|  |
| --- |
|  |

**MUCP Local Installation MANUAL**

**by**

**Kirodh Boodhraj**

**Handed over to DFFE on 2 October 2025**

|  |  |
| --- | --- |
| Executive Summary | |
| Title: | MUCP Local Installation Manual (Kirodh Boodhraj) |
| Abstract: | How to install the MUCP web tool locally. |
| Keywords: |  |
| Objective: | Manual written for DFFE and future users. |
| Purpose: |  |
| Methodology: |  |
| Conclusions: |  |
| Limitations or Risks: |  |
| Benefits: |  |
| Contract Context: | This document is the result of a research effort funded by NRM and CSIR PG internal funding. |

|  |  |
| --- | --- |
| Approval Page | |
| Project Name | MUCP |
| Project No |  |
| Title |  |
| Prepared for |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Authored for | CSIR | Signature | Date |
| By | Kirodh Boodhraj | Kirodh | 29 Sept. 25 |
| Designation |  | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Change History | | | | |
| Change Control No | Date | Name | Description |
| Version 1 | 29 Sept. 25 |  | Original release. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[Abbreviations 4](#_Toc210040961)

[1 Introduction 5](#_Toc210040962)

[2 Get the code 6](#_Toc210040963)

[2.1 Install Git (if not already installed) 6](#_Toc210040964)

[2.2 Clone the MUCP Tool Repository 6](#_Toc210040965)

[2.3 Navigate into the Project Folder 7](#_Toc210040966)

[3 Downloading Python 8](#_Toc210040967)

[3.1 Check if Python is already installed 8](#_Toc210040968)

[3.2 Download Python 8](#_Toc210040969)

[3.3 Setting Up a Virtual Environment 9](#_Toc210040970)

[3.3.1 Create the virtual environment 9](#_Toc210040971)

[3.3.2 Activating the Virtual Environment 9](#_Toc210040972)

[3.3.3 Deactivating the Virtual Environment 10](#_Toc210040973)

[3.4 Quick Workflow Summary 10](#_Toc210040974)

[4 Setting up the environment (installing Python packages) 12](#_Toc210040975)

[5 Initial Program setup 13](#_Toc210040976)

[5.1 Preparing the Django Environment 13](#_Toc210040977)

[5.2 Applying Migrations 13](#_Toc210040978)

[5.2.1 Make migrations (create migration files if any new models were added or modified): 13](#_Toc210040979)

[5.2.2 Apply migrations (update the database structure): 13](#_Toc210040980)

[5.3 Creating a Superuser (Admin Account) 13](#_Toc210040981)

[5.4 Collecting Static Files (If Required) 14](#_Toc210040982)

[6 Running and accessing the viewer on your web browser 15](#_Toc210040983)

[7 Support and debugging 16](#_Toc210040984)

Abbreviations

|  |  |
| --- | --- |
| CSIR | Council for Scientific and Industrial Research |
| DFFE | Department of Forestry, Fisheries and Environment |
| MUCP | Management Unit Control Plan |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

Le Maitre et al. (2012) developed a generic species and area prioritization model (MUCP Tool) for use in prioritising invasive alien plant control operations in South Africa using spatial data and Forsyth et al. (2012) . It schedules treatments of invasions in the catchment taking into account the current state of the invasions, benefits of clearing, treatments that are required and the resources provided in its budget. The MUCP tool does not generate a detailed annual schedule of annual operations but the DEA-Natural Resources Management (NRM) programme has an Annual Plan of Operations tool which serves that purpose.

This manual will guide the user on installing the Web version of the MUCP tool on a local machine (Windows or Linux, untested for MacOS).

Outlined process consists of:

* Get the code
* Downloading Python
* Setting up the environment (installing Python packages)
* Initial Program Setup: Setting up the program with a superuser, performing migrations and collecting static files (if necessary)
* Running and accessing the viewer on your web browser

# Get the code

Downloading the MUCP Tool Code

Before setting up the Python environment, you need to obtain the source code for the MUCP tool from the official repository.

## Install Git (if not already installed)

Windows:

Download Git from <https://git-scm.com/download/win>

git --version

Linux (Debian/Ubuntu):

sudo apt update

sudo apt install git

After installation, confirm Git is available by running:

git --version

You should see an output like git version 2.43.0.

## Clone the MUCP Tool Repository

Navigate to the folder where you want to download the project. For example:

cd ~/projects

Then run the following command to download the MUCP code:

git clone <https://gitlab.com/kirodh/mucp-viewer-django-web.git>

This will create a new folder named mucp-viewer-django containing all the project files.

