3. **Self-reflection** — feedback loop inspired by *CodeCoR* [14], where agents critique their own outputs and adjust.
4. **Optimization** — refinement of inefficient or incorrect modules until predefined quality thresholds are met.

This approach mirrors experiential learning models such as *MAEL* [9], where agents refine future decisions based on prior execution history. Over time, this allows CodeCodez to build an adaptive knowledge base of common design patterns, error resolutions, and optimized workflows.

System Architecture

The proposed solution is architected as a layered multi-agent system:

- **Input Layer** → accepts natural language requirements and parses them into structured goals.
- **Decomposition Layer** → applies hierarchical orchestration strategies [3], [12], [18].
- **Execution Layer** → role-specialized agents [4], [15], [16] collaboratively execute subtasks.
- **Reasoning Layer** → debate, consensus, and belief-update mechanisms ensure correctness [2] , [5] , [7].
- **Validation Layer** → iterative testing and self-reflection loops refine outputs [14] , [16].
- **Output Layer** → produces complete project deliverables including source code, configuration files, test suites, and documentation.

This architecture ensures modularity, scalability, and adaptability, making CodeCodez well-suited for diverse project domains such as web applications, machine learning pipelines, and data dashboards.

Justification and Novelty

The novelty of CodeCodez lies in its integration of multiple state-of-the-art techniques into a unified framework: