# **Sentiment Analysis API & Dashboard**

## **Technical Documentation**

## **Table of Contents**

1. Setup Guide
2. API Reference
3. Usage Examples
4. Security
5. Troubleshooting
6. Best Practices

## **Setup Guide**

### **Prerequisites**

* Python 3.8+
* Node.js 14+
* Docker and Docker Compose (optional)
* Git

### **Local Development Setup**

## **Backend Setup**

***# Clone repository***

**git clone https://github.com/harshavardhan-9/sentimentAI**

**cd sentiment-analysis/backend**

***# Create virtual environment***

**python -m venv venv**

**source venv/bin/activate *# Windows: venv\Scripts\activate***

***# Install dependencies***

**pip install -r requirements.txt**

***# Set environment variables***

**export SECRET\_KEY="your-secret-key"**

**export CORS\_ORIGINS="http://localhost:3000"**

***# Run migrations***

**python manage.py migrate**

***# Start server***

**uvicorn app.main:app --reload --port 8000**

## **Frontend Setup**

**cd sentiment-analysis/frontend**

***# Install dependencies***

**npm install**

***# Set environment variables***

**echo "REACT\_APP\_API\_URL=http://localhost:8000" > .env**

***# Start development server***

**npm start**

### **Docker Deployment**

***# Build and run containers***

**docker-compose up --build**

***# Run in detached mode***

**docker-compose up -d**

***# View logs***

docker-compose logs -f

## **API Reference**

### **Authentication**

## **Get Access Token**

**POST /token**

**Content-Type: application/x-www-form-urlencoded**

**username=admin&password=password**

Response:

**{**

**"access\_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",**

**"token\_type": "bearer"**

**}**

## **Analyze Sentiment**

**POST /analyze**

**Authorization: Bearer <your\_token>**

**Content-Type: multipart/form-data**

file=@your\_file.csv

Response:

**{**

**"results": [**

**{**

**"id": 1,**

**"text": "Great product!",**

**"sentiment": "positive",**

**"polarity": 0.8,**

**"timestamp": "2024-01-01T10:00:00"**

**}**

**],**

**"summary": {**

**"positive": 10,**

**"neutral": 5,**

**"negative": 2**

**}**

**}**

### **Error Responses**

**{**

**"detail": "Error message",**

**"status\_code": 400,**

**"error\_type": "ValidationError"**

**}**

## **Usage Examples**

### **CSV File Format**

**id,text,timestamp**

**1,"The product exceeded my expectations!",2024-01-01 10:00:00**

**2,"Service was average, could be better",2024-01-01 11:30:00**

**3,"Terrible experience, never again",2024-01-01 12:45:00**

### **cURL Examples**

***# Get token***

**curl -X POST http://localhost:8000/token \**

**-d "username=admin&password=password"**

***# Upload file for analysis***

**curl -X POST http://localhost:8000/analyze \**

**-H "Authorization: Bearer YOUR\_TOKEN" \**

**-F "file=@data.csv"**

### **Python Client Example**

**import requests**

**def analyze\_sentiment(file\_path, token):**

**headers = {'Authorization': f'Bearer {token}'}**

**files = {'file': open(file\_path, 'rb')}**

**response = requests.post(**

**'http://localhost:8000/analyze',**

**headers=headers,**

**files=files**

**)**

**return response.json()**

## **Security**

### **Authentication**

* JWT tokens expire after 30 minutes
* Refresh tokens not implemented - re-authenticate after expiry
* CORS configured for frontend origin only

### **File Upload Security**

* Maximum file size: 10MB
* Allowed formats: CSV only
* File content validation before processing

### **API Rate Limiting**

***# Configuration in main.py***

**RATE\_LIMIT = "100/minute"**

**RATE\_LIMIT\_BYPASS\_TOKENS = ["special-token"]**

## **Troubleshooting**

### **Common Issues**

1. Token Issues

**{**

**"detail": "Could not validate credentials",**

**"status\_code": 401**

**}**

Solution: Token expired - request new token

1. File Upload Issues

**{**

**"detail": "Invalid CSV format",**

**"status\_code": 400**

**}**

Solution: Verify CSV follows required format

1. CORS Issues

**Access-Control-Allow-Origin error**

Solution: **Add frontend origin to CORS\_ORIGINS**

## **Best Practices**

### **Data Preparation**

1. Clean text data:
   * Remove special characters
   * Normalize whitespace
   * Handle encoding issues
2. CSV Guidelines:
   * UTF-8 encoding
   * No missing values in required fields
   * Maximum 1000 rows per file

### **API Usage**

1. Token Management:
   * Store securely
   * Refresh before expiry
   * Don't share between users
2. Error Handling:

**try:**

**response = analyze\_sentiment(file\_path, token)**

**except requests.exceptions.RequestException as e:**

**logger.error(f"API Error: {e}")**

***# Implement retry logic***

1. Rate Limiting:
   * Implement exponential backoff
   * Cache results when possible
   * Batch requests when possible

### **Performance Optimization**

1. File Processing:
   * Chunk large files
   * Use streaming for uploads
   * Compress when possible
2. Client-Side:
   * Implement request caching
   * Use debouncing for user input
   * Optimize bundle size

## **Monitoring**

### **Logging**

***# Backend logging configuration***

**logging.config.dictConfig({**

**'version': 1,**

**'handlers': {**

**'console': {**

**'class': 'logging.StreamHandler',**

**'formatter': 'standard'**

**}**

**}**

**})**

### **Metrics**

* API response times
* Error rates
* File processing times
* Authentication success/failure rates

## **Development Workflow**

### **Local Development**

1. Start backend:

**uvicorn app.main:app --reload**

1. Start frontend:

**npm start**

1. Run tests:

***# Backend tests***

**pytest**

***# Frontend tests***

**npm test**

### **Code Quality**

***# Backend linting***

**flake8 .**

**black .**

***# Frontend linting***

**npm run lint**