## Navigate into the Project Folder

Once the repository has been cloned, move into the project directory:

cd mucp-viewer-django

# Downloading Python

## **Check** if **Python** is already installed

On Windows:

Open Command Prompt and run:

python --version

If you see a version number (e.g., Python 3.12.2), Python is already installed. If not, continue with the steps below.

## Download Python

Visit the official Python website: <https://www.python.org/downloads/>

Choose the latest Python 3.11 or 3.12 release ( or later, but avoid Python 2.x).

Download the installer for your operating system (Windows, or Linux).

Install Python following the instructions.

Windows:

Run the installer (python-3.x.x-amd64.exe).

On the first screen, check the box:

✅ Add Python 3.x to PATH

Click Install Now.

Linux (Ubuntu/Debian):

Python is usually pre-installed. If not, run:

sudo apt update

sudo apt install python3 python3-pip

## Setting Up a Virtual Environment

A virtual environment is an isolated workspace where you can install the required packages for the MUCP tool without interfering with system-wide Python.

Open Terminal / Command Prompt

Navigate to your project directory where the MUCP tool code is located:

cd path/to/your/project

### Create the virtual environment

Run:

python -m venv venv

This will create a folder called venv/ in your project.

Inside it, Python and pip (Python package manager) will be installed.

### Activating the Virtual Environment

Windows (Command Prompt)

venv\Scripts\activate

Windows (PowerShell)

.\venv\Scripts\Activate.ps1

Linux

source venv/bin/activate

When the environment is active, your terminal will show (venv) at the beginning of the line. Example:

(venv) C:\Users\YourName\project>

### Deactivating the Virtual Environment

When you are done working, deactivate the environment by running:

deactivate

The (venv) prefix will disappear, and you’ll return to the system’s default Python environment.

## Quick Workflow Summary

Open terminal in your project folder

Activate the environment:

Windows:

venv\Scripts\activate

Linux:

source venv/bin/activate

Work on the project (run commands, install packages, etc.)

When finished, type:

deactivate

# Setting up the environment (installing Python packages)

Before setting up the environment and installing packages, first activate the virtual environment as per the previous section.

Once the environment is active, you can install the required packages. For example:

pip install Django

But for convenience the MUCP tool comes with a requirements.txt file (list of all dependencies) already populated for you, first navigate to the folder where the requirements.txt file is then install everything at once with:

pip install -r requirements.txt

This will install all the necessary packages.

# Initial Program setup

Setting up the Django Project

## Preparing the Django Environment

Once your virtual environment is active and all required packages are installed, you need to configure Django for local use.

Make sure you are in the project root directory (the same folder as manage.py).

## Applying Migrations

Django uses a database to store information. Before running the project, you must set up the database schema.

### Make migrations (create migration files if any new models were added or modified):

python manage.py makemigrations

If there are no model changes, Django will tell you “No changes detected”.

If changes exist, new migration files will be created in each app’s migrations/ folder.

### Apply migrations (update the database structure):

python manage.py migrate

This command ensures your database is up to date with all the required tables.

## Creating a Superuser (Admin Account)

Django includes an administrative interface. To access it, you need a superuser account.

Run the following:

python manage.py createsuperuser

You will be prompted for:

Username

Email address

Password

⚠️ Remember your login details — you will need them to access the Django Admin panel.

## Collecting Static Files (If Required)

Django apps use static files (CSS, JavaScript, images). For local development, this is usually handled automatically. For production or when you want all static files in one place, run:

python manage.py collectstatic

This gathers all static files into the folder defined in your settings.py (STATIC\_ROOT).

You may be asked if you want to overwrite files — usually you can type yes.

# Running and accessing the viewer on your web browser

Once the initial Django program is setup, we can run the Development Server.

To test your MUCP tool locally:

python manage.py runserver

By default, this starts the server at:

👉 <http://127.0.0.1:8000/>

Open your browser and enter the above address. You should see the Django site.

To access the admin panel, go to:

👉 <http://127.0.0.1:8000/admin/>

Login with the superuser credentials you created earlier.

⚡ Tip (Optional):

If you want the server to be available on your local network (e.g., for testing on another device), run:

python manage.py runserver 0.0.0.0:8000

# Support and debugging

There are many resources out there for putting a Django program into production. Some include:

* ChatGPT and other advanced AI chatbots
* StackOverflow
* Youtube

Manual End