Think like a developer, write like a poet

System Design Document

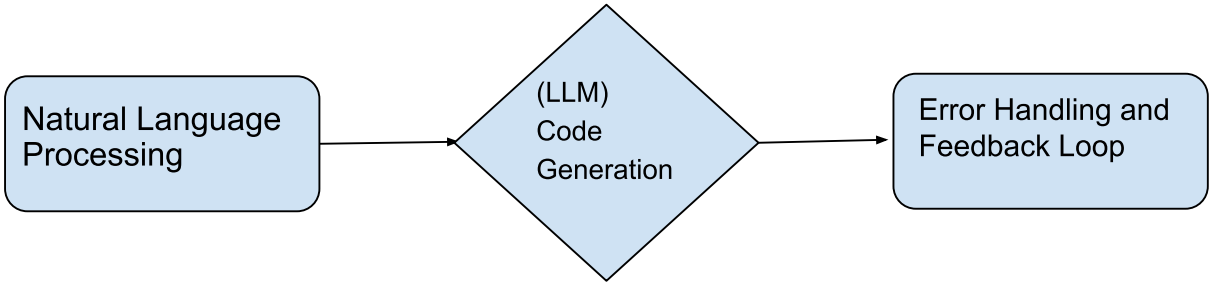
High-level Architecture

The architecture of the "Think like a developer, write like a poet" tool is designed to translate natural language descriptions into executable code using large language models (LLMs). The system consists of three main components:

1. Natural Language Processing (NLP) Module:
   - Responsible for parsing and understanding natural language descriptions provided by the user.

2. Large Language Model (LLM) Code Generation Module:
   - Utilises a pre-trained LLM to convert the structured representations from the NLP module into executable code.

3. Error Handling and Feedback Loop:
   - Monitors the output of the LLM for potential errors or unexpected behaviour, provides feedback to the user, and collects user feedback to improve the system's performance over time.



High Level Architecture

Low-level Architecture & Data FLow

Components:

User Interface (UI):
- Receives user Prompt input, Textual information.

Preprocessor:
- Cleans, tokenizes, and extracts keywords from the user input.

Context Processor :
- Analyses and processes information from the input prompt (surrounding text, user history, additional data) to generate relevant context prompts.

Prompt Assembler:

- Constructs a dynamic prompt by combining a base prompt template with extracted keywords and context prompts.

LLM Code Generation Module:

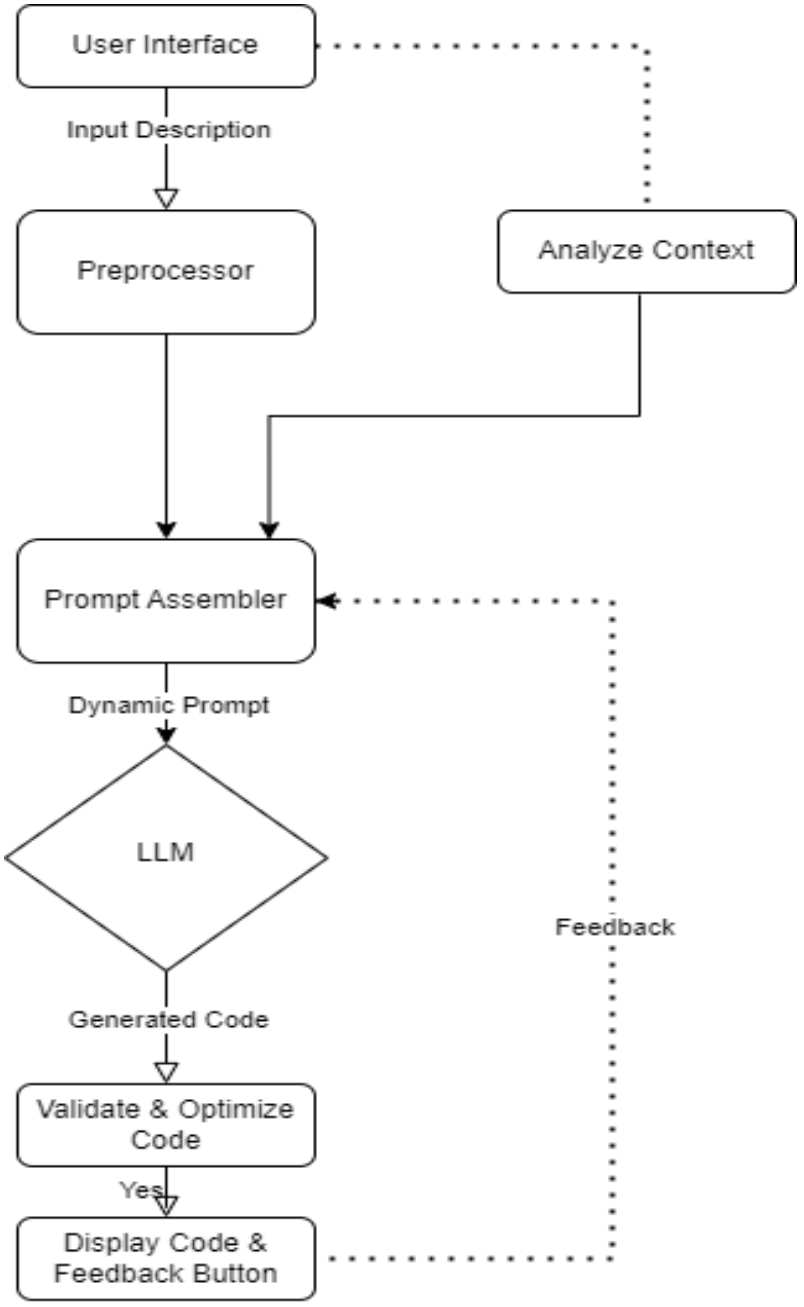
- Sends the dynamic prompt to a chosen LLM and receives the generated code.

Code Postprocessor:

- Validates and optimises the generated code for errors such as syntax, style, and efficiency.

Output: Presents the final code and a feedback mechanism to the user.

Data Flow Diagram



Data Flow

- User input flows from the UI to the Preprocessor and Context Processor (if used).
- Processed keywords and context prompts are sent to the Prompt Assembler.
- The dynamic prompt is sent to the LLM Code Generation Module, and the generated code is returned.
- The Code Post Processor refines the code, which is then displayed to the user.
- User feedback loops back to improve the system over time.

Key Characteristics:

- Modular design for flexibility and potential customization.
- Integration of contextual information for more relevant code generation.
- Error handling and feedback loop for continuous improvement.