**CIOBrain Deployment**

Requirements Documentation

Anton Horvath

Anuja Sahu

Scott Lorance

Kevin Lieng

Jose Colina Salas

Saige Wright

# Abstract

This requirements document captures the functional and nonfunctional requirements for the CIObrain deployment project. The document includes the use cases that correspond to a functional requirement including both graphical and textual descriptions of the use case. A list of nonfunctional requirements for the CIOBrain deployment capabilities project has also been included.

# 

# Table of Contents

[**Abstract**](#_ixg77vn48u6) **2**

[**Table of Contents**](#_2q703hhy3go1) **3**

[**List of Figures**](#_3kmcnwhhl6qs) **4**

[**List of Tables**](#_k4tqfpyuqndf) **5**

[**Introduction**](#_pd5757bxwu9) **6**

[**Use Case Model for Functional Requirements**](#_52hp04dsce4g) **7**

[Use Case Diagrams](#_xfgq7b5dku7n) 8

[Requirement 1](#_vcjrpm40whcb) 8

[Requirement 2](#_zg1nonuk52f) 9

[Requirement 3](#_wb05rlfl2zsy) 10

[Requirement 4](#_x5at0tmxkltn) 11

[Requirement 5](#_myqhm7llcdni) 12

[**Rationale for Use Case Model**](#_pyz92m90vdri) **13**

[**Nonfunctional Requirements**](#_lgl6icguzt95) **14**

[**Evidence Requirements Have Been Placed Under Configuration Management**](#_l12o9edobhpa) **16**

[**References**](#_xi3r96x38f6b) **17**

[**N/A**](#_yvgmok538slp) **17**

# 

# List of Figures

|  |  |
| --- | --- |
| [Figure 1. Requirement 1](#_vcjrpm40whcb) | 7 |
| [Figure 2. Requirement 2](#_zg1nonuk52f) | 8 |
| [Figure 3. Requirement 3](#_wb05rlfl2zsy) | 9 |
| [Figure 4 Requirement 4](#_x5at0tmxkltn) | 10 |

# 

# List of Tables

N/A

# Introduction

This document will outline and describe all functional and nonfunctional requirements associated with the CIOBrain Deployment project. It will also include use cases associated with the functional requirements that are presented. All requirements detailed will help the group establish a clear picture of what tasks and priorities need to be addressed over the semester. It will also ensure the group is working on the correct tasks the sponsor requires.

This requirements document includes the following:

* Use Case Models for Functional Requirements
  + This section contains all use cases for the semester project — each corresponding to a functional requirement. The section is broken into two subsections — diagrams representing the use cases and the corresponding textual descriptions
* Rationale for Use Case Model
  + This section describes the reasoning behind each use case and the justification on why the team has decided to approach the problem in that way
* Nonfunctional Requirements
  + This section describes all nonfunctional requirements the team will be prioritizing for the CIOBrain Deployment project
* Evidence Requirements Have Been Placed Under Configuration Management
  + Screenshot
* References

# Use Case Model for Functional Requirements

## Use Case Diagrams

### Requirement 1

CIOBrain shall be packaged into a .msi installer file and support one-click installation.

