1. User Input Layer

The process begins when the **user provides a product description**. Since large language models have a token input limitation, the system first checks if the input falls within the allowed token range.

- If it exceeds the limit, the system automatically **divides the input into smaller parts** to maintain coherence.
- This ensures scalability and allows even complex projects to be processed systematically without loss of context.

2. Problem Breakdown Layer

At this stage, the system transforms the product description into structured development tasks.

- **Extract High-Level Components**: The main features and modules of the product are identified.
- **Break Down into Sub-Problems**: Each component is further decomposed into smaller functional units for modular implementation.
- **Generate Smallest Functional Units**: The most granular tasks are defined, ensuring that each unit can be individually developed and tested.
- **Generate Test Cases and Expected Outputs**: Automated test cases are prepared in parallel to guide verification during code generation.

This structured breakdown guarantees **modularity, parallelism, and test-driven development**.

3. Code Generation Layer

This layer focuses on **actual code development and validation**.

- **Generate Code for Functional Units**: Source code is created for each functional block.
- **Write Function to File**: Generated code is systematically stored in structured files.
- **Execute in CLI & Test Functions**: Each unit is executed and validated against the pre-defined test cases.
- If outputs **do not match expectations**, the system enters an **Iterate & Refine loop**, improving the code until correctness is achieved.

This ensures a **fail-fast, self-correcting loop** where errors are caught and resolved immediately.