Yes, absolutely! That level of detail is crucial for making your README.md (and overall project documentation) truly comprehensive and impactful. It bridges the gap between "I did this cool technical thing" and "Here's *why* this matters and *what specific value I added*."

Let's expand on what should go into your README.md and other documentation to cover these aspects:

**Revised README.md Structure & Content (with a focus on context, your contributions, and improvements):**

**Project Title:** A Data-Driven Exploration Platform for Cs₂NaTlBr₆₋ₓClₓ Perovskites: From DFT Outputs to Design Insights & Predictive Interpolation

**Table of Contents (Good practice for long READMEs)**

1. **Introduction & Motivation**
   * **1.1. The Promise of Cs₂NaTlBr₆₋ₓClₓ Double Perovskites:**
     + Briefly explain (in layman's terms first, then a bit more technically) what these materials are (lead-free double perovskites).
     + Mention their potential applications as described in the source paper's abstract (e.g., "photovoltaics," "optoelectronic applications," "solar cells," "photodetectors").
     + Highlight *why* they are interesting (e.g., "tunable bandgap," "higher stability" than lead-based alternatives, "strong optical absorption").
   * **1.2. The Source Research (Hasan et al., Dryad Dataset):**
     + Acknowledge the foundational DFT study by Hasan et al. (cite it with DOI).
     + State that their work provided the raw computational data exploring the impact of anion variation (Cl/Br ratio, 'x') on these material properties.
   * **1.3. Project Goal & My Contribution:**
     + Clearly state your project's overall objective (as defined before – to transform raw DFT data into an interactive, predictive "Material Insights Platform").
     + **Crucially, articulate what your project *adds* or *improves* upon the raw dataset:** "While the raw DFT data is invaluable, its original format (numerous disparate files) presents challenges for holistic analysis, trend discovery across multiple properties, and estimation for untested compositions. This project addresses these gaps by..." (then you'll elaborate in the "Key Achievements" section).
2. **Key Achievements & Novel Contributions of This Project**
   * **(This is where you detail *your new work* based on the earlier "New Achievements" points):**
   * **2.1. Creation of a Unified & Queryable Perovskite Data Platform:**
     + *Problem Addressed:* Disparate, hard-to-access raw DFT output files.
     + *My Solution:* Detailed the data engineering pipeline (parsing diverse file types, structuring data, creating the SQLite database).
     + *Benefit/Improvement:* "Enabled systematic querying, comprehensive property exploration, and served as the foundation for all subsequent advanced analyses."
   * **2.2. Development of Predictive Surrogate Models for Unexplored Compositions:**
     + *Problem Addressed:* DFT data available only for 7 discrete compositions; exploring intermediate compositions requires new, costly simulations.
     + *My Solution:* Detailed the development of Polynomial Regression and Gaussian Process Regression models to predict key properties (bandgap, effective mass, etc.) as a continuous function of anion concentration 'x'.
     + *Benefit/Improvement:* "Provides a novel capability for rapid property estimation for untested compositions, potentially guiding and accelerating future experimental or computational research by identifying promising intermediate 'x' values. GPR models also offer uncertainty quantification."
   * **2.3. Unveiling Multi-Property Design Trade-offs via an Interactive Hyperspace Explorer:**
     + *Problem Addressed:* Understanding how multiple desirable material properties simultaneously change with composition, and identifying optimal balances.
     + *My Solution:* Detailed the creation of interactive 3D visualizations plotting, for example, Bandgap vs. Mobility vs. Stability.
     + *Benefit/Improvement:* "Offers new visual insights into material design trade-offs not easily apparent from individual property plots, aiding in the identification of 'Pareto-like' optimal compositions for specific application needs."
   * **2.4. Implementation of a Comprehensive, Interactive "Perovskite Insights Dashboard":**
     + *Problem Addressed:* Making complex scientific data accessible and explorable for a wider audience beyond DFT experts.
     + *My Solution:* Described the Streamlit/Dash application, its features (material explorer, trend analyzer, predictive interpolator, trade-off visualizer).
     + *Benefit/Improvement:* "Transforms static research data into a dynamic tool, enhancing data accessibility, fostering deeper understanding, and enabling user-driven exploration."
   * **(2.5. Optional: Quantified Application-Specific Figures of Merit):** Detail this if you implemented it.
3. **Technical Overview**
   * **3.1. Data Source:** Link to Dryad dataset.
   * **3.2. Data Engineering Pipeline:**
     + Brief diagram or step-by-step description of your ingestion process (unzip -> inventory -> parse -> structure -> SQLite).
     + Brief overview of the SQLite database schema (key tables and their purpose).
   * **3.3. Analytical Methods & Modeling:**
     + Mention EDA techniques used.
     + Describe the surrogate models (Polynomial Regression, GPR), including choices made (e.g., polynomial degree, GPR kernel if applicable).
   * **3.4. Dashboard Technology:** Streamlit/Dash.
   * **3.5. Tools & Libraries Used:** Python, Pandas, NumPy, Scikit-learn, Plotly, Matplotlib, Seaborn, Streamlit/Dash, SQLite.
4. **Key Findings & Visualizations**
   * **(Showcase, don't just tell):**
   * **4.1. Impact of Anion Variation ('x') on Key Properties:**
     + Embed your best plots showing Bandgap vs. 'x', Effective Mass vs. 'x', Bulk Modulus vs. 'x', etc., with regression lines.
     + Briefly interpret each plot, comparing with the abstract (e.g., "The HSE bandgap was found to increase from ~1.78 eV (x=0) to ~2.98 eV (x=6), closely following a quadratic trend (R²=0.99), which refines the 'approximately linear' description from the source abstract.").
   * **4.2. Cross-Property Correlations:**
     + Embed your correlation matrix heatmap.
     + Highlight 1-2 interesting correlations found (e.g., "A strong positive correlation (r=0.85) was observed between calculated Bulk Modulus and Debye Temperature, suggesting materials that are stiffer also exhibit higher characteristic lattice vibration frequencies.").
   * **4.3. Insights from Surrogate Models:**
     + Show a plot of your surrogate model's prediction (e.g., for HSE Bandgap) along with the original 7 DFT points and confidence intervals (if GPR).
     + Example interpretation: "The GPR surrogate model for HSE bandgap provides accurate interpolation between the DFT data points and allows for estimation at x=2.5, predicting a bandgap of 2.25 ± 0.03 eV."
   * **4.4. Multi-Objective Design Considerations:**
     + Embed a key screenshot or animated GIF of your interactive 3D trade-off plot.
     + Example interpretation: "Interactive exploration revealed that while Cs₂NaTlCl₆ (x=6) offers the highest bandgap, Cs₂NaTlBr₂Cl₄ (x=4) presents a more compelling balance between a reasonably high bandgap (~2.6 eV) and higher predicted electron mobility, making it a potentially interesting candidate for [specific application]."
5. **The "Perovskite Insights Dashboard" - An Interactive Tool**
   * Describe the main features of your dashboard.
   * Include screenshots of different sections of the dashboard.
   * Link to the deployed dashboard (if you deploy it, e.g., on Streamlit Cloud) or provide clear instructions to run it locally.
6. **Limitations**
   * "The DFT data is from a single study and computational setup."
   * "Surrogate models are interpolative and their accuracy for extrapolating far beyond the x=0-6 range is unknown; they are intended for estimation within this range."
   * "The 'trade-off' analysis is based on the 7 calculated points; a finer grid of DFT calculations could reveal more nuanced frontiers."
   * "The dashboard provides insights based on theoretical calculations; experimental validation is always required."
7. **Future Work / Potential Enhancements**
   * "Incorporate more DFT output properties if parsers are developed (e.g., detailed PDOS, more complex elastic tensor components)."
   * "Extend surrogate models to include more input features if available (e.g., if data for slightly different atomic constituents were added)."
   * "Integrate experimental data for these compounds if it becomes publicly available, to validate DFT and surrogate model predictions."
   * "Add more sophisticated multi-objective optimization algorithms if more data points were available."
8. **Setup & Usage**
   * **8.1. Prerequisites:** Python 3.x, pip.
   * **8.2. Installation:**
   * git clone [your\_repo\_url]
   * cd perovskite\_insights\_platform
   * python -m venv venv
   * source venv/bin/activate # or venv\Scripts\activate on Windows
   * pip install -r requirements.txt

