

Post Implementation Report

By Collin Walker

Contents

[Project Purpose 3](#_Toc69320180)

[Technical Description 3](#_Toc69320181)

[Datasets 5](#_Toc69320182)

[Data Product Code 7](#_Toc69320183)

[Hypothesis Verification 7](#_Toc69320184)

[Effective Visualizations and Reporting 8](#_Toc69320185)

[Accuracy Analysis 10](#_Toc69320186)

[Application Testing 12](#_Toc69320187)

[Application Files 13](#_Toc69320188)

[Client Files 14](#_Toc69320189)

[Server Files 15](#_Toc69320190)

[User’s Guide 17](#_Toc69320191)

[Client Installation 17](#_Toc69320192)

[Automatic 17](#_Toc69320193)

[Manual 17](#_Toc69320194)

[Client Administration 18](#_Toc69320195)

[Starting and Stopping Services 18](#_Toc69320196)

[Checking Logs 19](#_Toc69320197)

[Checking for updates 20](#_Toc69320198)

[Web Portal Access 21](#_Toc69320199)

[Authentication 21](#_Toc69320200)

[Navigation 21](#_Toc69320201)

[Forecasting 22](#_Toc69320202)

[Client Uninstallation 23](#_Toc69320203)

[Automatic 23](#_Toc69320204)

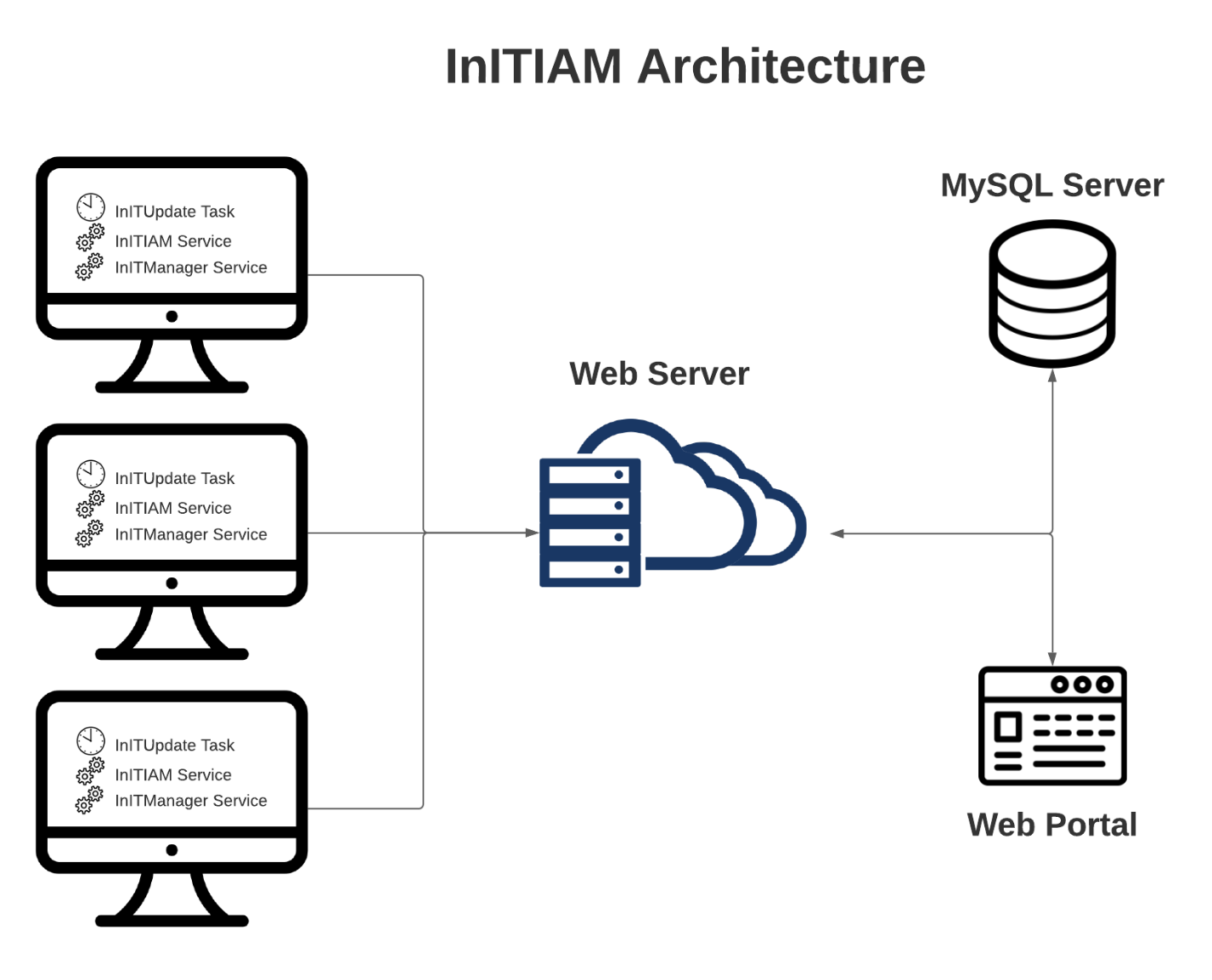
[Manual 24](#_Toc69320205)

[Summation of Learning Experience 25](#_Toc69320206)

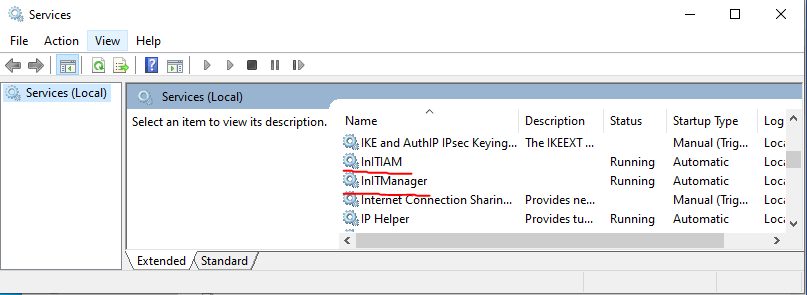
# Project Purpose

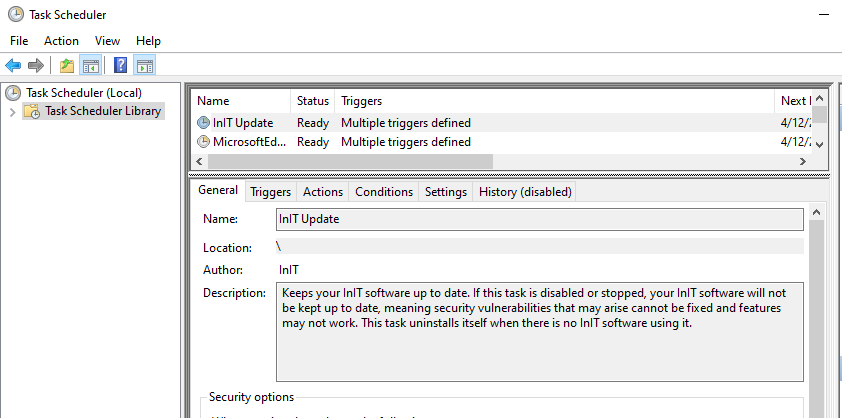
The project’s purpose was to provide an IT organization the means to monitor and manage their computer assets, as well as provide an interface that displays reports, predictions, and would allow clients to interact with IT services developed in the future. To effectively manage assets, it was imperative to include a method to deploy automatic updates to the software.

# Technical Description

The product architecture was built with security in mind, due to the value technology assets expose within an organization. At the heart of InITIAM is a secure server hosted on the cloud with only port 443 (https) opened. This server houses the web server and database server. The web server is the central hub of InITIAM, acting as the middleman between the clients, the database server, and user interaction.

