Slash Mark

esaip
École d'Ingénieurs La Salle

PAN ENCRYPTION & TOKENIZATION SERVICE

Name: Surya Hanuman KONJETI
Position: Cyber Security Intern
Company: Slash Mark IT Solutions (OPC) Pvt. Ltd**.**
Institution: ESAIP École d'Ingénieur
Technologies: FastAPI, Python, AES-GCM, SQLite, Uvicorn, Pydantic
Environment: VS Code, Windows OS

INDEX

1. INTRODUCTION

Financial institutions and digital payment platforms handle millions of customer Primary Account Numbers (PANs) every day. Because PANs are highly sensitive and regulated under PCI-DSS, storing them in plaintext exposes organizations to massive risk, including fraud, identity theft, and legal penalties.

This project implements a PAN Encryption & Tokenization Service—a secure backend system that transforms sensitive PAN data into encrypted, non-sensitive tokens. The service ensures:

- Confidentiality
- Integrity
- Controlled access
- Full regulatory compliance

The system uses AES-GCM, a modern authenticated encryption algorithm, and provides REST APIs for encrypting, decrypting, and retrieving token metadata.

This work demonstrates how financial-grade security mechanisms are implemented in real-world cybersecurity systems.

2. PURPOSE OF THE PROJECT

The primary purpose of this project is to provide a secure mechanism for storing and retrieving PAN data by converting it into a tokenized and encrypted form. The system allows only authorized administrators to decrypt PAN values.

The project aims to:

- Prevent data exposure in databases
- Protect financial information from cyberattacks
- Demonstrate tokenization workflow
- Provide a secure encryption engine
- Implement admin-level access control
- Showcase strong cryptographic practices

3. PROJECT OBJECTIVES

- Implement encryption using AES-GCM with Nonce & Ciphertext
- Generate secure tokens mapped to encrypted PANs
- Build FastAPI endpoints for handling encryption & decryption
- Store encrypted PAN data securely in an SQLite database
- Enforce admin-only access for sensitive operations
- Automatically generate masked PAN for safe display
- Provide API documentation via Swagger UI

4. SCOPE OF THE SYSTEM

In Scope

- PAN Encryption
- Token Generation
- Admin-only PAN Decryption
- Token Metadata Retrieval
- Secure database storage
- FastAPI-based backend

Out of Scope

- Multi-user authentication system
- Web UI front-end
- Cloud deployment
- Multi-key rotation system

5. SYSTEM ARCHITECTURE

The system consists of the following core layers:

1. Client Layer

- Swagger UI
- Postman
- CURL commands

2. API Layer (FastAPI)

- Handles routing
- Validates request bodies
- Verifies admin authentication
- Sends data to encryption engine

3. Encryption Engine

- AES-GCM encryption
- Generates nonce
- Authenticated decryption
- Masked PAN generation

4. Database Layer

- SQLite for secure token vault storage
- Stores Token • Ciphertext • Nonce • PAN metadata

6. WORKING FLOW OF THE SYSTEM

Encryption Flow

1. User submits a PAN
2. System validates PAN format
3. AES-GCM encryption applied
4. Token generated
5. Token + encrypted data stored in SQLite

6. Masked PAN returned as output

Decryption Flow

1. Admin provides token + API key
2. System validates admin privileges
3. Fetches encrypted payload
4. AES-GCM decryption applied
5. PAN returned & masked

Metadata Flow

1. User provides token
2. System returns token metadata (masked PAN, timestamp)

7. TECHNOLOGIES USED

Component	**Description**
Python	Core programming language
FastAPI	API framework
AES-GCM	Modern authenticated encryption
SQLite	Token storage database
Uvicorn	ASGI web server
Pydantic	Request validation
VS Code	Development environment
Windows OS	Execution platform

8. FUNCTIONAL DESCRIPTION

The system performs the following critical functions:

- Encrypts PAN into secure ciphertext
- Generates unique tokens
- Stores secure records with metadata

- Allows admin to decrypt PAN
- Provides token metadata lookup
- Implements detailed error handling
- Logs API operations

