ACMS Information Security Policy

Document Version: 1.0

Last Updated: December 22, 2025

Classification: Internal

---

1. Overview

This document describes the information security policies and procedures for ACMS (Adaptive Context Memory System), a personal financial management and AI assistant application.

Application Type: Personal use, single-user desktop application

Data Scope: Personal financial data, emails, chat history

Deployment: Local desktop with containerized backend services

---

2. Information Security Policy2.1 Security Objectives

ACMS is designed with the following security objectives:

1. **Confidentiality**: Protect sensitive financial data from unauthorized access
2. **Integrity**: Ensure data accuracy and prevent unauthorized modification
3. **Availability**: Maintain reliable access to personal data
4. **Privacy**: Prevent sensitive data from leaving the local environment

2.2 Risk Management

Security risks are identified and mitigated through:

- Regular code review for security vulnerabilities
- Encryption of sensitive data at rest
- Network isolation for financial data processing
- Separation of concerns between AI processing and financial data

---

3. Data Classification

Classification	Description	Examples
LOCAL_ONLY	Never leaves local system	Financial amounts, account numbers
CONFIDENTIAL	Encrypted, restricted access	OAuth tokens, API keys
INTERNAL	Standard protection	Chat history, preferences

PUBLIC	No restrictions	App metadata, documentation

4. Technical Security Controls4.1 Encryption at Rest

All sensitive financial data is encrypted before storage:

- **Algorithm:** Fernet (AES-256-CBC with HMAC-SHA256)
- **Key Management:** Environment variable with secure generation
- **Encrypted Fields:**
  - Plaid access tokens
  - Dollar amounts (market values, cost basis, transaction amounts)
  - Account identifiers

Implementation:
`src/integrations/plaid/`
,
`sync_service.py`
4.2 LLM Data Boundary

Financial data is explicitly excluded from external AI processing:

- Dollar amounts NEVER sent to external LLMs (Claude, GPT, Gemini)
- Financial queries processed with rule-based logic only
- Privacy level enforcement at query gateway layer

Implementation:
`src/gateway/orchestrator.py`

- privacy filtering4.3 Authentication & Access Control
- **OAuth 2.0** for third-party integrations (Plaid, Gmail)
- **Token encryption** before database storage
- **Single-user model** - no multi-tenant access concerns
- **Local deployment** - no network exposure of sensitive endpoints

4.4 Network Security

- Application runs on localhost only (ports 40080, 40432, etc.)
- No external network listeners for financial data
- CORS restricted to local origins only
- Docker network isolation between services

4.5 Database Security

- PostgreSQL with authentication required
- Credentials stored in environment variables
- Database runs in isolated Docker container
- No external port exposure in production mode

---

5. Secure Development Practices5.1 Code Security

- Input validation on all API endpoints
- Parameterized queries to prevent SQL injection
- No dynamic code execution from user input
- Dependencies regularly reviewed for vulnerabilities

5.2 Secret Management

- API keys and secrets stored in environment variables
- `.env`
- files excluded from version control (
- `.gitignore`
-)
- No secrets committed to source code
- Encryption keys generated with cryptographically secure methods

5.3 Audit Logging

All data access operations are logged:

- Data source access (Plaid API calls)
- Data transformations
- Privacy level changes
- Error conditions

Implementation:
`src/audit/logger.py`

---

6. Incident Response6.1 Detection

- Application logging monitors for:
    - Authentication failures
    - API errors
    - Unexpected data access patterns

6.2 Response Procedures

1. **Identify** - Review logs to understand scope
2. **Contain** - Revoke affected tokens/credentials
3. **Eradicate** - Update code, rotate secrets
4. **Recover** - Restore from backup if needed
5. **Document** - Record incident and remediation

6.3 Token Revocation

If credentials are compromised:

1. Revoke Plaid access via API (
2. `/api/plaid/disconnect`
3.)
4. Rotate
5. `PLAID_ENCRYPTION_KEY`
6. environment variable
7. Re-encrypt affected data with new key
8. Generate new Plaid connection

---

7. Data Retention

Data Type	Retention Period	Deletion Method
Financial transactions	Indefinite (user-controlled)	Manual deletion
Position snapshots	Indefinite (historical tracking)	Manual deletion
OAuth tokens	Until revoked	Secure deletion on disconnect
Audit logs	90 days	Automatic rotation

---

8. Third-Party Security8.1 Plaid Integration

- Uses Plaid's secure Link flow (no credential handling)
- Access tokens encrypted immediately after exchange
- Webhook validation for incoming notifications
- Regular sync with error monitoring

8.2 Dependency Security

- Python dependencies pinned to specific versions
- Regular review of security advisories
- No unnecessary dependencies included

---

9. Physical Security

- Application runs on personal hardware
- Full-disk encryption recommended for host machine
- Screen lock and user authentication on host OS

---

10. Compliance Considerations

This application is for **personal use only** and processes only the owner's financial data. As a personal finance tool:

- No customer data is processed
- No data is shared with third parties (except Plaid for data retrieval)
- User maintains full control and ownership of their data

---

11. Policy Review

This security policy is reviewed and updated:

- When significant changes are made to the application
- When new integrations are added
- At minimum annually

---

12. Contact

Application Owner: Rajan Yadav/rajan.conch@gmail.com

Last Security Review: December 2025