**CIOBrain Deployment**

Testing Plan

Anton Horvath

Anuja Sahu

Scott Lorance

Kevin Lieng

Jose Colina Salas

Saige Wright

# Abstract

This test plan document introduces the CIOBrain deployment’s test cases, their traceability to their respective use case, and the techniques that were used to generate them. Note that this document only provides the test cases that involve the additions or modifications that we made to the existing product. A test case shows its description, the priority of the test case, the precondition before beginning the test, the steps to test it itself, and the expected results of the test case. The document includes a traceability section that shows the respective connections between a test case and a use case. Finally, this document states the techniques that were used to generate the test cases.

# Table of Contents

[**Abstract**](#_x6rwfpx8ibno) **2**

[**Table of Contents**](#_oi6ud0wjlf0m) **3**

[**List of Figures**](#_n5yatnf7llxl) **4**

[**List of Tables**](#_f9g83v8zvuly) **5**

[**Introduction**](#_fol7u981ugxu) **6**

[**Specification-based System Level Test Cases**](#_j2t9pkqxa6d3) **7**

[**Traceability of Test Cases to Use Cases**](#_h0i4gexv9wfg) **12**

[**Techniques for Test Generation**](#_mqh3i59kdbc7) **13**

[**Evidence the Test Plan Has Been Placed Under Configuration Management**](#_35ubn5vjgh0p) **14**

[**References**](#_kb0iqtnj5ixl) **15**

# List of Figures

Figure 115

# List of Tables

Table 112

# Introduction

The purpose of the test plan for CIOBrain is to show an overview of the test cases as well as the techniques used to create them. Since we are adding and modifying onto an already made software, this test plan will focus on the test cases that involve our additions.

The scope of the test plan includes the following:

* Requirements/Specifications-based System Level Test Cases  
  This section introduces the test cases.
* Traceability of Test Cases to Use Cases  
  This section shows the connection between the test cases and the use cases.
* Techniques for Test Generation  
  This section specifies which techniques were used for generating the test cases and which criteria were used to measure the quality of our tests, as well as whether or not we used black-box or white-box based testing.

# Specification-based System Level Test Cases

**Test Case #:** 1

**Title:** Secure Login Successfully

**Description:** On startup and refresh of the application, the user should be prompted for a login that should lock out all features of CIOBrain until login has been successfully done.

**Priority [Very High, High, Medium, Low, Very Low]:** High

**Precondition(s):** A data connection into the server exists and a password has been set

**Test Steps:**

1. Open up the CIOBrain application locally on the desktop
2. Confirm the Login screen has shown up and everything else is untouchable
3. Attempt a successful login with the correct password
4. Refresh the application after a successful login to show the login screen reappears

**Expected Result:** A login screen should show on the screen and be able to be logged in once a correct password is entered.

**Test Case #:** 2

**Title:** Incorrect Login Blocked Successfully

**Description:** On startup and refresh of the application, the user should be prompted for a login that should lock out all features of CIOBrain and should reject any and all incorrect attempts of password.

**Priority [Very High, High, Medium, Low, Very Low]:** Medium

**Precondition(s):** None

**Test Steps:**

1. Open up the CIOBrain application locally on the desktop
2. Confirm the Login screen has shown up and everything else is untouchable
3. Attempt a unsuccessful login with an incorrect password
4. Refresh the application after a successful login to show the login screen reappears

**Expected Result:** The login screen failed and has rejected the login password.

**Test Case #:** 3

**Title:** Data transfer from local machine to Azure application

**Description:** Upon clicking the ‘data transfer’ button, the files chosen for transfer from the user’s local machine should move to the Azure application’s database after the user enters the URL for destination and the correct password set for the account. However, if the data files contain duplicates, those files should be rejected.

