ECOTRACK

Kompleksowa Dokumentacja Techniczna
Wersja Enterprise – Final

Spis treści

1. Overview

Ecotrack is an advanced CO_2 emission monitoring application designed for individual users.

The platform provides data collection, analytics, forecasting using machine learning,

security mechanisms (AES encryption, JWT authentication), and export capabilities.

This fully expanded technical documentation includes:

- System architecture
- Data model
- Security specification
- Full API reference
- ML algorithm specification
- UML diagrams (text-based)
- Deployment guide
- Future improvements

2. System Architecture

Ecotrack uses a clean 4-layer architecture:

1. Presentation Layer – HTML, CSS, JS, responsive layout, dark mode, custom charts.
2. Application Layer – Flask controllers, authentication, CRUD operations, logic.
3. Security Layer – PBKDF2, AES/Fernet encryption, JWT.
4. Data Layer – MariaDB on port 3307, SQLAlchemy ORM.

This modular structure increases maintainability and reliability.

3. UML Architecture Diagram

```
+--------------------------------------+

|          User           |
+----------------+----------------------+
               |
               v
+--------------------------------------+
|       Frontend (HTML/JS/CSS)         |
| - Dashboard, Charts, Forms           |
+----------------+----------------------+
               |
               v
+--------------------------------------+
|     Flask Backend (Controllers)      |
| - Routes, Logic, Validation          |
+----------------+----------------------+
               |
               v
+--------------------------------------+
| Security & Processing (AES/JWT/ML)    |
+----------------+----------------------+
               |
               v
+--------------------------------------+
|        MariaDB (port 3307)           |
+--------------------------------------+
```

4. Class Diagram (UML)

```
+----------------------+
|       User           |
+-----------------------+

| id                    |

| email_encrypted       |

| password_hash         |

| created_at            |

+-----------------------+


+---------------------------+

|    EmissionRecord         |

+---------------------------+

| id                        |

| user_id                   |

| category                  |

| value                     |

| note_encrypted            |

| created_at                |

+---------------------------+


+---------------------------+

| PasswordResetToken        |

+---------------------------+

| id, user_id, token        |

| created_at                |

+---------------------------+
```

5. Sequence Diagrams

Login:

User → Frontend → Backend → DB → Backend → Frontend (redirect + JWT)

Add Emission:

User → Form → Backend → DB → Frontend (success)

6. Data Model

USER(id, email_encrypted, password_hash, created_at)

EMISSION_RECORD(id, user_id, category, value, note_encrypted, created_at)

PASSWORD_RESET_TOKEN(id, user_id, token, created_at)

All sensitive fields (email, notes) are AES encrypted using Fernet.

7. Security Model

Passwords:

- PBKDF2-HMAC via Werkzeug

Encryption:

- AES-128 Fernet encryption for emails and notes
- Key stored securely in .env

API:

- JWT HS256 authorization
- Errors logged to error.log

8. API Specification

Endpoint	Method	Auth	Description
/api/emissions	GET	JWT	Returns list of emissions

/api/emissions	POST	JWT	Inserts new emission
/api/emissions/<id>	PUT	JWT	Updates emission
/api/emissions/<id>	DELETE	JWT	Deletes emission
/api/calc	POST	-	Computes CO_2
/api/stats/category	GET	JWT	Category statistics
/api/predict	GET	JWT	ML forecast
/api/docs	GET	-	Swagger UI

9. API Examples

```
POST /api/emissions

Authorization: Bearer <token>

{
  "category": "transport",
  "amount": 15
}

→ Response: {"id": 12, "co2": 3.15}
```

10. Machine Learning Engine

Forecast model steps:

```
1. Aggregate daily sums
2. If 0 entries → predicted = 0
3. If 1 entry → predicted = x * 365
4. Else:
   a, b = polyfit(range(n), values, 1)
   next = a*(n+1) + b
   predicted = next * 365
```

11. Deployment Guide

Requirements:

- Python 3.11+
- MariaDB on port 3307
- pip install -r requirements.txt

Setup:

- Create .env with FERNET_KEY
- Configure DB URI in app.py

Run:

```
python app.py
```

12. Future Improvements

- Admin panel
- Advanced ML models
- CI/CD pipeline
- Email notifications
- Docker deployment for production