PulseNet - Real-Time Global Emotion Map

Complete Project Requirements & Technical Structure

Version: 1.0

Last Updated: October 20, 2025

Repository: https://github.com/vixtor-e86/emotion-map-project

---

🎯 PROJECT VISION

PulseNet is the world's first real-time emotional weather map. It shows how the planet feels right now by analyzing public social media posts, news headlines, and discussions using AI.

Core Concept:

> "Just like checking the weather, people will check: How does the world feel today?"

Key Features

- **Real-time emotion tracking** across 50+ countries
- **5 Core Emotions:** Joy, Anger, Fear, Hope, Calmness
- **Interactive global map** with color-coded emotions
- **Search & filter** by location, emotion, or keyword
- **Time replay** to see emotional changes over time
- **Live updates** every 60 minutes

📊 CORE EMOTIONS SYSTEM

Emotion Color Mapping

```
Joy      → Yellow/Gold   (#FFC107)
Anger    → Red           (#F44336)
Fear     → Purple/Dark   (#9C27B0)
Hope     → Green         (#4CAF50)
Calmness → Blue          (#2196F3)
```

Emotion Classification Rules

- **Joy:** happiness, celebration, excitement, love, gratitude
- **Anger:** frustration, outrage, hate, fury, irritation
- **Fear:** anxiety, worry, terror, panic, dread
- **Hope:** optimism, faith, aspiration, confidence, belief
- **Calmness:** peace, serenity, relaxation, tranquility, neutrality

---

🛠️ TECHNICAL STACK

Backend

- **Framework:** Flask 3.0.0 (Python)
- **Database:** SQLite (development) → PostgreSQL (production)
- **Sentiment Analysis:** VADER + Custom emotion classifier
- **Location Extraction:** Regex patterns + geopy + manual mapping
- **Task Scheduling:** APScheduler (60-minute intervals)
- **API Framework:** Flask-CORS for cross-origin requests

Frontend

- **Map Visualization:** Plotly.js (choropleth maps)
- **UI Framework:** HTML5, CSS3, Vanilla JavaScript
- **Charts:** Chart.js for trend visualization
- **Responsive Design:** Mobile-first approach

Data Sources

1. **Twitter/X API** (primary - real-time social sentiment)
2. **Reddit API** (PRAW) - 54 global subreddits
3. **NewsData.io API** - 32 countries coverage
4. **RSS Feeds** - 20 international news sources

Deployment

- **Development:** Local Flask server
- **Production:** Heroku / Railway.app / DigitalOcean
- **Domain:** TBD
- **CDN:** Cloudflare (optional)

📁 PROJECT STRUCTURE

```
pulsenet/
│
├── backend/
│   ├── app.py                  # Main Flask application
│   ├── config.py               # Configuration & API keys
│   │
│   ├── data_collection/
│   │   ├── __init__.py
│   │   ├── twitter_collector.py    # Twitter API (TODO)
│   │   ├── reddit_collector.py     # Reddit posts ✅
│   │   ├── news_collector.py       # NewsData.io API ✅
│   │   └── rss_collector.py        # RSS feeds ✅
│   │
│   ├── processing/
│   │   ├── __init__.py
│   │   ├── emotion_analyzer.py     # 5-emotion classifier (TODO)
│   │   ├── location_extractor.py   # Extract locations (TODO)
│   │   └── aggregator.py           # Aggregate by location (TODO)
│   │
│   ├── database/
│   │   ├── __init__.py
│   │   ├── init_db.py              # Database setup ✅
│   │   └── db_manager.py           # CRUD operations ✅
│   │
│   ├── scheduler/
│   │   ├── __init__.py
│   │   └── background_tasks.py     # Automated pipeline (PARTIAL)
│   │
│   └── routes/
│       ├── __init__.py
│       └── api_routes.py           # REST API endpoints ✅
```

```
│
├── frontend/
│   ├── static/
│   │   ├── css/
│   │   │   └── style.css          # Styling (TODO)
│   │   ├── js/
│   │   │   ├── map.js             # Plotly map logic (TODO)
│   │   │   ├── dashboard.js       # Stats & charts (TODO)
│   │   │   └── api.js             # API calls (TODO)
│   │   └── images/
│   │
│   └── templates/
│       └── index.html             # Main page (BASIC)
│
├── tests/
│   ├── test_api.py
│   ├── test_collectors.py
│   └── test_processing.py
│
├── .env                       # Environment variables ✅
├── .gitignore                 # Git ignore rules ✅
├── requirements.txt           # Python dependencies ✅
├── README.md
└── emotion_map.db             # SQLite database ✅
```