* + **8.3. Running the Data Pipeline (if needed by user to replicate DB):**
  + python ingestion\_scripts/01\_unzip\_and\_inventory.py
  + python ingestion\_scripts/02\_extract\_and\_structure\_data.py
  + python ingestion\_scripts/03\_build\_database.py

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

*(Or note if the SQLite DB is included in the repo and these are for understanding only).*

* + **8.4. Running the Dashboard:**
  + streamlit run app/main\_dashboard.py

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

By structuring your documentation this way, you clearly define the context, explicitly state your novel contributions and improvements, detail your technical approach, showcase your results, and provide clear instructions. This tells a compelling story of how you took existing data and created something new and valuable with it. This is what makes a portfolio project stand out!

# OBJECTIVE

To develop a "Perovskite Insights Platform" that transforms raw DFT simulation outputs (from Hasan et al.) into an interactive and predictive tool. This platform will allow users to:

Easily explore and visualize fundamental material properties (electronic, optical, structural, mechanical) for 7 different Cs₂NaTlBr₆₋ₓClₓ compositions.

Understand how these properties change as the Chlorine/Bromine ratio ('x') is varied.

Identify key design trade-offs between multiple desirable material properties.

Obtain rapid estimations of properties for new, untested intermediate compositions using predictive surrogate models.

Ultimately, make complex scientific data more accessible and actionable for understanding and potentially guiding the design of new optoelectronic materials.

Mapping Project Aims to Specific Data Needs & File Types:

Let's break down what data you need from which files to achieve each part of this aim:

Aim 1: Explore & Visualize Fundamental Material Properties for each of the 7 Compositions

Objective 1a: Visualize Electronic Band Structure.

Data Needed: Energy bands vs. k-path, locations of high-symmetry k-points, labels for these points.

Files to Use (from BAND or HSE folder for each material 'x'):

REFORMATTED\_BAND.dat (preferred) or BAND.dat: Contains the (k-distance, Energy Band 1, Energy Band 2, ...) data for the lines on the plot.

HIGH\_SYMMETRY\_POINTS: For marking where Γ, X, M, etc., are on the k-path.

KLABELS: To get the actual 'Γ', 'X', 'M' string labels.

KLINES.dat (optional but helpful): Helps to properly segment the x-axis of the band plot between high symmetry points.

Objective 1b: Visualize Density of States (DOS).