**Priority [Very High, High, Medium, Low, Very Low]:** High

**Precondition(s):** User enters the URL and password for data transfer

**Test Steps:**

1. Import a chosen Excel database from the computer’s local machine onto the local CIOBrain application
2. Click on the Upload button
3. Login in using the correct Azure URL and password
4. Confirm that the application is starting data transfer to the database
5. Navigate to the Azure CIOBrain application
6. Reject any duplicate files during the data transfer
7. Login into the cloud application and confirm the files appear on the database of the application

**Expected Result:** The Azure application database should hold the Excel files that the user intended to transfer from their local machine.

**Test Case #:** 4

**Title:** Deploying from Powershell on a Windows system

**Description:** User authenticates github account while trying to deploy from powershell

**Priority [Very High, High, Medium, Low, Very Low]:** High

**Precondition(s):**

**Test Steps:**

1. The user opens powershell to deploy application
2. The user is prompted to input password to authenticate github account
3. The one time code generated is input incorrectly by user

**Expected Result:** Powershell displays message indicating incorrect code

**Test Case #:** 5

**Title:** Correct password for data transfer

**Description:** When the user enters the password during data transfer of files from the local machine to Azure’s database, the application must verify that it is the password set for the account. However, if the incorrect password is set, the data transfer is canceled and the user is prompted to re-enter the correct password.

**Priority [Very High, High, Medium, Low, Very Low]:** Low

**Precondition(s):** The user clicks the ‘data transfer’ button

**Test Steps:**

1. Open the CIOBrain application.
2. Click on the Upload button.
3. Login in using the correct Azure password.
4. Confirm that the password is correct.

**Expected Result:** The data transfer from the user’s local machine to the Azure application will begin.

**Test Case #:** 6

**Title:** URL user input while deploying on Windows

**Description:** A URL will be created during the deployment process, and the input needs to copy it into the github account

**Priority [Very High, High, Medium, Low, Very Low]:** Very High

**Precondition(s):**

**Test Steps:**

1. The user opens powershell to deploy application
2. A URL will be generated
3. The user needs to copy and paste URL on github

**Expected Result: URL is copied correctly and the deployment proceeds.**

**Test Case #:** 7

**Title:** Remove duplicate files in data transfer

**Description:** Upon clicking the ‘data transfer’ button, the files chosen for transfer from the user’s local machine to the Azure application should check for any duplicate files. If there exists any, then those files should be rejected.

**Priority [Very High, High, Medium, Low, Very Low]:** Low

**Precondition(s):** User enters the correct URL and password for Azure application

**Test Steps:**

1. Open the CIOBrain application.
2. Click on the Upload button.
3. Enter the URL for the data transfer destination.
4. Login in using the correct Azure password.
5. Confirm that the password is correct.
6. Confirm that the application is starting data transfer to the database

**Expected Result:** Any duplicate files should be rejected during data transfer.

**Test Case #:** 8

**Title:** Deployment attempts

**Description:** The user attempts to deploy the application more than once.

**Priority [Very High, High, Medium, Low, Very Low]:** Low

**Precondition(s):** User has already deployed application.

**Test Steps:**

1. Deploy the CIOBrain application successfully
2. User follows all steps to deploy the application again.
3. Azure already created resource groups, and the web app.

**Expected Result:** Once the user deploys the application correctly, second attempt will not work.

**Test Case #:** 9

**Title:** Native Program Run

**Description:** Once the MSI installer has successfully installed the program, the program should run successfully on the machine from the shortcut that was created on the desktop.

**Priority [Very High, High, Medium, Low, Very Low]:** Very High

**Precondition(s):** The CIOBrain installer has installed the program onto the Windows machine.

**Test Steps:**

1. Click on the CIOBrain application shortcut on the desktop.
2. Verify that the application works.
   1. Able to select assets.
   2. Able to view node information via long-press (hover equivalent).
   3. Able to see all relationships, implicit and explicit, between selected nodes.
   4. Able to import more Excel spreadsheets.
   5. Able to transfer data to an Azure instance.