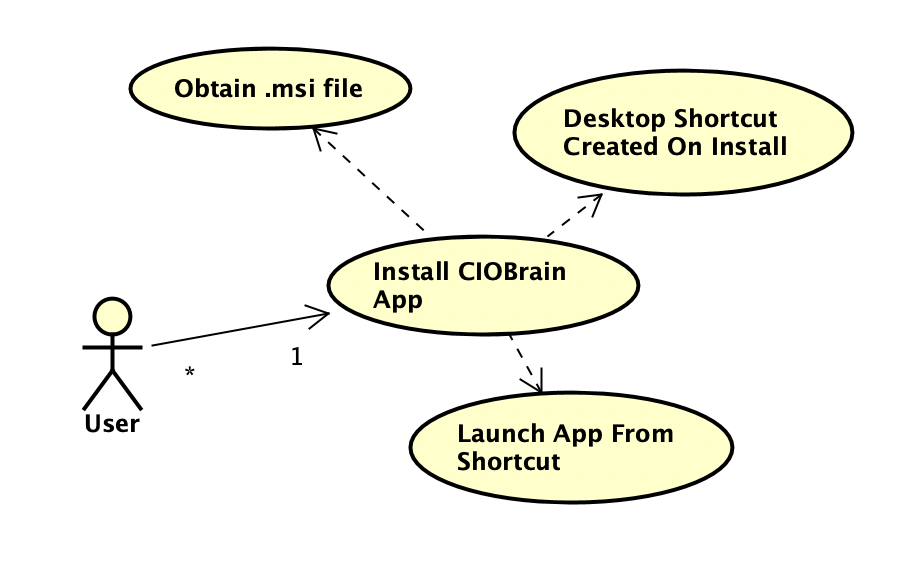


Figure 1

1. The desktop app should have a .msi installer file that can be run to instantly install the application to the Windows machine. A desktop shortcut should be created where the user can double-click to launch the application.
   1. Actors: User
   2. Entry condition: User obtains the .msi installer file.
   3. Normal Flow: User obtains .msi installer and runs it. Installer will install the desktop application and load a shortcut on the desktop. Users will be able to launch the application by double-clicking the shortcut.
   4. Exit Condition: User does not run the .msi file after obtaining it. User completes installation with desktop app installed and shortcut located on desktop.
   5. Exceptions: None
   6. Special Requirements: .msi installer is strictly Windows-compatible and cannot be run on Mac or Linux. Dedicated installers for those operating systems can be built using the ciobrain-native installer builder.

### Requirement 2

CIOBrain shall have a one click transfer capability from desktop app to Microsoft Azure