🗄️ DATABASE SCHEMA

Table 1: `raw_posts`

Stores every collected post before processing.

sql

```sql
CREATE TABLE raw_posts (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    text TEXT NOT NULL,
    source TEXT NOT NULL,            -- Twitter, Reddit, NewsAPI, RSS
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
    city TEXT,
    country TEXT,
    continent TEXT,
    emotion TEXT,                    -- joy, anger, fear, hope, calmness
    emotion_score REAL,              -- Confidence score (0-1)
    sentiment TEXT,                  -- positive, negative, neutral (legacy)
    sentiment_score REAL             -- VADER score (-1 to 1)
);

-- Indexes for performance
CREATE INDEX idx_raw_posts_timestamp ON raw_posts(timestamp);
CREATE INDEX idx_raw_posts_country ON raw_posts(country);
CREATE INDEX idx_raw_posts_emotion ON raw_posts(emotion);
```

Table 2: `aggregated_emotions`

Summarized emotion data by location for map visualization.

```sql
```

```
CREATE TABLE aggregated_emotions (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    location_name TEXT NOT NULL,
    location_type TEXT NOT NULL,       -- continent, country, city
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
    joy_count INTEGER DEFAULT 0,
    anger_count INTEGER DEFAULT 0,
    fear_count INTEGER DEFAULT 0,
    hope_count INTEGER DEFAULT 0,
    calmness_count INTEGER DEFAULT 0,
    total_posts INTEGER DEFAULT 0,
    dominant_emotion TEXT,             -- Most common emotion
    avg_emotion_score REAL,            -- Average confidence

    -- Legacy sentiment fields (can remove later)
    positive_count INTEGER DEFAULT 0,
    negative_count INTEGER DEFAULT 0,
    neutral_count INTEGER DEFAULT 0
);

CREATE INDEX idx_aggregated_location ON aggregated_emotions(location_name, location_type);
CREATE INDEX idx_aggregated_timestamp ON aggregated_emotions(timestamp);
```

🔌 API ENDPOINTS

Base URL

```
Development: http://localhost:5000/api
Production: https://pulsenet.io/api (TBD)
```

Endpoints

1. Health Check

```
GET /api/health

Response:
{
  "status": "ok",
  "timestamp": "2025-10-20T15:30:00Z"
}
```

2. Map Data

```
GET /api/map-data/<zoom_level>?hours=24

Parameters:
- zoom_level: continent | country | city
- hours: data from last N hours (default: 24)

Response:
{
  "locations": ["USA", "UK", "France"],
  "emotions": ["joy", "anger", "calmness"],
  "emotion_scores": [0.8, 0.6, 0.7],
  "hover_text": ["USA: Joy (1500 posts)", ...],
  "post_counts": [1500, 800, 600],
  "timestamp": "2025-10-20T15:30:00Z"
}
```

3. Location Details

```
GET /api/location/<location_name>

Response:
{
  "location": "United States",
  "total_posts": 1500,
  "joy_count": 600,
  "anger_count": 400,
  "fear_count": 200,
  "hope_count": 200,
  "calmness_count": 100,
  "dominant_emotion": "joy",
  "avg_score": 0.75,
  "sample_posts": [...],
  "timestamp": "2025-10-20T15:30:00Z"
}
```

4. Global Statistics

```
GET /api/stats

Response:
{
  "total_posts": 25000,
  "world_joy": 35,
  "world_anger": 20,
  "world_fear": 15,
  "world_hope": 18,
  "world_calmness": 12,
  "dominant_emotion": "joy",
  "countries_tracked": 45,
  "last_updated": "2025-10-20T15:30:00Z"
}
```

5. Emotion Trends

```
GET /api/trends?hours=24&emotion=joy

Response:
{
  "timestamps": ["15:00", "16:00", "17:00"],
  "emotion_scores": [0.65, 0.70, 0.68],
  "post_counts": [200, 250, 230]
}
```

6. Search

```
GET /api/search?query=climate&emotion=fear

Response:
{
  "query": "climate",
  "emotion_filter": "fear",
  "results": [...],
  "total": 245
}
```