Data Needed: Total density of electronic states vs. Energy. (Partial DOS is a bonus).

Files to Use (from DOS folder for each 'x'):

tdos.dat: For Total DOS.

(Optional/Advanced): PDOS\_USER1.dat etc. for Partial DOS.

Objective 1c: Visualize Key Optical Properties.

Data Needed: Absorption coefficient vs. Photon Energy, Reflectivity vs. Photon Energy, Refractive Index vs. Photon Energy.

Files to Use (from OPTIC or Optic folder for each 'x'):

ABSORPTION.dat

REFLECTIVITY.dat

REFRACTIVE.dat

(And others like EXTINCTION.dat, ENERGY\_LOSSSPECTRUM.dat, real.in, IMAG.in if you want to be comprehensive).

Objective 1d: List Key Scalar Structural, Mechanical, and Thermodynamic Properties.

Data Needed (single values per material 'x'):

Accurate Bandgap (Direct & Indirect)

Lattice Parameters (a, b, c, angles) & Cell Volume

Binding Energy / Formation Energy / Total Energy (for stability assessment)

Bulk Modulus, Shear Modulus, Young's Modulus, Poisson's Ratio

Effective Mass (electron & hole)

Debye Temperature

Files to Use (from various folders for each 'x'):

HSE/BAND\_GAP: Crucial for the most accurate scalar bandgap.

OUTCAR (from SR, STRUC, or the main folder for a self-consistent run like STR): Your primary source for final lattice parameters, cell volume, and total energy (which you can use to discuss stability, or if binding energy is directly there). You'll need to write specific parsing logic for these lines in the OUTCAR.

Files in Elastic folder (e.g., specific .dat outputs or its OUTCAR): For elastic constants and derived moduli.

Files in eff\_M or Eff\_Mass folder (e.g., specific .dat or its OUTCAR): For effective masses.

STRUC/ENERGY\_VOLUME\_DFT.dat and STRUC/PRESSURE\_VOLUME\_DFT.dat can also provide structural energy and volume information.

Aim 2: Understand How Properties Change with Anion Ratio 'x'

Objective: Plot trends of the scalar properties (identified in 1d) versus the Chlorine concentration 'x' (from 0 to 6). Quantify these trends (e.g., with regression).

Data Needed: All the scalar properties extracted for each of the 7 materials from Aim 1d. These will form your material\_summary\_properties.csv or the Materials table in your SQLite database.

Files to Use: You've already parsed the necessary files in Aim 1d. Here, you're using that aggregated summary data.

Aim 3: Identify Key Design Trade-offs

Objective: Visualize relationships between multiple scalar properties to see how optimizing one might affect another.

Data Needed: Again, the aggregated scalar properties from your material\_summary\_properties.csv or Materials SQLite table.

Files to Use: No new file parsing. You are using the data consolidated in Aim 2. You'll use tools like Plotly to create 3D scatter plots from this summary data.

Aim 4: Estimate Properties for Untested Compositions (Predictive Interpolation)

Objective: Build simple regression models (surrogate models) based on the 7 data points to predict properties for fractional 'x' values.

Data Needed: Key scalar properties vs. 'x' from your material\_summary\_properties.csv or Materials table.

Files to Use: No new file parsing. You're training models on the data consolidated in Aim 2.

Aim 5: Make Data Accessible & Actionable via an Interactive Dashboard

Objective: Create a Streamlit/Dash application that presents all the above.

Data Needed: The fully populated SQLite database (containing both scalar summary data and detailed spectral/band data) and your trained surrogate models.

Files to Use: Your dashboard will read from the SQLite database and use your trained surrogate models.

In Essence, Your File Focus Strategy Should Be:

Core Scalar Properties (for trends, trade-offs, surrogate models, and dashboard summaries):

HSE/BAND\_GAP (Absolute must for bandgap).

Relevant OUTCAR files (from SR, STRUC, STR, Elastic, eff\_M) to get final lattice parameters, volume, total energy/binding energy, elastic moduli, effective masses. If the Elastic or eff\_M folders have simpler .dat summary files for these, prioritize those over parsing complex OUTCAR sections, but OUTCAR is often the ultimate source.

The ENERGY\_VOLUME\_DFT.dat in STRUC is also a good source for energy and volume.

Core Spectral/Band Data (for detailed plots in dashboard and EDA):

BAND/REFORMATTED\_BAND.dat (or HSE/...) + HIGH\_SYMMETRY\_POINTS + KLABELS (for Band Structure plots).

DOS/tdos.dat (for Total Density of States plots).

OPTIC/ABSORPTION.dat, OPTIC/REFLECTIVITY.dat, OPTIC/REFRACTIVE.dat, OPTIC/real.in, OPTIC/IMAG.in (for Optical Properties plots).

What to Initially De-prioritize (or skip if too complex for first pass):

Deeply parsing all details from OUTCAR files unless a critical scalar isn't found elsewhere.

Parsing all the individual PDOS\_USERN.dat files (focus on tdos.dat first).

Parsing every single auxiliary file in the BAND or OPTIC folders if its purpose isn't immediately clear for your main plots (e.g., KLINES.dat is helpful for bands but you can make a plot without it initially; ENERGY\_LOSSSPECTRUM.dat is interesting but absorption/reflectivity are more common).

The very large vasprun.xml – avoid unless absolutely necessary.

