

- **Integrated Testing and Error Correction Module**

Another key deliverable will be the implementation of an auto-verification loop where generated code is immediately tested using auto-generated test cases. Errors will be captured, analyzed, and corrected through iterative feedback loops powered by the LLM agents. This process will validate the feasibility of self-healing code generation workflows.

- **Contextual Documentation Generation**

Each code module will be accompanied by detailed, context-aware documentation. The system will generate Markdown-based files outlining the purpose, structure, inputs, outputs, and dependencies of each component. This will improve maintainability and usability, especially in collaborative or enterprise scenarios.

- **Graph and Tree-Based Task Decomposition Engine**

The system will include a core engine capable of breaking down high-level software specifications into a tree of subtasks and subsequently converting it into a DAG (Directed Acyclic Graph) for dependency-aware code generation. This feature ensures logical sequencing and modular integrity throughout the build process.

- **Deployment-Ready Output**

The generated projects will include configuration files compatible with popular development workflows such as GitHub Actions, Docker containers, and CI/CD pipelines. The deliverables will be readily deployable on cloud or local environments, with minimal post-processing required by the developer.

- **Extensibility and Scalability**

The architecture will be designed to allow easy integration of additional LLMs, support for new programming languages, and adaptation for various software development templates. This will ensure that the system is not rigid or domain-specific, but rather generalizable for a wide range of real-world use cases.