DOCUMENTATION
PYTHON DEVELOPMENT
WEEK-4

PROJECT NAME: Real-Time FinTech Fraud Detection System (SentinelStream)
Week 4 – Finalization, Security & Deployment
Organization -Zaalima Development Pvt. Ltd.

PITCHE ESHWAR-DOCUMENTATION.worked in testing and research.

AKMAL-Used docker as container for whole project.
Implemented json web tokens

SIVANANDANA-Prepared and executed test cases to validate overall system functionality

1. Problem Statement:

After implementing core transaction processing and the intelligence layer, the system must be prepared for real-world deployment. A production-ready FinTech system requires secure APIs, containerized services, proper testing, and protection against common vulnerabilities.

Without deployment readiness, security hardening, and quality assurance, even a well-designed fraud detection system cannot be safely used in real environments.

2. Objective:

The objective of Week 4 is to finalize the system for production readiness by:

Containerizing services for consistent deployment

Implementing security mechanisms

Performing vulnerability checks

Ensuring high code quality and test coverage

Preparing the system for final presentation and evaluation

3. Deployment Architecture Overview:

Week 4 focuses on transforming SentinelStream into a deployable, secure, and maintainable system using industry-standard deployment practices.

Key Focus Areas:

- Containerization
- Reverse proxy & load handling
- API security
- Testing & quality assurance

4. Containerization Using Docker:

4.1 Docker Implementation

Containerized the FastAPI backend using Docker

Created Dockerfiles for consistent runtime environments

Ensured portability across development and deployment systems

4.2 Benefits

Environment consistency

Easier deployment

Simplified scaling

5. Reverse Proxy & Load Distribution:

5.1 Nginx Configuration

Configured Nginx as a reverse proxy

Enabled:

- Request routing
- Load distribution
- SSL termination

5.2 Outcome

- Improved request handling
- Enhanced security at the network layer
- Better performance under concurrent access

6. Security Implementation:

6.1 JWT Authentication

- Implemented JWT (JSON Web Token) based authentication
- Secured API endpoints
- Ensured only authorized clients can access sensitive operations

6.2 API Security Measures

- Input validation using Pydantic
- Protection against:
- SQL Injection
- Unauthorized access
- Token misuse

7. Testing & Quality Assurance:

7.1 PyTest Implementation

Wrote comprehensive PyTest test cases

Covered:

- API endpoints
- Rule Engine
- ML scoring logic

7.2 Code Coverage

- Achieved code coverage above 80%
- Ensured reliability and maintainability of the system

8. Security Audit & Vulnerability Checks:

8.1 Audit Focus

Checked for common vulnerabilities:

- SQL Injection
- Improper authentication
- Insecure endpoints

8.2 Outcome

- All critical vulnerabilities addressed
- System verified for secure operation

9. Performance Validation:

Verified that system performance remains stable after:

- Containerization
- Security layers
- Confirmed that latency requirements remain within acceptable limits

10. Technologies Used (Week 4):

- Docker – Containerization
- Nginx – Reverse proxy and load handling
- JWT – API authentication
- PyTest – Testing framework
- FastAPI – Backend framework

11. Deliverables:

- Dockerized backend services
- Nginx reverse proxy configuration
- JWT-secured APIs
- PyTest test suite with >80% coverage
- Security audit results
- Final system demonstration readiness

12. Challenges & Mitigation:

Challenges:

- Securing APIs without affecting performance
- Managing multiple deployment components
- Ensuring test coverage for all critical modules

Mitigation:

- Lightweight JWT authentication
- Modular Docker setup
- Automated testing with PyTest

13. Learning Outcomes:

By completing Week 4:

- Learned real-world deployment practices
- Understood API security mechanisms
- Gained experience with Docker and Nginx
- Implemented professional testing and quality assurance
- Prepared a system for production-level usage

14. Conclusion:

Week 4 successfully finalized SentinelStream as a secure, deployable, and production-ready fraud detection system. With containerization, security hardening, extensive testing, and performance validation completed, the project is now ready for final evaluation and real-world deployment scenarios.