Input files like INCAR, POSCAR, POTCAR, KPOINTS, KPATH.in, VPKIT.in (unless, as in OPTIC, a .in file actually is the data output).

Your Workflow for Section 2 (Parsers) Should Be:

For each AIM above, ask: "What is the most direct file mentioned in the Dryad README or typically used in the field that will give me this piece of information?" Start with those.

For example:

"I need the accurate scalar bandgap for Aim 1d, 2, 3, 4." -> "Focus on parsing HSE/BAND\_GAP."

"I need to plot optical absorption for Aim 1c." -> "Focus on parsing OPTIC/ABSORPTION.dat."

Okay, let's dive into **SECTION 1: Project Setup & Foundational Data Ingestion** with detailed step-by-step instructions. We'll get your project environment and initial data access scripts up and running using VS Code.

# SECTION 1: Project Setup & Foundational Data Ingestion

**Overall Goal for Section 1:** Establish the project environment, acquire raw data, and develop initial scripts to understand the file structure and identify target data files. By the end, you'll have a script that unzips the data and creates an inventory of files, tagging them with material composition information.

**T1.1: Environment & Repository Setup**

* **Step 1: Create the Main Project Folder**
  + Open your system's terminal (or the integrated terminal in VS Code once you open a temporary folder).
  + Navigate to where you want to store your projects.
  + Create the project directory:
  + mkdir perovskite\_insights\_platform
  + cd perovskite\_insights\_platform

* **Step 2: Initialize Git Repository**
  + In the terminal, inside your perovskite\_insights\_platform directory:
  + git init

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

* + This creates a hidden .git folder, initializing version control for your project.
* **Step 3: Open Project in VS Code**
  + Launch VS Code.
  + Go to File > Open Folder... and select your perovskite\_insights\_platform directory.
* **Step 4: Create Python Virtual Environment**
  + Open the integrated terminal in VS Code (Terminal > New Terminal or `Ctrl+`` (backtick)).
  + Ensure you are in the perovskite\_insights\_platform directory.
  + Create the virtual environment (named venv):
  + python -m venv venv

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

*(If python points to Python 2 on your system, try python3 -m venv venv)*

* **Step 5: Activate the Virtual Environment**
  + **On macOS/Linux (bash/zsh):**
  + source venv/bin/activate

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

* + **On Windows (PowerShell):**
  + .\venv\Scripts\Activate.ps1

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Powershell

IGNORE\_WHEN\_COPYING\_END

*(If you get an execution policy error, you might need to run Set-ExecutionPolicy Unrestricted -Scope Process in PowerShell first for that session, then try activating again).*

* + **On Windows (Command Prompt - cmd.exe):**
  + venv\Scripts\activate.bat

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

* + Your terminal prompt should now show (venv) at the beginning, indicating the virtual environment is active. VS Code might also automatically detect and prompt you to select this new environment as the interpreter for your workspace. If it asks, say yes. If not, you can select it later (Ctrl+Shift+P -> "Python: Select Interpreter" -> find the one with (venv)).
* **Step 6: Install Initial Core Libraries**
  + With venv active in the VS Code terminal:
  + pip install pandas numpy matplotlib seaborn jupyterlab streamlit plotly scikit-learn openpyxl

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

*(Added openpyxl as Pandas sometimes needs it for Excel compatibility, though not strictly for this project unless you decide to output something to Excel).*

* **Step 7: Create requirements.txt**
  + With venv active in the VS Code terminal:
  + pip freeze > requirements.txt

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

* + This file lists all installed packages and their versions, crucial for reproducibility.
* **Step 8: Create .gitignore File**
  + In VS Code's File Explorer (left sidebar), right-click inside your project root and select New File.... Name it .gitignore.
  + Add the following lines to .gitignore to prevent tracking unnecessary files:
  + # Byte-compiled / optimized / DLL files
  + \_\_pycache\_\_/
  + \*.py[cod]
  + \*$py.class
  + # C extensions
  + \*.so
  + # Distribution / packaging
  + .Python
  + build/
  + develop-eggs/
  + dist/
  + downloads/
  + eggs/
  + .eggs/
  + lib/
  + lib64/
  + parts/
  + sdist/
  + var/
  + wheels/
  + pip-wheel-metadata/
  + share/python-wheels/
  + \*.egg-info/
  + .installed.cfg
  + \*.egg
  + MANIFEST
  + # PyInstaller
  + # Usually these files are written by a script, but they may be committed by
  + # accident.
  + \*.manifest
  + \*.spec
  + # Installer logs
  + pip-log.txt
  + pip-delete-this-directory.txt
  + # Unit test / coverage reports
  + htmlcov/
  + .tox/
  + .nox/
  + .coverage
  + .coverage.\*
  + .cache
  + nosetests.xml
  + coverage.xml
  + \*.cover
  + .hypothesis/
  + .pytest\_cache/
  + # Environments
  + .env
  + .venv
  + venv/
  + ENV/
  + env/
  + env.bak/
  + venv.bak/
  + # Spyder project settings
  + .spyderproject
  + .spyproject
  + # Rope project settings
  + .ropeproject
  + # mkdocs documentation
  + /site
  + # mypy
  + .mypy\_cache/
  + .dmypy.json
  + dmypy.json
  + # Jupyter Notebook
  + .ipynb\_checkpoints
  + notebooks/.ipynb\_checkpoints/
  + # VS Code
  + .vscode/
  + # SQLite databases
  + \*.sqlite
  + \*.sqlite3
  + \*.db
  + \*.db3
  + # Data files (you might want to commit small \*processed\* sample CSVs for testing later, but ignore raw/large ones)
  + data/raw/\*
  + data/interim/unzipped\_data/
  + # data/interim/parsed\_property\_collections/\* # Decide later if you want to commit small processed files
  + # data/processed/\*.csv # Decide later for summary CSV
  + # data/processed/\*.sqlite # Usually not committed, generated by script
  + # ML Models (can be large)
  + \*.pkl
  + \*.joblib
  + # OS-specific files
  + .DS\_Store
  + Thumbs.db