*Figure 1: InITIAM Architecture*

The clients are built on three pieces of software: two background windows services and an executable tied to a scheduled task with the purpose of managing updates. The background windows services were built with C++, chosen for efficiency and ease of Operating System integration. InITIAM is the primary IT Asset Management service, written to be able to monitor the workstation with the ability to add administration capabilities in the future. InITManager is the secondary service that functions as a controller and watch-dog over the InITIAM service. InITManager’s entire purpose is to execute functions related to ensuring InITIAM’s integrity. An example would be executing a remote administrative request to restart the InITIAM service. The final component to the client software is InITUpdate, an executable that upon installation creates a Windows Scheduled Task with logon and time-based triggers. Upon being triggered, InITUpdate will communicate with the web server, ask if any updates exist for this specific client, and download and install if so. A Windows application installer was created in both executable and MSI formats to install InITManager, InITIAM, and InITUpdate.

*Figure 2: InIT Client Services*

*Figure 3: InITUpdate scheduled task*

The web server was built in PHP. The primary function of the web server, in relation to the clients, is to receive GET and POST requests for the various client functionality like checkins, updates, and client logging. Once these GET and POST requests are received, the web server can retrieve any needed data from the database server, invisible to any outside eyes. All communication takes place over encrypted https to ensure data security.

The final purpose of the web server is to provide a separate web portal for administrative user interaction. To access the web portal, users must authenticate with a username and password, which is then checked against a secure password hash stored in the database. Hashing the password will ensure user passwords stay safe in the event of a data breach. Upon successful logon, the user is presented with an administrative dashboard featuring three pre-configured charts about the InITIAM client data. In this portal the user can navigate to a database administration page, and a data forecasting page. The database administration page is simply a link to the Adminer home page for the InIT MySQL database. The forecasting page hosts a chart that initially shows the disk space usage for a client (selected by a drop-down menu of all clients from the database). After filling out time parameters and the usage increase or decrease, the page allows the user to project when the client’s diskspace will reach capacity and be unusable.

*Figure 4: InITIAM Web Authentication Portal*

The database server is a MySQL database server configured with the Adminer front end. The Adminer front end allows for a graphical user interface over https, which is helpful given the security configuration of the server.

# Datasets

The data collected by the IT Asset Manager consists of three categories: client activity, asset logging, and update requests.

Clients check in with the web server every 5 minutes. This allows for a high-level view of a computer’s power state, or at least internet connectivity. The client check-in is identified by the ClientID, a 128-bit GUID assigned to the asset, and the service name identifier. Upon check-in, the current date and time is recorded.

Asset logging is achieved by InITIAM executing a long series of Windows Management Instrumentation API calls, retrieving detailed information about the asset’s hardware, software, and user information. Below is a list of all the data being polled at 10-minute intervals.

Hardware:

* Computer Name
* Manufacturer
* Model
* Domain
* Workgroup
* IPv4 Address
* IPv6 Address
* MAC Address
* CPU
* CPU Cores
* RAM (in bytes)
* C:\ Drive Capacity (in bytes)
* C:\ Drive Space Remaining (in bytes)
* Operating System
* Operating System Version
* Operating System Architecture
* Operating System Serial Number
* Operating System Install Date
* Last Boot Up Time
* Number of Users

Software:

* Entry Date (when the log was recorded)
* Software Name
* Software Vendor
* Software Version
* Install Date
* Install State

Users:

* User’s Disk Path
* User SID
* Last Used Time
* User Status
* User Health Status
* Desktop Folder Size (in bytes)
* Documents Folder Size (in bytes)
* Pictures Folder Size (in bytes)
* Music Folder Size (in bytes
* Videos Folder Size (in bytes)
* Favorites Folder Size (in bytes)
* Downloads Folder Size (in bytes)
* Links Folder Size (in bytes)
* Searches Folder Size (in bytes)
* Saved Games Folder Size (in bytes)
* Appdata/Roaming Folder Size (in bytes)

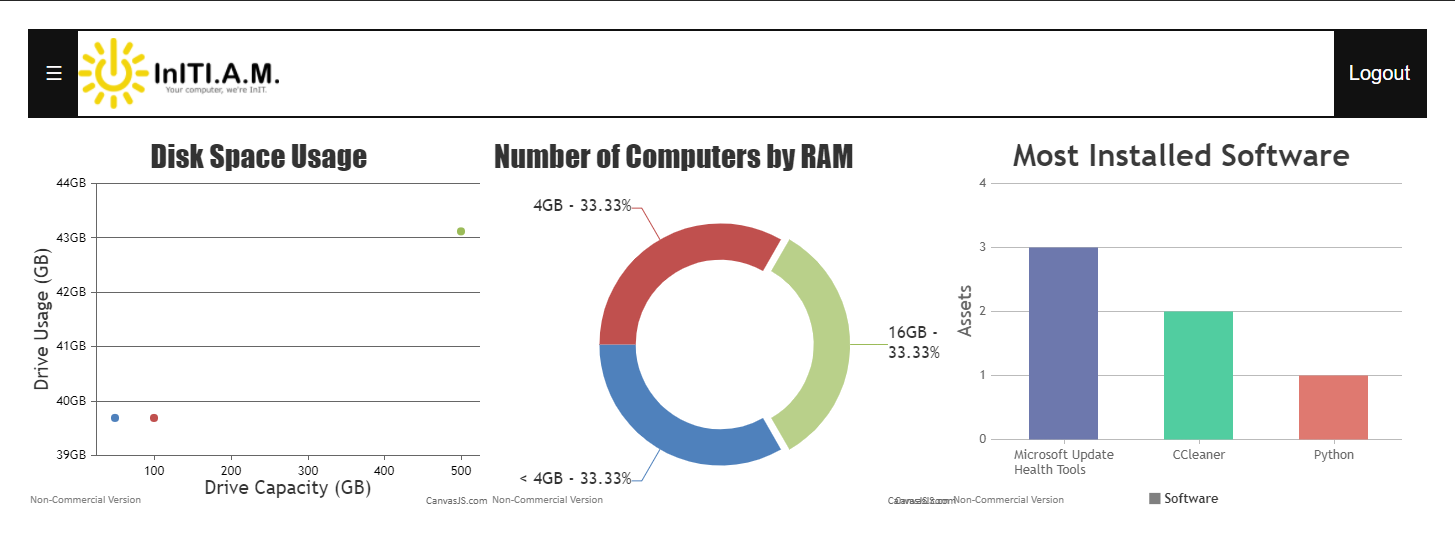
For this implementation, three virtual machines with varying hardware specifications were generated and installed with InITIAM. In a production implementation, an inventory of 100-2000 assets would not be unexpected, and would yield a much fuller data set.

Advantages of polling data within the client and posting to the server means periodically gives you records that show change over time. This allows administration to see trends in the data across all clients, or individual clients.

Disadvantages and potential issues with the method of data collection include the fact that there must be a working internet connection between the server and client. A relevant risk is that there are three points of failure to consider: the client software, the network connection, and the server software. Without implementing unnecessarily complicated block chain technology, this risk cannot be avoided.

# Data Product Code

One of the primary reasons the InITIAM User Web Portal was built for was to enable administration to view their collected data through visual representations. This allows management to see trends across hardware, software, and users throughout both the totality of the IAM’s assets as well as the individual assets. Upon successful authentication, the administration home page will be displayed, hosting a variety of charts. The User Web Portal was created using a hybrid of PHP, HTML, and javascript. The PHP performs backend server functions, injecting the necessary data into the clientside HTML and javascript. The charts are created utilizing the javascript CanvasJS framework through an educational license. These charts are simple, effective, beautiful, and interactive.



*Figure 5: InITIAM User Web Portal Dashboard*

Due to the limited number of virtual machines I could host, the dashboard appears sparse, but you can easily imagine once you hit upwards of 50 technology assets, how useful such charts would be.

