# Offline Backup Scheduler Desktop App with Versioning

Offline Backup Scheduler Desktop App with Versioning – VaultKeeper: Scheduled, Encrypted, Restorable Backups

## Introduction / Background

Everyone knows backups matter; almost nobody keeps them current. Cloud backup demands subscriptions and uploads, while manual copying is forgotten within a week — and when a drive fails, the ‘backup’ is usually months stale or missing the one folder that mattered. The failure is rarely technical; it is behavioural. People need backups that happen automatically, keep history, and make restore a one-click act rather than an archaeology project.

VaultKeeper is a desktop app that schedules offline backups with full versioning: jobs bind source folders to the user's own drives, run on triggers, and store every run as a restorable snapshot. AES-256 encryption, checksum verification and retention policies make it operationally complete — and everything runs locally, with no account and no cloud.

## Problem Statement

Manual and cloud-dependent backup habits leave users with stale, unverified or inaccessible copies. What is needed is automatic scheduled backup to the user's own media, with version history, encryption, integrity checks and genuinely simple restore — all working offline.

## Objectives

1. Schedule backup jobs (daily/weekly/hourly/on-shutdown) without OS cron knowledge.

2. Implement full/incremental/differential versioning with space-efficient chains.

3. Provide a version timeline with per-run change stats and one-click restore of any snapshot.

4. Encrypt backup sets with AES-256 and verify integrity via checksums.

5. Enforce retention policies and surface failures loudly instead of failing silently.

## Proposed Methodology

Python + PyQt6 desktop app. APScheduler drives per-job triggers. The engine diffs the source tree against the last version, copies changed files with xxhash checksums into a version store, and references unchanged files (incremental chains). Each version is catalogued in SQLite with timestamp, type, size delta and file count, then verified by re-reading checksums. Restore replays the chain to the chosen version with verification on write. Retention prunes outside policy with confirmation for full baselines.

## System Architecture / Working

Scheduler (APScheduler) → engine (snapshot, diff, copy, verify) → version store + SQLite catalogue → timeline UI → restore path. Encryption wraps the store with a password-derived AES-256 key. Failure handling: missed runs (drive absent) are marked and alerted, never silently skipped.

## Hardware / Software Requirements

Software: Python 3, PyQt6, SQLite, APScheduler, cryptography, xxhash, PyInstaller (packaging). Hardware: a desktop plus a destination drive (external HDD/partition/share) for real backup runs. No special hardware required.

## Expected Outcomes

A desktop app running scheduled versioned backups to a local drive, with an encrypted set, a browsable timeline and verified one-click restore of any version. Expected secondary outcomes: the incremental-chain design document and measured space savings of incremental vs full on the demo data.

## Applications

Students and professionals protecting project work; small offices without IT staff; photographers with large local archives; labs needing auditable data retention.

## Future Scope

Block-level deltas for huge files; disk imaging and bare-metal restore; optional encrypted cloud mirror as a secondary target; and backup-set portability verification across machines.