Software Design Document

**Web Application**

O365 Provisioning Utility

**Team Members**

Full-Stack Developer: Jacob Fanelli

Back-End Developer: Michael Rapaport

Scrum Master/Application Architect: Dion Q. Stovall

**Location**

RTX Headquarters

**Date**

May 4, 2020

Version 001

Table of Contents

[**1.0 Introduction** 2](#_Toc36814707)

[**2.0 Purpose** 3](#_Toc36814708)

[**3.0 Scope** 3](#_Toc36814709)

[**4.0 Development Cadence** 3](#_Toc36814710)

[**5.0 Definitions, Acronyms, Abbreviations** 4](#_Toc36814711)

[**6.0 Design Overview** 5](#_Toc36814712)

[6.1. Description of the Problem 5](#_Toc36814713)

[6.2. Technologies Used 5](#_Toc36814714)

[6.3.0 System Architecture 5](#_Toc36814715)

[6.3.1 Connectivity Architecture 7](#_Toc36814716)

[6.3.2 System Operation 8](#_Toc36814717)

[6.2.3 Source Code Repository 9](#_Toc36814718)

[6.3.4 Cosmos DB Schema 9](#_Toc36814719)

[6.3.5 SQL DB Schema 9](#_Toc36814720)

[**7.0 References** 9](#_Toc36814721)

# **1.0 Introduction**

**Currently, there is a manual process for enabling users that are either new or existing members within the Commercial tenant(C Cloud) into the new tenant (GccH). To resolve this challenge, the development of am automated web application was proposed to streamline the migration process by creating a centralized location that displays user migration data that is specific to each Business Unit. The Business Unit Lead will be able to select the users that will be migrated and export the list to the designated program.**

# **2.0 Purpose**

The web application has been created to automate the migration process by importing user data from specific data sources and display the data through a GUI. Business Unit Leads will select and export the users that are to be migrated. The file that contains the list of users will be sent to Program Managers. Once the Program Managers update each user profile, the web application will refresh, retrieve the updated information and display the updated user data through the GUI in continuous refresh iterations of 24 hours.

# **3.0 Scope**

1. Import user data from various data sources (email, AD, HR, SafeNet etc.) into the back-end of the enhancement which is separated by BU (Collins Aerospace, PW, HQ, UTRC).

2. Create a GUI that gives users the ability to access data based on the BU they belong to.

a. Ex: A user that is employed at Collins Aerospace should only have access to Collins Aerospace data; otherwise, their access to other BU data is restricted.

3. Once the user has access to their BU’s data, they are able to filter the data by Department, Floor, Building, Location etc. as well as see which users have been provisioned a license/activated in GccH.

4. The user is then able to export that list of data to specific teams such as Cyber, Software Development, Communications, AD, etc.

5. Lastly, there must be an exception policy for specific international employees.

# **4.0 Development Cadence**

The development team adopted the Agile Framework to complete the web application. This development cadence increased visibility supported by MVPs (Most viable products) while providing a productive and efficient development environment. Below is list of the Sprint schedule:

**Development Status: 8 Sprints (2 weeks)**

**Sprint 1:** Jan. 20th – Jan. 31st (MVP 1)

**Sprint 2:** Feb 3rd – Feb. 14th

**Sprint 3:** Feb. 17th – Feb. 28th (MVP 2)

**Sprint 4:** Mar. 2nd – Mar. 13th

**Sprint 5:** Mar. 16th – Mar. 27th (MVP 3)

**Sprint 6:** Mar. 30th – Apr. 10th

**Sprint 7:** Apr. 13th – Apr. 24th

**Sprint 8:** Apr. 27th – May 1st (*accelerated*)

**Rollout Date:**

Monday, May 4th (Go-Live)

# **5.0 Definitions, Acronyms, Abbreviations**

**Active Directory (AD**): a Microsoft product that consists of several services that run on Windows Server to manage permissions and access to networked resources.

**Application Programming Interface (API):** a set of routines, protocols, and tools for building software applications that specifies how software components should interact when programming a GUI.

