🌥️ PM2.5 Air Quality Prediction Dashboard

Real-time PM2.5 monitoring and 24-hour prediction system using Transformer neural network, deployed on Railway with auto-updating hourly crawler.

📋 Features

- ✅ **Real-time PM2.5 Monitoring** - Live data from Taiwan EPA API
- 🔮 **24-Hour Predictions** - Transformer model forecasting
- 📊 **Interactive Visualizations** - Chart.js powered charts
- 💬 **AI Chatbot** - Gemini-powered RAG for historical queries
- ⏰ **Auto-Update** - Hourly crawler + model inference
- 🎨 **Modern UI** - Glassmorphism design, light mode

🏗️ Architecture

```
User → Railway (Flask API) → SQLite Database (ALL historical data)
             ↓
        APScheduler (Hourly)
             ↓
      EPA API Crawler → Model Inference (last 720h)
```

- **Frontend**: HTML/CSS/JavaScript + Chart.js
- **Backend**: Flask + APScheduler
- **Database**: SQLite with Railway Volume (ALL data 2018-2025, ~61,000+ hours)
- **Model**: Transformer (uses last 720h → predicts 24h)
- **RAG**: Gemini API (queries full 2018-2025 history)
- **Deployment**: Railway with Docker

📁 Project Structure

```
pm25_dashboard/
```

```
├── backend/
│   ├── app.py              # Flask main app
│   ├── config.py           # Configuration
│   ├── database.py         # SQLite utilities
│   ├── init_db.py          # Initialize DB from CSV
│   ├── crawler.py          # EPA API crawler
│   ├── prediction_service.py # Model inference
│   ├── rag_service.py      # Gemini chatbot
│   └── scheduler.py        # APScheduler
├── frontend/
│   ├── index.html
│   └── static/
│       ├── css/style.css
│       └── js/
│           ├── dashboard.js
│           └── chat.js
├── models/
│   └── best_model.keras    # YOUR TRAINED MODEL
├── Dockerfile
├── railway.toml
├── requirements.txt
├── .env.example
└── README.md
```

🚀 DEPLOYMENT INSTRUCTIONS

Step 1: Prepare Your Data & Model

1. **Prepare your full historical CSV**:
   - Merge all PM2.5 data files (2018-2025) into one CSV
   - Format: `createdAt, pm25` columns
   - Example: 61,000+ rows = ~7 years of hourly data
2. **Initialize database with ALL data**:

```bash
# On your local machine
python -m backend.init_db --csv /path/to/all_pm25_data.csv
```

Expected output:

```
📊 Data Range:
  Start: 2018-01-01 00:00
  End:   2025-11-23 14:00
  Total: 61,320 hours (2,555 days)

✅ Database initialized successfully!
  Total measurements: 61,320
  Database size: ~3 MB (estimated)
```

This creates `data/pm25_data.db` with ALL historical data.

3. **Copy your trained model**:

```bash
cp /path/to/best_model.keras models/
```

Step 2: Get API Keys

1. **Taiwan EPA API Key**:
   - Visit: https://data.moenv.gov.tw/
   - Register account → Get API key
2. **Google Gemini API Key**:
   - Visit: https://makersuite.google.com/app/apikey
   - Create API key

Step 3: Railway Setup

3.1 Create Railway Account

1. Go to https://railway.app
2. Sign up with GitHub (required for verification)
3. Verify your account to enable **Full Trial**

3.2 Create New Project

1. Click "**New Project**"
2. Select "**Deploy from GitHub repo**"

3. Connect your GitHub account
4. Push this project to your GitHub repository
5. Select the repository in Railway

3.3 Configure Environment Variables

In Railway dashboard → **Variables** tab, add:

```env
EPA_API_KEY=your_epa_api_key_here
GEMINI_API_KEY=your_gemini_api_key_here
SECRET_KEY=your_random_secret_key
DATABASE_PATH=/data/pm25_data.db
MODEL_PATH=./models/best_model.keras
SITE_NAME=板橋
TZ=Asia/Taipei
FLASK_ENV=production
```

3.4 Create Volume

1. In Railway dashboard, press ⌘**K** (Cmd+K)
2. Type "**Volume**" → Select "**New Volume**"
3. Set:
   - **Mount Path**: `/data`
   - **Name**: `pm25-data` (any name)
4. Attach volume to your service

3.5 Set RAILWAY_RUN_UID

Add this variable to fix volume permissions:

```
RAILWAY_RUN_UID=0
```

3.6 Upload Database

Since Railway volumes can't be accessed directly, you have two options:

