# AnantaNetra - AI-Powered Environmental Monitoring System

## 🌍 Project Overview

**AnantaNetra** is a comprehensive AI-powered environmental monitoring system designed to address India's critical air pollution crisis affecting 2+ million lives annually. The system provides real-time AQI monitoring, predictive analytics, and AI-powered health advisories for all 732 districts of India.

### 🎯 Key Features

- **Real-time AQI monitoring** for 50+ major Indian cities
- **24-48 hour AQI predictions** with 89-94% accuracy (95% CI)
- **AI-powered health advisories** using Google Gemini integration
- **Interactive dashboard** with mobile-responsive design
- **Comprehensive fallback system** ensuring high availability
- **Multi-source data integration** from 15+ heterogeneous sources

### 🏆 Achievements

- **92% Prediction Accuracy** using Hybrid LSTM+XGBoost ensemble models
- **Production-Ready Architecture** with Docker containerization
- **Hackathon Winner** - Recognized for innovative environmental monitoring solution
- **Research Publication** - IEEE-format paper with comprehensive technical validation

---

## 📁 Project Structure

```
India_Environment_Project/
├── AnantaNetra_AQI_Project/     # 🏠 Main application codebase
│   ├── backend/                 # FastAPI backend services
│   │   ├── app/
│   │   │   ├── api/            # API endpoints (AQI, forecast, health, map)
│   │   │   ├── services/       # Business logic (prediction, caching, external
APIs)
│   │   │   ├── models/         # Data models and schemas
│   │   │   ├── utils/          # Utilities (config, logging, error handling)
│   │   │   └── main.py         # FastAPI application entry point
│   │   ├── requirements.txt    # Python dependencies
│   │   └── Dockerfile          # Backend containerization
│   ├── frontend/               # React/TypeScript dashboard
│   │   ├── src/
│   │   │   ├── components/     # React components (Dashboard, Map, Health
Advisory)
│   │   │   ├── services/       # API integration layer
│   │   │   ├── types/          # TypeScript type definitions
│   │   │   └── App.tsx         # Main React application
│   │   ├── package.json        # Node.js dependencies
│   │   └── vite.config.ts      # Vite build configuration
│   ├── data/                   # Application-specific datasets
│   ├── ml/                     # ML models and training scripts
│   ├── ml_models/              # Trained model artifacts
│   ├── deployment/             # Docker Compose and deployment configs
│   ├── scripts/                # Utility scripts (start_system.sh/.bat/.ps1)
│   ├── docker-compose.yml      # Multi-container orchestration
│   └── README.md               # Application-specific documentation
├── docs/                       # 📚 Research and documentation
│   ├── AnantaNetra_Research_Paper.tex    # IEEE research paper
│   ├── RESEARCH_PAPER_DOCUMENTATION.md  # Paper documentation
│   ├── HACKATHON_INSTRUCTIONS.md         # Hackathon guidelines
│   └── test_architecture.tex             # Architecture documentation
├── shared/                     # 🔗 Shared resources
│   ├── data/                   # Common datasets and databases
│   ├── ml/                     # Shared ML utilities and models
│   └── real_aqi_model.pkl     # Trained model file
├── .env                        # Environment variables (API keys)
├── .env.example                # Environment template
├── requirements.txt            # Root-level Python dependencies
└── README.md                   # This comprehensive guide
```

---

## 🚀 Quick Start Guide

### 📋 Prerequisites

**Required Software:**

- **Python 3.8+** (Backend development and ML)
- **Node.js 16+** (Frontend development)
- **Docker & Docker Compose** (Containerized deployment)
- **Git** (Version control)

**System Requirements:**

- **RAM:** 8GB minimum, 16GB recommended
- **Storage:** 5GB free space
- **OS:** Windows 10+, macOS 10.15+, Ubuntu 18.04+

### 🔧 Step-by-Step Installation

#### 1. Clone and Navigate

```
git clone <repository-url>
cd India_Environment_Project
```

#### 2. Environment Setup

```
# Copy environment template
cp .env.example .env

# Edit .env file with your API keys (see API Keys section below)
notepad .env  # Windows
# or
nano .env     # Linux/Mac
```

### 3. Backend Setup

```
cd AnantaNetra_AQI_Project

# Create Python virtual environment
python -m venv venv
source venv/bin/activate  # Linux/Mac
# or
venv\Scripts\activate     # Windows

# Install Python dependencies
pip install -r requirements.txt
pip install -r backend/requirements.txt
```

### 4. Frontend Setup

```
cd frontend

# Install Node.js dependencies
npm install

# Build the application (optional for development)
npm run build
```

