

RecAI -Smart Emotion Based Place Recommender System

1. Introduction

The **Places Emotion Recommender System** is an AI-driven solution designed to suggest personalized locations to users based on their emotional preferences and real-time geographical location. Unlike traditional recommendation systems that rely solely on ratings or popularity, this system integrates:

- Online place category datasets
- Emotion mapping using NLP
- Transformer-based semantic understanding
- Real-time geospatial data
- Intelligent recommendation ranking

The system builds a connection between **user emotions** and the **emotional context of places**, enabling emotionally meaningful recommendations for travel, leisure, and exploration.

2. System Pipeline

The complete workflow of the project consists of the following stages:

2.1 Dataset Acquisition

An online dataset was used that contains:

- Place names
- 5 hierarchical category levels
(Example: Hockey → Sports & Adventure → Club → ...)

This hierarchical taxonomy provides rich semantic context for each place.

2.2 Emotion Mapping in Google Colab

The dataset categories were processed in **Google Colab** using:

- Sentence-BERT embeddings
- Emotion lexicons
- Category → Emotion mapping logic

Each category was classified into emotional labels such as:

- Calm
- Excited
- Social
- Spiritual
- Adventurous
- Curious
- Relaxed
- Energetic

This produced a **classified emotion database**.

2.3 Emotion Database Creation

The output of the Colab stage was stored as:

- Emotion-labeled dataset
- Category–Emotion mapping table
- Embedding vectors for reference

This database serves as the **training reference** for the live system.

2.4 Transformer-Based Emotion Classifier (VS Code)

In VS Code, a **Sentence-Transformer model (all-mpnet-base-v2)** was integrated.

1. Takes place descriptions fetched from APIs
2. Generates embeddings
3. Matches them with the classified emotion database
4. Predicts dominant emotions for each place

This enables real-time emotion classification for any location.

2.5 Backend Recommendation Engine

A FastAPI backend processes:

- User emotions
- Geolocation data
- Predicted place emotions

and generates personalized recommendations.

3. Technologies Used

Backend Framework & API

- FastAPI – high-performance API development
- Uvicorn – ASGI server for deployment
- Pydantic – request/response validation

Database

- MongoDB – storing place and emotion data
- PyMongo – Python MongoDB driver

AI / Machine Learning

- PyTorch
- Sentence-Transformers
- all-mpnet-base-v2 model
- Cosine similarity for emotion matching

Data Processing

- Pandas
- NumPy
- openpyxl (via pandas) for Excel dataset handling

External Services & APIs

- Overpass API (OpenStreetMap)
- Requests HTTP client

Tools & Development Environment

- Google Colab – emotion mapping and dataset processing
 - VS Code – backend development and model integration
 - Python Standard Libraries:
os, threading, time, logging, math, tempfile, random
-

4. Functionality Overview

4.1 Distance & Location Functions

- `calculate_distance_km()` – Haversine formula
- `get_cached_location()` – retrieves stored location
- `is_within_proximity()` – 1km cache validation
- `save_location_cache()` – caching mechanism

These functions optimize performance by avoiding repeated API calls.

4.2 Place Data Fetching

- `get_place_details_from_overpass()`

Features:

- Multiple Overpass endpoints
 - Tag extraction (amenity, tourism, shop, leisure)
 - Context description generation
 - Reliability handling
-

4.3 Emotion Analysis Functions

PlaceEmotionAnalyzer Class

Includes:

- `_load_model()` – loads transformer model
- `_load_reference_data()` – loads emotion dataset
- `create_context_aware_text()` – builds rich descriptions
- `predict_emotions_for_place()` – main classifier
- `_get_emotions_from_similarity()` – embedding similarity
- `_get_enhanced_emotions_from_context()` – rule-based fallback

This component represents the **AI core** of the system.

4.4 Recommendation Engine

- `calculate_emotion_match_score()`
- `rank_places_by_emotion_priority()`

Scores include:

- Exact match
- Weighted match
- Coverage score
- Dominance boosting

Final output: Ranked personalized recommendations

4.5 API Endpoints (FastAPI)

Key endpoints include:

- `/fetch_places`
- `/recommend_places`
- `/places`
- `/test_api`
- `/cache_status`
- `/unique_emotions`

These enable full system interaction, debugging, and monitoring.

5. Data Processing Pipeline

1. Online dataset imported
 2. Category hierarchy extracted
 3. Emotion mapping performed in Colab
 4. Emotion database generated
 5. Transformer model trained with reference embeddings
 6. Real-time place fetching from Overpass API
 7. Emotion classification per place
 8. Recommendation scoring
 9. Ranked output returned to user
-

6. Results and Verification

The system successfully:

- Classified emotions for dataset categories
- Generated emotional embeddings
- Fetched real-time places based on user location
- Produced personalized recommendations

Performance outcomes:

- API caching reduced calls by 50–70%
 - Emotion prediction accuracy improved using similarity matching
 - Response time improved due to caching and parallel processing
-

7. Key Features

- Emotion-based recommendations
 - Category–emotion mapping
 - Transformer-based emotion classifier
 - Real-time location integration
 - Intelligent caching
 - Multi-level fallback strategies
 - Debugging and analytics endpoints
-

8. Future Enhancements

- Cloud deployment (AWS / Hugging Face Spaces)
 - User profiles and learning preferences
 - Google Maps UI integration
 - Voice-based recommendation assistant
 - Multi-language support
 - Larger datasets with reviews and ratings
 - Hybrid deep learning sentiment models
-

9. Conclusion

The Places Emotion Recommender System successfully combines:

- Geospatial analysis
- Transformer models
- Real-time APIs
- Recommendation algorithms

to provide emotionally meaningful location suggestions.

The integration of an online dataset, emotion mapping in Colab, and a transformer-based classifier in VS Code forms a complete, innovative pipeline with strong real-world applicability.