9. SECURITY MECHANISMS IMPLEMENTED

- AES-GCM Encryption (Authenticated Encryption)
- Secure Nonce generation
- Admin API Key authentication
- Masked PAN output to avoid exposure
- Input validation using Pydantic
- Local token vault with restricted access
- Prevention of replay and tampering attacks

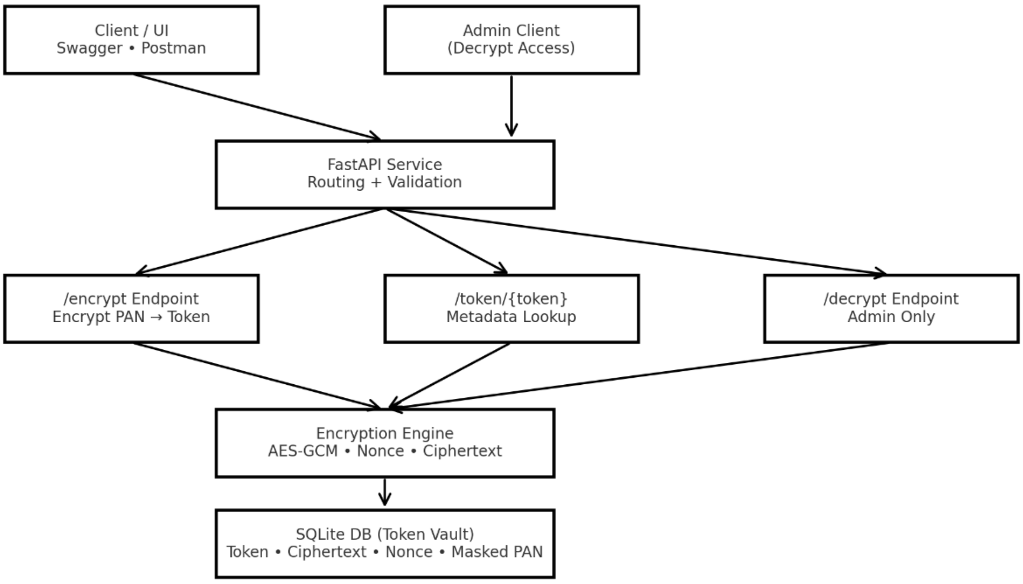
---

10. API ENDPOINTS OVERVIEW

Endpoint	Method	Description
/encrypt	POST	Encrypt PAN → Return token + masked PAN
/decrypt	POST	Admin-only decryption of PAN
/token/{token}	GET	Retrieve metadata
/health	GET	System health status
/docs	GET	Swagger documentation