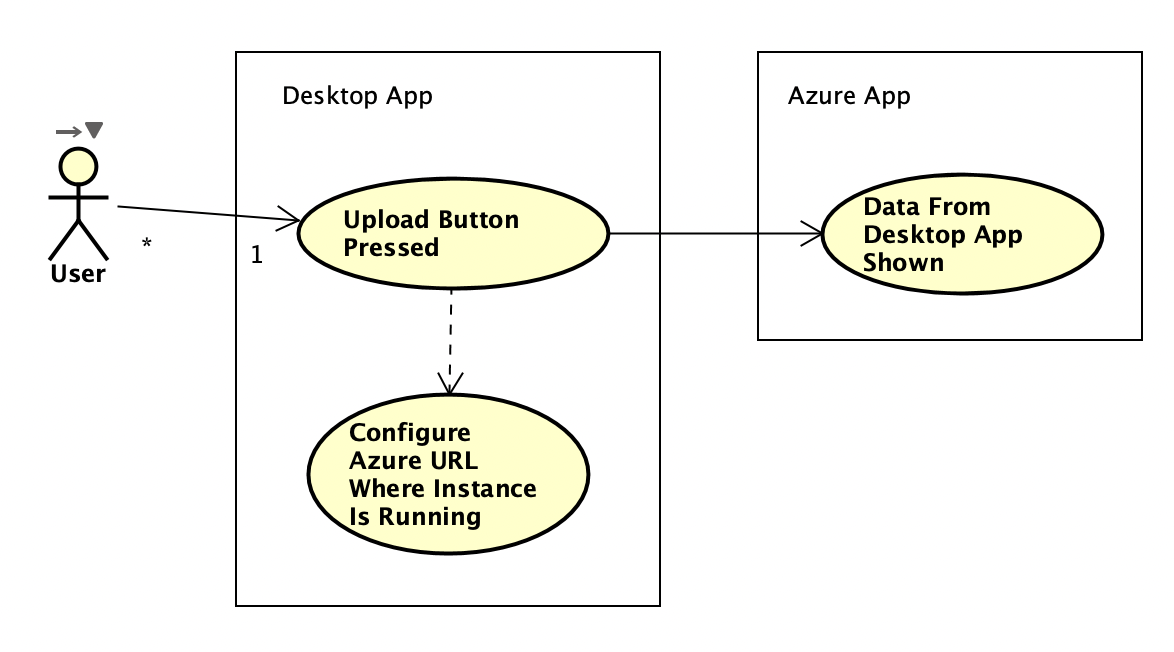


Figure 2

1. The desktop app should have an upload button to send the current relationships and data to the running app instance on Azure.
   1. Actors: User
   2. Entry condition: The user presses the upload button
   3. Normal Flow: The desktop CIOBrain takes the data currently in use for the desktop application and sends it to Azure. The CIOBrain API running on Azure will receive that data and add the assets to the existing dataset. CIOBrain running on Azure will show the newly received assets.
   4. Exit Condition: Data is successfully transferred to Azure CIOBrain API instance.
   5. Exceptions:
      1. Exception #1 - CIOBrain API not found at configured URL
         1. The user is prompted from a popup window when pressing the **Upload** button on the desktop app to input the URL where the API instance is running on Azure.
         2. The user selects **Upload** from the popup window after configuring the URL.
         3. A popup window will be displayed indicating the API could not be contacted or accessed at the given URL if not found.
   6. Special requirements: None.

### Requirement 3

CIOBrain shall support login functionality utilizing a password system.

