

- Support for step-by-step interaction (e.g., asking for confirmation before creating modules).
- Colored terminal outputs to highlight file types, warnings, or errors.
- Error handling and recovery messages for incomplete or invalid prompts.

### 2.2.3.2 Software Interfaces

The system must integrate smoothly with both internal components and external developer tools.

- **Operating System Compatibility:**
  - Cross-platform support for **Linux, macOS, and Windows** command-line environments.
  - Runs within standard shells such as Bash, Zsh, and PowerShell.
- **Programming Language Environment:**
  - Core implementation is in **Python**, enabling integration with LLM frameworks like LangChain.
  - Support for external package managers (e.g., pip, npm, Maven) to scaffold dependencies inside generated projects.
- **External APIs and Libraries:**
  - Integration with **OpenAI/LLM APIs** for text-to-code generation.
  - Optional use of **Ollama** or other lightweight local inference backends for offline mode.
- **Interoperability:**
  - Generated code can be exported to GitHub/GitLab repositories directly through CLI commands.
  - Supports Dockerfile and CI/CD pipeline generation to integrate into DevOps workflows.
- **Error and Exception Interfaces:**
  - The CLI provides structured error codes and logs when failures occur in project generation.
  - Compatibility with monitoring/logging systems like **Prometheus or ELK stack** can be extended in future iterations.