Slash Mark

11. ARCHITECTURE DIAGRAM & EXPLANATION

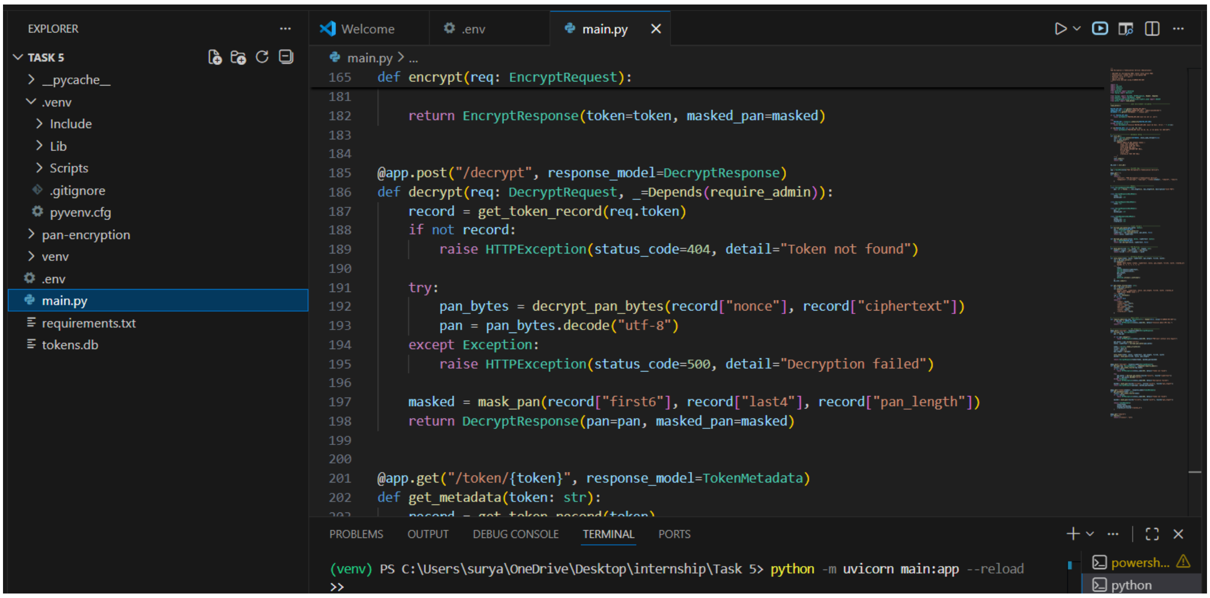


Explanation:

- Clients interact with the FastAPI service
- FastAPI routes requests to encryption, token lookup, or decryption modules
- The encryption engine performs AES-GCM operations
- SQLite token vault stores secure data
- Admin clients use API keys for restricted decryption

12. SCREENSHOTS & EXPLANATION

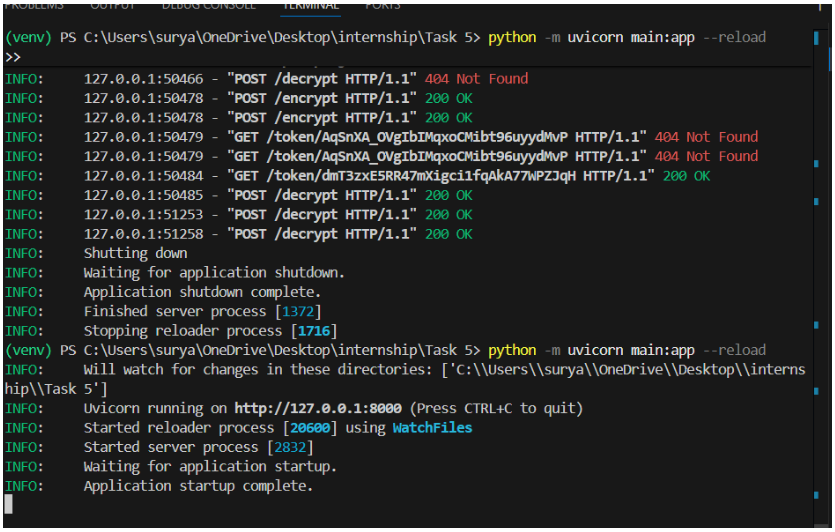
Screenshot 1 — VS Code Project View



Shows the full backend structure:

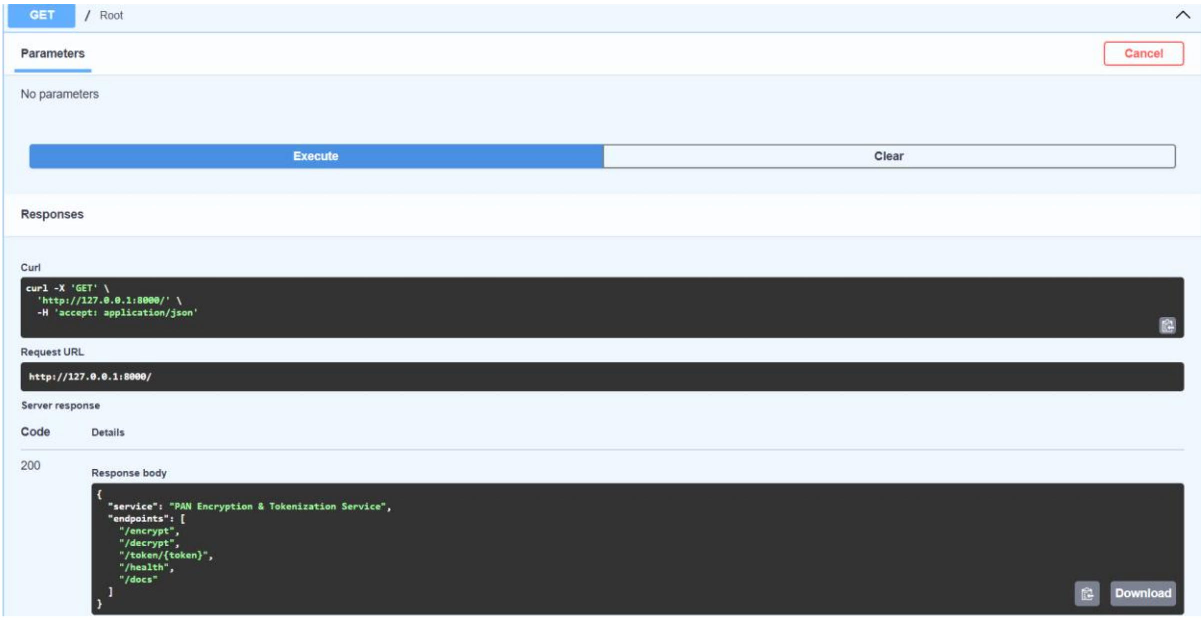
- main.py
- .env
- tokens.db
- Dependencies