**Expected Result:** The native application works correctly when launched on a Windows machine.

**Test Case #:** 10

**Title:** MSI Installer Builder

**Description:** The native application needs to be installed using an MSI installer. This MSI installer is built using a builder application that should successfully package together the API and React applications into a single native application.

**Priority [Very High, High, Medium, Low, Very Low]:** Very High

**Precondition(s):** None

**Test Steps:**

1. Clone the project onto a local machine.
2. Run ‘npm install’ to install all dependencies of the project.
3. Run ‘npm run electron:package:win’ to build the MSI installer for Windows.
4. Check the ‘dist’ folder once complete.

**Expected Result:** The MSI installer should be present within the ‘dist’ folder of the project.

**Test Case #:** 11

**Title:** Dynamic Fields On Excel Spreadsheets

**Description:** The Excel spreadsheets that are imported into the CIOBrain application should be able to support custom-named fields along with the preset fields needed to import.

**Priority [Very High, High, Medium, Low, Very Low]:** Medium

**Precondition(s):** CIOBrain application is running.

**Test Steps:**

1. Import Excel spreadsheet with new custom fields.
2. Check to make sure the fields do not break the graph.
3. Fields should be shown on the UI.

**Expected Result:** The custom fields should not break the application and should appear on the application UI.

# Traceability of Test Cases to Use Cases

| **Test Case #** | **Associated Use Case** |
| --- | --- |
| 1,2 | CIOBrain shall support login functionality utilizing a password system. |
| 3,7 | CIOBrain shall have a one click transfer capability from desktop app to Microsoft Azure |
| 6 | The instance of the app running on Azure shall be able to handle multiple users logging in and running CIOBrain at once. |
| 4,5,8 | Deployment |
| 9, 10, 11 | There should be a one click MSI installer for smooth and easy installation |

Table 1: Test Cases and Their Associated Use Cases

# Techniques for Test Generation

* **Boundary value analysis:** Testing the successful scenario, then the bordering unsuccessful scenarios (examples: T01, T02).
* **Specification-based technique**: Test cases are generated in part from the formally defined requirements specification. (T03, T04, T05, T10)
* **Black-box testing:** Test cases that will simply be tested from outside of the development environment (T06)

# Evidence the Test Plan Has Been Placed Under Configuration Management

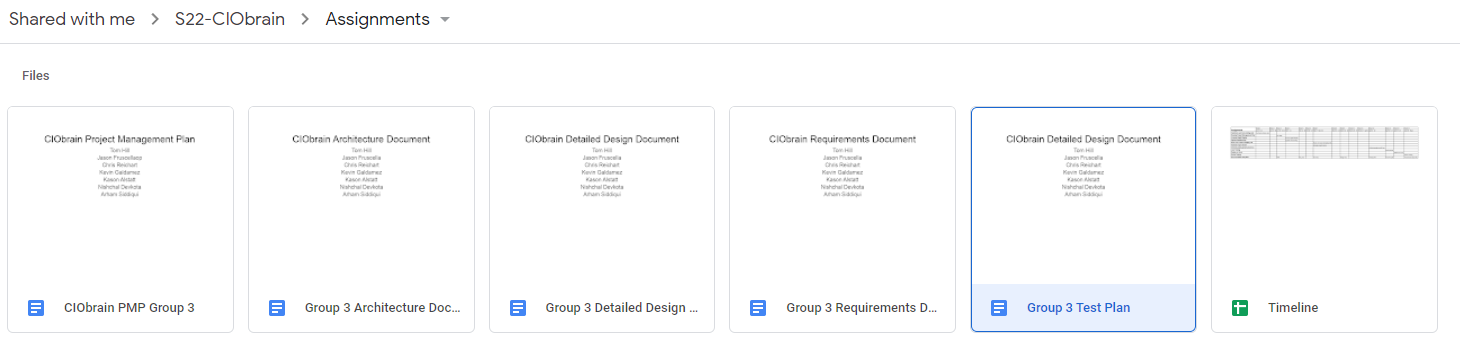


Figure 1: Evidence of Configuration Management

# References

N/A