### 5. Start the Application

#### Option A: Automated Startup (Recommended)

```
# From AnantaNetra_AQI_Project directory
./start_system.sh    # Linux/Mac
start_system.bat     # Windows
# or
./start_system.ps1   # Windows PowerShell
```

#### Option B: Manual Startup

*Terminal 1 - Backend:*

```
cd AnantaNetra_AQI_Project
source venv/bin/activate  # Linux/Mac
# or
venv\Scripts\activate     # Windows

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

*Terminal 2 - Frontend:*

```
cd AnantaNetra_AQI_Project/frontend
npm run dev
```

**6. Access the Application**

- **Frontend Dashboard:** http://localhost:5173
- **Backend API Docs:** http://localhost:8000/docs
- **API Base URL:** http://localhost:8000/api

---

## 🔑 API Keys & Configuration

### Required API Keys

You need to obtain and configure the following API keys in your `.env` file:

```
# Weather API (Primary data source)
WEATHER_API_KEY=your_weatherapi_key_here

# OpenWeatherMap API (Backup weather data)
OPENWEATHER_API_KEY=your_openweather_key_here

# OpenCage Geocoding API (Location services)
OPENCAGE_API_KEY=your_opencage_key_here

# Google Gemini AI (Health advisories)
GEMINI_API_KEY=your_gemini_key_here
```

### How to Get API Keys

1. **WeatherAPI:** https://www.weatherapi.com/ (Free tier: 1M calls/month)
2. **OpenWeatherMap:** https://openweathermap.org/api (Free tier: 60 calls/minute)
3. **OpenCage:** https://opencagedata.com/ (Free tier: 2,500 calls/day)
4. **Google Gemini:** https://makersuite.google.com/app/apikey (Free tier available)

### ⚠ API Limits & Costs

| API Service | Free Tier Limits | Paid Tier | Cost |
|---|---|---|---|
| **WeatherAPI** | 1M calls/month | 10M calls/month | $10/month |
| **OpenWeatherMap** | 60 calls/minute | 600 calls/minute | $40/month |
| **OpenCage** | 2,500 calls/day | 100K calls/day | $50/month |
| **Google Gemini** | 60 requests/minute | Higher limits | Free tier sufficient |

**Rate Limiting in Application:**

- Gemini AI: 10 requests/minute, 250 requests/day
- Weather APIs: 3.6 second delay between calls
- Automatic fallback to cached/demo data when limits exceeded

---

## 🔌 API Documentation

### Base URL

```
http://localhost:8000/api
```

### Core Endpoints

#### 🌡 AQI Monitoring

```
GET /api/aqi/{pincode}
```

**Parameters:**

- `pincode` (string): 6-digit Indian pincode

**Response:**

```
{
  "aqi": 180,
  "category": "Unhealthy",
  "pincode": "110001",
  "timestamp": "2025-01-04T10:30:00Z",
  "source": "weatherapi",
  "pm25": 85.2,
  "pm10": 142.5,
  "temperature": 28.5,
  "humidity": 65.0,
  "wind_speed": 12.5
}
```

### 📈 AQI Forecasting

```
GET /api/forecast/{pincode}?hours=24&confidence_interval=true
```

**Parameters:**

- `pincode` (string): 6-digit Indian pincode
- `hours` (integer): Forecast duration (1-168 hours)
- `confidence_interval` (boolean): Include prediction confidence

**Response:**

```json
[
  {
    "timestamp": "2025-01-04T11:00:00Z",
    "aqi": 185,
    "confidence_lower": 165,
    "confidence_upper": 205,
    "category": "Unhealthy"
  }
]
```

### 🏥 Health Advisories

```
GET /api/health/advisory?aqi=180
```

**Response:**

```json
{
  "category": "Unhealthy",
  "message": "Air quality is unhealthy. Sensitive groups should avoid outdoor activities.",
  "precautions": [
    "Wear N95 masks when outdoors",
    "Limit outdoor activities",
    "Keep windows closed",
    "Use air purifiers indoors"
  ],
  "risk_groups": ["Children", "Elderly", "People with respiratory conditions"],
  "aqi_range": "151-200",
  "health_effects": "May cause respiratory symptoms in sensitive individuals"
}
```

### 🗺 Map Data

```
GET /api/map/data
```

**Response:**

```json
{
  "cities": [
    {
      "name": "Delhi",
      "lat": 28.6139,
      "lng": 77.2090,
      "aqi": 180,
      "category": "Unhealthy",
      "pm25": 85.2,
      "temperature": 28.5
    }
  ]
}
```

### 📊 System Status