# Hypothesis Verification

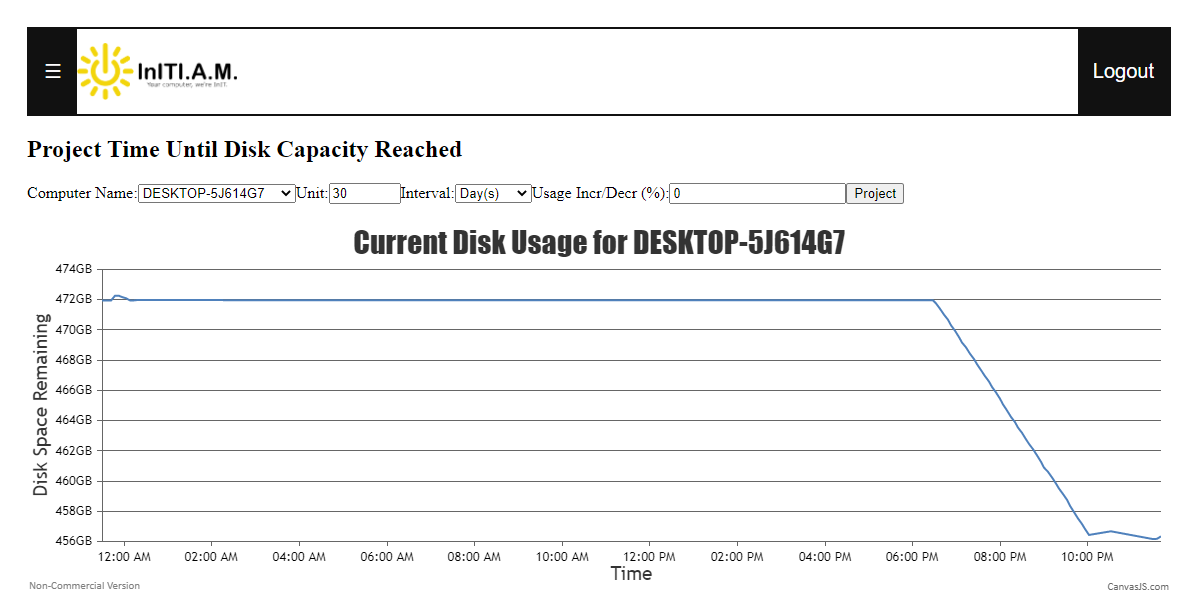
*If InITIAM successfully collects and displays hardware, software, and user data according to the objectives, then Walker Company will be able to employ greater technology management, enabling increased agility in responding to trends and environmental risks.*

The hypothesis was ultimately accepted due to two factors. Having access to comprehensive hardware, software, and user data resulted in Walker Company’s ability to anticipate problems and solve them before they caused employee downtime. In addition, the IT department was able to create effective maintenance and purchasing cycles due to the ability to categorize and organize their assets. Eliminating problems and improving services have led to more productive employees, and thus a greater organization-wide task turnaround.

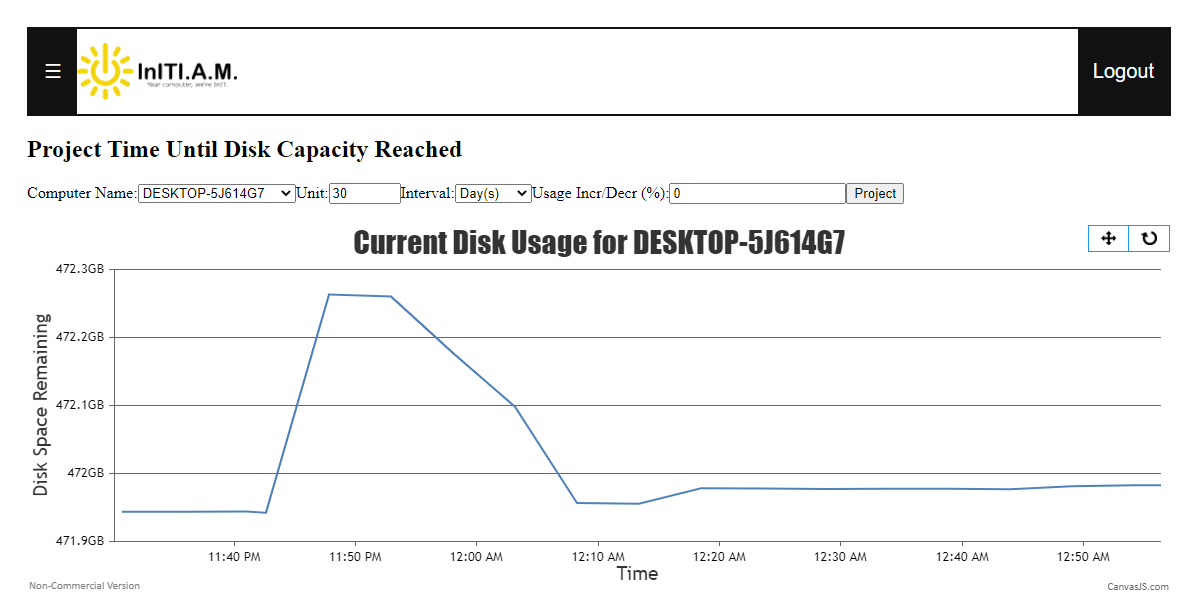
# Effective Visualizations and Reporting

Upon entering the InITIAM Administrator dashboard, a series of charts displayed a bird’s eye view of the organization’s asset data. The charts were able to effectively tell a story to the administrative user about how the level of technology was being employed by the organization, and how it was being used. This data allowed the IT department to plan on where and how to focus their efforts to improve their service, adding meaning and value to the service chain. See figure 4.

Additionally, a disk usage chart for individual assets was provided to highlight trends and tell a story about that asset’s history of usage. The landscape of disk usage can be a wild or a tame place, depending on how the user utilizes their technology asset. Because of this, it was particularly important to be able to visualize a machine’s change over time. Not only was it helpful to visualize the change over time, but also to be able to zoom in and look smaller sections of the trend more closely. Figure 5 below shows the current disk usage for an InITIAM client. Figure 6 demonstrates the ability to select an area in the chart and drill deeper into the data.



*Figure 6: Computer Disk Usage Chart*



*Figure 7: Chart Zoom Demonstration*

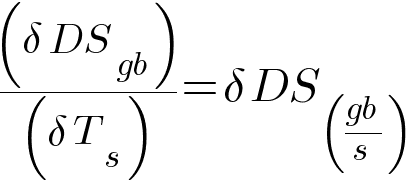
The process of data preparation for these visualizations involved client-side preparation and server-side preparation. The client-side preparation primarily consisted of ordering the data and formatting the string and integer values in a way that would be accepted by the database. Once the server received the data, after validating the important fields like the ClientGUID with Regex, it would pass the data into the database. Server-side preparation was also demonstrated when retrieving data for visualization or reporting. Time data had to be converted from UTC to Unix Timestamps or local datetimes, bytes had to be converted to the more meaningful format of gigabytes, and software states had to be converted from integers to their associated text definitions.

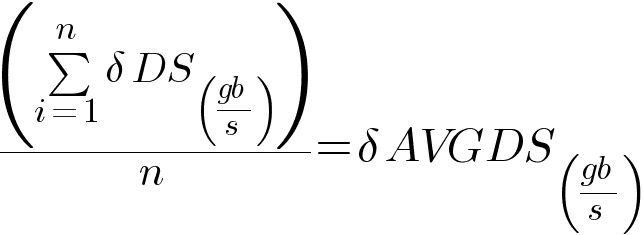
Data analysis had to take place when forecasting disk usage and then displaying those results visually. The disk usage data had to be analyzed to determine the rate of change, in both gigabytes and seconds, between each data record. Once this had been determined, then the average delta of gigabytes per second could be determined, utilizing a linear equation to determine at what time the disk would reach 0 bytes.

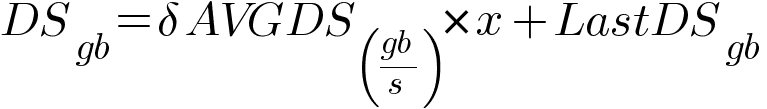
To summarize the projected disk usage, the original disk usage data between a selected interval was selected from the database. Once the date in which the disk would reach 0 bytes was determined, two rows were added to the previously selected data. One row included NULL data listed one second prior to the disk capacity epoch. The other row contained the zero-byte integer value with the time at which disk capacity would be reached. The union of these three datasets was converted into a JSON array and piped into the chart for visual display and interaction.

