**Enhancing Code Generation through**

**TDD and GenAI**

**Methodology**

I used three different prompting strategies to evaluate how TDD principles can guide LLM code generation.

**Dataset and Tasks**

In this project, I used the HumanEval benchmark from OpenAI (Chen et al., 2021). This dsimataset is often used to test if LLMs can generate correct Python code. It includes 164 programming problems. Each problem gives a function signature and a short text description of what the function should do. The tasks include text understanding, writing basic algorithms, and solving small math problems. The difficulty is low to medium. It’s similar to an introductory programming course or simple coding interview.

Each HumanEval problem also provides a reference solution and human-written unit tests. In this project, I included both the problem description and the test cases in the prompt given to the model. The model then generated code based on this input.

All HumanEval problems are in Python, which fits the model I used. The focus of this project is on the process of code generation with different prompting methods(baseline vs. TDD-guided). It does not aim to build a full automated evaluation system. Other datasets like APPS(Hendrycks et al., 2021) are more complex, but HumanEval is simpler and more consistent. It is a pretty good starting point to see the effects of TDD guidance on code generation.