- **LLM Agent Network:** Assigns subtasks to specialized LLMs (frontend, backend, testing, documentation).
- **Testing and Error Correction Module:** Performs validation of generated code and resolves errors automatically.

3. **Output Interfaces**
   - **Codebase Generator:** Produces complete directory structures, configuration files, modular code, and reusable components.
   - **Documentation Generator:** Creates detailed, developer-friendly documentation alongside generated code.
   - **Integration APIs:** Allows export to Git repositories, Docker containers, or CI/CD pipelines.

Relationship with Other Systems

- **Complementary to Existing IDEs and Tools:**
  CodeCodez does not aim to replace established IDEs (e.g., VS Code, PyCharm) but rather enhances them by automating initial project scaffolding and repetitive coding tasks.
- **Interoperability with Development Ecosystem:**
  - Compatible with **version control systems** like GitHub and GitLab.
  - Supports **containerization** and deployment with Docker/Kubernetes.
  - Can be extended to integrate with **CI/CD pipelines** (e.g., Jenkins, GitHub Actions).
- **AI Model Dependence:**
  The system relies on **pre-trained LLMs** for task execution. It does not train models from scratch but uses existing AI capabilities (e.g., OpenAI GPT models, Ollama for lightweight deployment) to generate outputs.

Constraints and Assumptions

- Requires stable internet connectivity for accessing cloud-based LLMs (unless deployed with local models via Ollama).
- The system assumes users provide **clear and unambiguous requirements** for effective interpretation.
- Hardware resources (GPU/CPU) may influence performance, though future work aims to optimize resource usage.