**Business Unit (BU):** subsidiary units of UTC such as Pratt & Whitney, UTC Research Center (UTRC), Collins Aerospace and UTC Headquarters (UTCHQ)

**Commercial Cloud (C Cloud):** Microsoft Azure commercial data center, also known as the “public cloud”

**CosmosDB:** Microsoft’s globally distributed, multi-model database service for operational and analytics workloads.

**Excel CSV:** a simple file format used to store tabular data, such as a spreadsheet or database.

**Government Cloud (G Cloud/GccH/G3):**  a comprehensive cloud platform designed expressly for U.S. Federal, State, and Local Governments to meet the U.S. Government's thorough security and compliance regulations.

**Graphical User Interface (GUI):** displays objects that convey information, and represent actions that can be taken by the user.

**MongoDB:** a cross-platform document-oriented database program that is classified as a NoSQL database program that uses JSON-like documents with schema.

**Single Sign-On (SSO):**  is a property of access control where a user logs in with a single ID and password to gain access to any of several related systems.

**SQL Database (SQL DB):**

**U.S. Person/Non-U.S. Person (USP/NUSP):** An employee that is a legal resident of the United States or is not a legal resident of the United States.

# **6.0 Design Overview**

## 6.1. Description of the Problem

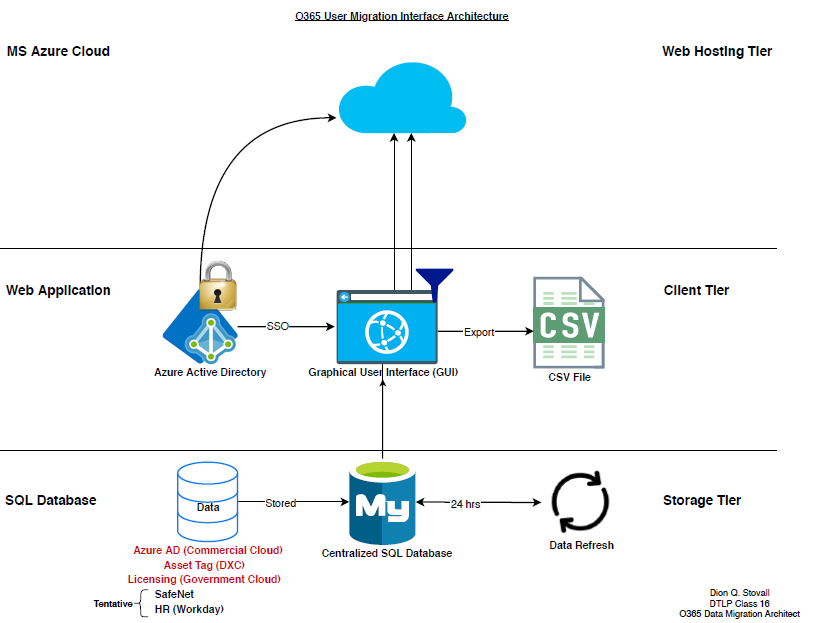
Currently, Business Units provide the following information in a spreadsheet: Users Full Name,Users UPN, User UPN full, User is USP \Export compliant, Email, Business Unit, Location, Existing O365 Users (Commercial at this time), Asset Tag, Executive Flag (Sr. Director and above). The spreadsheet is put into a folder on the SharePoint site under the Early Adopter folders based on Wave rollout. The information is then validated by the program, put into file and highlighted. The accurate highlighted information within the file is then sent to programs such as Stub Creation, Licensing, Software Distribution, Communications and MFA to indicate the users that need to be migrated.

# 6.2. Technologies Used

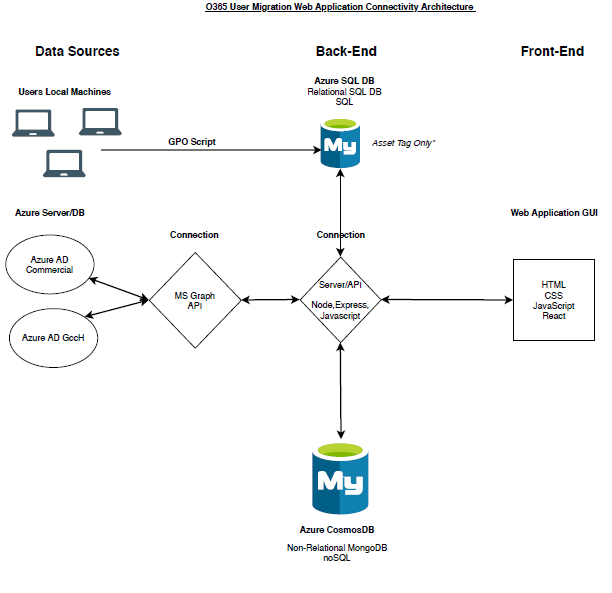
The application has three tiers of operation. Top down, the web hosting tier is where the application and majority of its elements will be hosted in Azure. The client tier is where the GUI will be accessible for users to engage and interact with the application. The final tier is storage where the data that is being retrieved from Azure will be stored within the CosmosDB and SQL databases which are refreshed every 24 hours. An API will make a request to Azure Commercial and Government Cloud which will post the request into the CosmosDB. A separate API will make a request to the CosmosDB will be posted to the GUI. Lastly, there will be an API request to a SQL database that will post the results to the GUI as well. *(The API’s and GUI are hosted on a local sever until go-live into the production environment)*