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Gitignore

IGNORE\_WHEN\_COPYING\_END

**T1.2: Data Acquisition**

* **Step 1: Download the Dataset**
  + Go to the Dryad dataset page: [https://doi.org/10.5061/dryad.8gtht770d](https://www.google.com/url?sa=E&q=https%3A%2F%2Fdoi.org%2F10.5061%2Fdryad.8gtht770d)
  + Download the HBD\_Mixed\_anions.zip file.
* **Step 2: Create Directory Structure in VS Code**
  + In VS Code's File Explorer:
    - Right-click on the project root, create New Folder... named data.
    - Inside data, create New Folder... named raw.
    - Inside data, create New Folder... named interim.
    - Inside data, create New Folder... named processed.
    - (You'll also create ingestion\_scripts/, notebooks/, src/, app/ later as per the overall structure).
* **Step 3: Place Downloaded File**
  + Move the downloaded HBD\_Mixed\_anions.zip file into the perovskite\_insights\_platform/data/raw/ directory.

**T1.3: Develop Initial Ingestion Script (ingestion\_scripts/01\_unzip\_and\_inventory.py)**

* **Step 1: Create the Script File**
  + In VS Code's File Explorer, create a folder named ingestion\_scripts in your project root.
  + Inside ingestion\_scripts, create a new file named 01\_unzip\_and\_inventory.py.
* **Step 2: Write the Python Code**
* import zipfile
* import os
* import pathlib
* import pandas as pd # We'll use pandas to save the inventory CSV easily
* import re # For more complex pattern matching if needed
* # --- Configuration ---
* RAW\_DATA\_DIR = pathlib.Path(\_\_file\_\_).resolve().parent.parent / "data" / "raw"
* ZIP\_FILE\_NAME = "HBD\_Mixed\_anions.zip"
* UNZIPPED\_BASE\_DIR\_NAME = "unzipped\_HBD\_Mixed\_anions" # Name of the folder inside data/interim
* INTERIM\_DATA\_DIR = pathlib.Path(\_\_file\_\_).resolve().parent.parent / "data" / "interim"
* INVENTORY\_FILE\_NAME = "file\_inventory.csv"
* def unzip\_data(zip\_path, extract\_to\_path):
* """Unzips the dataset."""
* print(f"Unzipping {zip\_path} to {extract\_to\_path}...")
* if extract\_to\_path.exists():
* print(f"Output directory {extract\_to\_path} already exists. Skipping unzipping.")
* print("Delete it if you want to re-unzip.")
* return
* extract\_to\_path.mkdir(parents=True, exist\_ok=True)
* with zipfile.ZipFile(zip\_path, 'r') as zip\_ref:
* zip\_ref.extractall(extract\_to\_path)
* print("Unzipping complete.")
* def identify\_material\_from\_path(relative\_path\_parts):
* """
* Identifies material composition (x\_value, formula) based on folder path parts.
* This needs to be adapted based on the EXACT folder structure from the Dryad README.
* """
* x\_value = None
* formula = None
* # Example logic - you'll need to refine this based on actual unzipped structure
* # relative\_path\_parts is a list like ['HBD\_Mixed\_anions', 'Cs2NaTlCl6', 'OPTIC']
* if not relative\_path\_parts or len(relative\_path\_parts) < 2:
* return None, None, "Unknown"
* # The root folder from the zip is usually the first part, let's assume its HBD\_Mixed\_anions
* root\_folder\_in\_zip = relative\_path\_parts[0] # e.g., 'HBD\_Mixed\_anions'
* if len(relative\_path\_parts) >= 2:
* main\_material\_folder = relative\_path\_parts[1]
* if main\_material\_folder == "Cs2NaTlCl6": # Pure Chlorine
* x\_value = 6
* formula = "Cs2NaTlCl6"
* elif main\_material\_folder == "Cs2NaTlBr6": # Pure Bromine
* x\_value = 0
* formula = "Cs2NaTlBr6"
* elif main\_material\_folder == "ClxBr6-xData33":
* if len(relative\_path\_parts) >= 3:
* sub\_folder = relative\_path\_parts[2]
* if sub\_folder == "Cl2Br4":
* x\_value = 2
* formula = "Cs2NaTlCl2Br4"
* elif sub\_folder == "Cl4Br2":
* x\_value = 4
* formula = "Cs2NaTlCl4Br2"
* elif main\_material\_folder == "ClxBr1-xpart4": # Dryad README seems to have a typo here "ClxBr1-xpart4"
* if len(relative\_path\_parts) >= 3:
* sub\_folder = relative\_path\_parts[2]
* if sub\_folder == "ClBr5":
* x\_value = 1
* formula = "Cs2NaTlClBr5"
* elif sub\_folder == "Cl3Br3":
* x\_value = 3
* formula = "Cs2NaTlCl3Br3"
* elif sub\_folder == "Cl5Br":
* x\_value = 5
* formula = "Cs2NaTlCl5Br1" # or Cs2NaTlCl5Br
* # Guess property type from the deepest folder if possible
* potential\_property\_type = "Unknown"
* if len(relative\_path\_parts) >= 3 : # if main\_material\_folder is Cs2NaTlCl6
