**Step 1 — Create the project folder and structure**

📍 Location: Your laptop → OneDrive\Desktop\Semester One\Agile Processes\MyWeatherChannel

**What you did:**

* Made a folder for your project: MyWeatherChannel.
* Inside it, you created subfolders:
  + scripts/ → Python code lives here.
  + data/ → raw and cleaned weather files.
  + web/ → later for website (charts, HTML).
  + docs/ → documentation (checklists, burndown, screenshots).

**Why it matters:**

* In real projects, **organisation is key**. You don’t want code, data, and notes all mixed in one folder.
* This mirrors what professional teams do — separation of **logic, data, presentation, and documentation**.

**Step 2 — Open PowerShell in the project root**

📍 You navigated into:  
C:\Users\ddree\OneDrive\Desktop\Semester One\Agile Processes\MyWeatherChannel

**What you did:**

* Used cd (change directory) to move into your project folder.
* Ran commands from there.

**Why it matters:**

* Git and Python commands always act on “the current working directory”.
* Being in the root means Git will track the whole repo, not just one folder.

**Step 3 — Initialise Git**

**Command:**

git init

**What it does:**

* Creates a hidden .git/ folder inside your project.
* This folder holds your commit history, settings, and branches.

**Why it matters:**

* You’ve just told Git: “Track this project.”
* From now on, every snapshot (commit) you take will live here.

**Step 4 — Create a README file**

**Command:**

echo "My Weather Channel" > README.md

**What it does:**

* Creates a file README.md and writes “My Weather Channel” inside it.
* .md = Markdown (formatting language).

**Why it matters:**

* Every GitHub project has a README explaining what it is.
* It gives you a file to commit as your first test.

**Step 5 — Configure Git identity**

**Commands:**

git config --global user.name "Darren Dreeling"

git config --global user.email "ddreeling@hotmail.com"

**What it does:**

* Tells Git who you are globally on your computer.
* Every commit will have your name + email.

**Why it matters:**

* Commits must have an author.
* On GitHub, this connects commits to your profile.

**Step 6 — Save first snapshot (commit)**

**Commands:**

git add .

git commit -m "Initial commit"

**What it does:**

* git add . → stages all files (tells Git “I want to include these in the next snapshot”).
* git commit -m "message" → creates a snapshot with your message.

**Why it matters:**

* Commits are like **save points in a video game**.
* You can always roll back to them.

**Step 7 — Rename branch to main**

**Command:**

git branch -M main

**What it does:**

* Changes default branch name from master → main.

**Why it matters:**

* main is the modern standard.
* Makes your repo consistent with GitHub’s default.

**Step 8 — Link local repo to GitHub**

**Command:**

git remote add origin https://github.com/ddreeling/my-weather-channel.git

**What it does:**

* Adds a remote connection named origin.
* That remote points to your GitHub repository.

**Why it matters:**

* Without this, your local repo doesn’t know where to push code.
* This is the “bridge” between local and cloud.

**Step 9 — Try pushing (hit flag issue)**

**Command:**

git push -u origin main

**What happened:**

* Git tried to authenticate via browser (OAuth).
* Your browser opened → GitHub showed “account flagged” banner.

**Why it matters:**

* The push failed because flagged accounts can’t complete browser OAuth.
* You appealed this and also learned SSH/PAT are alternatives.

**Step 10 — Write the weather script**

**File:** scripts/fetch\_weather.py

**What it does (step by step):**

1. **Imports libraries**:
   * requests → fetch data from the internet.
   * pandas → handle data tables.
   * json, os, datetime → handle files and dates.
2. **Sets location**: Cork coordinates (latitude & longitude).
3. **Builds API URL**: Open-Meteo forecast endpoint with temperature, precipitation, wind.
4. **Fetches JSON**: Downloads raw data from Open-Meteo.
5. **Saves raw file**: Writes response to /data/cork\_raw.json.
6. **Cleans data**: Picks useful fields → timestamp, temp, precipitation, wind.
7. **Formats rows**: Converts strings into real datetime, rounds numbers.
8. **Stores CSV**: Writes clean dataset to /data/cork\_clean.csv.
9. **Prints preview**: Shows first few rows in console for proof.

**Why it matters:**

* This delivers the **Data Input + Cleaning + Storage** part of Sprint 1.
* It gives you raw + clean versions → vital Agile evidence.

**Step 11 — Install Python packages**

**Command:**

pip install pandas requests

**What it does:**

* Downloads the libraries your script needs.

**Why it matters:**

* Python doesn’t include everything by default.
* You install only what you need for each project.

**Step 12 — Run the script**

**Command:**

python scripts\fetch\_weather.py

**Result:**

* data/cork\_raw.json created (raw messy API).
* data/cork\_clean.csv created (tidy table).
* First few rows printed to terminal.

**Why it matters:**

* Proves your script works.
* Gives files for Sprint 2 to visualise.

**Step 13 — Create Sprint 1 documentation**

Inside docs/ you made three things:

1. **Checklist (Markdown)**
   * User stories, acceptance criteria, evidence.
   * Shows what Sprint 1 was supposed to achieve → and that you did it.

@"

# Sprint 1 – Data Input (My Weather Channel)

## Sprint Goal

Get weather data into the project from an API, clean it into a usable table, and store both raw and clean copies.

## User Stories Completed

1) As a user, I want to fetch Cork weather from an API so that I can see the latest forecast.

- Acceptance: Running `python scripts/fetch\_weather.py` completes without error and downloads JSON for Cork.

2) As a user, I want the raw data saved in `/data` so I can verify the source later.

- Acceptance: A file exists at `data/cork\_raw.json` after the script runs.

3) As a user, I want a cleaned CSV (timestamp, temperature\_c, precipitation\_mm, wind\_speed\_mps) so I can visualise it in Sprint 2.

- Acceptance: A file exists at `data/cork\_clean.csv` with those 4 columns and at least 10 rows.

## Definition of Done (DoD)

- Script fetches data for Cork without API keys using Open-Meteo.

- Raw JSON and cleaned CSV are saved under `/data`.

- The script prints a preview (head) of the cleaned table.

- Code and docs are committed to the repo.

## Evidence to Attach

- `Sprint1\_proof.png` (screenshot of terminal run + `/data` folder view).

- `Sprint1\_burndown.csv` (simple progress table).

"@ > docs\Sprint1\_checklist.md

1. **Burndown (CSV)**
   * Simple table showing how many story points remained each day.
   * Agile proof → shows you tracked progress.

@"

Day,Planned Points,Remaining Points,Notes

1,8,8,Sprint planning + repo setup

2,8,5,Script skeleton + first API fetch

3,8,2,CSV cleaning + validation

4,8,0,Screenshots + docs + commit

"@ > docs\Sprint1\_burndown.csv

1. **Proof screenshot (PNG)**
   * Console output + /data folder screenshot.
   * Visual evidence → great for demo/review.

**Step 14 — Commit Agile docs**

**Commands:**

git add docs/\*

git commit -m "Add Sprint 1 Agile docs: checklist + burndown + proof"

**What it does:**

* Stages + commits docs into Git.