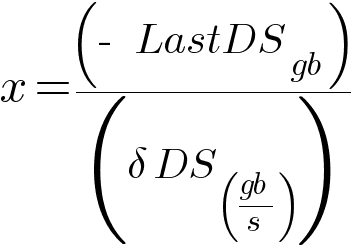
# Accuracy Analysis

InITIAM offers a method of predicting when a client will reach disk capacity. This method takes in data from a selected interval, containing a trend of decreasing space, and extrapolates the current disk usage model. The prediction model functions as follows:

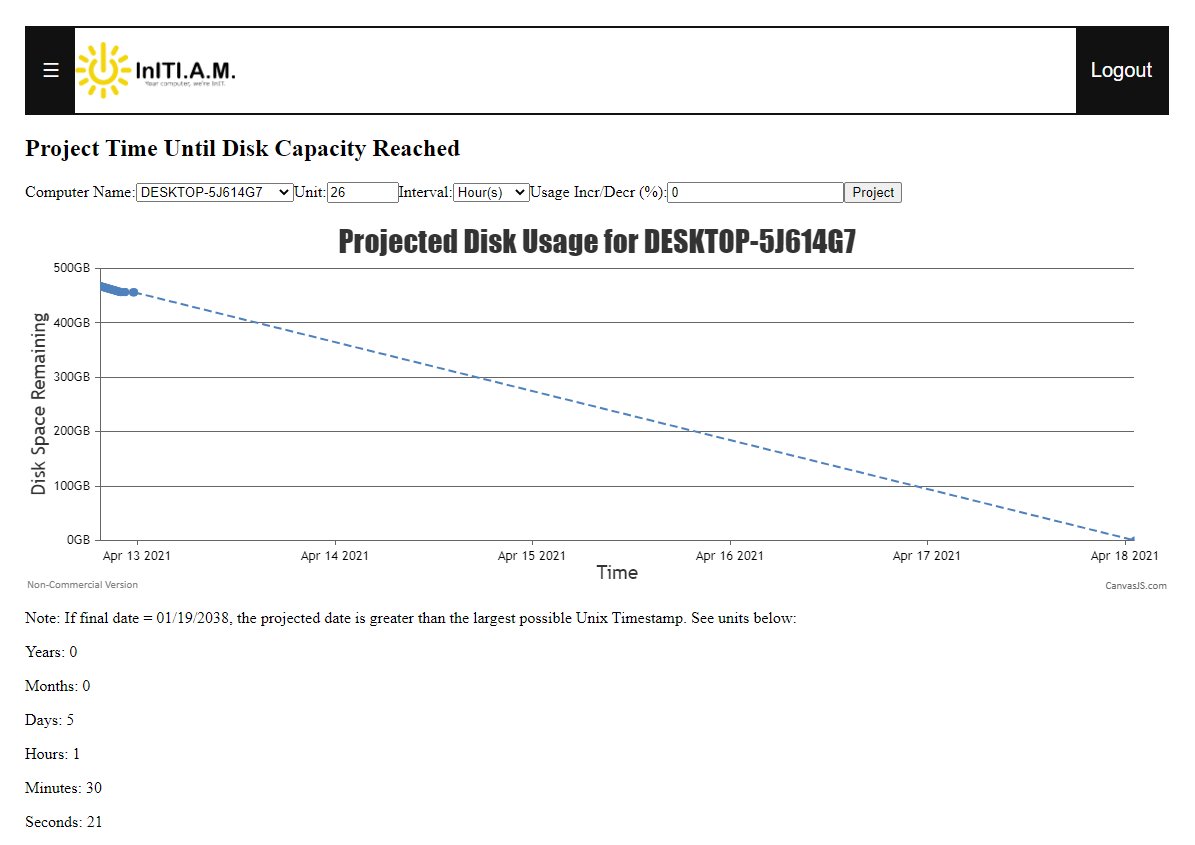
First, the change in drive space gigabytes and the change in time seconds was calculated for each record based on the prior one. Those rates of change were combined to get the rate of change in gigabytes per second.

Next, the average drive space change in gigabytes per second is computed:

The average drive space is then plugged into the linear equation, where “x” is seconds.

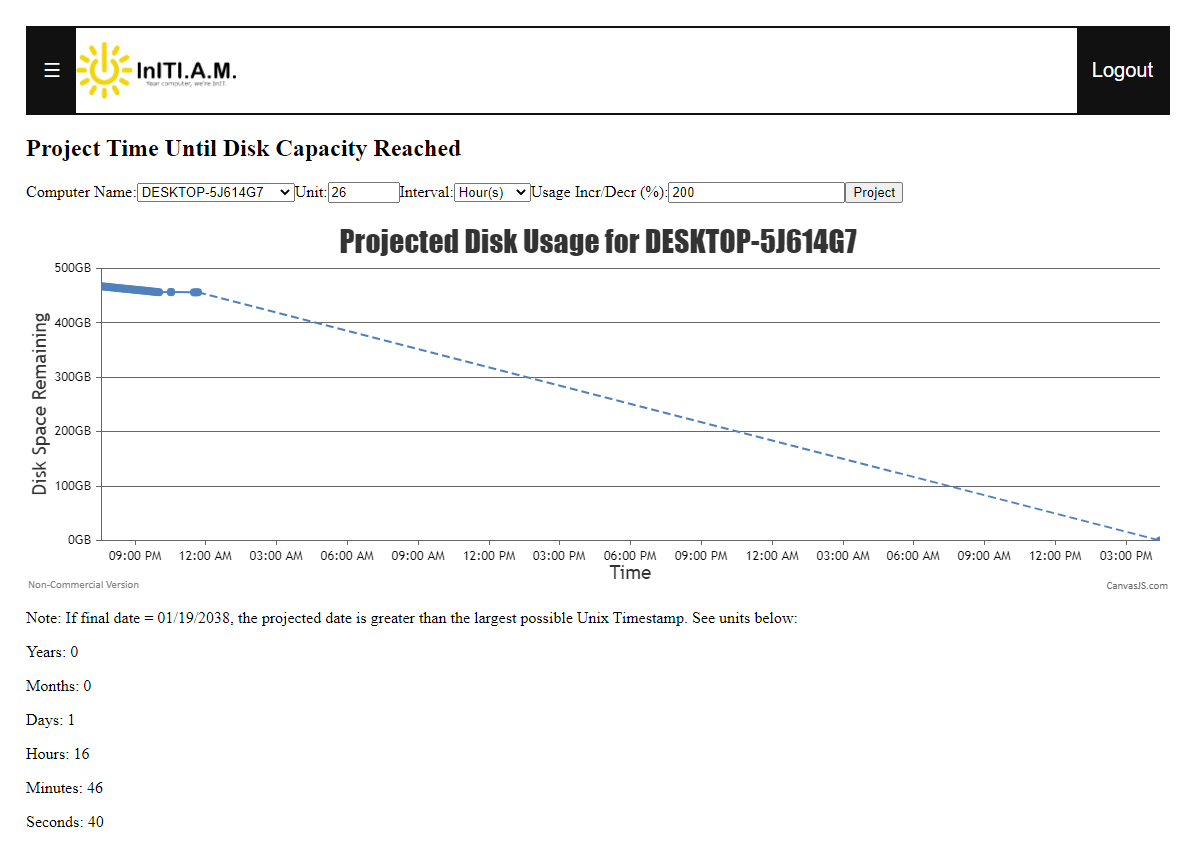
Finally, we set “DS” to 0 and solve for x:

Once the time at which disk capacity is determined, by calculating “x”, the data summary described in the previous section takes place. The data summary results in a prediction that looks like the figure below.



*Figure 8: Disk Projection Demonstration*

Additionally, if the user would like to calculate a prediction based on a given usage increase or decrease, they can do so.



*Figure 9: Disk Projection Demonstration w/ Usage Multiplier*

Due to the relatively consistent data given in the interval, the examples shown can be taken to be an accurate representation of when these machines will reach disk capacity given the current trend (or a customized trend multiplier). Given the context, a user’s disk usage is expected to remain relatively steady considering it is used for work 5 days a week. This makes prediction by linear regression very effective.

As with most prediction methods, inconsistent data and usage will be the primary disadvantage of InITIAM’s disk forecasting. Inconsistent usage will result in a poor set of sample data and that bad sample data will result in a linear prediction that is unlikely to align with what is to occur in the future. Thankfully, despite the accuracy disadvantages presented by InITIAM’s disk forecasting method, if the problem existed, it could be remedied by selecting a larger sample through the interval modification interface.

