Affiliated to Savitribai Phule Pune University

A PROJECT REPORT ON

"VAULTLINK SECURITY SUITE"

SUBMITTED TO

Savitribai Phule Pune University

SUBMITTED BY

Om Hanumant Shinde

Bachelor of Computer Science (BSc CS)

Semester VI

UNDER THE GUIDANCE OF

Prof. Preti Jadhav

Department of Computer Science

Academic Year 2025-2026

Swaraj College of Arts, Commerce and Science, Pune

Gurudatta Housing Society, Satara Rd, Behind Khushabu Hotel,
Balaji Nagar, Dhankawadi, Pune, Maharashtra 411043

CERTIFICATE

This is to certify that Mr. Om Hanumant Shinde, student of Division T.Y. B.Sc. (Computer Science), has satisfactorily completed their project work entitled "VaultLink" and submitted the project for the fulfilment of Bachelor of Computer Science as per syllabus of Savitribai Phule University of Pune, during the academic year 2025-26

Date:

Exam Seat No: 60589

Subject Teacher **Head of Department**

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

Working on VaultLink has been one of the most challenging and rewarding experiences of my academic journey. What started as an idea for a simple file encryption tool grew into a full security suite over many late evenings, and that growth would not have been possible without the encouragement and support of several people I genuinely want to thank.

My deepest gratitude goes to Prof. Preti Jadhav, my project guide, whose patient guidance kept me on the right path every time I got stuck or went off track. Her feedback was always precise and practical, and she pushed me to think beyond just making things work — to truly understand the why behind every design decision. I am very grateful for her time, her patience, and her consistent encouragement throughout this project.

I would also like to thank the faculty and staff of the Department of Computer Science at Swaraj College of Arts, Commerce and Science, Pune, for creating an environment where curiosity is encouraged. The Principal and College Management deserve recognition for providing the infrastructure and resources that made this project possible.

Finally, I want to thank my family and friends for their patience and moral support during the long hours spent building and debugging this project. This work is as much theirs as it is mine.

Om Hanumant Shinde

BSc Computer Science, Semester VI

Swaraj College of Arts, Commerce and Science, Pune

DECLARATION

I hereby declare that the present final year project work "VaultLink Security Suite" is original work carried out under the guidance of Prof. Preti Jadhav, Department of Computer Science, Swaraj College of Arts, Commerce and Science, Pune. It has not been submitted by me in part or full to any University for any examination before. This work has been carried out by me at Savitribai Phule Pune University during the academic session 2025-2026.

Date: _____

Sign: _____

Om Hanumant Shinde

BSc Computer Science, Semester VI

Swaraj College of Arts, Commerce and Science

Gurudatta Housing Society, Pune,

Satara Rd, behind Khushabu Hotel,

Balaji Nagar, Dhankawadi, Pune, Maharashtra 411043

ABSTRACT

VaultLink Security Suite is a comprehensive Java-based desktop security application that unifies encrypted file storage with active threat protection under a single, modern interface. The system addresses a growing gap in personal security software: most encryption tools offer only static file protection without real-time threat monitoring, while antivirus solutions typically offer no privacy-preserving encrypted storage.

VaultLink combines AES-256-GCM file encryption, OAuth 2.0 authentication with PKCE (Proof Key for Code Exchange), PBKDF2-based cryptographic key derivation, ClamAV antivirus integration, Google Safe Browsing URL malware detection, and a crypto-mining process monitor into one cohesive application.

The application follows a modular architecture built on JavaFX with a futuristic dark-themed interface. Files stored in the vault are encrypted using a 256-bit AES key in Galois/Counter Mode, providing both confidentiality and tamper detection. The encryption key is never stored directly — it is re-derived on each login using the user's OAuth identity combined with a per-device random salt, processed through 310,000 iterations of PBKDF2WithHmacSHA256.

The result is a security suite that demonstrates how modern cryptographic techniques, identity-based authentication, and real-time threat detection can be practically integrated into a user-facing application with genuine utility.

INDEX

1. INTRODUCTION

1.1 Problem Definition

In an era of increasing digital threats, individuals face two distinct but related security challenges: protecting sensitive files from unauthorized access, and defending their systems from active threats such as malware, phishing websites, and resource-hijacking software. Current solutions treat these problems in isolation — encryption utilities protect stored data but cannot detect threats, while antivirus software monitors for threats but provides no mechanism for secure file storage.

The absence of an integrated solution forces users to manage multiple separate tools, creating gaps in their security posture and increasing the cognitive overhead of maintaining personal digital safety. Furthermore, existing encryption solutions often rely on static passwords, which are vulnerable to brute-force attacks and poor user practices.

VaultLink Security Suite was conceived to address this gap by providing a unified platform that combines robust encryption, identity-based key derivation, antivirus scanning, URL threat detection, and process-level monitoring for crypto-mining attacks in a single, cohesive desktop application.

1.2 Objectives of the System

- Provide AES-256-GCM encrypted file storage accessible only through authenticated login
- Implement OAuth 2.0 with PKCE for secure, password-free identity verification
- Derive encryption keys from user identity using PBKDF2 to resist brute-force attacks
- Integrate ClamAV for real-time antivirus scanning of files and folders
- Implement URL malware checking using Google Safe Browsing API v4
- Monitor running processes for known crypto-mining malware signatures
- Provide a modern dark-themed UI with animated feedback
- Ensure encrypted files are completely unreadable without authenticated access
- Support peer-to-peer encrypted file transfer with PIN-based authentication

1.3 Scope of the Project

VaultLink is designed as a standalone desktop security application targeting Windows, macOS, and Linux. The scope encompasses: full OAuth 2.0 login with Google and GitHub, encrypted vault storage for any file type, on-demand antivirus scanning powered by ClamAV, single and batch URL threat checking against Google Safe Browsing, background process monitoring with configurable intervals, a unified dashboard with all protection metrics, and peer-to-peer file transfer between VaultLink instances.

2. NEED OF COMPUTERIZATION

The rapid digitization of personal and professional data has made security an inescapable concern for all computer users. Manual approaches to data protection are inadequate against modern threats including malware, ransomware, phishing, and unauthorized system access. A computerized security suite addresses this need in several critical ways:

- Automated encryption eliminates human error in applying file protection
- Real-time scanning detects threats faster than any manual process
- Cryptographic key derivation provides mathematically stronger protection than passwords
- Centralized management reduces the overhead of using multiple disparate security tools
- Process monitoring runs continuously in the background without user intervention

The increasing prevalence of crypto-mining malware, which silently consumes system resources, and phishing attacks, which harvest credentials through deceptive websites, demonstrates that reactive security is insufficient. VaultLink addresses this through proactive, automated protection that requires minimal user intervention while providing maximum coverage.

3. FACT-FINDING TECHNIQUES

The following fact-finding techniques were employed during the requirements gathering phase of the project:

Literature Review

Existing security software including BitLocker, VeraCrypt, Windows Defender, and Malwarebytes were studied to understand current approaches to file encryption and threat detection. Their architectures, user interfaces, and technical implementations informed the design of VaultLink.

Technical Documentation Review

Official documentation for the following technologies was studied in depth: OAuth 2.0 RFC 6749, PKCE RFC 7636, AES-GCM (NIST SP 800-38D), PBKDF2 (RFC 8018), ClamAV documentation, and Google Safe Browsing API v4 specification.

Prototyping

Incremental prototyping was used extensively. The crypto layer was built and tested independently before integration with the authentication system. Each module was verified in isolation through unit tests before being wired into the complete application.

Internet Research

Current threat statistics, OWASP security recommendations for key derivation iteration counts, and best practices for OAuth implementation in desktop applications were researched through security-focused publications and standards bodies.

4. STUDY OF EXISTING SYSTEM

4.1 Existing Systems

Several existing products provide partial solutions to the security challenges addressed by VaultLink:

System	Primary Function	Limitation
BitLocker	Full disk encryption	Windows-only, no threat detection, no per-file granularity
VeraCrypt	Encrypted containers	Complex UI, no cloud auth, no antivirus integration
Windows Defender	Antivirus and threat detection	No encrypted storage, Windows-only, no URL checker
Malwarebytes	Malware scanning and removal	No file vault, premium features require subscription
LastPass	Password management with encryption	No file storage, no antivirus, cloud-dependent

4.2 Drawbacks of Existing Systems

- No existing solution combines encrypted file storage with integrated antivirus scanning
- Most encryption tools rely on static passwords vulnerable to brute-force attacks
- URL threat checking is absent from all standalone encryption utilities
- Process-level crypto-mining detection is not integrated with file security in any existing tool
- Identity-based key derivation using OAuth is not used by any comparable open-source tool

5. PROPOSED SYSTEM

VaultLink Security Suite is proposed as a unified, cross-platform desktop application that addresses all identified shortcomings of existing systems. It integrates four independent but complementary security layers into a single application with a cohesive user interface.