🤖 EMOTION ANALYSIS SYSTEM

Recommended Approach (Simple & Effective)

Option 1: Enhanced VADER + Keyword Mapping ⭐ RECOMMENDED

```
python

# Step 1: Use VADER for basic sentiment
# Step 2: Use emotion keywords to classify 5 emotions

EMOTION_KEYWORDS = {
  'joy': ['happy', 'love', 'amazing', 'great', 'wonderful', 'excited'],
  'anger': ['hate', 'angry', 'furious', 'outrage', 'terrible', 'worst'],
  'fear': ['scared', 'afraid', 'worry', 'anxiety', 'panic', 'terrified'],
  'hope': ['hope', 'optimistic', 'believe', 'confident', 'faith', 'positive'],
  'calmness': ['calm', 'peace', 'relax', 'tranquil', 'serene', 'okay']
}

# Combine VADER score + keyword matches for final classification
```

Option 2: Use Pre-trained Emotion Model

```python
# Use HuggingFace: j-hartmann/emotion-english-distilroberta-base
# Classifies: joy, sadness, anger, fear, surprise, neutral, disgust
# Map: sadness→calmness, surprise→hope, neutral→calmness
```

Option 3: Train Custom Model (Future)

```python
# Collect labeled dataset of 10,000+ posts
# Fine-tune BERT model specifically for 5 emotions
# Deploy as microservice
```

📍 LOCATION EXTRACTION STRATEGY

Method 1: Regex Patterns (Fast & Simple) ⭐ RECOMMENDED

```python
PATTERNS = [
    r'in ([A-Z][a-z]+(?:\s+[A-Z][a-z]+)*)',  # "in Paris"
    r'from ([A-Z][a-z]+)',                   # "from Nigeria"
    r'#([A-Z][a-z]+)',                       # "#Lagos"
]

LOCATION_DATABASE = {
    'Paris': {'country': 'France', 'continent': 'Europe'},
    'Lagos': {'country': 'Nigeria', 'continent': 'Africa'},
    # ... 200+ major cities
}
```

Method 2: Source-Based Mapping (Guaranteed)

```python
# Reddit: r/france → France
# NewsAPI: country parameter → Direct mapping
# Twitter: User location (if available)
```

Method 3: geopy Verification (Fallback)

```python
from geopy.geocoders import Nominatim

geolocator = Nominatim(user_agent="pulsenet")
location = geolocator.geocode("Paris")
# Returns: country, coordinates
```

⚙️ DATA PIPELINE WORKFLOW

Every 60 Minutes:

```
1. DATA COLLECTION (15 minutes)
   ├─ Twitter API → Fetch 1000+ recent tweets
   ├─ Reddit API → Scrape 54 subreddits (175+ posts)
   ├─ NewsData.io → Fetch from 32 countries (500+ articles)
   └─ RSS Feeds → Parse 20 sources (700+ headlines)

2. PROCESSING (10 minutes)
   ├─ Clean text (remove URLs, hashtags, special chars)
   ├─ Analyze emotion (classify into 5 categories)
   ├─ Extract location (regex + mapping + geopy)
   └─ Save to raw_posts table

3. AGGREGATION (5 minutes)
   ├─ Group by country/continent
   ├─ Count emotion frequencies
   ├─ Calculate dominant emotion
   ├─ Compute average scores
   └─ Save to aggregated_emotions table

4. CLEANUP (2 minutes)
   └─ Delete data older than 7 days
```

🎨 FRONTEND DESIGN SPECIFICATIONS

Color Scheme

```css
--joy: #FFC107 (Yellow/Gold)
--anger: #F44336 (Red)
--fear: #9C27B0 (Purple)
--hope: #4CAF50 (Green)
--calmness: #2196F3 (Blue)
--bg-dark: #1a1a2e
--bg-light: #f5f5f5
--text-primary: #333333
```