# Application Testing

Application Testing was accomplished through integration, system, and beta testing. Integration testing was configured such that everywhere different parts of the InITIAM solution came into contact, JSON encoded result text was passed between applications to verify successful integration. These integration tests were built as the product was developed and executed each time application communication occurred. Upon integration test failure, the root problem would be identified and addressed before moving on to the next product development. Occasionally integrating one product would cause another previously passing test to fail, which would help identify problems that at first were hidden.

System testing was accomplished through defining a set of tests beforehand and executing them once the InITIAM solution had completed development. Below is a table of tests and criteria used to evaluate their success.

|  |  |
| --- | --- |
| **System Test** | **Success Criteria** |
| Set InITIAM version to “.3”. Set database Service Release version to “1.0”. Test InITUpdate’s ability to execute automatic update. | InITIAM version “1.0” shows in logs and registry. |
| Set InITManager version to “.3”. Set database Service Release version to “1.0”. Test InITUpdate’s ability to execute automatic update. | InITManager version “1.0” shows in logs and registry. |
| Configure 3 virtual machines with varying hardware specifications. Install InITIAM on each one. Test multiple client check-in capability, computer logs, software logs, and user logs. | Verify all three different ClientGUIDs are showing up in the database’s “Checkins” table with regularly updated “LastCheckinTime”. Verify All three different MachineGUIDs are regularly showing up in the database’s “ComputerInformation”, “SoftwareInformation”, and “UserInformation” table. |
| Test InITIAM Web Portal authentication by logging in with ‘initadmin’ username and password. | Verify logging in with ‘initadmin’ username and password successfully begins session and redirects to the InITIAM Administrator Dashboard. Verify logging in with an invalid password fails to begin a session and notifies the user of failure to login. |
| Test default disk usage forecasting method by selecting computer name and an interval of 26 hours. | Verify the prediction visually appears to model the sample data’s disk usage trend. Verify the math taking place by hand. |
| Test usage multiplier disk usage forecasting method by selecting computer name, an interval of 26 hours, and a 200% increase. | Verify the prediction visually appears to model the sample data’s disk usage trend increased by 200%. Verify the math taking place by hand. |

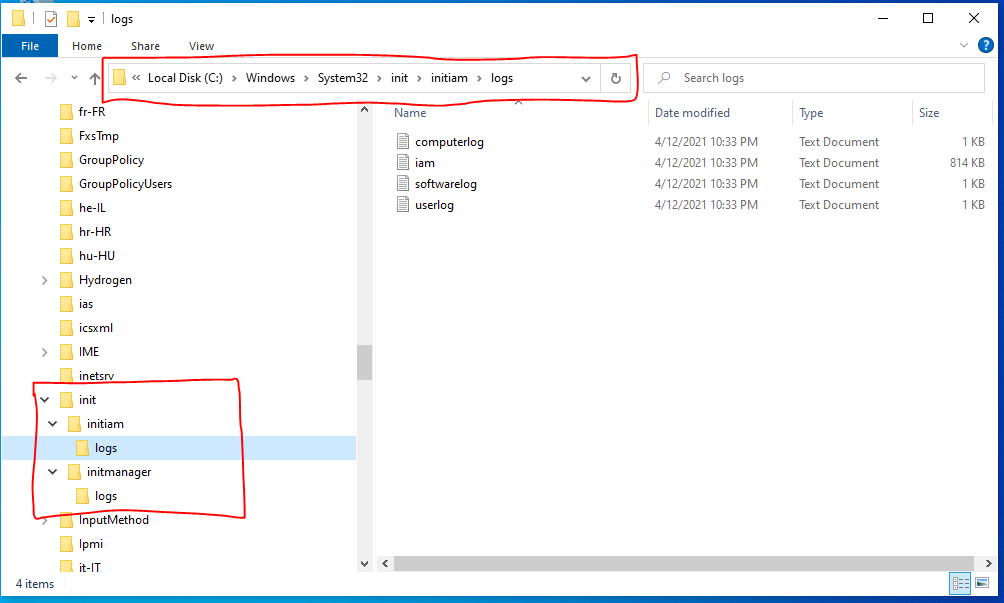
Finally, beta testing was accomplished by selecting two participants: one technically skilled user, and one technically unfamiliar user. Each was asked to install InITIAM onto a virtual machine given the windows installer, log into the InITIAM user web portal, and forecast a computer’s disk usage with a custom trend multiplier after being given the computer name. No additional instructions were given to either participant. Each participant was able to successfully complete their assigned tasks without receiving any help. This indicated that a user-friendly interface was successfully employed in each place that required interaction with a user.

# Application Files

When discussing the application files, their relationships, and their dependencies, the discussion should be separated into two categories: client files and server files.

### Client Files

When the InIT Application Installer is executed, it runs the InITManager, InITIAM, and InITUpdate’s Install function. This creates a folder at C:\windows\System32\init. Within this folder are two sub folders, InITIAM and InITManager, each with a subfolder called “logs” containing the debugging logs for each service.



*Figure 10: Client Log File Structure*

By default, the InIT Application Installer creates a folder called “InIT” in the Program Files folder with three subfolders: InITManager, InITIAM, and InITUpdate. These programs are not required to reside in this location, so long as their service folders InITManager, InITIAM, and InITUpdate are sibling child folders to the same parent directory. The reason for this is because InITUpdate must update the services according to the path “../SERVICENAME/”.

The InITManager service folder holds the following files:

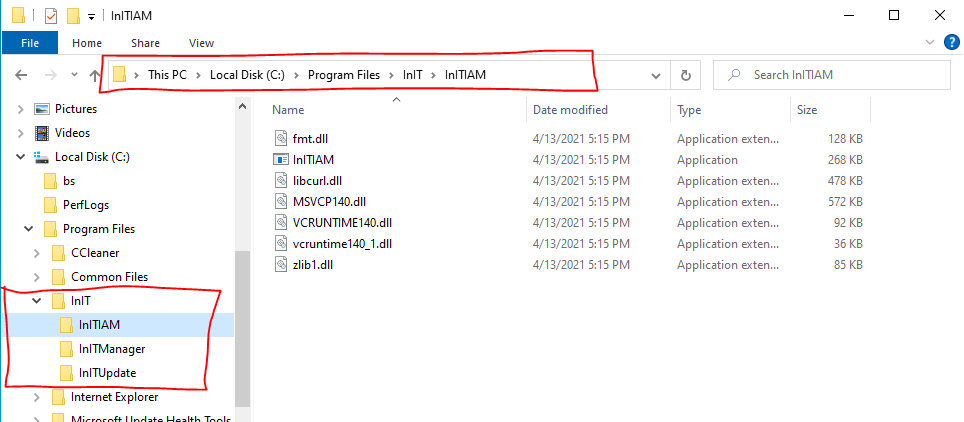
* fmt.dll
* InITManager.exe
* libcurl.dll
* MSVCP140.dll
* VCRUNTIME140.dll
* vcruntime140\_1.dll
* zlib1.dll

The InITIAM service folder holds the following files:

* fmt.dll
* InITIAM.exe
* libcurl.dll
* MSVCP140.dll
* VCRUNTIME140.dll
* vcruntime140\_1.dll
* zlib1.dll

The InITUpdate service folder holds the following files:

* bz2.dll
* fmt.dll
* InITUpdate.exe
* libcurl.dll
* MSVCP140.dll
* VCRUNTIME140.dll
* vcruntime140\_1.dll
* zip.dll
* zlib.dll



*Figure 11: Client Program Files*

### Server Files

The InITIAM Web Server’s root file begins with the folder initiam.net:

**initiam.net**

* **adminer**
  + Contains the adminer framework used to manipulate the MySQL database through a web-based GUI.
* **bin**
  + Contains images used for the web portal
* **canvasjs**
  + Contains the CanvasJS charting framework
* **forecasting**
  + Contains the page used to house forecasting and prediction functions
* **include**
  + Contains PHP functions available for use in all pages.
* **srv**
  + Contains the server side functions that interact with the client and database
* **favicon.ico**
  + The icon used in the web browser tabs.
* **index.php**
  + The home page, or administrator dashboard, for the InITIAM User Web Portal.
* **login.php**
  + The page which checks authentication credentials and creates an expirable access session.
* **logout.php**
  + The page which cleans and removes the session when user logs out.