5.1 Features of Proposed System

- OAuth 2.0 + PKCE authentication with Google and GitHub — no password stored on device
- AES-256-GCM file encryption with per-file initialization vectors
- PBKDF2WithHmacSHA256 key derivation at 310,000 iterations (OWASP 2023 recommendation)
- Per-device random salt ensures vault files are non-portable without re-encryption
- ClamAV antivirus integration for scanning files, folders, and downloaded content
- Google Safe Browsing API v4 integration for single and batch URL malware detection
- ProcessHandle-based crypto-mining detector with database of 15+ known miner signatures
- Network connection monitoring for known mining pool domains
- Dashboard with animated protection ring, layer bars, and real-time statistics
- Peer-to-peer file transfer with PIN authentication over local network
- SQLite metadata store for file index management
- All encryption keys zeroed in memory on logout

5.2 Objectives of Proposed System

- No plaintext file content is ever written to disk — encryption occurs before storage
- No user passwords stored or transmitted — OAuth tokens held in memory only
- Encrypted .vault files appear as random byte sequences without authentication
- Any tampering with encrypted files detected by AES-GCM authentication tag
- Brute-force attacks on key derivation face 310,000 iterations per attempt

5.3 System Specifications

Hardware Requirements

Component	Minimum Specification
Processor	Dual-core 2.0 GHz or higher (x64 architecture)
RAM	4 GB (8 GB recommended)
Storage	500 MB free disk space plus vault storage
Network	Internet connection required for OAuth login and URL checking
Operating System	Windows 10/11, macOS 10.14+, or Linux (Ubuntu 20.04+)

Software Requirements

Component	Specification
Programming Language	Java 21 (OpenJDK 25 tested)
UI Framework	JavaFX 21.0.2
Build Tool	Apache Maven 3.8+
Encryption	AES/GCM/NoPadding, PBKDF2WithHmacSHA256
Authentication	OAuth 2.0 + PKCE (Nimbus JOSE+JWT 9.37.3)
Antivirus Engine	ClamAV 1.x (must be installed separately)

Component	Specification
URL Threat Detection	Google Safe Browsing API v4
Database	SQLite 3.45 (via SQLite JDBC 3.45.2.0)
JSON Processing	Jackson Databind 2.17.0
IDE	VS Code with Extension Pack for Java

6. SYSTEM DESIGN

6.1 Data Flow Diagram — Level 0 (Context Diagram)

The Context Diagram shows VaultLink as a single process interacting with three external entities: the OAuth Provider (Google/GitHub), the User, and External APIs (ClamAV and Google Safe Browsing). At this level, all system complexity is abstracted into a single process bubble to show the high-level data flows.

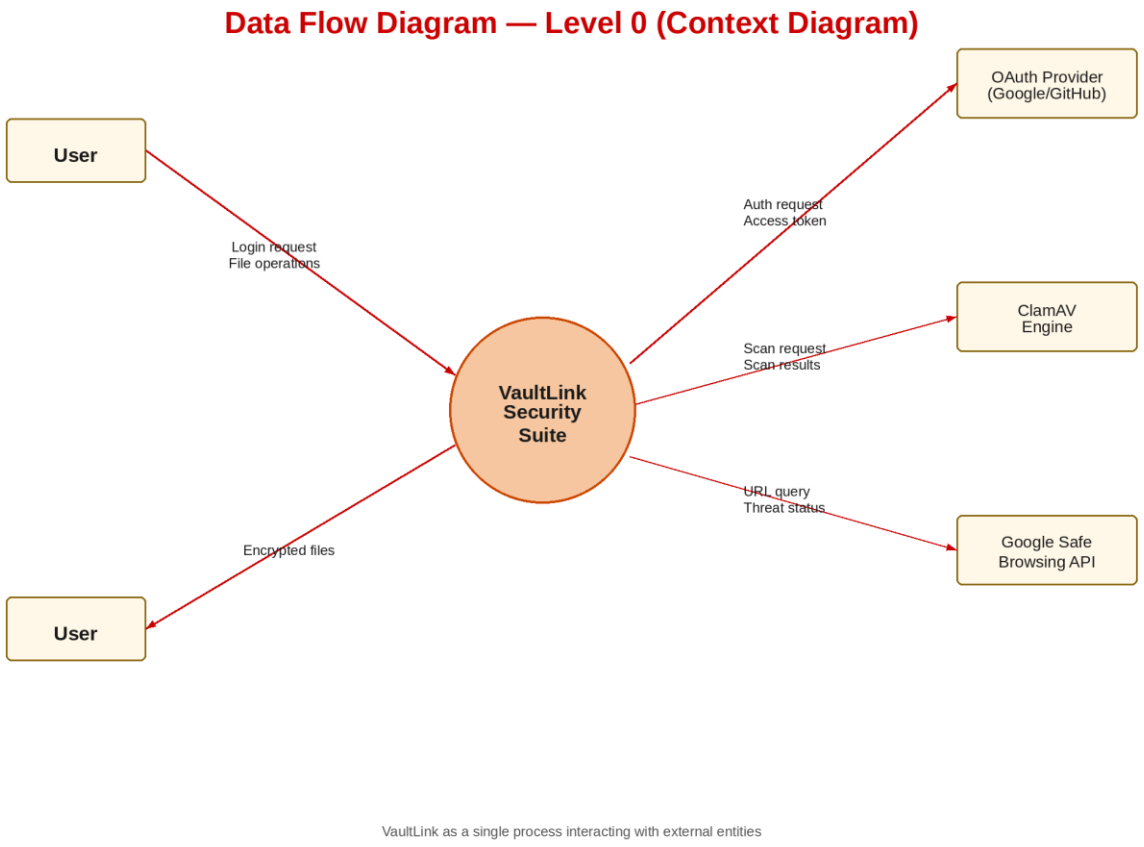

Figure 6.1 — Data Flow Diagram Level 0 (Context Diagram)

6.2 Data Flow Diagram — Level 1

The Level 1 DFD decomposes the system into five primary processes showing the major data transformations within VaultLink: Authentication, Vault Operations, Antivirus Scan, URL Check, and Process Monitor.

