This is very limiting as it is the equivalent of only utilizing an AND logical conjunction between each term. While we could implement our own custom methods for allowing other logical conjunctions, such as OR, NAND, and NOR, tools already exist for these functions.

Namely, the team was interested in implementing Yara, an open-source tool by VirusTotal used for identifying and classifying malware samples. While Omniscient File Assistant is not designed for malware analysis, all the features of Yara perfectly suit the needs of Omniscient File Assistant. Along with the features already present in Omniscient File Assistant, Yara also includes the abilities for binary inspection, logical grouping of rules, and the aforementioned advanced logical conditions. With Yara, Omniscient File Assistant, would be a more capable system allowing administrators to create patterns of greater complexity.

Conclusion

Throughout the project's development, the team was challenged with tasks no one had experienced before. The team consisted of zero classically trained developers, and as such, the

project was a constant learning experience that required collaboration to overcome the unforeseen obstacles of developing from the ground up. The following section covers the lessons learned and skilled developed that the team will take with them because of this project.

Lessons Learned

Throughout this project we learned about the preliminary research needed to complete a development project from scratch. Due to the interconnectivity for each of the three products, web portal, SQL database, and Windows service, each part needed to be developed co-dependently. As such, different restraints that originally inhibited only one of the products now inhibited all. To mitigate constant obstructions, substantial time and effort was added to the early development period to ensure all features, restraints, characteristic, and designs were full realized.

Skills Developed During the Project

During the development, the team was exposed to new technologies. When building the web portal, new skills were developed in utilizing NodeJS, JavaScript, middleware such as ExpressJS, and templating engines. Skills were refined and built within Azure SQL, as much of the setup required heavy interconnection of tables, lengthy SQL commands, and access control. C# programming skills were developed during the building of the Windows service.

References

Herhold, Kristen. “Working From Home During the Coronavirus Pandemic: The State of Remote Work.” Clutch, 16 Apr. 2020, clutch.co/real-estate/resources/state-of-remote-work-during-coronavirus-pandemic.

“Security Threats in Employee Misuse of IT Resources.” Computer Economics , Mar. 2020, www.computereconomics.com/article.cfm?id=1436.