# 6.3.0 System Architecture

* **Web Hosting Tier:** The application will be deployed through a Virtual Machine in Microsoft Azure Cloud where the application will be hosted. User access privileges are also granted from Azure through SSO.
* **Client Tier:** Users are able to engage with the application and export CSV files after selecting the list of employees.
* **Storage Tier:** The application will connect to various data sources and store information within a CosmosDB and SQL database. The application will query the data at the time of login. The data is then refreshed on a scheduled basis.

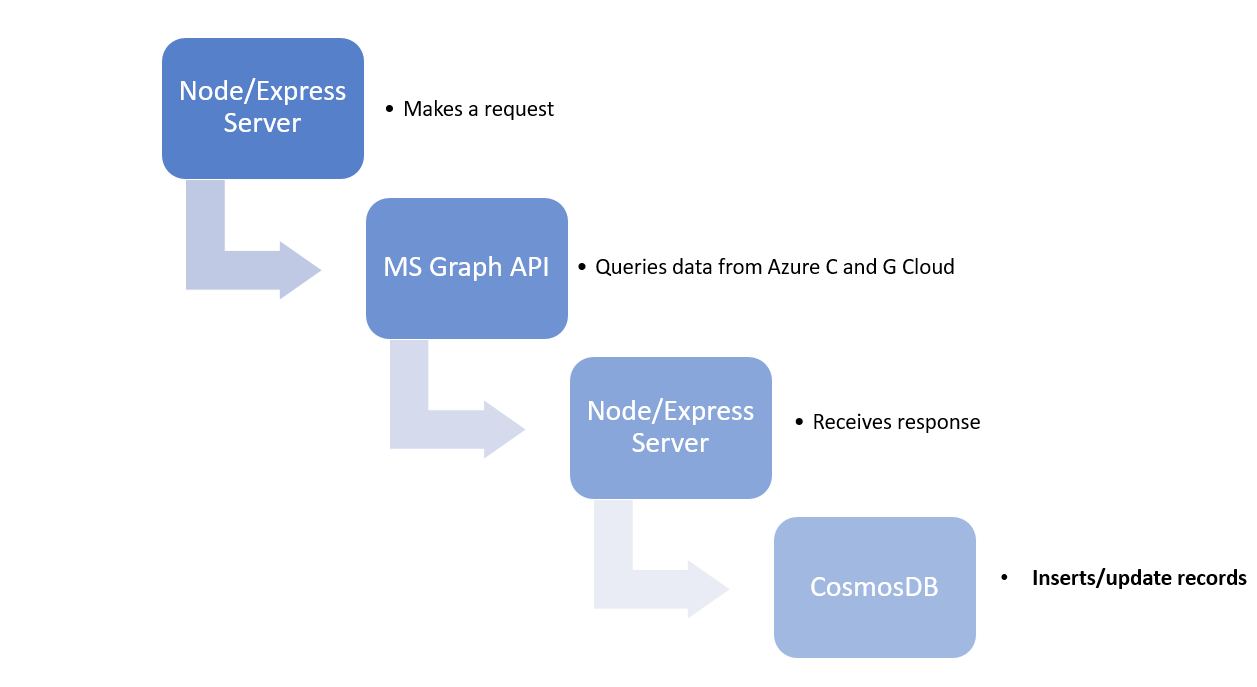


# 6.3.1 Connectivity Architecture

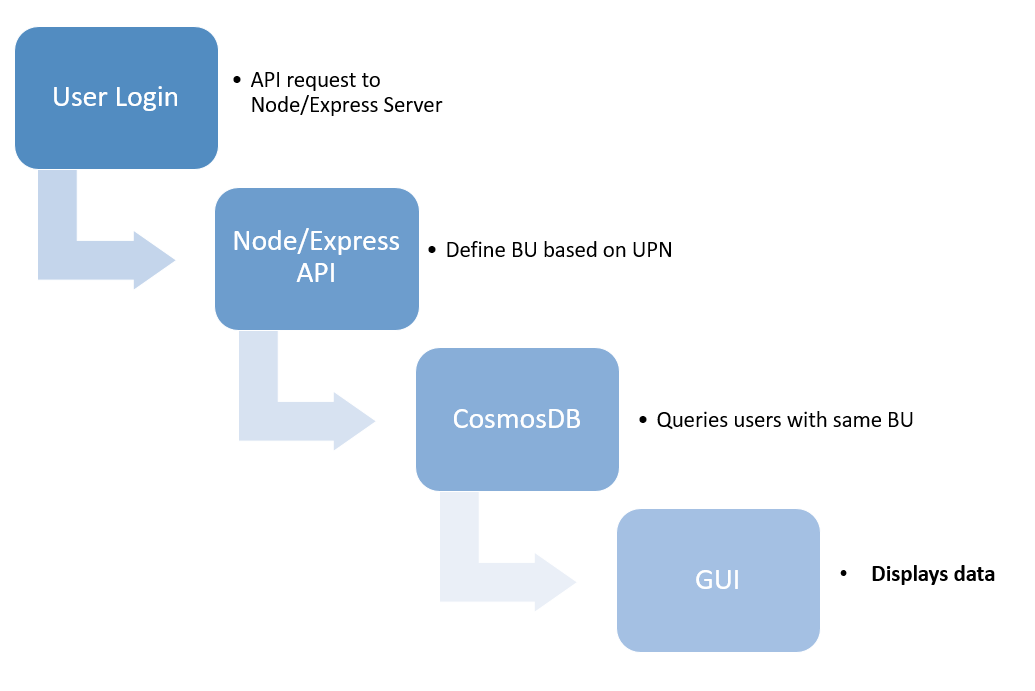
* **Front-End:** The user interacts with the data displayed in the GUI.
* **Back-End:** The data is being gathered from specific data sources and stored into two separate databases. Information based on individual Business Unit employees, their location, and their status within the license provisioning process is stored in the CosmosDB. Information based on an employees’ device is stored within the SQL Database.
* **Data Sources:** The back-end connections query data from Azure AD and store the data into the CosmosDB. The GPO script that will deployed through DXC will query user asset data from their local machines (domestic and international machines) and store the data into the SQL DB. The back-end connection will query that data from the SQL DB and display it in the GUI.