```
GET /api/status/health
```

**Response:**

```json
{
  "status": "healthy",
  "timestamp": "2025-01-04T10:30:00Z",
  "services": {
    "prediction": "operational",
    "cache": "operational",
    "external_apis": "operational"
  },
  "uptime": "2h 15m",
  "version": "1.0.0"
}
```

---

## 🏗 Backend Architecture (FastAPI)

## Core Components

#### API Layer (`/api/`)

- `aqi.py` - Current AQI data retrieval and caching
- `forecast.py` - ML-based AQI predictions with confidence intervals
- `health_advisory.py` - AI-powered health recommendations
- `map_data.py` - Geographic data for interactive maps
- `status.py` - System health monitoring and diagnostics

#### Services Layer (`/services/`)

- `prediction.py` - Hybrid LSTM+XGBoost ensemble model orchestration
- `caching.py` - Redis-based caching with TTL management
- `external_apis.py` - Third-party API integration with fallback mechanisms

#### Utilities (`/utils/`)

- `config.py` - Environment configuration and API key management
- `logging.py` - Structured logging with different log levels
- `error_handling.py` - Comprehensive error handling and recovery

## Key Features Implemented

#### 🔄 Intelligent Caching System

- Redis-based caching with configurable TTL
- Automatic cache invalidation and refresh
- Fallback to stale cache during API outages

#### 🤖 AI/ML Pipeline

- Hybrid LSTM+XGBoost ensemble for superior accuracy
- Real-time feature engineering (temporal, meteorological, spatial)
- Confidence interval estimation for predictions

#### 🛡️ Robust Error Handling

- Multi-level fallback systems (API → Cache → Demo data)
- Graceful degradation during service outages
- Comprehensive logging and monitoring

#### 🔐 Rate Limiting & Security

- API rate limiting to respect third-party limits
- Input validation and sanitization
- CORS configuration for frontend integration

---

## 🎨 Frontend Architecture (React + TypeScript)

### Technology Stack

- **React 18** - Modern component-based UI framework
- **TypeScript** - Type-safe JavaScript development
- **Material-UI** - Professional component library
- **React Query** - Server state management and caching
- **Leaflet** - Interactive maps and geospatial visualization
- **Recharts** - Data visualization and charting
- **Vite** - Fast build tool and development server

### Core Components

#### Dashboard Components

- `Dashboard.tsx` - Main dashboard with AQI overview and charts
- `MapView.tsx` - Interactive map with city markers and AQI visualization
- `HealthAdvisory.tsx` - Health recommendations based on AQI levels
- `Search.tsx` - Pincode-based location search functionality
- `Navigation.tsx` - Responsive navigation and routing

#### Services & Integration

- `api.ts` - Axios-based API client with error handling and fallbacks
- **Type definitions** - Comprehensive TypeScript interfaces for all data models

### Key Features Implemented

#### 📱 Responsive Design

- Mobile-first approach with Material-UI components
- Adaptive layouts for desktop, tablet, and mobile
- Dark/light theme support

#### 🗺️ Interactive Mapping

- Real-time AQI visualization on maps
- City markers with color-coded AQI categories
- Click-to-explore functionality for detailed city data

#### 📊 Data Visualization

- Real-time charts and graphs using Recharts
- Historical trends and forecast visualization
- Comparative analysis across cities

#### 🔄 State Management

- React Query for server state management
- Optimistic updates and background refetching
- Intelligent caching and error boundaries

---

## 🐳 Docker Deployment

### Single-Command Deployment

```
cd AnantaNetra_AQI_Project
docker-compose up -d
```

### Services in Docker Compose

```
services:
  backend:
    build: ./backend
    ports: ["8000:8000"]
    environment:
      - REDIS_URL=redis://redis:6379
    depends_on: [redis]

  frontend:
    build: ./frontend
    ports: ["5173:5173"]
    depends_on: [backend]

  redis:
    image: redis:7-alpine
    ports: ["6379:6379"]
```

### Individual Container Builds

```
# Backend
cd backend
docker build -t anantanetra-backend .

