REMHART Digital Twin - Complete Setup Guide

Table of Contents

Project Structure

```
remhart-digitaltwin/
├── backend/                    # FastAPI Backend
│   ├── app/
│   │   ├── __init__.py         # CREATED
│   │   ├── main.py             # CREATED
│   │   ├── database.py         # CREATED
│   │   ├── models/
│   │   │   ├── __init__.py
│   │   │   └── db_models.py    # CREATED
│   │   ├── schemas/
│   │   │   ├── __init__.py
│   │   │   └── grid_data.py    # CREATED
│   │   ├── routers/
│   │   │   ├── __init__.py
│   │   │   ├── auth.py         # NEXT
│   │   │   ├── grid_data.py    # NEXT
│   │   │   └── websocket_router.py  # NEXT
│   │   ├── services/
│   │   │   ├── __init__.py
│   │   │   ├── data_generator.py    # CREATED
│   │   │   └── ml_models.py    # FUTURE
│   │   └── utils/
│   │       ├── __init__.py
│   │       └── security.py     # CREATED
│   ├── requirements.txt        # CREATED
│   └── .env                    # CREATED
```

```
│
└── frontend/                    # Django Frontend
    ├── smartgrid/
    │   ├── __init__.py
    │   ├── settings.py          # NEXT
    │   ├── urls.py              # NEXT
    │   └── wsgi.py
    ├── dashboard/
    │   ├── __init__.py
    │   ├── views.py             # NEXT
    │   ├── urls.py              # NEXT
    │   ├── templates/
    │   │   ├── base.html        # NEXT
    │   │   ├── landing.html     # NEXT
    │   │   ├── login.html       # NEXT
    │   │   └── dashboard.html   # NEXT
    │   └── static/
    │       ├── css/
    │       │   └── style.css    # NEXT
    │       ├── js/
    │       │   ├── charts.js    # NEXT
    │       │   └── websocket.js # NEXT
    │       └── images/
    │           └── logo.png     # Use your green logo
    ├── manage.py
    └── requirements.txt         # NEXT
```

Backend Setup

Step 1: Create Project Structure

bash

```
# Create main directory
mkdir remhart-digitaltwin
cd remhart-digitaltwin

# Create backend structure
mkdir -p backend/app/{models,schemas,routers,services,utils}
cd backend

# Create __init__.py files
touch app/__init__.py
touch app/models/__init__.py
touch app/schemas/__init__.py
touch app/routers/__init__.py
touch app/services/__init__.py
touch app/utils/__init__.py
```

Step 2: Copy Backend Files

Copy all the files I've created:

1. requirements.txt
2. .env (update with your database credentials)
3. app/database.py
4. app/models/db_models.py
5. app/schemas/grid_data.py
6. app/utils/security.py
7. app/services/data_generator.py
8. app/main.py

Step 3: Install Dependencies

```
bash
```

```
# Create virtual environment
python -m venv venv

# Activate it
# Windows:
venv\Scripts\activate
# macOS/Linux:
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt
```

Step 4: Configure Database

Update .env file:

```
env

DATABASE_URL=mysql+pymysql://your_user:your_password@localhost:3306/remhart_db
SECRET_KEY=your-super-secret-key-at-least-32-characters-long
```

Frontend Setup

Step 1: Create Django Project

```
bash

cd ..  # Back to remhart-digitaltwin/
mkdir frontend
cd frontend

# Create Django project
django-admin startproject smartgrid .

# Create dashboard app
python manage.py startapp dashboard

# Create necessary directories
mkdir -p dashboard/templates dashboard/static/{css,js,images}
```

Step 2: Frontend Requirements

Create frontend/requirements.txt:

```
Django==4.2.7
httpx==0.25.2
python-dotenv==1.0.0
```

Install:

```bash
pip install -r requirements.txt
```

Database Setup

Create MySQL Database

```sql
CREATE DATABASE remhart_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

-- Optional: Create dedicated user
CREATE USER 'remhart_user'@'localhost' IDENTIFIED BY 'secure_password';
GRANT ALL PRIVILEGES ON remhart_db.* TO 'remhart_user'@'localhost';
FLUSH PRIVILEGES;
```

Initialize Tables

The backend will automatically create tables on first run, or you can manually run:

```bash
cd backend
python -c "from app.database import init_db; init_db()"
```

Running the System

Terminal 1: Start Backend

```bash
cd backend
source venv/bin/activate  # or venv\Scripts\activate on Windows
uvicorn app.main:app --reload --port 8001
```

Access:

- API: http://localhost:8001
- Docs: http://localhost:8001/docs

Terminal 2: Start Frontend

```bash
cd frontend
python manage.py runserver 8000
```

Access:

- Frontend: http://localhost:8000

Terminal 3: Generate Test Data (Optional)

```bash
cd backend
python
>>> from app.services.data_generator import grid_generator
>>> data = grid_generator.generate_time_series(100)
>>> print(f"Generated {len(data)} data points")
```

Default Login Credentials

After first run, these users are created:

Username	Password	Role	Purpose
admin	admin123	admin	Full system access
operator	operator123	operator	Grid operations
analyst	analyst123	analyst	Data analysis
viewer	viewer123	viewer	Read-only access

WARNING: Change these passwords in production!

API Documentation

Authentication

Login:

```bash
```

```
POST /api/auth/login
Content-Type: application/json

{
  "username": "admin",
  "password": "admin123"
}

Response:
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGc...",
  "token_type": "bearer",
  "user": {...}
}
```

Use Token:

```
bash

GET /api/grid/data
Authorization: Bearer <your_token>
```

Next Steps

Immediate (Next files to create):

1. Backend router files (auth.py, grid_data.py, websocket_router.py)
2. Django settings.py configuration
3. Django views.py
4. HTML templates (landing, login, dashboard)
5. CSS styling
6. JavaScript for charts and WebSocket

Phase 2 (After basic setup works):

1. Implement AI/ML models with comments
2. Add more graph visualizations
3. Implement WebSocket real-time updates
4. Add info icons to graphs
5. Create mobile-responsive design

Troubleshooting

Database Connection Error

```
Check .env DATABASE_URL
Ensure MySQL is running
Verify credentials
```

Import Errors

```
Ensure all __init__.py files exist
Check virtual environment is activated
Reinstall requirements
```

CORS Errors

```
Check FRONTEND_URL in backend .env
Verify Django is running on port 8000
Check browser console for specific error
```

Support

For issues or questions:

- Check /docs endpoint for API documentation
- Review error logs in terminal
- Ensure all dependencies are installed
- Verify database is running

Ready for next phase? Confirm and I'll provide all remaining router, view, and template files.