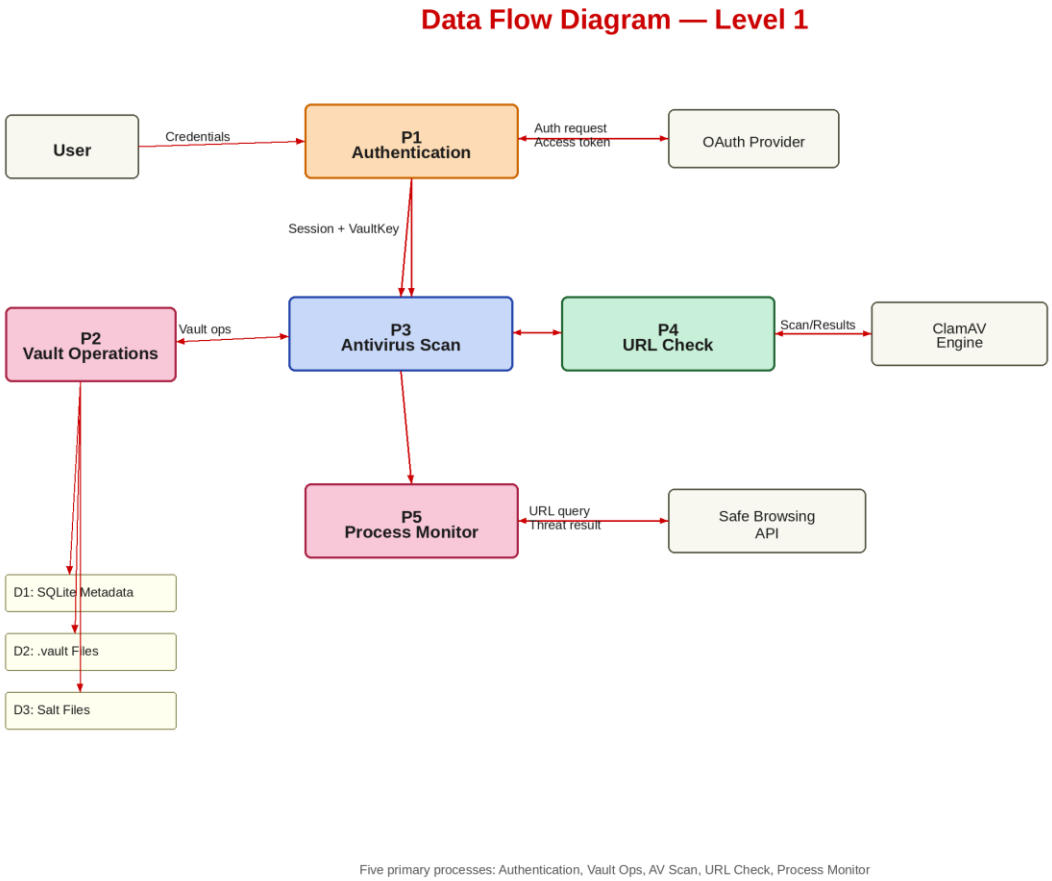

Figure 6.2 — Data Flow Diagram Level 1

6.3 Use Case Diagram

The Use Case Diagram shows all interactions between the authenticated user and the VaultLink system. The primary actor is the Authenticated User. All use cases require successful OAuth login as a precondition.

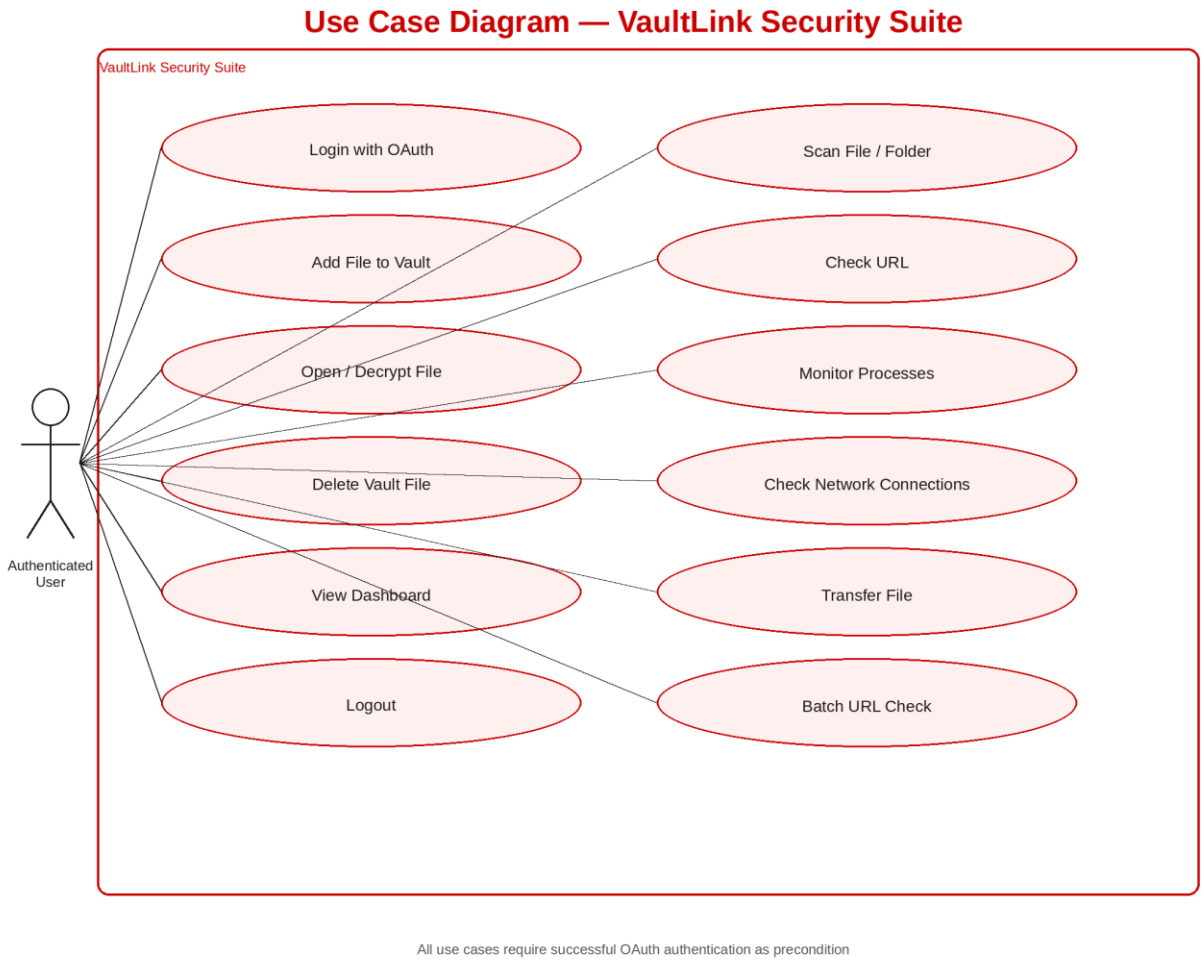

Figure 6.3 — Use Case Diagram

6.4 Architecture Diagram

VaultLink follows a layered modular architecture with clear separation of concerns. Each layer communicates only with adjacent layers through well-defined interfaces, enabling independent testing and future enhancement of individual modules.

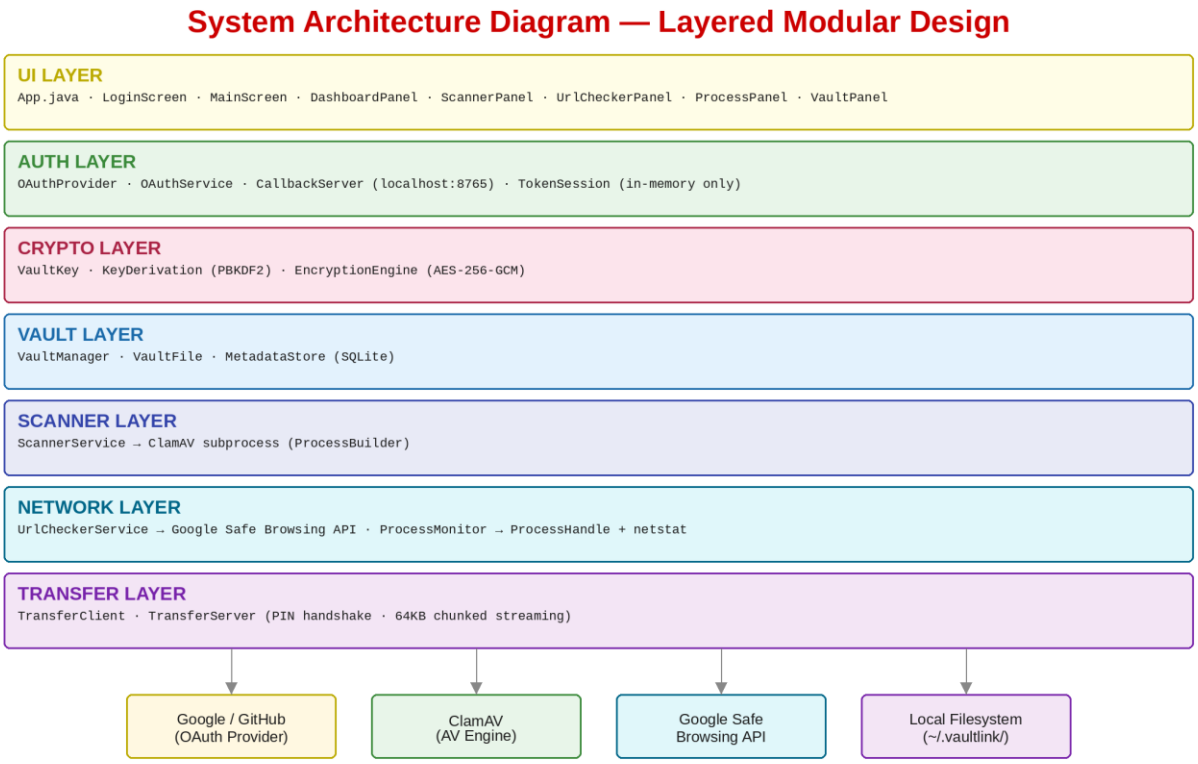

Figure 6.4 — System Architecture Diagram

6.5 Entity-Relationship Diagram

The ER Diagram shows the data entities and their relationships within VaultLink. The primary stored entity is VaultFile, managed in SQLite. The User entity is derived from the OAuth TokenSession and is never persisted. Device Salt is stored as a file keyed by userId.

