- Backend logic is handled by Python/Node.js-specific LLMs.
- Test case generation is delegated to models trained on unit testing patterns.
- Documentation is produced using instruction-following LLMs.

This multi-agent approach circumvents the limitations of monolithic LLMs by reducing cognitive drift and increasing domain precision. It also introduces redundancy and fallback mechanisms, where failure from one agent can be mitigated by another.

4. Prompt Orchestration and Result Aggregation

A **central orchestration layer** manages the flow of prompts, gathers responses from LLMs, and verifies structural and syntactical correctness. If any subtask's result is invalid or incomplete, an automatic re-prompting mechanism is triggered, either with modified instructions or using a different model. This orchestration ensures both **quality assurance** and **consistency** across modules.

Additionally, prompts include contextual cues from adjacent modules to preserve inter-component cohesion—for instance, ensuring that the frontend and backend use matching data models or naming conventions.

5. Code Integration, Testing, and Correction

After all individual modules are generated, the system performs **automated integration** into a unified project structure. This includes creating directories, updating import paths, generating configuration files (e.g., .env, package.json, requirements.txt), and writing readme documentation.

A test suite is then executed to evaluate functional correctness. If test failures occur, the orchestrator re-engages the relevant LLM with detailed error traces, enabling **iterative correction**.

6. Documentation and Final Packaging

Each generated module is documented using markdown files, covering its purpose, inputs/outputs, and usage guidelines. These are compiled into a developer handbook stored alongside the codebase.

Finally, the complete project is packaged with support for version control (Git), Dockerization (if required), and deployment readiness (CI/CD configuration templates), allowing the developer to clone, run, and scale the system effortlessly.