REQUIREMENT ANALYSIS

2.1 Literature Survey

2.1.1 Theory Associated With Problem Area

In recent years, the field of code generation has witnessed a paradigm shift with the integration of Large Language Models (LLMs) into software engineering workflows. These models have demonstrated remarkable capabilities in natural language understanding, code synthesis, and autonomous task planning. However, the leap from generating isolated code snippets to producing complete, production-grade software systems remains a significant theoretical and practical challenge.

A key theoretical foundation lies in multi-agent collaboration among LLMs, wherein multiple specialized models interact to solve distinct subtasks—such as frontend design, backend logic, testing, and documentation—in a coordinated manner. The Chain-of-Agents framework demonstrates how LLMs can operate collaboratively over long-context tasks, simulating human-like division of labor to enhance reasoning capabilities and task coherence [1]. Building on this, Corexintroduces the notion of collaborative reasoning with diverse LLMs, where agents exchange intermediate reasoning steps to arrive at more reliable outcomes [2].

Another foundational concept is hierarchical task decomposition, often represented through trees or directed acyclic graphs (DAGs). This structure is pivotal in CodeCodez, where a root task (e.g., “Build a financial tracker app”) is broken down into manageable subtasks (UI, authentication, database integration, etc.) and further mapped into a DAG to handle interdependencies and generation order. Research such as HALO [3] and TDAG [12] support the notion of using hierarchical and graph-based representations to enhance agent orchestration and avoid logical inconsistencies.

In the context of reasoning fidelity, studies like ReConcile [6] and Debate to Equilibrium [5] argue that LLMs can reach more factually accurate and semantically consistent conclusions when modeled as debating or consensus-seeking agents. This directly supports CodeCodez’s self-verification module, where generated code is reviewed and corrected through consensus-driven loops among agents.