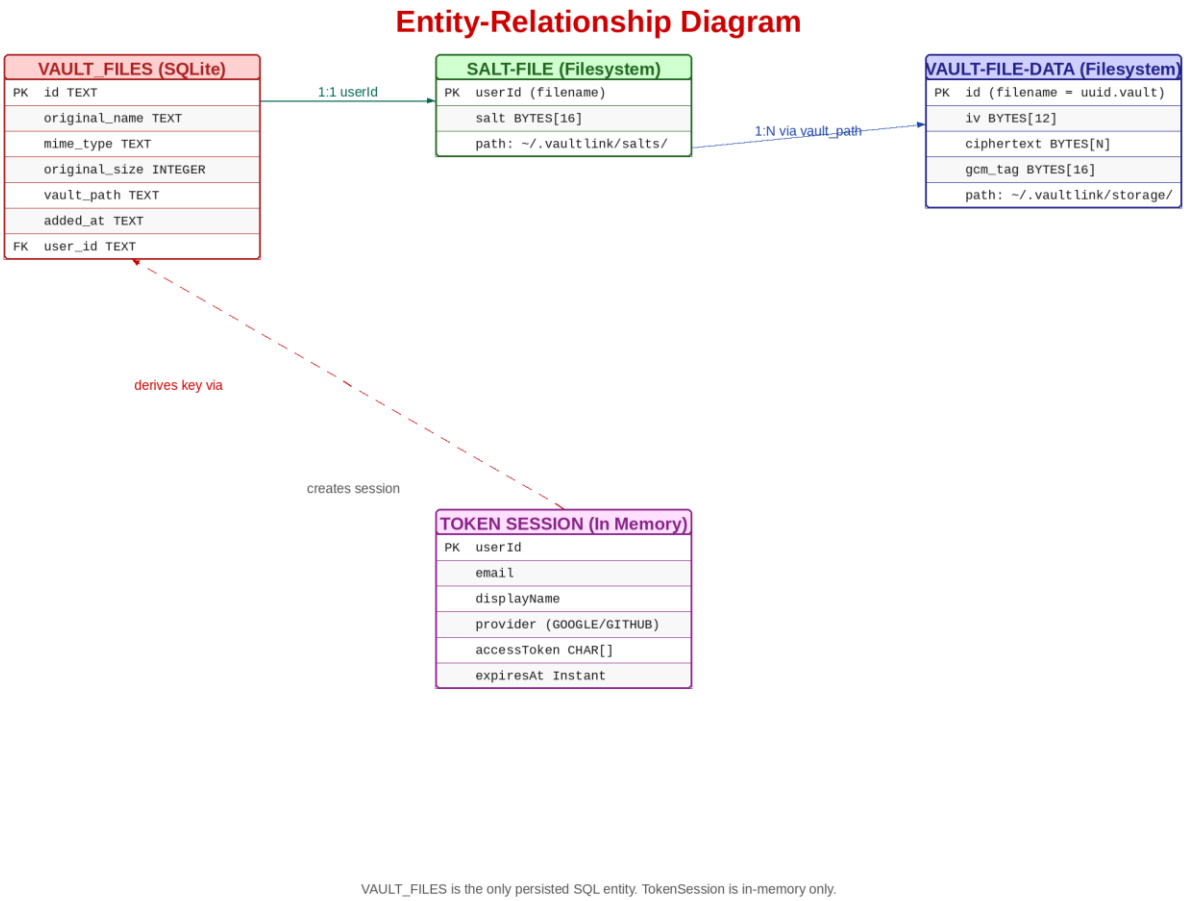

Figure 6.5 — Entity-Relationship Diagram

6.6 Class Diagram

The Class Diagram shows the primary classes across all modules, their attributes, methods, and relationships. Dependencies flow downward: the UI layer depends on all service layers, while the crypto layer has no dependencies on application code.

Figure 6.6 — Class Diagram

6.7 Sequence Diagram

The Sequence Diagram illustrates the runtime message flow for the most critical use case: OAuth login followed by vault file encryption. This diagram shows the exact order of method calls, external API interactions, and object lifetimes during a complete login-and-add-file session.

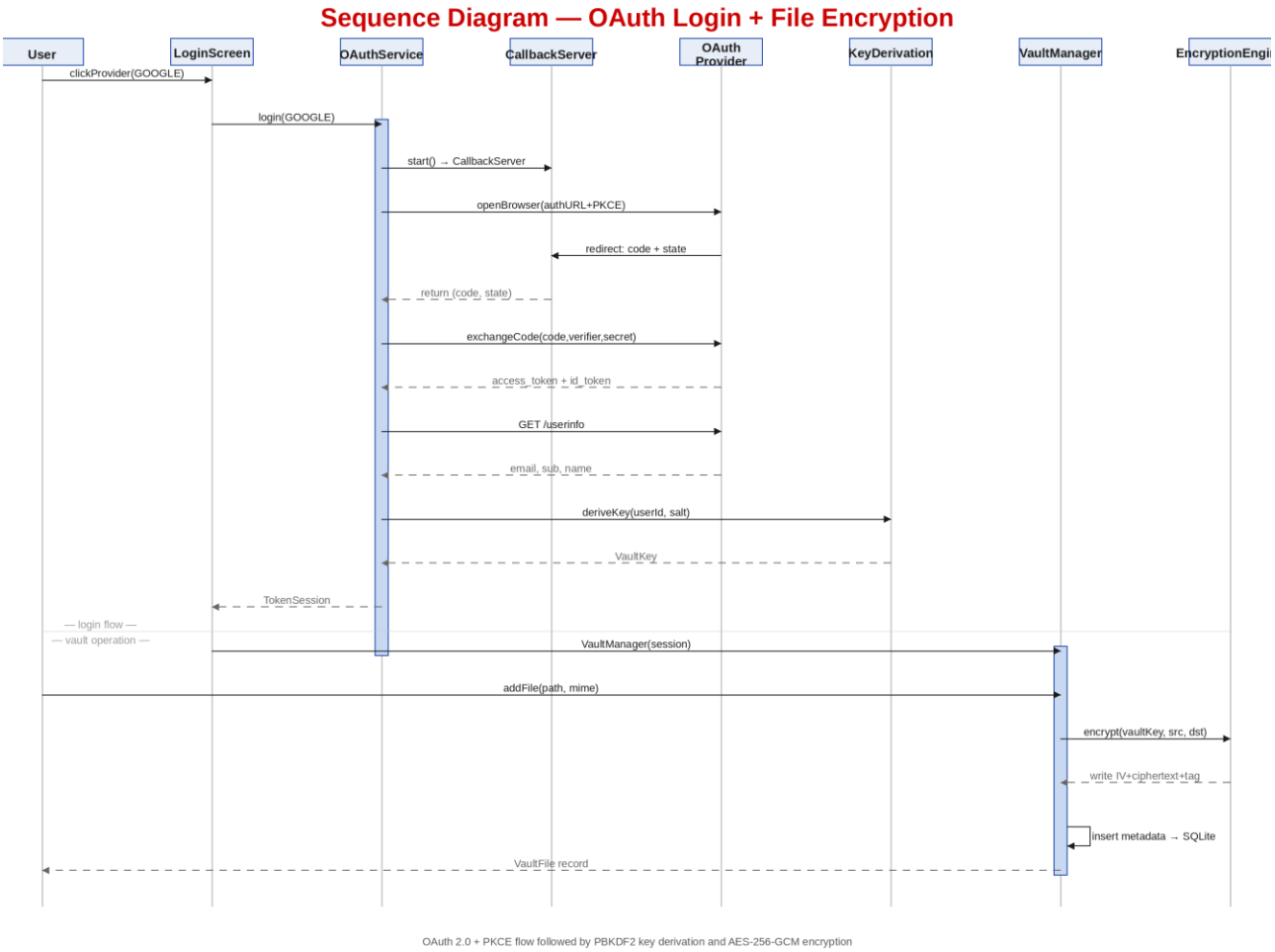

Figure 6.7 — Sequence Diagram (OAuth Login + File Encryption)

6.8 Activity Diagram

The Activity Diagram shows the complete workflow for the file scanning process, from user input through ClamAV execution to result presentation. Decision nodes represent branching logic based on scan results and user selections.

