Homework 1

Due Sep 26

Extending the Hogwarts Artifacts Management System

# Background

You are continuing development of the Hogwarts Artifacts Management System. The starter code already supports basic CRUD operations for wizards and artifacts. In this assignment, you will extend the system with owner display, unassign, search, transfer history, and artifact condition management + JSON persistence—all while building on top of the provided codebase.

# Constraints

Do not rename or remove existing public classes, methods, packages.

# Features & Requirements

## 1) Show Artifact Owners (15 pts)

* Add an Owner column to the Artifacts table.
* Display the owning wizard’s name or — if unowned.
* Must update automatically after assignment/unassignment.
* Sorting by Owner should work (unowned last).

## 2) Unassign Artifacts (15 pts)

* Add an Unassign button (or row context menu).
* Enabled only if the selected artifact has an owner.
* On click: confirmation → unassign in service → refresh table.
* Unassigning an already-unowned artifact should be a no-op without errors.

## 3) Artifact Search (15 pts)

* Add a search bar above the Artifacts table to filter by artifact name as the user types.

## 4) Assignment History (15 pts)

* Record a transfer event on every assign or unassign: (artifactName, fromWizard, toWizard, timestamp)
* Provide a History view (dialog or tab) listing transfer events for the selected artifact.
* History must update immediately after changes.

## 5) Repair & Condition Management (15 pts)

* Add a Repair action for artifacts:
  + Users can increase an artifact’s condition by an entered amount.
  + Condition is clamped to 0–100.
* Assignment rule: if condition < 10, assignment is blocked with a clear message.
* Wear rule: each assignment reduces condition by 5 (minimum floor 0).
* Show condition in the UI and ensure tables refresh after changes.

## 6) JSON Persistence (15 pts)

* On exit: save all data into JSON files under data/ folder:
  + wizards.json
  + artifacts.json
  + transfers.json
* On startup:
  + If files exist, load them.
  + If not, seed from the provided DataStore.

# Deliverables

* Updated source code in the starter project (builds and runs).
* JSON files generated by your program (wizards.json, artifacts.json, transfers.json).