

LLM-Based C++ Grading Agent

Project Overview

In this project, you will build a **proof-of-concept automated grading system** for C++ programming exercises using **Large Language Models (LLMs)**.

Your system will take as input:

- A **C++ problem description**
- A **teacher reference solution**
- A **grading rubric** (structured criteria + points)
- A **student C++ code submission**

Your agent should analyze the student's code, compare it with the reference solution, apply the rubric, and generate:

- A **final numerical grade**
- A **score breakdown per rubric criterion**
- A **short explanation for the grade**

You will also build a **simple web interface** that allows users to input the problem, reference solution, rubric, and student code, and view the generated grade.

This is a **one-month, hands-on project** focusing on LLM reasoning, prompt engineering, and building a small benchmark to evaluate the system

Project Objectives

A. Build an LLM-powered grading agent capable of:

- Reading and interpreting a problem statement
- Comparing student code with a teacher reference solution
- Applying a rubric to compute a score
- Producing structured outputs (JSON)

B. Experiment with multiple prompting and agentic techniques:

You are required to try **several approaches and choose the best one**, such as:

- Chain-of-Thought prompting
- Few-shot Chain-of-Thought Prompting
- Agentic workflows such as Parallelization (Voting) and Evaluator-optimizer
- **Optional:** Tool-use steps for checking syntax errors and running test cases

Your goal is to **compare techniques and use which yields the most accurate grading.**

C. Build a benchmark dataset (required):

You must create a **small evaluation dataset** containing:

- **5 C++ questions (Very Easy, Easy, Medium, Hard, Very Hard)**
- For each question:
 - The teacher reference solution
 - A clear grading rubric
 - Several sample student answers (correct, partially correct, incorrect)
 - A human-graded score for each sample

You will use this dataset to **test and measure your agent's accuracy.**

D. Build a simple web application:

Your web app should:

1. Provide input fields for the problem, reference solution, rubric, and student code
2. Call your LLM-based grading agent on the backend
3. Display the grade, breakdown, and explanation

Deliverables

1. A presentation detailing the development process, including:
 - a. Prompt engineering and agentic workflow strategies
 - b. Challenges encountered and solutions implemented
 - c. Benchmark evaluation results
 - d. Interpretation of the result
2. A live demonstration of the web app grading at least one example problem