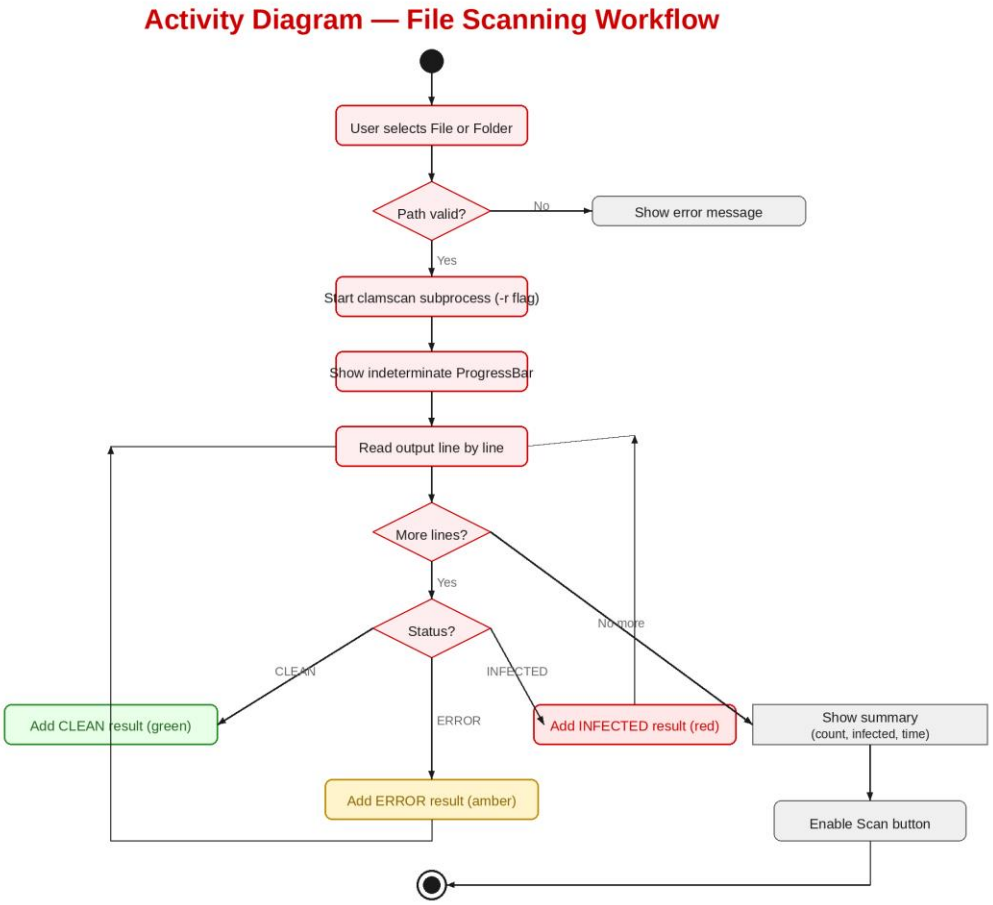

Figure 6.8 — Activity Diagram (File Scanning Workflow)

7. MODULE DESCRIPTION

VaultLink is organized into seven independent modules, each encapsulating a distinct area of functionality. Modules communicate through well-defined interfaces, enabling independent testing and future replacement.

7.1 Authentication Module (com.vaultlink.auth)

Implements the complete OAuth 2.0 Authorization Code flow with PKCE. A 96-byte random code verifier is generated using SecureRandom, and its SHA-256 hash forms the code challenge. A temporary HTTP server starts on localhost:8765 to receive the redirect. A random state parameter prevents CSRF attacks. The authorization code is exchanged for an access token, user profile is fetched, and the resulting TokenSession is held only in memory — never serialized to disk.

Key classes: OAuthProvider (enum), OAuthService (full PKCE flow), CallbackServer (ephemeral localhost server), TokenSession (immutable in-memory session).

7.2 Cryptography Module (com.vaultlink.crypto)

KeyDerivation uses PBKDF2WithHmacSHA256 at 310,000 iterations to derive a 256-bit AES key from the user's OAuth identity string combined with a per-device 128-bit random salt stored at ~/.vaultlink/salts/. EncryptionEngine implements AES/GCM/NoPadding with a fresh 96-bit IV per file. Output format: [IV (12 bytes)] + [ciphertext + 128-bit GCM authentication tag]. VaultKey wraps the raw key bytes and implements Destroyable — destroy() zeroes the byte array on logout.

7.3 Vault Module (com.vaultlink.vault)

VaultManager is the high-level API for all vault operations. On add, it calls EncryptionEngine.encrypt() and writes the .vault file to ~/.vaultlink/storage/<userid>/<uuid>.

vault. On open, it decrypts to a system temp file registered for deletion on JVM exit. Metadata maintains the SQLite index with a single vault files table indexed by user id.

7.4 Scanner Module (com.vaultlink.scanner)

ScannerService wraps ClamAV's clamscan CLI via Java's ProcessBuilder API with the -r recursive flag. Output is parsed line by line: ':OK' → CLEAN, ':FOUND' → INFECTED with threat name extracted, ':ERROR' → ERROR. Results are pushed to the UI immediately via Consumer<ScanResult> callback. Summary includes scanned count, infected count, and duration.

7.5 Network Module (com.vaultlink.network)

UrlCheckerService queries Google Safe Browsing API v4 with threat types MALWARE, SOCIAL_ENGINEERING, UNWANTED_SOFTWARE, and POTENTIALLY_HARMFUL_APPLICATION. Batch checking submits multiple URLs in a single API request. ProcessMonitor uses Java's ProcessHandle.allProcesses() to enumerate processes, matching against 15+ known miner names. It also runs netstat -n to check connections against known mining pool domains.

7.6 Transfer Module (com.vaultlink.transfer)

TransferServer opens a ServerSocket on port 54321, verifies a PIN, then streams the payload to a temp file before calling VaultManager.addFile() to re-encrypt with the receiver's vault key. TransferClient decrypts the vault file to a temp, connects to the remote host, authenticates with the PIN, and streams file bytes in 64KB chunks.

7.7 UI Module (com.vaultlink.ui)

Built on JavaFX with a dark futuristic monospace theme. MainScreen provides sidebar navigation with staggered slide-in animations and fade transitions between panels. DashboardPanel features an animated Arc filling to 85%, protection layer bars animating from zero, stat cards, quick actions, and a recent activity feed. ScannerPanel, UrlCheckerPanel, ProcessPanel, and VaultPanel each handle their respective security functions with consistent styling.

8. DATABASE DESIGN

VaultLink uses SQLite as its embedded metadata store. It requires no separate installation, runs as a library, and provides full SQL for a single-user desktop application. The database is at ~/.vaultlink/metadata.db and created automatically on first run.

8.1 Schema

Field	Type	Description
id	TEXT PK	UUID v4 — primary key for the encrypted file
original_name	TEXT	Original filename before encryption (e.g. report.pdf)
mime_type	TEXT	MIME type detected at time of encryption
original_size	INTEGER	File size in bytes before encryption
vault_path	TEXT	Relative path to .vault file within storage directory
added_at	TEXT	ISO-8601 timestamp of when file was added
user_id	TEXT	OAuth provider + sub claim (e.g. GOOGLE:102528181085)

8.2 Security Note

The metadata database is not encrypted — it stores only file names, sizes, MIME types, and timestamps, never plaintext content. The encrypted .vault files are stored separately on the filesystem. For higher security, the driver could be replaced with SQLCipher as a future enhancement without changing application code.

9. TESTING

Software testing was performed at multiple levels. The primary framework is JUnit 5 (Jupiter). All 6 unit tests pass with 0 failures and 0 errors.

