spip: A State-of-the-Art Version-Controlled Pip Replacement

Antigravity

January 2026

1 Introduction

Current Python package management using `pip` and `venv` suffers from environment drift and a lack of primitive versioning. `spip` introduces a pure implementation that bypasses `pip` architecture entirely and provides a rich, informative user interface and deep dependency analysis.

2 The spip Architecture

`spip` addresses these via a novel architecture combining:

- **Centralized Vault**: All environments are stored in `~/.spip/envs/`, identified by project hashes.
- **Git-as-Database**: A central Git repository (`~/.spip/repo`) tracks all environment states.
- **Custom Installer**: Bypasses `pip` by implementing a recursive dependency resolver and a wheel-based installation engine directly in C++23.
- **Dependency Tree Analysis**: Built-in command (`spip tree`) to visualize complex sub-dependency hierarchies recursively.
- **Interactive UI**: Real-time progress bars for both individual package downloads and global installation progress.
- **Deterministic Uninstaller**: Precisely removes packages by parsing the `RECORD` metadata file.
- **Local Registry (DB)**: A partitioned, versioned Git repository in `~/.spip/db/` containing metadata for 730,000+ packages.
- **Atomic Commits**: Every single package extraction or removal is an atomic Git commit.

3 Formal Verification

The isolation strategy has been formally verified in Coq (`safety.v`), proving that recursive dependency installation only ever targets the local user branch and never contaminates the global system state.

4 Implementation

`spip` is implemented in high-performance C++23. It utilizes multi-threading for the registry crawler and advanced shell escaping for risk-free command execution.