Screenshot 2 — Running Uvicorn Server



Shows successful API execution and live endpoints.

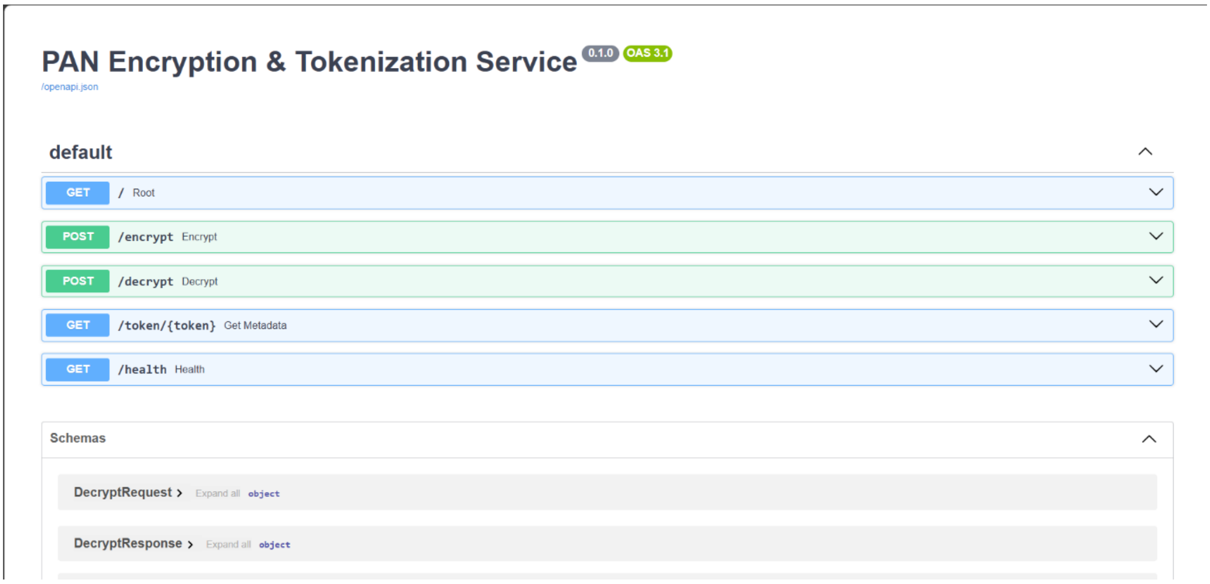
Screenshot 3 — Root Endpoint Response



This output screen displays the response of the Root (/) API endpoint of the PAN Encryption & Tokenization Service. When the endpoint is executed through Swagger UI, the server returns a successful HTTP 200 OK response, confirming that the application is running correctly. The response contains the name of the service and a list of available API endpoints supported by the system, such as encryption, decryption, token retrieval, health check, and documentation access. No input parameters are required for this endpoint, and no sensitive information is exposed. This screen is mainly used to verify service availability and to provide an overview of the system's API structure.

