2. Application Layer

At the core of the system is the Application Layer, which handles planning and orchestration:

- **The Task Breakdown Engine** decomposes the project specification into smaller, manageable subtasks organized as a dependency tree (e.g., database → API → tests → documentation).
- **The LLM Orchestrator** coordinates execution by routing each subtask to the most suitable specialized agent, while also managing retries, caching, and cost efficiency.
- **Specialized LLM Agents** (Backend, Testing, Documentation) are responsible for generating domain-specific outputs, ensuring that the codebase, test suite, and documentation evolve consistently.

3. Processing Layer

Outputs from the Application Layer flow into the Processing Layer, which serves as a quality control and repair loop:

- Testing & Validation executes unit and integration tests, static analysis, and security scans.
- The Error Correction Loop analyzes failures and generates targeted fixes, which are re-routed back through the Orchestrator for patching. This creates an autonomous self-healing cycle that reduces the need for manual debugging.

4. Infrastructure Layer

The File Generator consolidates all artifacts—source code, test cases, documentation, and configuration files—into a consistent project structure. It ensures formatting, dependency management, and packaging, so the project is production-ready and can be directly integrated with external version control or CI/CD pipelines.

5. External Integrations

The system is extensible through External LLM Models and Open-source Plugins. These integrations allow for scaling, domain-specific customization, and adoption of organization-specific coding standards or deployment patterns. This modularity provides future-proofing and adaptability.