3. Role-Based LLM Assignment for Specialized Outputs

One of the standout achievements of CodeCodez is its multi-agent LLM architecture. Subtasks identified in the decomposition stage are routed to LLMs fine-tuned or prompted for specific roles:

- Frontend LLMs handle UI generation.
- Backend LLMs manage route logic and database operations.
- Testing LLMs write unit and integration tests.
- Documentation LLMs generate markdown-based guides and API references.

This domain-specific delegation significantly improves the contextual relevance, correctness, and quality of generated code.

4. Embedded Real-Time Testing and Correction Mechanism

The system includes automated test case generation and execution pipelines that validate the functional integrity of generated code. If discrepancies or errors are identified:

- The system triggers an error correction loop that re-prompts or revises the LLM's output.
- Logs and execution traces are passed back to the responsible agent to guide correction.

This loop ensures not only syntactic correctness but also operational validity and alignment with user intent.

5. Automated Documentation Generation

CodeCodez also achieves auto-documentation across all modules. For every function, API route, or logic block, the system generates:

- Descriptive comments in the code,
- Markdown documentation (e.g., README.md, docs),
- Setup and deployment guides for the end user.

This enhances project transparency and maintainability while supporting easy onboarding for future developers or collaborators.