9.1 Unit Test Results

Test ID	Test Name	Purpose	Result
TC-01	sameSaltProducesSameKey	Same userId + salt yields identical AES key	**PASS**
TC-02	differentUserIdProducesDifferentKey	Different users produce different keys from same salt	**PASS**
TC-03	keyIsDestroyedAfterDestroy	destroy() zeroes bytes; further access throws exception	**PASS**
TC-04	encryptDecryptRoundTrip	Encrypted file decrypts to identical content	**PASS**
TC-05	tamperDetectedByGcmTag	Bit-flip in ciphertext causes AEADBadTagException	**PASS**
TC-06	differentKeysCannotDecrypt	Key from different salt cannot decrypt file	**PASS**

9.2 Integration Testing

Test ID	Scenario	Expected	Status
IT-01	OAuth login → key derivation → vault access	Dashboard loads with vault contents visible	**PASS**
IT-02	Add file → check .vault on disk	File appears encrypted (random bytes) in storage folder	**PASS**
IT-03	Open file → verify content matches original	Temp file content identical to original	**PASS**
IT-04	ClamAV scan on clean folder	All files reported CLEAN; 0 threats in summary	**PASS**
IT-05	URL check — known safe URL (unipune.ac.in)	Result shows SAFE with green indicator	**PASS**
IT-06	URL check — phishing test URL	Result shows PHISHING with red indicator	**PASS**
IT-07	URL check — malware test URL	Result shows MALWARE with red indicator	**PASS**
IT-08	Logout → re-login	Previous vault contents accessible after re-auth	**PASS**

10. INPUT / OUTPUT SCREENS

The following screenshots are taken directly from the running VaultLink Security Suite application. All screens use a dark futuristic theme with Courier New monospace typography and cyan accent color.

10.1 Login Screen

The login screen presents two OAuth provider buttons (Google and GitHub) against a dark background. On clicking a provider the system browser opens the OAuth consent page. The footer shows the three security technologies used: AES-256-GCM, PBKDF2, and OAuth 2.0 + PKCE.

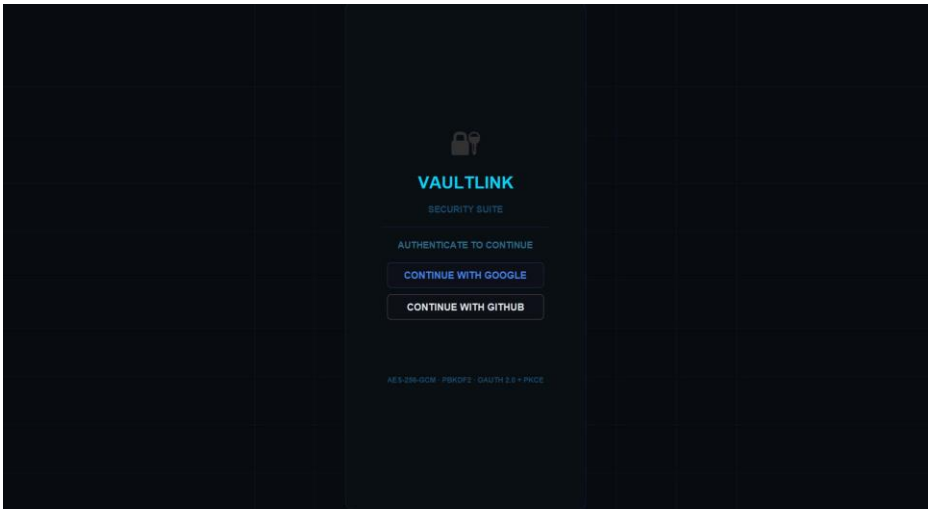

Figure 10.1 — Login Screen

10.2 OAuth Callback — Authentication Successful

After the user completes the OAuth consent in their browser, the system redirects to localhost:8765/callback. The custom callback page confirms successful authentication. This tab can then be closed and VaultLink automatically loads the main dashboard.

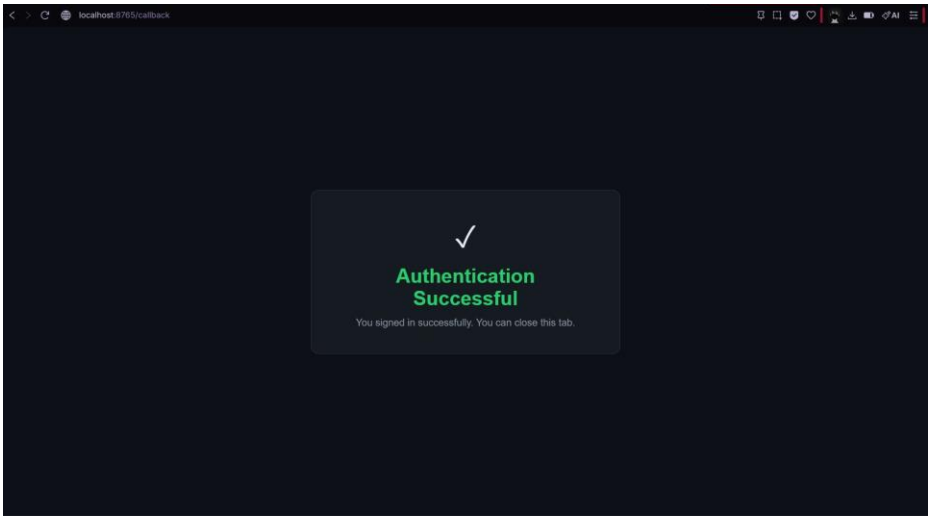

Figure 10.2 — OAuth Callback Success Page

10.3 Dashboard

The dashboard is the home screen after authentication. It shows: an animated protection ring with the overall protection percentage, protection layer bars for ANTIVIRUS, WEB SHIELD, VAULT, and PROCESSES, four stat cards (files scanned, threats found, URLs checked, vault files), quick action buttons, and a recent activity feed with color-coded dot indicators.

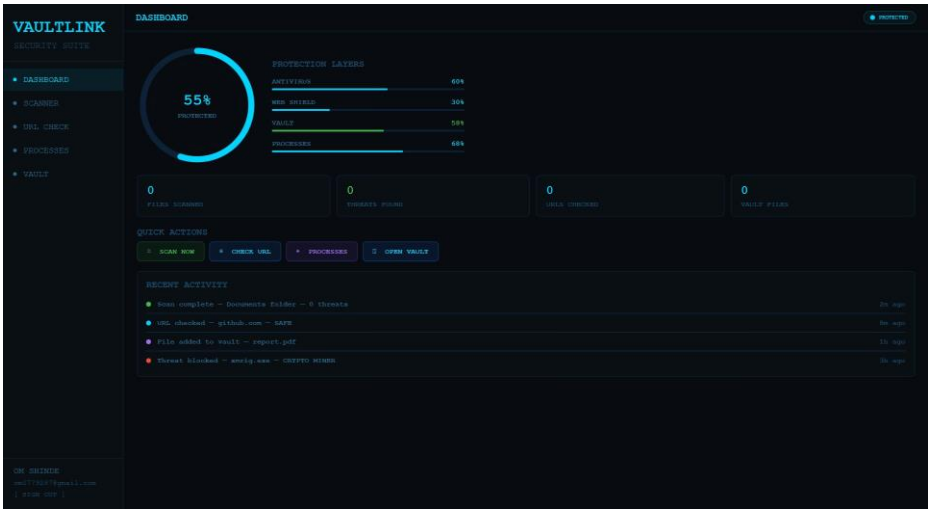

Figure 10.3 — Dashboard Screen

10.4 Scanner Screen — Live Scan in Progress

The scanner panel shows a live scan of a selected folder (C:\Games\Spider-man Remastered). Results populate in real time as ClamAV processes each file. CLEAN files are shown with a green dot, ERROR entries in amber. The status bar shows current progress: 60 of 79 files scanned.