The “srv” folder is structured as follows:

**srv**

* **checkin**
  + **index.php** – The page used for client check-ins (synonym for ping).
* **files**
  + **tmplink.php** – The page used to download updates. This page takes a file-tag, verifies it against the database, and returns the appropriate file to the client which is found in the “srv/release” folder.
* **log**
  + **computers**
    - **index.php** – The page used for posting client hardware data.
  + **software**
    - **index.php** – The page used for posting client software data.
  + **users**
    - **index.php** – The page used for posting client user data.
* **release**
  + **initiam** – Folder which will hold service files for updating.
    - **1.0** – Folder which will hold service version files for updating.
      * **package.zip** – The zip file containing the DLL and executable files for the service version. Downloaded by accessing “srv/files/tmplink.php”.
  + **initmanager** – Folder which will hold service files for updating.
    - **1.0** – Folder which will hold service version files for updating.
      * **package.zip** – The zip file containing the DLL and executable files for the service version. Downloaded by accessing “srv/files/tmplink.php”.
* **shared**
  + **db.php** – Database connection initialization. Included in other pages.
  + **user.php** – Database connection initialization. Included in other pages.
  + **webdb.php** – Database connection initialization. Included in other pages.
* **update**
  + **index.php** – The page used for checking if updates for client services exist. If so, it creates a release request and gives client a file-tag which can access necessary file download at “srv/files/tmplink.php”.
* **index.php** – An empty page that does nothing.

In the file submission, the root folder has been separated into two child folders: “Documents” and “Product”. “Documents” contains all the associated capstone documents such as the project proposal, letter of transmittal, and project recommendation. “Product” contains all the data product libraries, executables, and source files. The “Product” folder is separated into three child folders: “initclient”, “initserver”, and “initsetup”.

Inside the “initclient” directory is two sub directories: “system32” and “InIT”. The directory “system32” holds the filestructure found in the client’s “C:\windows\system32” folder (see description above). The directory “InIT” is the root program folder, typically hosted at “C:\Program Files\InIT” (see description above). Finally, the “automated\_installer” folder contains two files, an MSI and an executable. Both are windows automated installers.

Within the “initserver” folder is a single child folder called “initiam.net”. This is the same root folder described above.

Inside the “initsourcefiles” contains all C++ source and header files for each client application.

# User’s Guide

## Client Installation

Installation can be performed via 2 different methods, automatic or manual. Note: InITIAM currently is only designed for 64-bit Windows operating systems. Installation or run-time functionality will fail on a 32-bit operating system.

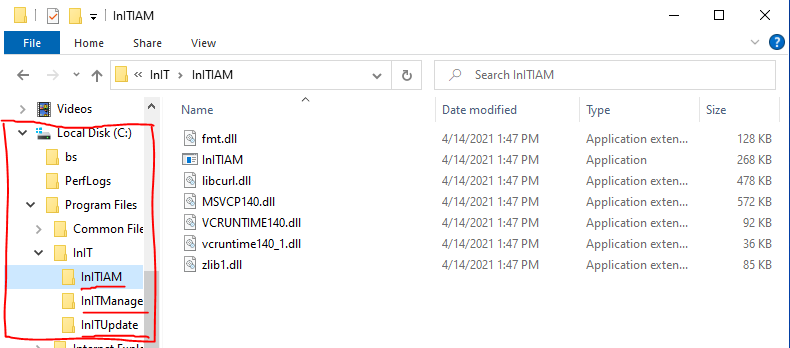
### Automatic

To execute the automated installer, retrieve the MSI and executable files from the “Capstone Submission/Product/initclient/automated\_installer” folder. Pick one (it doesn’t matter which), and right click -> run as administrator.

Click “next” through all the prompts, beginning the install. The installer will create the necessary directory structure, run the “/Install” argument on InITIAM.exe, InITManager.exe, and InITUpdate.exe. Upon successful installation, the installer will commit the changes to the computer and then run the “/Start” argument on InITIAM.exe and InITManager.exe. If this completes successfully, InITIAM has been installed.

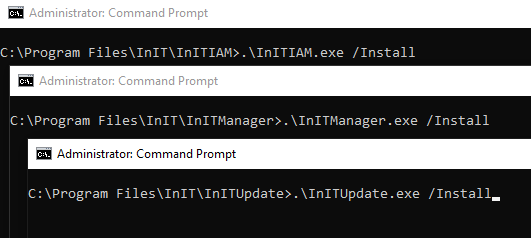
### Manual

To manually install the InITIAM client, four directories must be created on your local computer. First, a directory called “InIT” must be created in “C:\Program Files\”. The new “C:\Program Files\InIT” directory then needs to have all directories and child items copied from “Capstone Submission\Product\initclient\InIT\”. The local computer should have the following file structure shown below.



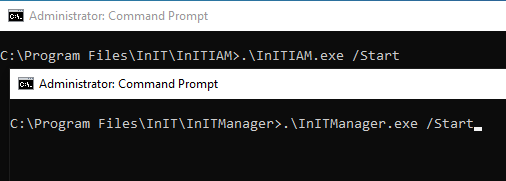
*Figure 12: Client Program File Structure*

Next, run command prompt as admin and navigate to the InITIAM program files folder. Type “.\InITIAM.exe /Install”. Repeat this for InITManager and InITUpdate.



*Figure 13: Manual Client Installation*

Once all three programs have been installed, start the InITIAM and InITManager services by doing the same thing but with the “/Start” argument.



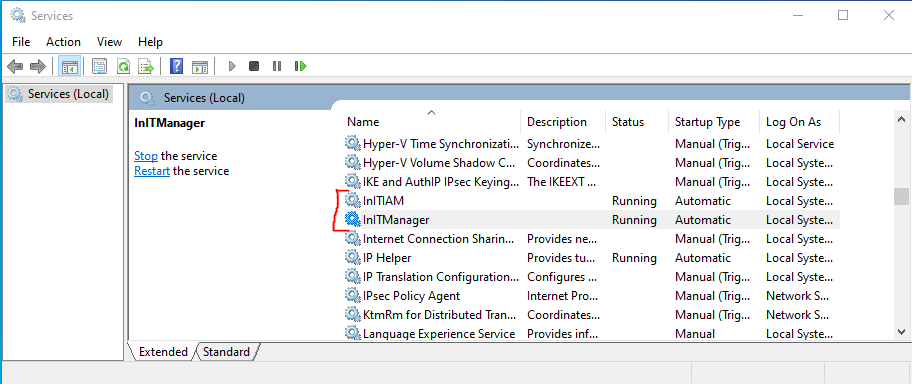
*Figure 14: Manual Client Service Start*

InITIAM has now been manually installed.

## Client Administration

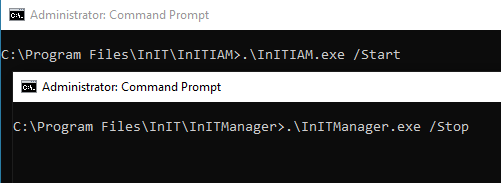
### Starting and Stopping Services

Services can be stopped one of two different ways. The first way is through the Windows Services interface. This can be accessed by searching “Services” in the windows search bar, or by opening task manager and navigating to the “Services” tab. Once the window is opened, scroll to the “InITIAM” and “InITManager” services. From here you may select a service, right click it, and start or stop the service. The services will automatically start on system boot by default.



