A green and black logo

Description automatically generated

New Reporting Tool Implementation Using Python Streamlit

ICM Computer Systems Limited – VER 20241118

I7DW: new Reporting Tool Implementation Using Python Streamlit

ICM COMPUTER SYSTEMS LIMITED

2024

Table of Contents

[1. Overview 1](#_Toc182820015)

[2. Installation Instructions 1](#_Toc182820016)

[2.1 Choose an Installation Directory: 1](#_Toc182820017)

[2.1.1 Decide where you want the application files to be installed. 1](#_Toc182820018)

[2.1.2 Move and Extract the ZIP Folder: 1](#_Toc182820019)

[2.2.3 Navigate to the Extracted Folder: 1](#_Toc182820020)

[3. Installing Python: 1](#_Toc182820021)

[4 Set up the Config.toml and Screts.toml file 2](#_Toc182820022)

[4.1 Open the secrets.toml file. 2](#_Toc182820023)

[4.2 Update the secrets\_tla.toml 2](#_Toc182820024)

[5.Run the Setup Batch File: 3](#_Toc182820025)

[6. Access the Application: 3](#_Toc182820026)

[7. Setting Up of Streamlit Apps on Windows 3](#_Toc182820027)

[7.1 Step 1: Create a Batch Script to Run Streamlit Apps in the Background 3](#_Toc182820028)

[7.1.1 Creating the .bat file: 3](#_Toc182820029)

[7.1.2. Save the file. 4](#_Toc182820030)

[7.2 Step 2: Set Up Task Scheduler to Run the Script at Startup 4](#_Toc182820031)

[7.2.1. Opening the task Scheduler. 4](#_Toc182820032)

[7.3 Step 3: Create a New Task 4](#_Toc182820033)

[7.4 Step 4: Configure Triggers 4](#_Toc182820034)

[7.5 Step 5: Configure Actions 4](#_Toc182820035)

[7.6 Step 6: Test the Task 4](#_Toc182820036)

[8. Configure Inbound Rules in Windows Firewall 5](#_Toc182820037)

[9. Required Software and Libraries 5](#_Toc182820038)

[9.1. Python: 6](#_Toc182820039)

[9.2 pip (Python Package Installer): 6](#_Toc182820040)

[9.3. Pre-downloaded Python Dependencies (.whl files): 6](#_Toc182820041)

[9. 4. Streamlit: 6](#_Toc182820042)

[10. Python Naming Conventions and Best Practices 6](#_Toc182820043)

[10.1 Summary of Naming Conventions 6](#_Toc182820044)

[11. Streamlit Application 0](#_Toc182820045)

[http://localhost:8501 0](#_Toc182820046)

[12. Documentation 1](#_Toc182820047)

[http://localhost:8502 1](#_Toc182820048)

A screenshot of a computer

Description automatically generated

# 1. Push the Git Code to AzureDevops

## Step 1: Generate a New GitHub Personal Access Token (PAT)

Go to Settings > Developer settings > Personal access tokens in GitHub.

Click Generate new token.

Name the token (e.g., "Azure DevOps Migration").

Select the repo scope.

Click Generate token and copy it for later use.

Step 2: Clone Your GitHub Repository with the New Token

Open a terminal or Visual Studio Code.

Clone your private GitHub repository using your GitHub username and new PAT in the URL:

git clone https://<your-username>:<your-PAT>@github.com/nicolasSilvaBR/Infeed700Plus.git

Replace <your-username> with your GitHub username and <your-PAT> with the token you just created.

      PAT = Token

## Step 3: Set Up Your New Project and Repository in Azure DevOps

Log in to Azure DevOps with your company account.

Create a New Project in Azure DevOps:

Click New Project and name it.

Set the visibility and select Git for version control.

Create a Repository:

In your new project, go to Repos > Files.

If a repository isn’t automatically created, click New Repository.

## Step 4: Add Azure DevOps as a Remote in Your Local Git Repository

In the terminal, navigate to your cloned GitHub repository folder:

cd Infeed700Plus

Add the Azure DevOps repository as a second remote:

git remote add azure <http://icmdevops1:8080/tfs/DefaultCollection/Infeed700%20Python%20%20SSRS/_git/Infeed700%20Python%20%20SSRS>

## Step 5: Push Your Code to Azure DevOps

Push all branches from your local repository to the new Azure DevOps repository:

git push azure --all

git push azure –tags

## Step 6: Verify in Azure DevOps

Go to Repos > Files in your Azure DevOps project to confirm that all code and branches were successfully uploaded.

# 11. Streamlit Application

## <http://localhost:8501>