* potential\_property\_type = relative\_path\_parts[-2] if len(relative\_path\_parts) > 1 else "General" # Last folder is property
* if len(relative\_path\_parts) >= 4 : # if main\_material\_folder is ClxBr6-xData33 etc
* potential\_property\_type = relative\_path\_parts[-2] if len(relative\_path\_parts) > 1 else "General"
* # if material folder itself implies a type
* if main\_material\_folder in ["SR", "STR", "STRUC", "Band", "HSE", "DOS", "eff\_M", "Elastic", "OPTIC"]:
* potential\_property\_type = main\_material\_folder
* # This implies a different folder structure than initially assumed by material\_folder logic above
* # Need to re-think how to get x\_value in this case if property is at level 1 after root\_folder\_in\_zip
* # For now, let's assume the structure: HBD\_Mixed\_anions/MATERIAL\_FOLDER/PROPERTY\_FOLDER/file.dat
* return x\_value, formula, potential\_property\_type
* def create\_file\_inventory(unzipped\_dir\_path, inventory\_output\_path):
* """Scans the unzipped directory and creates an inventory of .dat and OUTCAR files."""
* print(f"Creating file inventory for {unzipped\_dir\_path}...")
* file\_list = []
* # The unzipped data usually has a single root folder inside, let's find it
* # Assuming the zip extracts to something like: .../unzipped\_HBD\_Mixed\_anions/HBD\_Mixed\_anions/Cs2NaTlCl6/...
* # We need to start walking from the \*actual\* root of the content, not the `unzipped\_dir\_path` itself.
* # Find the true content root (likely the first and only item in unzipped\_dir\_path)
* content\_roots = [d for d in unzipped\_dir\_path.iterdir() if d.is\_dir()]
* if not content\_roots:
* print(f"Error: No subdirectories found in {unzipped\_dir\_path}. Check unzipping.")
* return
* actual\_content\_root = content\_roots[0] # Assuming first dir is the main content
* print(f"Scanning content root: {actual\_content\_root}")
* for file\_path\_obj in actual\_content\_root.rglob('\*'): # rglob finds files in subdirectories
* if file\_path\_obj.is\_file():
* file\_name = file\_path\_obj.name
* if file\_name.endswith(".dat") or file\_name == "OUTCAR" or file\_name == "BAND\_GAP" \
* or file\_name.endswith(".in") or file\_name in ["KLABELS", "HIGH\_SYMMETRY\_POINTS"]: # Include .in for optic files too
* # Get relative path parts to identify material and property
* # Relative to the `actual\_content\_root`
* relative\_path = file\_path\_obj.relative\_to(actual\_content\_root)
* relative\_path\_parts = list(relative\_path.parts)
* # remove filename from parts for identifying folders
* folder\_path\_parts = relative\_path\_parts[:-1]
* x\_value, formula, potential\_property\_type = identify\_material\_from\_path(folder\_path\_parts)
* file\_list.append({
* "file\_path\_absolute": str(file\_path\_obj.resolve()), # Store absolute for direct access
* "file\_path\_relative\_to\_content\_root": str(relative\_path),
* "material\_x\_value": x\_value,
* "material\_formula": formula,
* "potential\_property\_type": potential\_property\_type, # Based on parent folder of file
* "file\_name": file\_name
* })
* if not file\_list:
* print("No relevant files (.dat, OUTCAR, BAND\_GAP etc.) found. Check rglob pattern or directory structure.")
* return
* inventory\_df = pd.DataFrame(file\_list)
* inventory\_df.to\_csv(inventory\_output\_path, index=False)
* print(f"File inventory saved to {inventory\_output\_path}")
* print(f"Found {len(inventory\_df)} relevant files.")
* # print("Sample of inventory:")
* # print(inventory\_df.head())
* def main():
* zip\_full\_path = RAW\_DATA\_DIR / ZIP\_FILE\_NAME
* unzipped\_full\_dir\_path = INTERIM\_DATA\_DIR / UNZIPPED\_BASE\_DIR\_NAME
* inventory\_full\_path = INTERIM\_DATA\_DIR / INVENTORY\_FILE\_NAME
* if not zip\_full\_path.exists():
* print(f"ERROR: Zip file not found at {zip\_full\_path}")
* return
* unzip\_data(zip\_full\_path, unzipped\_full\_dir\_path)
* if unzipped\_full\_dir\_path.exists(): # Only create inventory if unzipping happened or dir exists
* create\_file\_inventory(unzipped\_full\_dir\_path, inventory\_full\_path)
* else:
* print(f"ERROR: Unzipped directory not found at {unzipped\_full\_dir\_path}. Cannot create inventory.")
* if \_\_name\_\_ == "\_\_main\_\_":
* main()