Option A: Include in Docker Image (Recommended for initial setup)

1. Keep `data/pm25_data.db` in your repository (remove from .gitignore)

2. Modify Dockerfile to copy database:

```dockerfile
COPY data/pm25_data.db /data/pm25_data.db
```

Option B: Use Railway File Browser Template

1. Deploy the "File Browser" template from Railway
2. Upload `pm25_data.db` to `/data/` directory
3. Delete File Browser service after upload

Step 4: Deploy

1. Railway will automatically detect `Dockerfile`
2. Click "**Deploy**"
3. Wait for build (3-5 minutes)
4. Once deployed, Railway provides a public URL like:

```
https://your-app.railway.app
```

Step 5: Verify Deployment

1. Visit your Railway URL
2. Check console logs in Railway dashboard:

```
⏰ SCHEDULER STARTED
📡 Fetching data from EPA API...
✅ Stored X measurements
🧠 Running model inference...
```

3. Test the dashboard:
   - Current PM2.5 displays
   - Charts load
   - Chatbot responds

🛠️ Troubleshooting

Database Not Persisting

- Ensure `RAILWAY_RUN_UID=0` is set
- Verify volume mount path is `/data`
- Check logs for permission errors

Model Not Loading

- Ensure `best_model.keras` is in `models/` directory
- Check file size < 100MB (Railway limit)
- Verify `MODEL_PATH` environment variable

Crawler Fails

- Verify `EPA_API_KEY` is correct
- Check EPA API status: https://data.moenv.gov.tw/
- Review logs for API errors

Out of Credits

- Railway free trial: $5 for 30 days
- After trial: Upgrade to Hobby plan ($5/month)
- Monitor usage in Railway dashboard

🔄 How It Works

Hourly Cycle

```
00:00 → Crawler fetches new data from EPA API
     → Clean & forward-fill missing values
     → Insert into SQLite (preserves ALL historical data)
     → Model: Read last 720 hours → Predict 24 hours
     → Store predictions in database
     → Frontend auto-refreshes via JavaScript
```

Data Flow

1. **Initialization**: ALL historical data (2018-2025) → SQLite
2. **Hourly Update**: EPA API → New hour → Append to database
3. **Model Inference**: Last 720 hours → Transformer → 24h forecast
4. **RAG Queries**: Full database (2018-current) for chatbot

🧪 Local Development

Setup

```bash
# Clone repository
git clone <your-repo-url>
cd pm25_dashboard

# Create virtual environment
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Create .env file
cp .env.example .env
# Edit .env with your API keys

# Initialize database
python -m backend.init_db --csv data/your_pm25.csv --site 板橋

# Run Flask app
python -m backend.app
```

Access

- Dashboard: http://localhost:5000
- API: http://localhost:5000/api/current

📊 API Endpoints

Endpoint	Method	Description
`/api/current`	GET	Current PM2.5 + next hour prediction
`/api/predictions`	GET	24-hour forecast
`/api/history?hours=168`	GET	Historical data (default 7 days)
`/api/chat`	POST	Chatbot query
`/api/status`	GET	AI health advice
`/api/stats`	GET	Database statistics

📈 Model Details

- **Architecture**: Dual-layer Transformer
- **Input**: 720 hours (30 days) of PM2.5 data
- **Output**: 24-hour predictions
- **Parameters**: 1.36M (~5.18 MB)
- **Training**: Google Colab T4 GPU
- **Performance**: R^2=0.42, MAE=4.60 μg/m^3

🤝 Support

- Railway Docs: https://docs.railway.app
- EPA API Docs: https://data.moenv.gov.tw/
- Gemini API: https://ai.google.dev/

📝 License

MIT License - Feel free to use for your projects!

✅ Deployment Checklist

- [ ] CSV data merged (2018-2025, all years)
- [ ] Database initialized locally (`init_db.py`)
- [ ] Model file ready (`best_model.keras`)

- [ ] EPA API key obtained
- [ ] Gemini API key obtained
- [ ] GitHub repository created
- [ ] Railway account verified
- [ ] Environment variables configured
- [ ] Volume created and mounted to `/data`
- [ ] `RAILWAY_RUN_UID=0` set
- [ ] Database uploaded to Railway
- [ ] Deployment successful
- [ ] Dashboard accessible via Railway URL
- [ ] Hourly scheduler running
- [ ] RAG queries working (test with "What was PM2.5 in 2020?")

Deployment Time: ~20-30 minutes

Good luck! 🚀