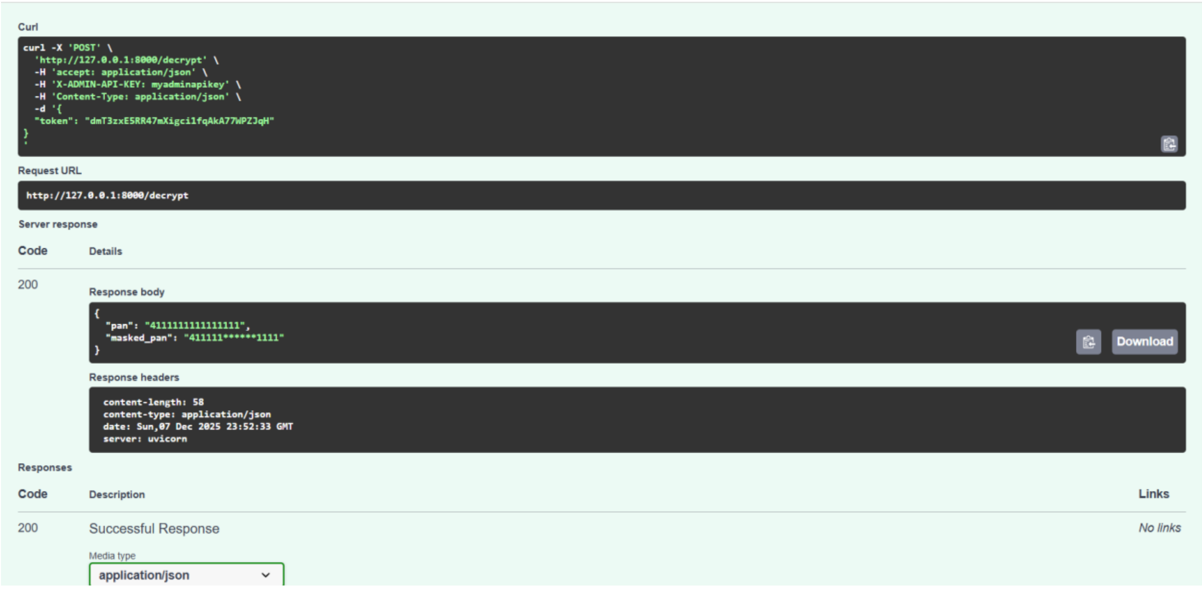
Screenshot 4 — Swagger UI Documentation

Shows all interactive API routes (/encrypt, /decrypt, /token/...).



This output screen displays the Swagger API documentation page of the PAN Encryption & Tokenization Service. It shows the list of all available API endpoints provided by the system along with their HTTP methods. The screen includes endpoints for root access, PAN encryption, PAN decryption, token metadata retrieval, and health checking. Each endpoint can be expanded to view request and response details. This interface helps users test the APIs easily and understand the functionality of the system.

Screenshot 5 — Successful Decryption Output



This output screen shows the result of the Decrypt (/decrypt) API. A valid token is provided in the request body along with the required admin API key in the request header. The system verifies the admin access and retrieves the encrypted PAN associated with the token. After successful verification, the PAN is decrypted securely and returned in the response along with its masked form. The response confirms that only authorized users can access the original PAN and that the decryption process works correctly.

13. CHALLENGES FACED & SOLUTIONS

Challenge	Solution
Incorrect PAN input	Added strict digit validation
Invalid admin access	Implemented API key authentication
Token not found errors	Added detailed exceptions
Decryption failures	Enhanced AES-GCM error handling
Database corruption	Implemented safe DB writes

14. RESULTS & OUTCOMES

- Successfully built secure tokenization backend
- Demonstrated AES-GCM usage in real applications
- Achieved full separation of client and admin access
- Implemented professional API documentation
- Ensured secure storage of sensitive financial data

15. FUTURE ENHANCEMENTS

- Add user authentication system
- Implement key rotation methodology
- Deploy system on cloud with secrets manager
- Add audit logging & monitoring
- Build front-end dashboard for admins

16. CONCLUSION

The ***PAN Encryption & Tokenization Service*** provides a robust and secure solution for protecting sensitive financial data. By combining modern cryptographic techniques, strong access control, and API-driven architecture, the system demonstrates real-world cybersecurity implementation suitable for enterprise environments.

This project enhances practical understanding of encryption, API security, and secure database handling—valuable skills for professional cybersecurity engineering.