PROJECT ABSTRACT

Desktop TOTP Authenticator App

A desktop authenticator generating RFC 6238 time-based one-time passwords offline, with an AES-256 encrypted vault and backup. Delivered built-to-order with desktop source, web demo, report, PPT and viva kit.

01. Title

Desktop TOTP Authenticator App

02. Introduction / Background

Two-factor authentication is only as convenient as the device holding the codes, and students working on laptops keep reaching for a phone app they cannot use mid-task. A desktop TOTP authenticator solves this by implementing RFC 6238 — base32 secret decoding, HMAC-SHA1, dynamic truncation and the 30-second time counter — so every code is genuinely computed. Secrets rest in an AES-256 encrypted vault under a user-set password, with QR-based enrolment, one-click copy and encrypted backup for migration.

03. Problem Statement

Desktop users lack a trustworthy local 2FA code generator, and most authenticator internals are black boxes students cannot explain. The project addresses both: a desktop app implementing TOTP from the RFC with documented mathematics, AES-256 vault encryption, and a live web demo so the algorithm, countdown and backup flows are all demonstrable.

04. Objectives

- Implement RFC 6238 TOTP (6 digits, 30 s step, SHA-1) from first principles with documentation.
- Store secrets in an AES-256-GCM vault keyed by PBKDF2 from a user-set password.
- Render live codes with a 30-second countdown ring and one-click clipboard copy.
- Support enrolment via 2FA QR scan or manual base32 secret entry.
- Export and restore an encrypted .vkenc backup file.
- Deliver the complete student kit: desktop source, web demo, report, PPT and viva Q&A.;

05. Proposed Methodology

The build has two parts. (1) Desktop app (Python): a TOTP core module implements base32 decoding, HMAC-SHA1, dynamic truncation and modulo reduction; a vault module wraps secrets with AES-256-GCM under a PBKDF2-derived key; the UI shows accounts with countdown rings and copy buttons, plus enrolment and backup screens. (2) Web demo: the same algorithm in JavaScript with six sample accounts demonstrates live code

generation, the add-account flow and backup UI.

06. System Architecture / Working

On enrolment the secret is encrypted into the vault immediately. Every 30 seconds the app computes the time counter from Unix epoch, HMAC-SHA1 hashes it with the account secret, truncates dynamically and reduces modulo 1,000,000 to the 6-digit code. The UI shows the code with its countdown ring; copy places it on the clipboard with automatic clearing. Backup exports the encrypted vault as one file; restore decrypts it after password verification.

07. Hardware / Software Requirements

- Windows 10+ or Ubuntu 22.04+ desktop (design target)
- Python 3, cryptography library (AES-256-GCM, PBKDF2)
- qrcode + OpenCV (QR enrolment)
- Tkinter/PyQt (desktop UI)
- Modern browser (web demo)

08. Expected Outcomes

- A working desktop authenticator generating verifiably correct TOTP codes.
- An encrypted vault with documented key-derivation parameters.
- A live web demo with six accounts for viva demonstration.
- A complete report, PPT and viva Q&A; on TOTP mathematics and vault encryption.

09. Applications

- Everyday 2FA code generation on the desktop.
- Teaching applied cryptography: HMAC, AES-GCM, PBKDF2.
- Demonstrating RFC implementation from specification.

10. Future Scope

- FIDO2/WebAuthn hardware-key support.
- Encrypted cloud sync across a user's own devices.
- TOTP over shorter custom time steps for enterprise tokens.
- Browser-extension companion for auto-fill.