*Figure 15: Client Services*

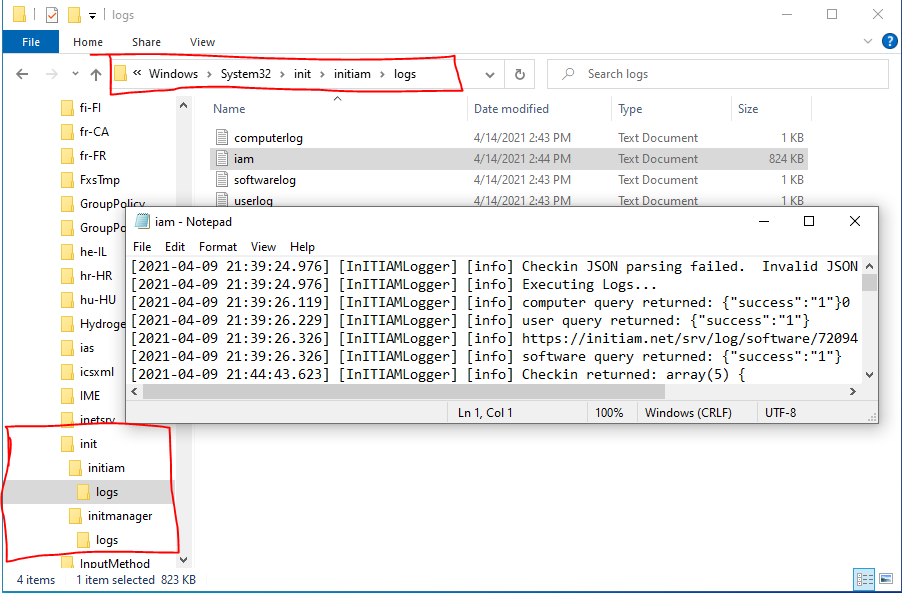
The second way to start and stop services is to run the “/Start” and “/Stop” arguments on the service executables. To do this, open command prompt as administrator, navigate to the service program files directory, and type “.\<ServiceName>.exe /<StartOrStop>”. See the figure below.



*Figure 16: Manual Client Service Stop*

### Checking Logs

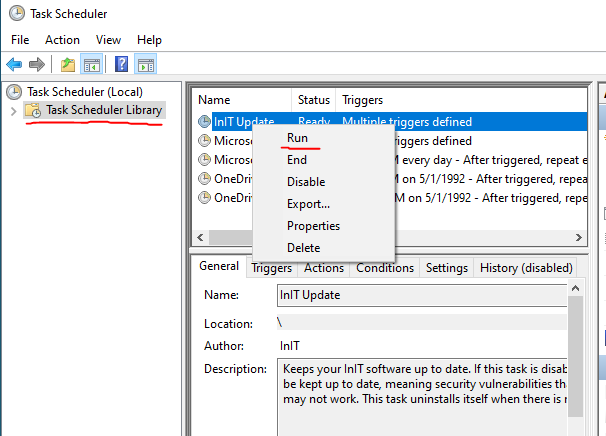
To view the client background service’s logs, navigate to “C:\windows\system32\init\<servicename>\logs”. Open the selected log in your preferred text editor/viewer.



*Figure 17: Client Logs*

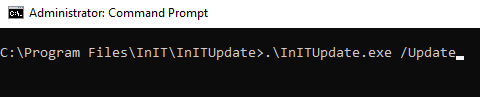
### Checking for updates

While updating InITIAM services is handled automatically, there could eventually be some reason that dictates the need to execute updates manually. This can be done one of two ways, executing the scheduled task dedicated to InIT updates, or running the InITUpdate executable. The first and easiest way is to open the Windows Task Scheduler interface, which can be done by typing “Task Scheduler” into the windows search bar. Once opened, select “Task Scheduler Library” on the left side and scroll to InITUpdate. Select and right click, pressing “Run”.



*Figure 18: Run Scheduled Task*

Alternatively, open command prompt as administrator, navigate to the InITUpdate program files directory, and type “.\InITUpdate.exe /Update”. See the figure below.



*Figure 19: Check for Updates From Executable*

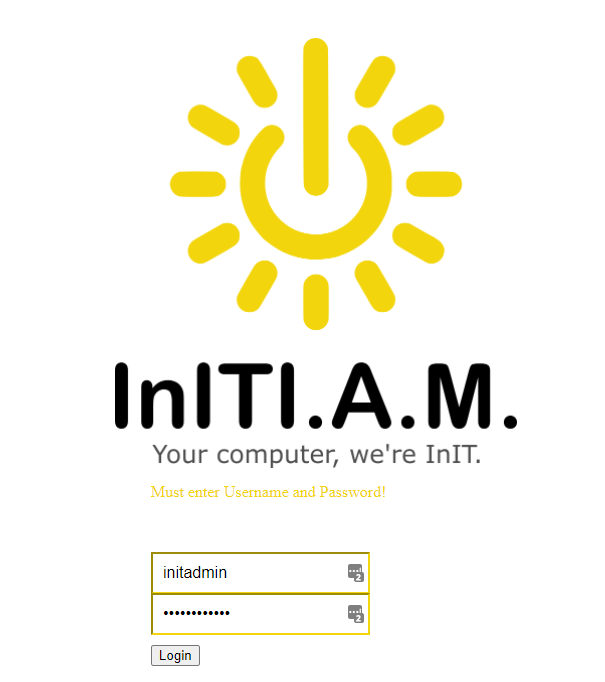
## Web Portal Access

### Authentication

To log into the InITIAM Web Portal, navigate to <https://initiam.net>. Enter the following credentials:

Username: initadmin

Password: 77TxfFI24FyQ

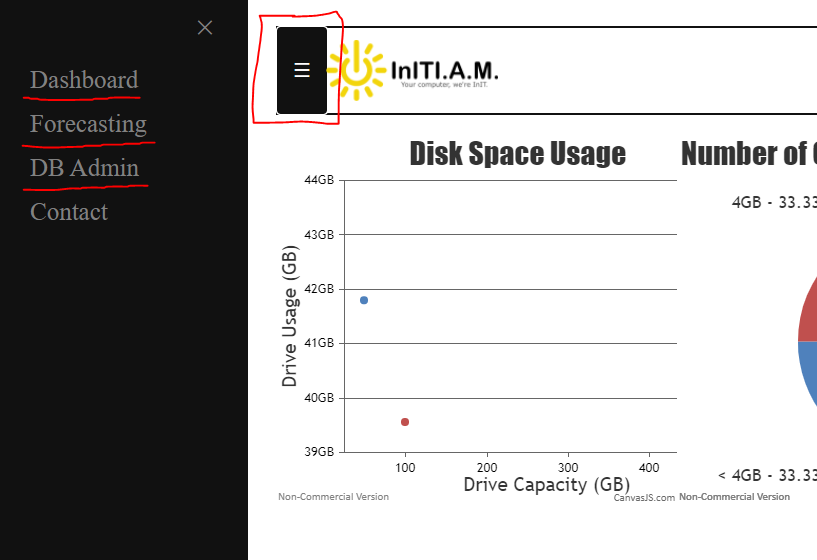


*Figure 20: InITIAM Login Page*

Upon successful authentication, you will be redirected to the InITIAM Administrator Dashboard.

### Navigation

To open the InITIAM Web Portal’s navigation menu, press the hamburger button at the top left.

