VAULTIFY — COMPLETE SYSTEM OVERVIEW

A secure, multithreaded, blockchain-inspired credential vault built in pure Java.

WHAT VAULTIFY IS

Vaultify is a local-first, encrypted storage system for personal documents (IDs, certificates, records).

It provides:

- **Secure credential storage**
  Files are encrypted using AES + RSA hybrid encryption.
- **Asynchronous operations**
  All heavy tasks (encryption, ledger writing, decryption, logging) run in background threads.
- **Immutable audit history**
  Every vault operation creates a cryptographically linked "ledger block" (similar to a blockchain) ensuring tamper-evidence.
- **CLI-first interface**
  Users interact through a terminal dashboard — perfect for academic demonstrations and real-world usability.
- **JDBC persistence**
  Data is stored in PostgreSQL with a clean DAO layer.
- **Complete OOP architecture**
  Strong encapsulation, abstraction, polymorphism, inheritance, and modular design.

Vaultify demonstrates serious system design competence — far beyond typical student projects.

VAULTIFY ARCHITECTURE OVERVIEW

com.vaultify

- `cli/` → CLI UI & command handling
- `controller/` → (Optional) REST APIs
- `service/` → Business logic
- `model/` → Domain data classes
- `crypto/` → Encryption engines
- `dao/` → JDBC persistence

- `ledger/` → Immutable hash-linked ledger
- `thread/` → Concurrency layer
- `util/` → File I/O and helper utilities
- `config/` → Global configuration

SYSTEM LAYER BREAKDOWN

CLI LAYER (PRIMARY INTERFACE)

Text-based UI using Java Scanner.

Commands:

- register
- login
- add-credential
- list
- share
- verify
- verify-ledger
- exit

Non-blocking UX: encryption shows progress dots while running in background.

MODEL LAYER (CORE OOP ENTITIES)

A table describing core OOP entities:

Class	Purpose
User	Represents a vault owner (RSA keys, vault link)
Vault	Contains credentials
Credential	Encrypted file + metadata
ShareToken	Temporary access token
LedgerBlock	Blockchain-style record

Encapsulation + clear relationships:

- User → Vault (composition)
- Vault → Credentials (aggregation)

- LedgerBlock has hash-linked chain

CRYPTO LAYER (ABSTRACTION + POLYMORPHISM)

`public interface CryptoEngine { ... }`

Two implementations:

- `RSAEngine` → key encryption + signing
- `AESEngine` → data encryption

Vaultify uses a hybrid crypto model (real-world approach):

- AES encrypts file
- AES key encrypted by RSA public key

DAO LAYER (JDBC PERSISTENCE)

DAO pattern separates storage from business logic.

DAOs:

- UserDAO
- CredentialDAO
- TokenDAO
- LedgerDAO

Uses:

- JDBC
- PreparedStatements
- Transactions
- ConnectionManager

Database: PostgreSQL

SERVICE LAYER (BUSINESS LOGIC)

A table describing service responsibilities:

Service	Responsibilities
VaultService	Encrypt, store, retrieve, decrypt credentials
UserService	Manage registration + keys

Service	Responsibilities
VerificationService	Validate shared tokens
LedgerService	Append/verify ledger chain

This is where all modules integrate.

LEDGER LAYER (BLOCKCHAIN-INSPIRED)

Vaultify includes a local immutable ledger for audit logging.

Each operation creates a LedgerBlock:

Fields:

- prevHash
- currentHash (SHA-256)
- action
- credentialId
- timestamp

Blocks are linked like:

- block0.hash → block1.prevHash
- block1.hash → block2.prevHash
- ...

Ledger tamper is detectable via:

```
ledgerService.verifyIntegrity();
```

THREAD LAYER (CONCURRENCY BACKBONE)

Vaultify uses multi-threading everywhere appropriate:

Threaded Components:

Task	Thread Type	Purpose
EncryptionTask	Callable + ExecutorService	Async AES/RSA encryption
LedgerWriter	Runnable	Append new ledger block
ActivityLogger	Daemon Thread	Continuous logging

Task	Thread Type	Purpose
TokenExpiryScheduler	ScheduledExecutorService	Remove expired share tokens
VaultLoader	FixedThreadPool	Parallel decryption

ThreadManager centralizes all concurrency.

UTILITY LAYER

Contains reusable helpers:

- FileStorageUtil → encrypted file I/O
- QRCodeUtil → token QR generation
- LoggerUtil → logging
- JSONUtil → JSON serialization

SYSTEM FLOW (END-TO-END)

EXAMPLE: ADD CREDENTIAL FLOW

User → CLI → VaultService.addCredential()

- → ThreadManager.submit(EncryptionTask)
  → AES encrypt file
  → RSA encrypt AES key
  → Save binary file to disk
  → CredentialDAO.insert(...)
- → ThreadManager.submit(LedgerWriter)
  → Append block to ledger.json
- ActivityLogger logs event

CLI prints “Encrypted and Saved”

Everything non-critical runs asynchronously.

SECURITY MODEL

Vaultify applies:

- **Strong cryptography**
  RSA-2048 keys per user
  AES-256 for file-level encryption

SHA-256 ledger hashing
RSA digital signatures

- **Tamper detection**
  Ledger chain breaks if file is modified.

PERSISTENCE (DATABASE)

PostgreSQL tables:

- users
- credentials
- tokens
- ledger

This provides relational integrity and auditability.

TESTING

JUnit 5:

- Service layer tests
- DAO tests (with local DB)
- Ledger integrity test (tampering → detect failure)

CLI manual tests:

- Add / remove credentials
- Ledger verification command

WHAT VAULTIFY DEMONSTRATES ACADEMICALLY

A table mapping concepts to their appearance:

Concept	Where It Appears
OOP	Model + Service layers
Abstraction	CryptoEngine, DAO, Services
Polymorphism	AES vs RSA crypto engines
Encapsulation	Credential, User, Vault classes
JDBC	DAO layer
File I/O	Encrypted file storage & ledger

Concept	Where It Appears
Multi-threading	EncryptionTask, Logger, Scheduler
Daemon threads	ActivityLogger
Blockchain Concepts	LedgerBlock + Hash chain
CLI Interaction	CLI layer

This covers 100% of typical Java project evaluation criteria. But Vaultify is far more advanced than standard student submissions.

WHY VAULTIFY IS UNIQUE

Most students build:

- Inventory apps
- CRUD apps
- E-commerce clones
- Simple APIs

Vaultify stands out because:

- Uses real cryptography
- Implements multi-threading correctly
- Includes a blockchain-style ledger
- Offers both a CLI and backend services
- Has a clean, enterprise-style architecture
- Demonstrates real-world secure system design

This is the type of project that gets top grades, impresses interviewers, and becomes portfolio gold.

FINAL SUMMARY

Vaultify is a secure, multithreaded, blockchain-inspired credential vault built entirely in Java using strong OOP, cryptography, JDBC, file I/O, and concurrency patterns.

It is architected like a professional system, operates asynchronously, logs every action immutably, and provides a clean user experience via a CLI interface.