# Frontend
cd frontend
docker build -t anantanetra-frontend .
```

---

## 🔬 Machine Learning Pipeline

### Model Architecture

#### Hybrid Ensemble Model

- **LSTM Component:** Captures temporal dependencies and seasonal patterns
  - Architecture: 2 LSTM layers (128, 64 units) with 0.2 dropout
  - Input sequence: 24-hour historical data
  - Output: Time-series predictions
- **XGBoost Component:** Handles non-linear feature interactions
  - Configuration: 100 estimators, max depth 6, learning rate 0.1
  - Features: 25+ engineered features
  - Output: Feature importance-based predictions
- **Ensemble Strategy:** Weighted combination (70% XGBoost, 30% LSTM)

### Feature Engineering

#### Temporal Features

- Cyclical encoding for hour/month (sin, cos transformations)
- Lag features (1h, 6h, 12h, 24h historical AQI)
- Rolling averages and statistical measures

#### Meteorological Features

- Temperature, humidity, wind speed, atmospheric pressure
- Weather condition categories and severity scores

#### Geospatial Features

- Population density, industrial activity indices
- Forest cover ratios and urban development metrics

### Performance Metrics

| Model | MAE | RMSE | $R^2$ Score | MAPE |
|---|---|---|---|---|
| Linear Regression | 45.2 | 62.1 | 0.72 | 28.5% |
| Random Forest | 38.7 | 54.3 | 0.79 | 24.1% |
| XGBoost (Individual) | 32.1 | 47.8 | 0.85 | 19.7% |
| LSTM (Individual) | 35.4 | 51.2 | 0.82 | 21.8% |
| **Hybrid Ensemble** | **28.9** | **42.3** | **0.92** | **17.2%** |

---

## 🧪 Testing & Quality Assurance

### Backend Testing

```
cd backend
pytest tests/ -v --cov=app
```

### Frontend Testing

```
cd frontend
npm test
```

### API Testing

```
# Using the provided demo script
cd AnantaNetra_AQI_Project
python demo_test.py
```

### Load Testing

```
# Test API endpoints under load
ab -n 1000 -c 10 http://localhost:8000/api/status/health
```

---

## 📈 Monitoring & Analytics

### System Health Monitoring

- Real-time API response times
- Cache hit/miss ratios
- External API usage statistics
- Error rates and fallback activations

### Performance Analytics

- Prediction accuracy tracking
- User interaction patterns
- Geographic usage distribution
- System resource utilization

---

## 🤝 Contributing

### Development Workflow

1. Fork the repository
2. Create feature branch: `git checkout -b feature/amazing-feature`

3. Make changes and test thoroughly
4. Commit with descriptive messages: `git commit -m 'Add amazing feature'`
5. Push to branch: `git push origin feature/amazing-feature`
6. Create Pull Request with detailed description

### Code Standards

- **Backend:** PEP 8 compliance, type hints, comprehensive docstrings
- **Frontend:** ESLint rules, TypeScript strict mode, component documentation
- **Testing:** 80%+ code coverage, integration tests for critical paths

---

## 📚 Documentation & Research

### Available Documentation

- **Research Paper:** `docs/AnantaNetra_Research_Paper.tex` (IEEE format)
- **Technical Docs:** `docs/RESEARCH_PAPER_DOCUMENTATION.md`
- **Hackathon Guide:** `docs/HACKATHON_INSTRUCTIONS.md`
- **API Documentation:** Auto-generated at `/docs` endpoint

### Research Contributions

- Novel hybrid AI architecture for environmental prediction
- Comprehensive multi-source data integration framework
- Production-ready implementation with robust fallbacks
- Policy-aligned solution supporting NCAP objectives

---

## 🐛 Troubleshooting

### Common Issues

#### Backend Won't Start

```
# Check Python version
python --version

# Verify virtual environment
source venv/bin/activate && which python

# Check API keys in .env
cat .env | grep -E "(API_KEY|KEY)"
```

#### Frontend Build Fails

```
```

```
# Clear node_modules and reinstall
rm -rf node_modules package-lock.json
npm install

# Check Node.js version
node --version
npm --version
```

#### API Connection Issues

```
# Test backend connectivity
curl http://localhost:8000/api/status/health

# Check Redis connection
docker ps | grep redis

# Verify API keys are loaded
docker logs anantanetra-backend
```

#### Docker Issues

```
# Stop all containers
docker-compose down

# Rebuild without cache
docker-compose build --no-cache

# Check container logs
docker-compose logs backend
```

---

## 📞 Support & Contact

### Project Team

- **Akshad Makhana** - Backend Development & ML Engineering
- **Yash Kolhe** - Frontend Development & UI/UX
- **Tejas Borkar** - Data Engineering & API Integration
- **Pranav Hadole** - DevOps & Infrastructure
- **Saurabh Turkane** - Research & Documentation

### Communication

- **Email:** [team-email@sanjivani.edu.in]
- **Institution:** Sanjivani University, Department of CSE (AI & DS)

- **Location:** Maharashtra, India

---

## 📜 License

---

## 🎉 Acknowledgments

- **Data Sources:** CPCB, IMD, Ministry of Environment
- **Research Support:** Sanjivani University faculty and infrastructure
- **Open Source:** FastAPI, React, XGBoost, and other amazing libraries
- **Hackathon Organizers:** For providing the platform and recognition

---

🌍 *AnantaNetra - Seeing the Invisible Threat: AI-powered environmental monitoring for a healthier India*

**Version:** 1.0.0 | **Last Updated:** November 2025 | **Status:** Production Ready