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Python

IGNORE\_WHEN\_COPYING\_END

* **Step 3: Run the Script from VS Code's Terminal**
  + Ensure your venv is active.
  + Navigate to your project root if you aren't already there.
  + Run: python ingestion\_scripts/01\_unzip\_and\_inventory.py
* **Step 4: Inspect the Output**
  + Check the data/interim/ folder. You should see:
    - unzipped\_HBD\_Mixed\_anions/ directory containing the unzipped data.
    - file\_inventory.csv.
  + Open file\_inventory.csv (VS Code handles CSVs well, or use Excel/Numbers/etc.).
  + **Crucially, verify:**
    - Does it list many .dat and OUTCAR files?
    - Are the material\_x\_value and material\_formula columns being populated correctly based on the folder names? You might need to **adjust the identify\_material\_from\_path function logic heavily** once you see the actual unzipped structure. This function is the most critical part of this script and will require careful inspection of how the folders are named.
    - Is potential\_property\_type making sense?
* **Step 5: Refine identify\_material\_from\_path (Iterative Process)**
  + Based on the Dryad README description:
    - "In HBD\_Mixed\_anions.zip, the different data types folder is for two pure double perovskites, such as Cs₂NaTlCl₆ & Cs₂NaTlBr₆." -> These are top-level material folders.
    - "Further, the Cl2Br4, Cl4Br2 (x = 2, 4) in ClxBr6-xData33 and ClBr5, Cl3Br3, Cl5Br (x= 1, 3, 5) in ClxBr1-xpart4 files are arranged..." -> These are material sub-folders *inside* generic folders.
  + You will need to examine the output of your first run of 01\_unzip\_and\_inventory.py and the actual unzipped folder structure to perfect the logic in identify\_material\_from\_path.
  + **Example debugging for identify\_material\_from\_path:**
    - When you run 01\_unzip\_and\_inventory.py, print out relative\_path\_parts inside the create\_file\_inventory loop before calling identify\_material\_from\_path. This will show you what path segments the identification function is working with for each file.
    - Then, adjust the if/elif conditions in identify\_material\_from\_path to correctly match these parts and assign the x\_value and formula.
    - It might be like HBD\_Mixed\_anions/Cs2NaTlCl6/OPTIC/ABSORPTION.dat or HBD\_Mixed\_anions/ClxBr6-xData33/Cl2Br4/OPTIC/ABSORPTION.dat.
    - The function provided in the code block above tries to anticipate this structure. **You will almost certainly need to debug and refine it based on what you see after the first unzipping.**
* **Step 6: Commit to Git**
  + Once 01\_unzip\_and\_inventory.py is working reasonably well and producing a good inventory:
    - In VS Code's terminal (or using its Git GUI):
    - git add .gitignore requirements.txt ingestion\_scripts/01\_unzip\_and\_inventory.py data/raw/HBD\_Mixed\_anions.zip # (If zip is small enough for your repo policy, else add data/raw/\* to .gitignore too)
    - git commit -m "Section 1: Initial project setup, unzip, and file inventory script"

IGNORE\_WHEN\_COPYING\_START

content\_copy download

Use code [with caution](https://support.google.com/legal/answer/13505487). Bash

IGNORE\_WHEN\_COPYING\_END

* + - You might choose *not* to commit the data/interim/ generated files directly, as they can be reproduced by your script. Add data/interim/\* to .gitignore if you prefer this. However, for now, having file\_inventory.csv might be useful to inspect.

**Deliverable for Section 1 Complete!**

You now have:

1. A fully set up project environment in VS Code.
2. The raw data acquired.
3. A Python script (01\_unzip\_and\_inventory.py) that:
   * Unzips the raw data.
   * Scans the unzipped structure.
   * Generates a CSV inventory (file\_inventory.csv) of relevant data files, importantly tagging each with its likely material\_x\_value, material\_formula, and potential\_property\_type.
4. An initial understanding of the folder structure and how material compositions are identified from file paths.

This inventory CSV is the direct input for your next set of scripts in Section 2, where you'll start parsing the actual content of these files. The quality of your identify\_material\_from\_path function is key here!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Phase A: Solidify Data Engineering & Preparation (Complete Sections 2 & 3 Properly)**

* **1. Finalize All Core Parsers (Your current focus - High Priority):**
  + **What to Do:** Ensure *all* the parsers for the key data files you've identified (BAND files, OPTIC files, DOS tdos.dat, Elastic\_Info, scalar BAND\_GAP, FERMI\_ENERGY, and if possible, effective mass and basic structural parameters from OUTCAR or simple files) are robust and working correctly.
  + **Impressive Skill:** Handling diverse, "ugly" real-world data formats. This is a huge plus over just using clean datasets.
* **2. Perfect 02\_extract\_and\_structure\_data.py:**
  + **What to Do:** Ensure this script smoothly calls all your parsers, correctly associates data with each material composition (x\_value, formula), handles potential missing files for a given material gracefully, and correctly generates:
    - material\_summary\_properties.csv (with ALL your target scalar properties as columns).
    - The set of consolidated spectral/band CSVs (e.g., all\_tdos\_data.csv, all\_absorption\_data.csv, etc., with material\_x\_value and material\_formula columns).
  + **Impressive Skill:** Data pipeline orchestration (even if simple), data aggregation, error handling.
* **3. Implement the SQLite Database (03\_build\_database.py) - Python Created:**
  + **What to Do:**
    - Define your SCHEMA\_SQL string with CREATE TABLE statements for Materials (from summary CSV) and all your spectral/band data types (from their respective consolidated CSVs). Define primary keys, foreign keys (conceptually linking to Materials via material\_x\_value), and appropriate data types (REAL, INTEGER, TEXT).
    - Write the script to:
      1. Connect to (and create if not exists) perovskite\_platform.sqlite.
      2. DROP TABLE IF EXISTS ...; CREATE TABLE ...; for all tables to ensure a clean build.
      3. Load your material\_summary\_properties.csv into the Materials table.
      4. Load each of your all\_...\_data.csv files into their respective tables.
  + **Impressive Skill:** Database design (relational thinking), SQL DDL execution via Python, ETL process (loading transformed data into a structured store), reproducibility, Infrastructure-as-Code for your data store. This is a BIG step up.