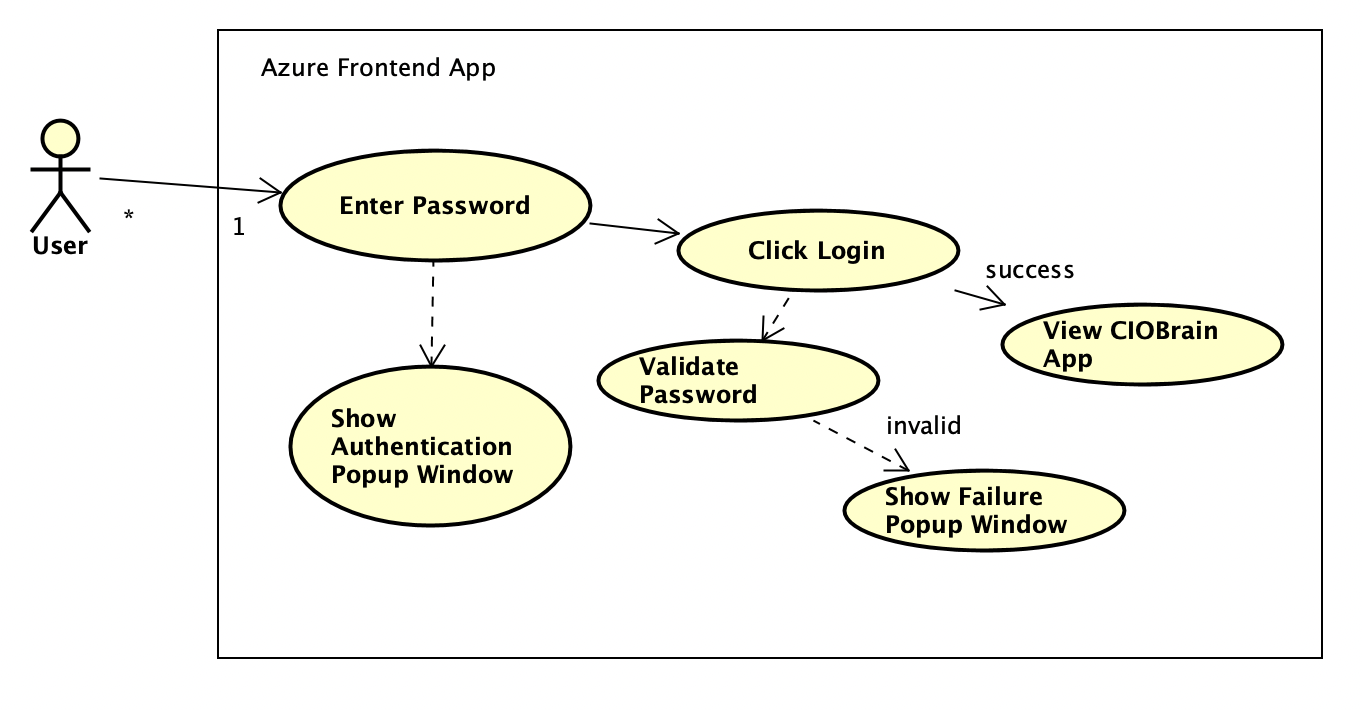


Figure 3

1. A user who attempts to access the CIOBrain application running on Azure will need to input a password to access the application and its data.
   1. Actors: User
   2. Entry condition: User enters login credentials on login screen
   3. Normal Flow: User enters login credential, CIOBrain authenticates login, redirects to main CIOBrain interface that displays all assets data
   4. Exit Condition: User either log outs or the program rejects login information.
   5. Exceptions:
      1. Exception #1: Password failure
         1. User is prompted to enter a password to access the application.
         2. Password is not validated successfully and a popup window appears indicating this to the user.
         3. User is able to submit another password attempt.
   6. Special Requirements: None

### Requirement 4

The instance of the app running on Azure shall be able to handle multiple users logging in and running CIOBrain at once.

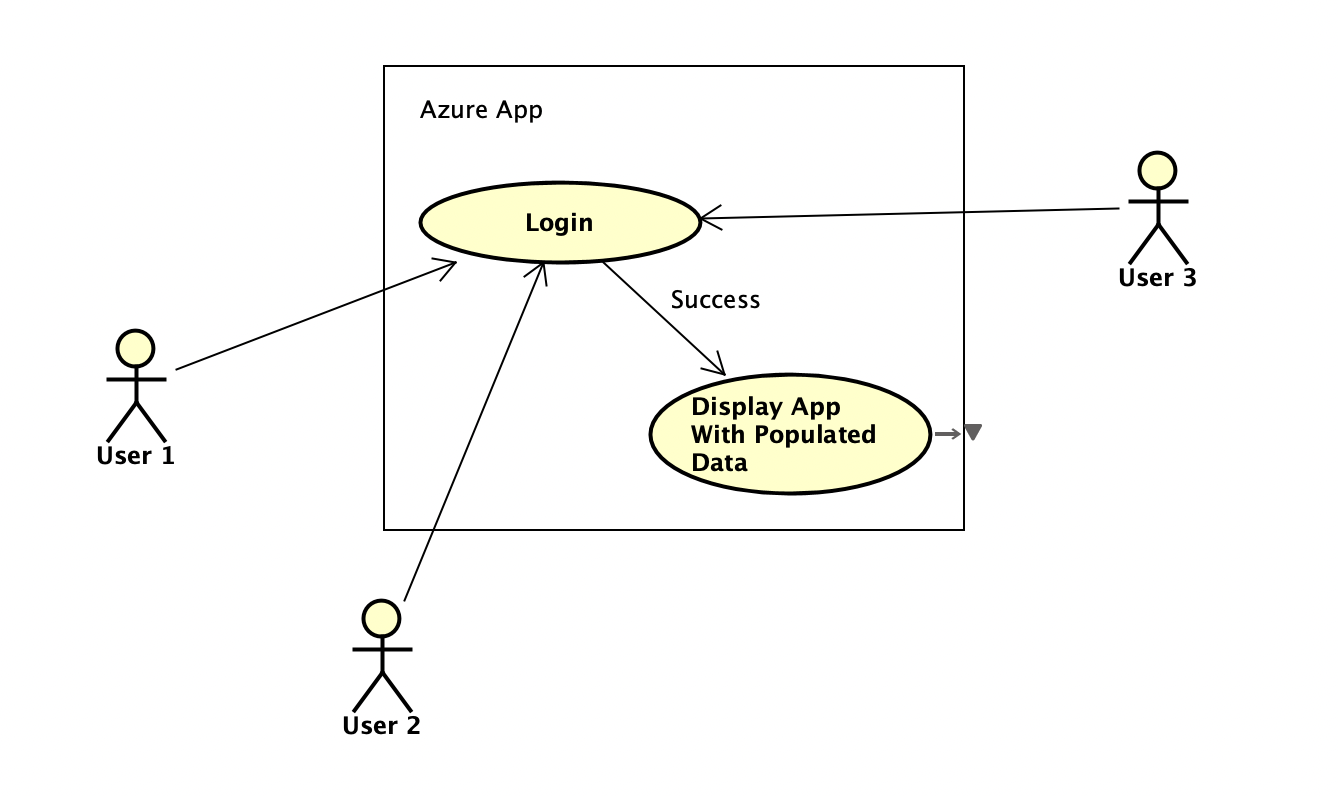


Figure 4

1. The app running on Azure shall permit multiple users to log in individually to the application before being presented with the application if authenticated successfully.
   1. Actors: User
   2. Entry condition: User or Users login at once
   3. Normal Flow: CIOBrain on the cloud takes in a login from a user, validates the password, and permits the user on that browser to view the CIOBrain application if the credentials were verified.
   4. Exit Condition: System goes offline.
   5. Exceptions: None
   6. Special Requirements: None