Layout Components

1. Header

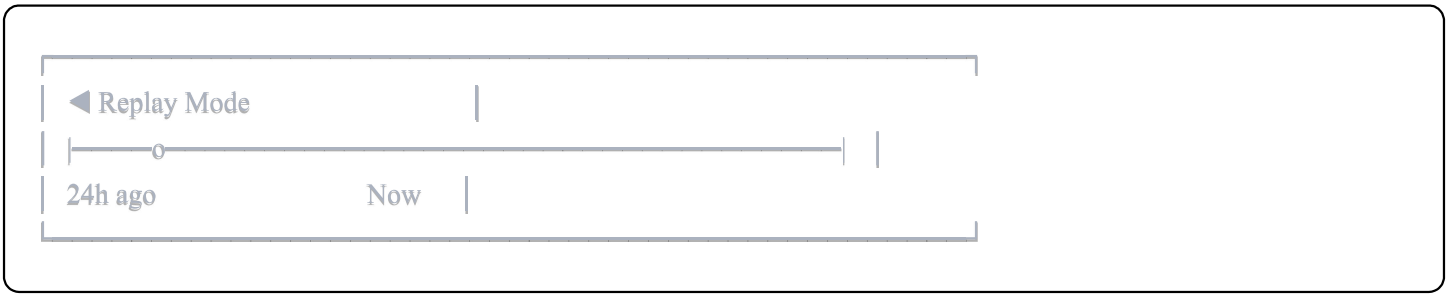
```
┌─────────────────────────────────────────────────────────┐
│ 🌎 PulseNet - The World's Emotional Pulse               │
│ [Search: _____] [Filter: All Emotions]                  │
└─────────────────────────────────────────────────────────┘
```

2. Global Emotion Meter

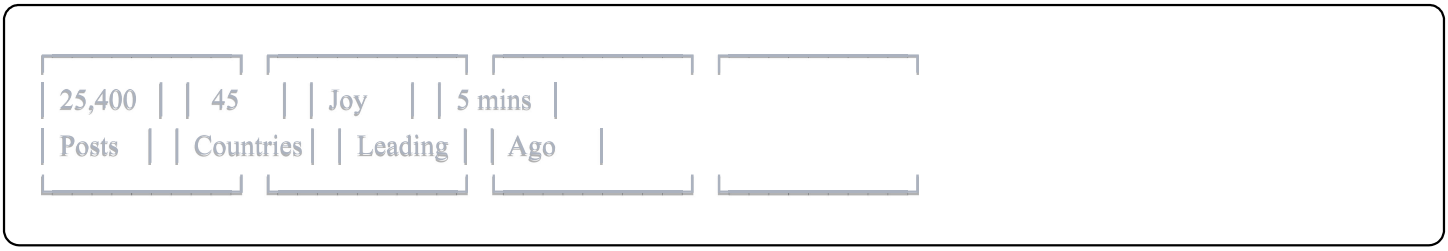
```
┌─────────────────────────────────────────────────────────┐
│ World Emotion Right Now:                                │
│ 😊 Joy: 35% ████████████                                │
│ 😠 Anger: 20% ██████                                    │
│ 😨 Fear: 15% ████                                       │
│ 🙏 Hope: 18% ██████                                     │
│ 😌 Calmness: 12% ███                                    │
└─────────────────────────────────────────────────────────┘
```

3. **Interactive Map** (Main Feature)

4. Time Slider



5. Dashboard Cards



🚀 DEVELOPMENT PHASES

MVP Phase (3 Weeks) - Current Focus

Week 1: Core Infrastructure ✅ COMPLETE

- Backend setup
- Database design
- API framework
- Data collectors (Reddit, News, RSS)

Week 2: Intelligence Layer 🔄 IN PROGRESS

- Emotion analyzer

- Location extractor
- Data aggregation
- Background scheduler integration

Week 3: Visualization ⏳ PENDING

- Frontend HTML/CSS
- Plotly.js map
- Dashboard components
- API integration
- User interactions

Beta Phase (2 Weeks)

- Twitter integration
- Real-time updates (WebSocket)
- Search & filter functionality
- Time replay feature
- Mobile optimization

Launch Phase (1 Week)

- Performance optimization
- Security hardening
- Production deployment
- Marketing website
- Public launch

📦 DEPENDENCIES & INSTALLATION

Backend Requirements

```
Flask==3.0.0
Flask-CORS==4.0.0
pandas==2.3.3
```

```
numpy==2.3.4
vaderSentiment==3.3.2
praw==7.7.1
feedparser==6.0.10
APScheduler==3.10.4
python-dotenv==1.0.0
requests==2.31.0
geopy==2.4.1
tweepy==4.14.0  # For Twitter (to be added)
```

Frontend Dependencies (CDN)

html

```
<!-- Plotly.js for maps -->
<script src="https://cdn.plot.ly/plotly-2.27.0.min.js"></script>

<!-- Chart.js for trends -->
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<!-- Font Awesome for icons -->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
```