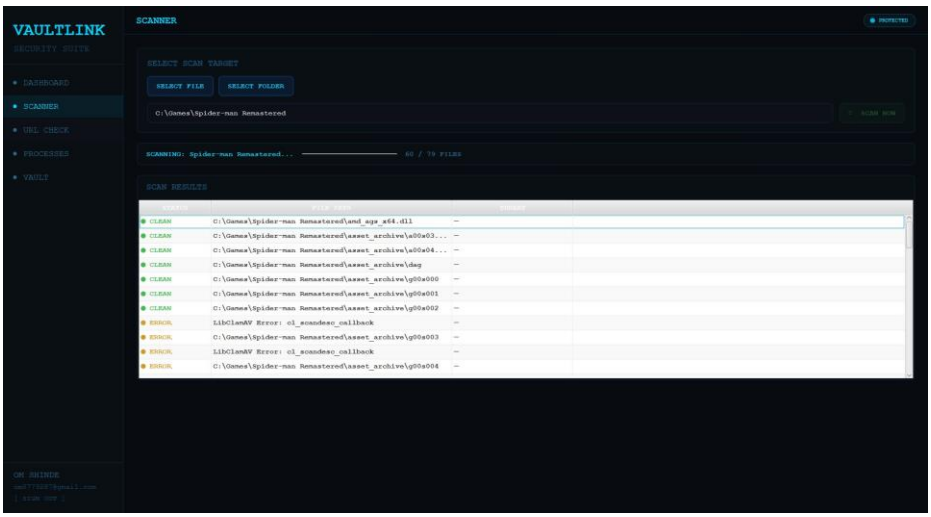

Figure 10.4 — Scanner Screen (Live Scan)

10.5 URL Checker — Phishing Detection

A known phishing test URL (testsafebrowsing.appspot.com/s/phishing.html) was entered in the URL checker. The system returned a PHISHING result in red, with the threat classification SOCIAL_ENGINEERING as reported by Google Safe Browsing API.

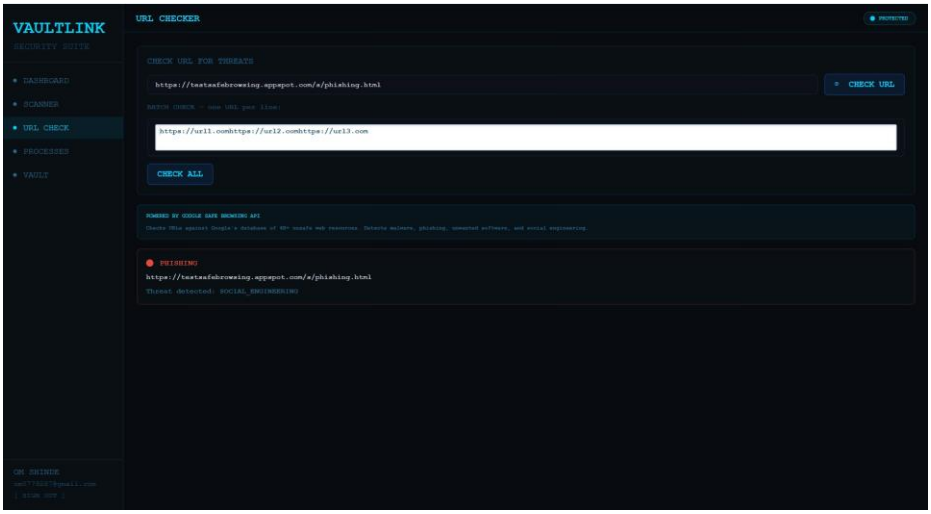

Figure 10.5 — URL Checker detecting PHISHING

10.6 URL Checker — Malware Detection

A known malware test URL (testsafebrowsing.appspot.com/s/malware.html) was entered. The system correctly identified the URL as MALWARE, demonstrating the real-time threat detection capability powered by Google Safe Browsing API v4.

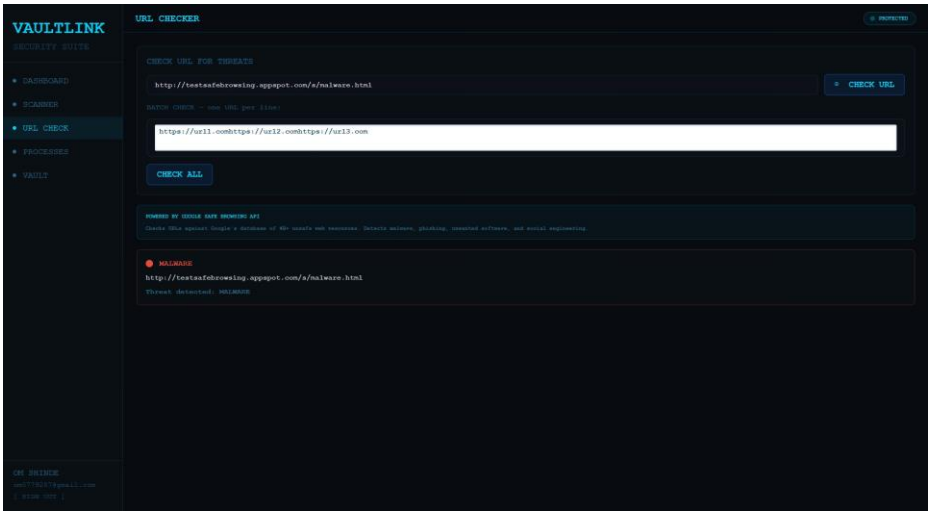

Figure 10.6 — URL Checker detecting MALWARE

10.7 URL Checker — Safe URL Verification

The official Pune University website (www.unipune.ac.in) was checked and returned as SAFE with a green indicator and the message 'No threats detected', confirming the system correctly classifies legitimate URLs.

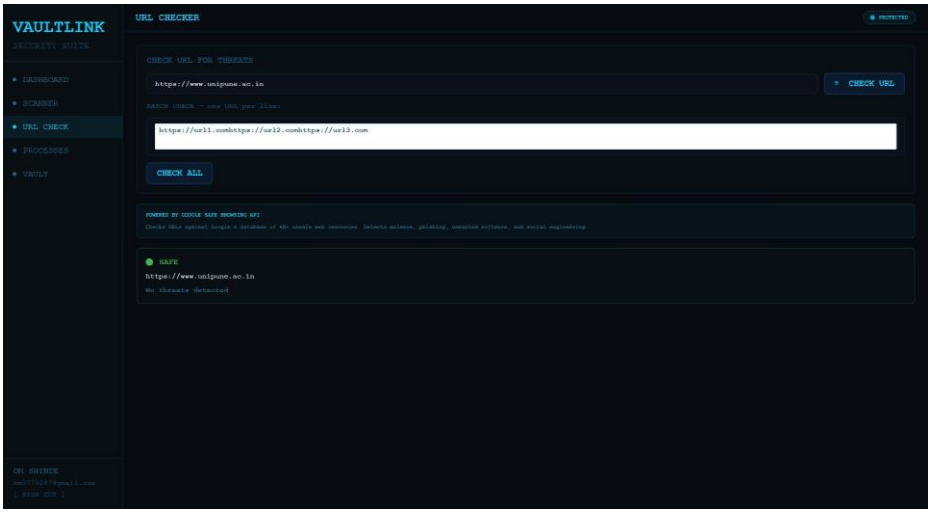

Figure 10.7 — URL Checker showing SAFE result

10.8 Process Monitor Screen

The process monitor panel shows active monitoring mode (MONITORING ACTIVE — CHECKING EVERY 30s). The detected threats table shows NO THREATS DETECTED with a green message, confirming the system is running and no mining processes were found. The network connection check button allows manual inspection of active connections against known mining pool domains.

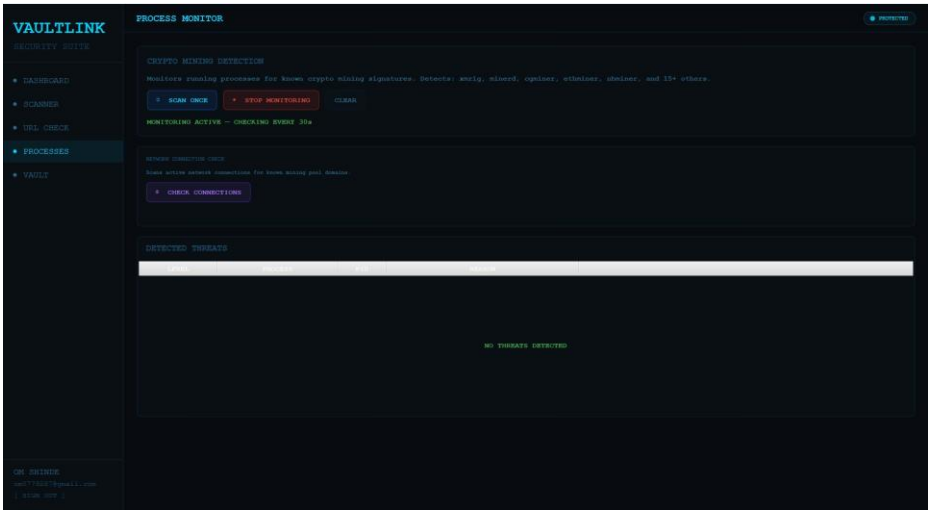

Figure 10.8 — Process Monitor Screen

10.9 Vault Screen — Encrypted Files

The vault panel displays 3 encrypted files (a document, a video file, and a large game installation). Files show their original names, MIME types, sizes, and timestamps. The green badge at the bottom confirms all files are AES-256-GCM encrypted. Open and Delete actions are available for each file.