**Phase B: Insightful Analysis & Predictive Modeling (Leverage your structured data - DA/DS skills)**

* **4. In-depth Exploratory Data Analysis (EDA) with SQL & Python (Section 4 - High Priority):**
  + **What to Do (in a Jupyter Notebook):**
    - Connect to your SQLite database.
    - Use SQL (via pd.read\_sql\_query) to fetch data for analysis. This showcases SQL skills!
    - Replicate and *expand upon* the findings from the paper's abstract using your structured data.
    - **Visualize trends:** Plot all key scalar properties from your Materials table against material\_x\_value. Use clear, publication-quality plots (matplotlib, seaborn).
    - **Visualize spectra:** For selected materials (e.g., x=0, an intermediate x, x=6), plot their band structures (with k-labels), DOS, and key optical spectra (absorption, reflectivity). Overlay spectra for different 'x' values to show evolution.
    - **Correlation analysis:** Create a heatmap of correlations between different scalar properties. Discuss interesting relationships (e.g., "does a higher bulk modulus correlate with a specific bandgap trend?").
  + **Impressive Skill:** SQL for data retrieval, scientific data visualization, statistical thinking, ability to connect data to scientific context, critical analysis, communication of findings.
* **5. Surrogate Modeling for Interpolation (Section 5 - High Impact):**
  + **What to Do (Jupyter Notebook, functions in src/surrogate\_models.py):**
    - For 1-2 key scalar properties (e.g., direct\_gap\_ev from your BAND/BAND\_GAP files, maybe Bulk Modulus), train surrogate models (Polynomial Regression definitely, Gaussian Process Regression if you want to really impress) using your 7 material\_x\_value data points.
    - Your input feature is material\_x\_value. Your target is the property.
    - Create functions to make predictions for fractional x\_values.
  + **Impressive Skill:** Applying machine learning (even simple regression) to a scientific problem, understanding of interpolation/surrogate modeling, ability to generate new "data" (predictions) from existing information. GPR shows advanced understanding.

**Phase C: Data Product & Presentation (Showcase your work - DA/Product/Communication Skills)**

* **6. Interactive Dashboard (Section 6 - Highest "Wow" Factor):**
  + **What to Do (Streamlit/Dash - app/main\_dashboard.py):**
    - Build an interactive web application that allows users to:
      * Select a material composition (x\_value 0-6) and see all its parsed scalar properties and its detailed spectral plots (bands, DOS, optics) fetched *from your SQLite database*.
      * View your key trend plots (e.g., Bandgap vs x).
      * **Integrate Surrogate Models:** Have an input field where users can type a fractional x\_value (e.g., 2.5) and see the *predicted* properties from your surrogate models. If using GPR, show the uncertainty.
      * **(Bonus Impressive Feature):** Include your 3D interactive "Trade-off Plot" (e.g., Plotly scatter3d: Bandgap vs. Mobility Proxy vs. Stability Proxy) where points are colored by 'x'.
  + **Impressive Skill:** Ability to create data products, UI/UX thinking, web app development (basic), full-stack data science capability (from data to interactive app), excellent communication of complex data.
* **7. Comprehensive README.md & GitHub Repo (Section 7 - Essential):**
  + **What to Do:** As detailed before: explain the science, your objectives, your data pipeline (with a simple diagram), your key analytical findings (with embedded plots or links to notebook figures), the surrogate modeling, the dashboard features (with screenshots/GIFs), and clear instructions on how to run everything.
  + Ensure clean, well-commented code throughout your src/, ingestion\_scripts/, and app/ directories.
  + **Impressive Skill:** Documentation, project organization, reproducibility, professionalism.

**What Makes it "More Impressive" from a Company Perspective:**

* **End-to-End Solution:** You're not just doing one piece. You're taking raw data, engineering it, storing it, analyzing it, modeling it, and productizing it (dashboard). This is what companies *do*.
* **Problem Solving with Real, Messy Data:** Not a clean Kaggle dataset.
* **Automation & Reproducibility:** Scripted database creation, clear data pipeline.
* **Quantitative Insights & Predictive Capability:** You're not just describing; you're quantifying trends and making predictions (even if interpolative).
* **Effective Communication:** A clear README.md and an interactive dashboard show you can explain complex work.
* **Technical Breadth:** Python, Pandas, SQL, basic ML, data viz, (basic) web app dev.
* **Initiative & Domain Learning:** Tackling a complex scientific dataset shows you're willing to learn and apply skills to challenging domains.

You are on a great path! Completing Section 2 fully (all parsers for files you identified as key for your project aims) and then moving methodically through Section 3 (DB), 4 (EDA/SQL), 5 (Modeling), and 6 (Dashboard) will result in a portfolio piece that is definitely "maximally impressive." Don't feel you have to do *every single optional thing*, but getting the core elements of each phase done well is key.