# 📘 Data Warehouse (DWH) Program – Technical Specification

## 1. Basic Idea

* The program designs a **data warehouse** either:
  + Directly in table form, or
  + In the future, by diagramming.
* **Design process**:
  + Create metadata for:
    - Stages
    - Artifacts/Nodes (files, tables, views, etc.)
    - Columns per artifact
  + From this metadata, generate **text blocks** called *lists*.
* **Lists**:
  + Can be one/multiple columns, or even a single word.
  + Used to substitute *list blocks* inside templates.
* **Templates**:
  + Text files (Databricks notebooks, Synapse notebooks, SQL scripts).
  + Contain *list blocks* enclosed in {}.
  + These blocks are substituted with generated *lists*.
  + Example:
    - Generated list → id integer, name string, …
    - Named list\_comma\_column\_datatype
    - Substitutes {list\_comma\_column\_datatype} in a template.
* **Artifacts**:
  + Produced as ready-to-run text files (DDL, ETL scripts, notebooks).
  + Saved in a dedicated folder.

## 2. Architecture

**2.1 Backend**

* Main focus.
* Language: **Python**.
* Organized in **modular code** (not monolithic).
* File naming convention: 1\_, 2\_, … A\_, B\_ … (for logical flow).
* Database: **Excel file** (future upgrade → DB).
  + Stores 3 main tables:
    - Stages
    - Artifacts
    - Columns

**2.2 Frontend**

* Very simple, independent from backend.
* Possible options:
  + Console UI
  + Simple form with tabs and buttons
  + (Future) Web interface
* If form-based, tabs should be:
  + **Tab A: Intro Project**
    - New project
    - Open existing project
  + **Tab B: Workbench**
    - Stages button (edit stages sheet)
    - Artifacts button (edit artifacts sheet)
    - Columns button (edit columns sheet):
      * Import/assign (load CSVs from 1\_source, match to artifacts, update columns sheet)
      * AI comments (generate descriptions)
      * Cascade (deterministic column filling based on relations)
    - Sync & check
    - Save
  + **Tab C: Create Artifacts**
  + **Tab D: Documentation**
    - Source GitHub artifacts
    - Generate customer documentation

**2.3 Tab Dependencies**

* Tab B is inactive until a project is selected (new or existing).
* Tab C requires at least one save in Tab B.

## 3. Usability & Functionality

### 3.1 Tab A – Intro Project

**New Project**

1. Ask project name.
2. Open save dialog → default folder \_DWH\_Projects.
3. Create project subdirectory: Project\_<name>
   * Subfolders:
     + 1\_sources (empty)
     + 2\_workbench
       - Excel file: workbench\_<name>
         * **Sheet: Stages**

Header: Stage ID, Stage Name, Stage Color, Stage Technical Columns, Stage DDL Default Templates, Stage ETL Default Templates

Rules:

Stage ID → s1, s2, s3 … (pattern fixed).

Default stage names → 0\_drop\_zone, 1\_bronze, 2\_silver, 3\_gold, 4\_mart, 5\_PBI\_Model, 6\_PBI\_Reports (editable).

Default stage colors (bronze → bronze, silver → silver, etc.).

Technical columns auto-added (e.g. source\_name, partition\_field).

DDL/ETL default templates (semicolon-separated).

* + - * + **Sheet: Artifacts**

Header includes:

Stage ID, Stage Name, Artifact ID, Artifact Name, Upstream Artifacts, Downstream Artifacts, Artifact Comment, Readable Column Name, Artifact Topology, Upstream Relations, Upstream Relation Types, Artifact Relation Direction, Artifact Domain, DDL Template, ETL Template, DDL Production File, ETL Production File

Rules:

Artifact ID → a1, a2, a3 … (pattern fixed).

Multiple upstream/downstream artifacts → separated with ;.

Relations:

Source → solid line

Lookup → dotted line

PBI lookup → diagram only

Relation direction (for Power BI): 1-m, m-m, single forward, both directions.