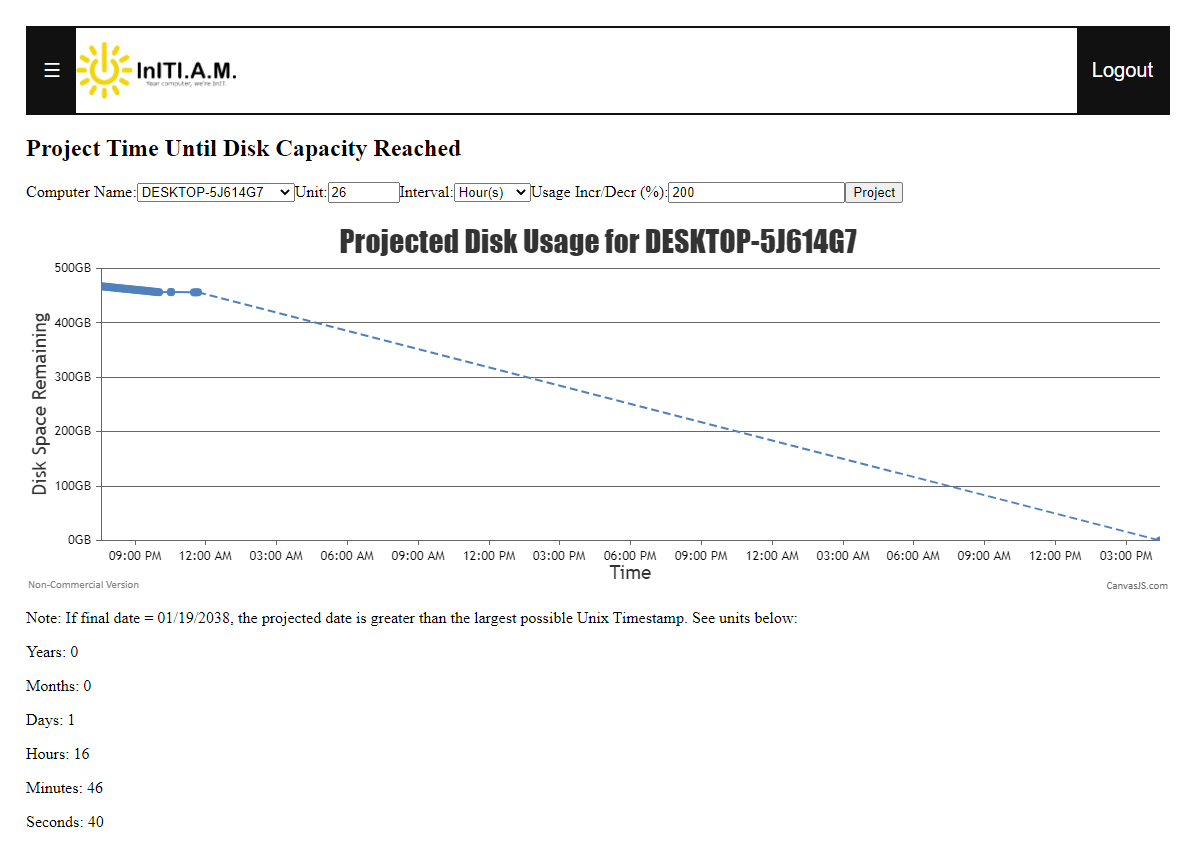


*Figure 21: InITIAM Navigation Menu*

This will allow you to navigate to the desired page.

### Forecasting

To use the disk usage forecasting method, navigate to the “Forecasting” page. Here you can select your computer name, specify how far back in time you want to poll for sample data, and then if desired include a usage increase or decrease multiplier.



*Figure 22: Disk Usage Forecasting User Interface*

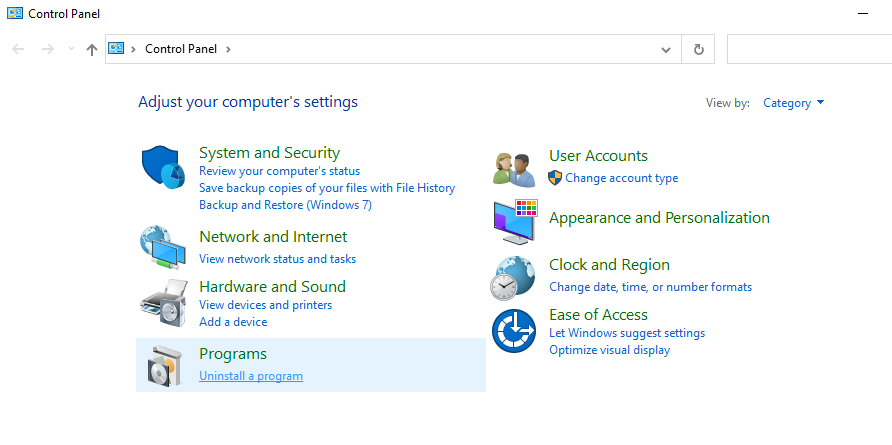
The forecasted results are displayed in a chart and text format. The chart is interactive, allowing the user to hover over data points and see their individual data in popup boxes. Additionally, if the user would like to examine a section of the chart further, they can drag and drop a selection over the specified area and the chart will zoom in to fit the selection. The text at the bottom of the page tells the user precisely when the system projects their disk to run out of space.

## Client Uninstallation

Like the client installation, client uninstallation can be accomplished two different ways.

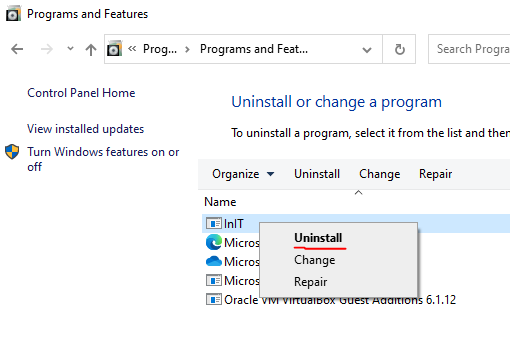
### Automatic

If InITIAM was installed using the windows installer, then that means it can be uninstalled with the windows installer. To uninstall the client, open Control Panel and navigate to “Uninstall a program”.



*Figure 23: Control Panel Uninstall Program*

Select “InIT”, right click the selection, and click “Uninstall”.

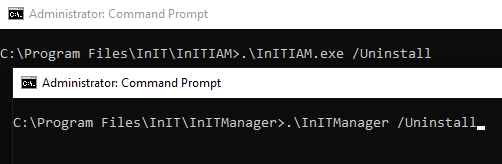


*Figure 24: Uninstall Program*

InITIAM is now uninstalled!

### Manual

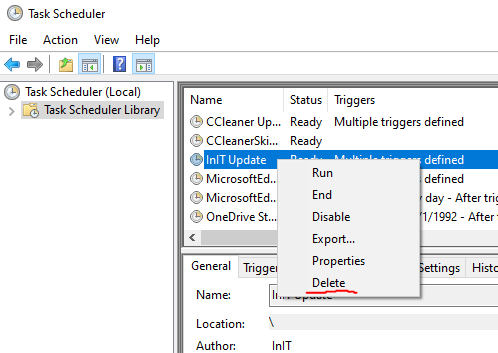
Run command prompt as admin and navigate to the InITIAM program files folder. Type “.\InITIAM.exe /Uninstall”. Repeat this for InITManager.



*Figure 25: Manual Client Uninstall*

Next, delete the folder “C:\Program Files\InIT” and all its children.

Finally delete the InITUpdate scheduled task. To do this, type “Task Scheduler” into the windows search bar. Once opened, select “Task Scheduler Library” on the left side and scroll to InITUpdate. Select and right click, pressing “Delete”.



*Figure 26: Manual Scheduled Task Removal*

# Summation of Learning Experience

This project represented both my experience, and an exploration of uncharted territory. Upon undertaking this project, I had been relatively experienced in C++ programming due to my prior education with Dixie State University. This is where my familiarity with the language came from, and my love for it pushed me to develop my capstone with it. Also, I am the database administrator for the City of St. George, so I am extremely well versed with relational databases, my primary skillset being with Microsoft SQL Server with a slightly lesser skill in MySQL. I was able to utilize my query scripting knowledge to quickly develop very complex queries for the project’s disk usage forecasting method.

I was somewhat familiar, but unskilled, in HTML and PHP. I often would have to work with the City’s systems engineer and web developer to create intranet website pages. Typically, they would do the PHP and HTML while I would handle the database queries. While my hands were not on the code, I became semi-familiar with the styles and functionality of both scripting languages. Because of this, I found development of the InITIAM web server to be interesting. Learning how the two languages were used in greater detail went a long way in teaching me how the client web browser, or client APIs, interacted with the server.

Completely new territory for me came in the form of creating a windows application that integrated with the Windows Service Manager. I had to become very familiar with Windows C++ libraries, including registry calls, filesystem calls, service control calls, and task scheduler calls. I know this is common in the software industry given the proliferous nature of the Windows operating system, so it excited me to be able to learn and use it for the first time.

This project also represented the first full stack application for me. Never have I created a product that required client, server, database, backend, and frontend programming. I found it very insightful to be able to develop all the different parts and watch them interact with each other. Doing this made me more aware of the importance of secure programming practices.

Overall, this project has added a lot of value to my journey in computer science and software development. Due to the constantly evolving nature of technology, lifelong learning is a goal I have to make number one in my career, like most others. This project has taught me a lot about full stack development that I won’t forget, as well as showed me that I can accomplish whatever it is I set my fingers to type… with a little help (a lot of help) from Google of course!