Spring 2026 CSE226.1 - Vibe Coding

Project 1 : The Dept. Admin "Audit Core"

Dr. Nabeel Mohammed
Demonstration due: 24th February 2026, in class.

1. Overview: Building for the Future

In the near future, Department Administration at NSU will be assisted by **Agentic AI**, autonomous assistants capable of handling student queries and graduation checks. However, the agentic systems still need tools to execute tasks.

Your goal in this project is to build the **Audit Engine** that powers this transition. While this version of the project **does not involve AI Integration**, you are developing the "Skill" or "Service" that a future AI Admin Agent would call to verify a student's graduation eligibility. Your product must be precise, handle messy data, and provide clear reports that a future agent could relay to a student or faculty member.

At this stage, your solution should be a tool with a Command Line Interface (CLI) that can be invoked from the terminal.

Deadline: This version of the product is due on **February 24th 2026, in class.**

Reassurance: We will be discussing the logic, data handling, and implementation strategies in class. This is a "vibe coding" friendly project—focus on the logic and the quality of the final service!

2. Resources & Flexibility

You are provided with two baseline files to build your solution:

- transcript.csv: A raw data export of a student's history.
- program_knowledge.md: The "Knowledge Base" markdown file structure containing rules and mandatory courses. The actual information in the files provided by me are there for proxy. They do not currently reflect NSU's actual requirements. You will need to "source" the information and create the markdown files with the proper information.

Note about content structure of these files: You have full flexibility to **modify the structure** of these files to incorporate the correct information and any semantic meaning you may wish to convey.

3. Implementation Levels & Marks Breakdown

You will develop the project in three different levels. For each level, you are responsible for providing the **actual CSV/Data files**, as well as the finished solutions that will demonstrate that it works. As this is a familiar domain for you all, you are expected to identify the edge cases that would normally trip up a human admin.

Level 1: The Credit Tally Engine (10 Marks)

- **The Task:** Read the student transcript and calculate total valid credits.
- **The Challenge:** A department admin only cares about *earned* credits. You need to decide which grades count and which do not. For example, how should your product treat an "F" versus a "W" or a "0-credit" lab?
- **Required:** Provide a test_L1.csv that proves your solution correctly identifies which credits are valid for graduation.

Level 2: The Logic Gate & Waiver Handler (10 Marks)

- **The Task:** Calculate weighted CGPA and handle program-specific waivers.
- **The Challenge:** Map letter grades to the NSU scale and handle "state" changes. The interface must ask the Admin: *"Waivers granted for ENG102 or BUS112?"* and adjust the requirements accordingly.
- **Required:** Provide a test_L2.csv that tests your math. You must ensure that non-grade entries (like waivers or withdrawals) do not break your CGPA calculation logic.

Level 3: The Audit & Deficiency Reporter (10 Marks)

- **The Task:** Compare the student's history against the program.md rules to find missing requirements.
- **The Challenge:** An Admin Agent must be able to tell a student *exactly* why they can't graduate. Your service should identify missing mandatory courses and flag "Probation" status (CGPA < 2.0).
- **Required:** Provide a **"Retake Scenario"** data file. If a student attempts a course multiple times, how does your product ensure the admin gets an accurate picture of their progress?

4. Deliverables

1. **The tool:** Functional CLI scripts, one per implementation level, that an Admin could run to process a student.
2. **Your Knowledge Files:** Your optimized versions of the .csv and your program knowledge file.
3. **Your Custom Test Cases:** The specific data files (test_L1.csv, test_L2.csv, etc.) that you designed to prove the product is robust enough for a future Agentic AI.

5. Usage

I envision running your CLI tool from the command line:

./your_cli_tool transcript.csv program_name program_knowledge.md