# **SentimentPulse Deployment Manual**

## **Prerequisites**

### **Required Software**

* **Node.js**: v16 or higher
* **Python**: v3.8 or higher
* **pip**: Latest version
* **Git**
* **Firebase CLI**

### **Required Accounts**

* **Firebase Account**
* **MarketAux API Account**
* **HuggingFace API Account**

## **Environment Setup**

### **Firebase Setup**

Install the Firebase CLI:

npm install -g firebase-tools

Log in to Firebase:

firebase login

Initialize the Firebase project:

firebase init

During initialization, configure Firebase with the following:

* Authentication (Email/Password and Google Sign-in)
* Hosting
* Functions (optional)

### **API Keys Configuration**

Create a .env file in the root directory with the following content:

MARKETAUX\_API\_TOKEN=your\_marketaux\_token

HUGGINGFACE\_API\_TOKEN=your\_huggingface\_token

PORT=3000

### **Frontend Setup**

Install frontend dependencies:

npm install

Update the Firebase configuration in firebase.js with your credentials:

const firebaseConfig = {

apiKey: "your-api-key",

authDomain: "your-domain.firebaseapp.com",

projectId: "your-project-id",

storageBucket: "your-bucket.appspot.com",

messagingSenderId: "your-sender-id",

appId: "your-app-id",

measurementId: "your-measurement-id"

};

### **Backend Setup**

Create a Python virtual environment:

python -m venv venv

Activate the virtual environment:

On Windows:  
venv\Scripts\activate

On Unix or macOS:  
source venv/bin/activate

Install Python dependencies:

pip install -r requirements.txt

## **Development Deployment**

### **Starting the Backend Server**

Run the following command from the project root:

python main.py

The backend will be available at: http://localhost:3000

### **Starting the Frontend Development Server**

Run the following command from the project root:

npm run dev

The frontend will be available at: http://localhost:5173

## **Production Deployment**

### **Backend Deployment (Using Docker)**

Build the Docker image:

docker build -t sentimentpulse-api .

Run the Docker container:

docker run -d -p 3000:3000 \

--env-file .env \

--name sentimentpulse-api \

sentimentpulse-api

### **Frontend Deployment**

Build production assets:

npm run build

Deploy the built assets to Firebase Hosting:

firebase deploy --only hosting

## **Configuration Files**

### **Docker Configuration**

Create a Dockerfile with the following content:

FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD ["python", "main.py"]

### **NGINX Configuration (Optional)**

Set up NGINX for proxying:

server {

listen 80;

server\_name your-domain.com;

location / {

proxy\_pass http://localhost:5173;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

}

location /api {

proxy\_pass http://localhost:3000;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

}

}

## **Security Considerations**

### **Environment Variables**

* Never commit .env files to version control.
* Use a secret management solution in production.
* Rotate API keys regularly.

### **CORS Configuration**

* Update allowed origins in main.py.
* Configure appropriate security headers.
* Enable HTTPS for all production environments.

### **API Rate Limiting**

* Set up rate limits to manage usage.
* Monitor API usage patterns and set alerts for abuse or unusual activity.

## **Monitoring**

### **Backend Monitoring**

* Configure logging for error tracking and debugging.
* Set up health checks to monitor service availability.
* Track API rate limits and performance metrics.

### **Frontend Monitoring**

* Use Firebase Analytics for user activity tracking.
* Implement error tracking for the frontend application.
* Monitor performance and user interaction metrics.

## **Backup and Recovery**

### **Database Backups**

* Schedule regular backups for cached data.
* Implement a secure API key backup strategy.
* Document recovery procedures for critical failures.

### **Deployment Rollback**

* Maintain previous versions of the application.
* Document and test rollback procedures.
* Ensure the recovery process works in various failure scenarios.

## **Troubleshooting**

### **Common Issues**

* API connection errors
* Rate limit exceeded
* Authentication failures
* CORS configuration issues

### **Solutions**

* Verify API keys and permissions.
* Check network connectivity.
* Validate Firebase and backend configurations.
* Review error logs for additional context.

## **Maintenance**

### **Regular Updates**

* Update dependencies to their latest stable versions.
* Apply security patches promptly.
* Optimize the application for performance and scalability.

### **Health Checks**

* Monitor the health of the system regularly.
* Verify API status and cache performance.
* Test the authentication system periodically to ensure reliability.