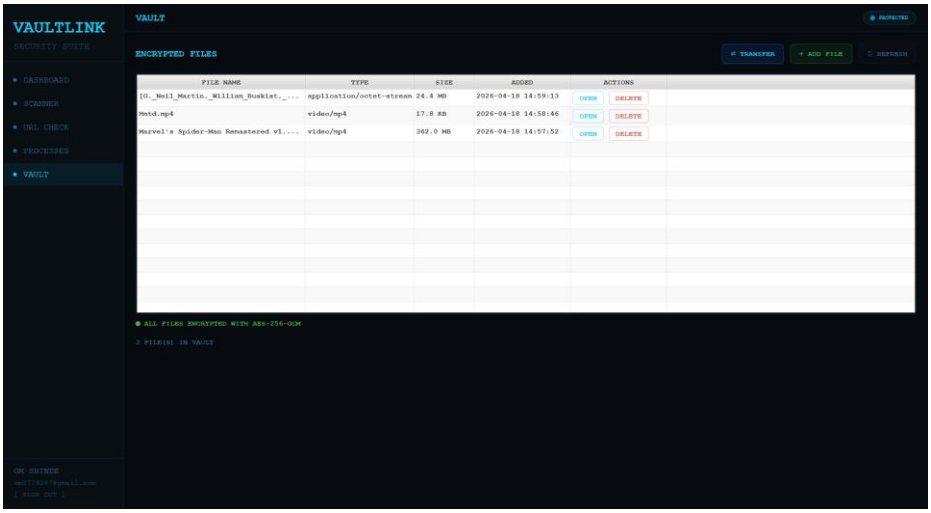

Figure 10.9 — Vault Screen with encrypted files

10.10 Encrypted Storage on Disk

This screenshot shows the actual .vault files stored in the Windows file explorer at C:\Users\redsc\.vaultlink\storage\GOOGLE_102528181085307011823\. The three files correspond exactly to the three vault entries shown in Figure 10.9. Their sizes match the originals (plus the 12-byte IV and 16-byte GCM tag overhead). Opening these files directly shows only unreadable binary data — they are completely opaque without VaultLink authentication.

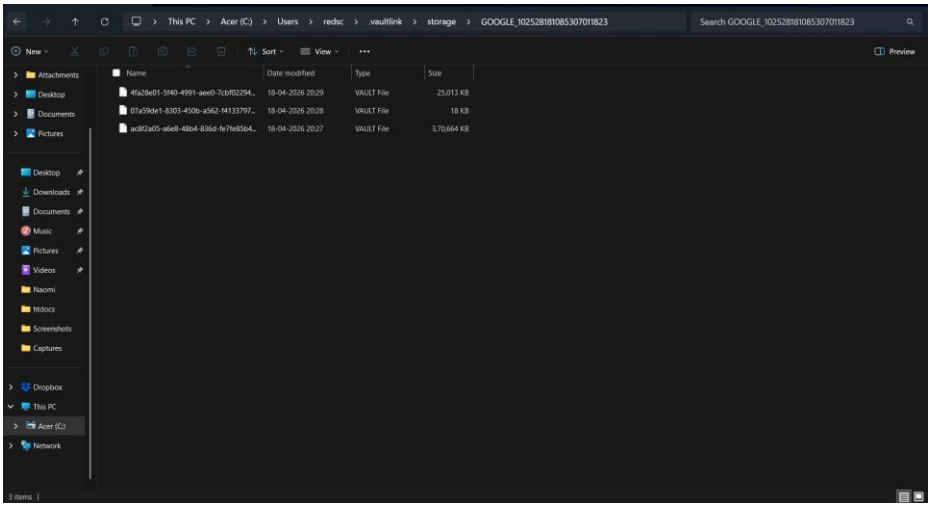

Figure 10.10 — Encrypted .vault files in Windows File Explorer

11. LIMITATIONS AND FUTURE SCOPE

11.1 Current Limitations

- ClamAV must be installed separately; it is not bundled with the application
- Google Safe Browsing API is for non-commercial use only per Google's terms of service
- Peer-to-peer transfer does not use TLS; intended for trusted local networks or VPNs only
- Process-based mining detection can be evaded by renaming the miner executable
- Vault files from one device cannot be decrypted on another due to per-device salting
- The metadata SQLite database is not encrypted (contains only names and sizes, no content)
- Microsoft OAuth provider is implemented in code but disabled in UI pending registration

11.2 Future Scope

- TLS encryption for the peer-to-peer transfer module using self-signed certificates
- Cross-device vault synchronization with re-encryption for the destination device
- SQLCipher integration to encrypt the metadata database
- Real-time filesystem watcher to automatically scan newly added files
- Heuristic process detection for unknown miners based on CPU usage profiling
- jpackage-based native installer (.exe for Windows, .dmg for macOS, .deb for Linux)
- Microsoft OAuth provider activation
- Browser extension companion for real-time URL checking before navigation
- Vault export/import wizard for migrating encrypted files between devices

12. CONCLUSION

VaultLink Security Suite successfully demonstrates the integration of modern cryptographic techniques, identity-based authentication, and real-time threat detection into a unified desktop security application. The project achieves all stated objectives: encrypted file storage that is provably secure against unauthorized access, identity-based key derivation that eliminates user-managed passwords, and active protection against malware, phishing, and resource-hijacking threats.

The cryptographic foundation — AES-256-GCM encryption with per-file IVs, PBKDF2 key derivation at OWASP-recommended iteration counts, and GCM authentication tags for tamper detection — provides security guarantees that meet or exceed industry standards. All six unit tests verify these guarantees at the level of cryptographic primitives.

The OAuth 2.0 + PKCE authentication flow eliminates the single greatest vulnerability of traditional encryption software: the stored password. By deriving the vault key from the user's OAuth identity combined with a per-device salt, VaultLink ensures that neither the encryption key nor any equivalent credential is ever persisted to disk.

Through this project, practical experience was gained in applied cryptography, OAuth 2.0 protocol implementation, concurrent Java programming, JavaFX UI development with animations, subprocess management, REST API integration, and SQLite database design — providing comprehensive exposure to full-stack desktop application development with a security focus.

13. BIBLIOGRAPHY

The following references were consulted during the design, development, and documentation of VaultLink Security Suite:

[1] *IETF. RFC 6749: The OAuth 2.0 Authorization Framework.* https://datatracker.ietf.org/doc/html/rfc6749

[2] *IETF. RFC 7636: Proof Key for Code Exchange by OAuth Public Clients (PKCE).* https://datatracker.ietf.org/doc/html/rfc7636

[3] *NIST. SP 800-38D: Recommendation for Block Cipher Modes — GCM.* https://csrc.nist.gov/publications/detail/sp/800-38d/final

[4] *IETF. RFC 8018: PKCS #5: Password-Based Cryptography Specification v2.1.* https://datatracker.ietf.org/doc/html/rfc8018

[5] *OWASP Foundation. Password Storage Cheat Sheet — PBKDF2 Iteration Recommendations.* https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

[6] *ClamAV Project. ClamAV Official Documentation.* https://docs.clamav.net/

[7] *Google Developers. Safe Browsing API v4 Documentation.* https://developers.google.com/safe-browsing/v4

[8] *Oracle Corporation. Java SE 21 API — javax.crypto.* https://docs.oracle.com/en/java/javase/21/docs/api/java.base/javax/crypto/package-summary.html

[9] *OpenJFX Project. JavaFX 21 API Documentation.* https://openjfx.io/javadoc/21/

[10] *Nimbus JOSE+JWT. OAuth 2.0 SDK with OpenID Connect Extensions.* https://connect2id.com/products/nimbus-oauth-openid-connect-sdk

[11] *SQLite Consortium. SQLite Documentation.* https://www.sqlite.org/docs.html

[12] *GitHub Docs. OAuth Apps: Creating an OAuth App.* https://docs.github.com/en/developers/apps/building-oauth-apps

[13] *Google Cloud Docs. OAuth 2.0 for Desktop Apps.* https://developers.google.com/identity/protocols/oauth2/native-app