🔐 ENVIRONMENT VARIABLES

env

```
# Flask Configuration
FLASK_ENV=development
FLASK_DEBUG=True
SECRET_KEY=your-secret-key-change-in-production

# Database
DATABASE_PATH=emotion_map.db

# API Keys
NEWS_API_KEY=pub_ec0b35a1ee7d473aaaf4e1078d716f00
REDDIT_CLIENT_ID=vz5CpcRQjC_jCugl84h9LQ
REDDIT_CLIENT_SECRET=dMloQ8iuHBtcg-hGWGuFeCVqN7GtqA
REDDIT_USER_AGENT=GlobalEmotionMapCollector
TWITTER_API_KEY=  # To be added
TWITTER_API_SECRET=  # To be added
TWITTER_BEARER_TOKEN=  # To be added

# Scheduler
UPDATE_INTERVAL_MINUTES=60

# Performance
MAX_POSTS_PER_COLLECTION=2000
CACHE_TIMEOUT=300
```

📈 SCALABILITY CONSIDERATIONS

Current Capacity

- **Storage:** SQLite (good for 100K+ posts)
- **Performance:** Single Flask server (100+ requests/sec)
- **Data Volume:** ~2,000 posts/hour

Production Scaling

```
Database: SQLite → PostgreSQL
Server: Single Flask → Gunicorn + Nginx
Caching: None → Redis
Queue: Sync → Celery + RabbitMQ
Hosting: Single server → Load-balanced cluster
```

CDN: None → Cloudflare
Monitoring: Logs → DataDog/Sentry

🎓 TEAM STRUCTURE & RESPONSIBILITIES

Total Team: 7 people

Roles

1. **Project Lead (1)** - Victor
   - Overall coordination
   - Backend architecture
   - API integration
2. **Backend Developers (2)**
   - Data collectors
   - Database management
   - Emotion analysis
3. **Frontend Developers (2)**
   - UI/UX design
   - Map visualization
   - Dashboard creation
4. **Data Scientist (1)**
   - Emotion model fine-tuning
   - Location extraction logic
   - Data validation
5. **DevOps/Tester (1)**
   - Deployment setup
   - Testing
   - Performance optimization

📚 LEARNING RESOURCES

For Backend

- Flask Tutorial: https://flask.palletsprojects.com/
- VADER Sentiment: https://github.com/cjhutto/vaderSentiment
- Reddit PRAW: https://praw.readthedocs.io/

For Frontend

- Plotly.js Maps: https://plotly.com/javascript/maps/
- JavaScript ES6: https://javascript.info/
- CSS Grid: https://css-tricks.com/snippets/css/complete-guide-grid/

For Deployment

- Heroku Tutorial: https://devcenter.heroku.com/articles/getting-started-with-python
- GitHub Actions: https://docs.github.com/en/actions

🎯 SUCCESS METRICS

MVP Success Criteria

- [ ] Collect 2,000+ posts per hour
- [ ] Classify emotions with 70%+ accuracy
- [ ] Map data for 40+ countries
- [ ] Load map in < 3 seconds
- [ ] Update data every 60 minutes reliably
- [ ] Mobile-responsive design
- [ ] Zero downtime for 24 hours

Beta Success Criteria

- [ ] 1,000+ daily active users
- [ ] 5-second average page load
- [ ] 99% uptime
- [ ] Twitter integration live
- [ ] Real-time updates working

🔮 FUTURE FEATURES (Post-MVP)

1. **AI Insights**
   - "Why is anger rising in Region X?"
   - Automatic event detection
   - Predictive emotion trends
2. **User Features**
   - Create custom alerts ("Notify when hope rises in Nigeria")
   - Save favorite locations
   - Export emotion reports
3. **Business Features**
   - Brand sentiment tracking
   - Crisis monitoring dashboard
   - Historical emotion analytics
4. **Social Features**
   - Share emotion snapshots
   - Compare regions
   - Emotion leaderboards

📞 SUPPORT & CONTACT

GitHub: https://github.com/vixtor-e86/emotion-map-project
Documentation: [Add Wiki link]
Issues: [Add Issues link]
Team Lead: Victor (vixtor-e86)

Last Updated: October 20, 2025
Document Version: 1.0
Next Review: After MVP completion