Day 2: Secure Memory & Context Systems - Implementation Guide

Quick Setup Commands

```bash
# 1. Run the implementation script
chmod +x secure-memory-agent/scripts/start.sh
cd secure-memory-agent
./scripts/start.sh
```

Manual Setup (Alternative)

Backend Setup

1. **Create Virtual Environment**

```bash
python3.12 -m venv venv
source venv/bin/activate  # On Windows: venv|Scripts|activate
```

2. **Install Dependencies**

```bash
cd backend
pip install -r requirements.txt
python -m spacy download en_core_web_sm
```

3. **Configure Environment**

```bash
# Edit .env file with your settings
cp .env.example .env
# Add your Gemini API key: GEMINI_API_KEY=your_api_key_here
```

4. **Start Backend Server**

```bash
uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

Frontend Setup

1. **Install Node Dependencies**

```bash
cd frontend
npm install
```

2. **Start Development Server**

```bash
npm run dev
```

Verification Steps

1. Health Check

```bash
curl http://localhost:8000/health
# Expected: {"status": "healthy"}
```

2. Database Connection

```bash
curl http://localhost:8000/api/security/health-check
# Expected: All checks should return true
```

3. Frontend Access

- Open http://localhost:3000
- Verify dashboard loads with metrics
- Navigate between pages

Testing the System

1. Create Test Conversation

```bash
curl -X POST "http://localhost:8000/api/memory/conversations?user_id=test-user&title=Demo" \
  -H "Content-Type: application/json"
```

2. Send Message with PII

```bash
curl -X POST "http://localhost:8000/api/memory/messages" \
  -H "Content-Type: application/json" \
  -d '{
    "content": "My email is john@example.com and my SSN is 123-45-6789",
    "role": "user",
    "conversation_id": "CONVERSATION_ID_FROM_STEP_1"
  }'
```

3. Test PII Detection

```bash
curl -X POST "http://localhost:8000/api/security/pii-analysis" \
  -H "Content-Type: application/json" \
  -d '{"text": "Contact me at jane@company.com or 555-123-4567"}'
```

4. Test Encryption

```bash
curl -X POST "http://localhost:8000/api/security/encryption-test" \
  -H "Content-Type: application/json" \
  -d '{
    "text": "Sensitive information here",
    "conversation_id": "test-conversation-123"
  }'
```

Expected Results

Dashboard Metrics

- **Conversations**: Shows total and active counts
- **Security**: Displays PII detection events
- **Performance**: Token usage statistics
- **Charts**: PII vs clean message ratios

PII Detection Output

```json
```

```
{
  "has_pii": true,
  "classification": {
    "high_sensitivity": [{"type": "ssn", "count": 1}],
    "medium_sensitivity": [{"type": "email", "count": 1}]
  },
  "redacted_text": "My email is john@example.com and my SSN is [REDACTED]",
  "confidence_score": 0.95
}
```


Encryption Test Output


```
json

{
  "original": "Sensitive information here",
  "encrypted": "gAAAAABh...encoded_data...",
  "decrypted": "Sensitive information here",
  "success": true
}
```


Testing with Docker (Optional)

1. Build Docker Images

```
bash

# Backend
docker build -t secure-memory-backend ./backend

# Frontend
docker build -t secure-memory-frontend ./frontend
```

2. Run with Docker Compose

```
bash

docker-compose up -d
```

3. Verify Services

```
bash

docker-compose ps
# All services should show "Up" status
```

Performance Testing

1. Load Test Messages

```bash
# Create 100 test messages
for i in {1..100}; do
  curl -X POST "http://localhost:8000/api/memory/messages" \
    -H "Content-Type: application/json" \
    -d "{\"content\": \"Test message $i\", \"role\": \"user\", \"conversation_id\": \"test-conv\"}"
done
```

2. Test Context Optimization

```bash
curl "http://localhost:8000/api/memory/conversations/test-conv/context?max_tokens=1000"
# Should return optimized context under token limit
```

Troubleshooting

Common Issues

Backend won't start:

- Check Python version (3.12+ required)
- Verify all dependencies installed
- Check port 8000 availability

Frontend build fails:

- Ensure Node.js 18+ installed
- Clear npm cache: `npm cache clean --force`
- Delete node_modules and reinstall

Database errors:

- Install SQLCipher: `pip install pysqlcipher3`
- Check file permissions in project directory
- Verify encryption key format (32 characters)

PII detection not working:

- Download spaCy model: `python -m spacy download en_core_web_sm`

- Check model path in configuration
- Verify regex patterns in PIIService

Debug Commands

```bash
# Check backend logs
tail -f backend/logs/app.log

# Test database connection
python -c "from backend.app.db.database import init_db; import asyncio; asyncio.run(init_db())"

# Verify encryption setup
python -c "from backend.services.encryption_service import EncryptionService; e=EncryptionService(); print('OK
```

Success Criteria Verification

✅ **Encrypted Storage**: Messages stored encrypted in SQLite
✅ **PII Detection**: Identifies emails, phones, SSNs with 95%+ accuracy
✅ **Context Optimization**: Reduces token usage by 40-60%
✅ **Audit Logging**: All security events logged with timestamps
✅ **Dashboard**: Real-time metrics and security monitoring
✅ **API Integration**: All endpoints respond correctly

Next Steps

1. **Extend PII Detection**: Add custom organizational patterns
2. **Performance Optimization**: Implement connection pooling
3. **Security Enhancement**: Add API rate limiting
4. **Integration**: Connect with Day 3 tool security systems

Assignment Solution Hints

Custom PII Pattern Extension:

```python
# Add to PIIService patterns
"employee_id": re.compile(r'\bEMP-\d{6}\b'),
"project_code": re.compile(r'\bPROJ-[A-Z]{2}-\d{4}\b')
```

Confidence Scoring Implementation:

```python
```

```
def calculate_pattern_confidence(self, matches, pattern_type):
    base_confidence = 0.8
    length_bonus = min(len(matches) * 0.1, 0.2)
    return min(base_confidence + length_bonus, 1.0)
```