* + - * + **Sheet: Columns**

Header: Artifact ID, Column ID, Column Name, Order, Data Type, Column Comment, Column Group, Simple Calculation

Rules:

Column groups: PKs, SKs, BKs, attributes, facts, technical fields.

Simple calculation allowed (e.g. Customer\_BK = id + '-' + name).

* + - 3\_templates (store DDL/ETL template text files).
    - 4\_artifacts (generated notebooks).
      * Subfolders by stage:
        + 1\_bronze → DDLs, ETLs
        + 2\_silver → DDLs, ETLs
        + 3\_gold → DDLs, ETLs
        + 4\_mart → DDLs
        + PBI
    - 9\_archive (previous versions of workbench).

1. Open Excel in **Stages** sheet for editing.
2. Setup GitHub project (email: bezas.a@dwh-bi.com).

**Open Existing Project**

* Open Excel in **Artifacts** sheet.
* (Future) Open canvas diagram for artifacts & relations.

### 3.2 Tab B – Workbench

**Stages Sheet**

* Open sheet.
* User fills stages manually.
* (Future) Stages filled via diagram front end.

**Artifacts Sheet**

* Open sheet.
* User fills artifacts & relations manually.
* (Future) Auto-fill from diagram front end.

**Columns Sheet**

* **Import/Assign** button:
  + Read files in 1\_source.
  + Match by name to artifact.
  + Extract column name, type, order.
  + Fill columns sheet.
* **AI Comments** button:
  + Generate artifact comments (if missing).
  + Generate column comments (if missing).
  + Generate human-readable column names (for business artifacts).
* **Cascade** button: (deterministic fill)
  + Logic:
    - If relation = source → copy all upstream columns.
    - If relation = lookup → copy first 3 upstream columns.
    - If relation = PBI lookup → no copy (diagram only).
    - Grouping & ordering respected.
    - Add technical columns from stage.
    - If no upstream → insert dummy row.

**Sync & Test**

* Sync all sheets.
* Run validations:
  + Every artifact in artifacts sheet has matching rows in columns sheet.
  + Unique IDs: stage ID, artifact ID, column ID.
  + Correct ordering of columns.
  + Report errors → allow manual fix.

**Save**

* Save workbench version.

### 3.3 Tab C – Create Artifacts

1. Create artifacts:
   * Match artifact templates with files in 3\_templates.
   * Replace {list blocks} with generated lists.
   * List generator produces only required lists.
   * Substitute into templates.
   * Save into 4\_artifacts.
2. (Future) AI check for syntax errors.
3. Push artifacts to GitHub:
   * No overwrite allowed.
   * Must delete manually before re-push.
   * Show warning.

### 3.4 Tab D – Documentation

* Retrieve real artifacts from GitHub → update production file columns in artifacts sheet.
* Generate markup documentation for customers.

## 4. General Rules

* **Workbench (Tab B)**:
  + Stage 1 → all columns = string.
  + Stages 0,1,2 = source-side naming.
  + Stages 3,4,5 = business-side naming.
* **Artifact Side**:
  + Source side → cryptic names (drop\_zone, bronze, silver).
  + Business side → human-readable names (gold, mart, PBI).
* **Column Groupings (default)**:
  + Stage 0 → none.
  + Stage 1 → source fields, technical columns.
  + Stage 2 → primary keys.
* **List Generator Function**:
  + Input: sequence of metadata (col name, type, order) + separators (e.g. ,, src., <=> dest.).
  + Output: aligned text lists for template substitution.
  + Example:
    - Input: (",", "src.", column, "= dest.", column)
    - Output:
    - , src.id = dest.id
    - , src.name = dest.name
    - , src.age = dest.age

**5. Coding Standards**

* Write **clear comments** throughout code.
* Add **tooltips** as inline help in UI forms.

Would you like me to now **turn this specification into a Python project skeleton** (folders + modules with docstrings & placeholder functions) so you can start coding on it?