# Rationale for Use Case Model

The use cases are all derived from functional requirements that we will be implementing this semester and they do not cover requirements that have already been implemented.

### Requirement 1

Users currently need to launch both the CIOBrain frontend React application and API application separately on a machine to view the application. Creating an easy to install .msi installer will allow users to install the application, have a desktop shortcut created, and click on the shortcut to run the application without having to run both frontend and backend apps separately.

Other installers for Mac and Linux should optionally also be supported alongside the .msi installer for Windows. This option will be available in the ciobrain-native installer builder application that is being developed.

### Requirement 2

Users who are maintaining the application on Azure would not have the option to transfer data from the desktop application to this running instance on the cloud. They would need to instead manually transfer the Excel spreadsheets to the Azure instance by importing. To improve convenience, an **Upload** button should be added to the desktop app to send data to a running CIOBrain Azure instance with a given URL where the API is hosted.

### Requirement 3

Users can access a running CIOBrain instance hosted on Azure without any authentication. When supporting multiple users that may visit the hosted application, a password system is needed to ensure unauthorized users are not accessing sensitive data. A password system when accessing the application will ensure unauthorized users are restricted from viewing the data.

### Requirement 4

A single log-in workflow would result in other users accessing the application to not have to input a password since verification was already permitted to a single user. The authentication system must be dependent on the client’s browser so multiple users attempting to access the application will all need to enter credentials before being permitted in.

# Nonfunctional Requirements

**Reliability**

* The native application installer shall run to completion without errors 99% of the time.
* The CIO brain application hosted on Azure shall be operational as long as Azure is operational.

**Compatibility**

* The backend API shall be compatible with any PC environment.
* The backend API shall be able to be deployed onto Microsoft Azure.
* The frontend shall be able to be deployed onto Microsoft Azure.
* The frontend shall be viewable using Internet Explorer, Microsoft Edge, Mozilla Firefox, Google Chrome, and Safari Web Browser.
* The native application installer shall be an .msi installer running on Windows.
* The native application shall run on Windows.

**Portability**

* The application hosted on Azure shall be accessible in full feature form on PC and mobile devices.

**Capacity**

* The application shall handle at least 20 concurrent users in its cloud deployed form.

**Robustness**

* The native installer shall provide an error message if there is an issue with installation.
* The system shall inform the user if the native application’s API process could not be run.
* The system shall inform the user if the native application’s React process could not be run.
* The system shall inform the user if the native application could not transfer data to the running cloud application
* The system shall inform the user if a password could not be set for the cloud application
* The system shall inform the user if the password inputted was not valid
* The system shall obscure source code from users and impede reverse engineering
* The Azure hosted version shall inform the user to the causes of any errors (eg. Server outage, improper format, spreadsheet not found)

**Online User Documentation and Help**

* There shall be a sample file provided to the end-user to be able to have a correctly-formatted blank file for assets to be imported.
* A link to this sample file shall be found immediately below the Import button.
* Documentation for Azure installation and run-time instructions shall be created so users can easily deploy the application to the cloud.
* Documentation for desktop app installation using the .msi file shall be created

**Performance**

* The front end application shall be run client side to reduce server usage.
* The Azure hosted version shall not slow down within acceptable traffic volumes (<40 concurrent users)

**Usability**

* The native installer shall be able to be ran without reading any documentation
* The native installer shall produce a desktop icon that launches the application
* The native application should have simple one-click data transfer capabilities to a running instance on Azure

# Evidence Requirements Have Been Placed Under Configuration Management

# 

# References

N/A