# 6.3.2 System Operation

***Application Process: Store Data within CosmosDB***



***Application Process: Display Data in GUI***



# 6.2.3 Source Code Repository

The application source code is stored in a GitHub Repo. A GitHub Repo can be defined as a technical folder for the project. It contains all of the O365 Provisioning Utility code files and stores each file's revision history. The source code has been optimized, tested and pushed to production.

Link: [O365 Provisioning Utility GitHub Repo](https://github.dx.utc.com/FANELLJA/o365-utility)

# 6.3.4 Cosmos DB Schema

*Non-Relational Document*

**Field Type Description**

1. \_id String Unique document identifier
2. fullName String User’s name - First Last
3. title String Title, such as: “Assoc dir, Project Mgmt”
4. stubCreated Boolean True if user has their azg stub ID created, false if yet to be
5. licensed Boolean True if user has a GCCH O365 License
6. commO365 Boolean True if user has a Commercial O365 License
7. upn String User’s UPN, i.e. “rapapomi”
8. fullUpn String Domain qualified UPN, i.e. [rapapomi@utccgl.com](mailto:rapapomi@utccgl.com)
9. usPerson Boolean True if user is US person, calculated based on whether

displayName contains “Export Licence Required”

1. email String User’s email address
2. bu String User’s BU – i.e. “Pratt & Whitney”
3. building String Location – Building i.e. “OBG1”
4. streetAddress String Location – Street Address i.e. “400 Main Street”
5. city String Location – City i.e. “East Hartford”
6. state String Location – State i.e. “CT”
7. postalCode String Location – Postal Code i.e. “06118”

# 6.3.5 SQL DB Schema

# **7.0 References**