# 📋 Electricity Cutting Down Management System - Implementation Guide

## 📑 Table of Contents

1. System Overview
2. Prerequisites
3. Installation Steps
4. Configuration
5. Running the System
6. Testing
7. Troubleshooting

---

## 🎯 System Overview

### Architecture

The system consists of 4 main components:

```
┌─────────────────────────────────────────────────────────┐
│   Electricity Cutting Down System        │
├─────────────────────────────────────────────────────────┤
│                     │
│  ┌──────────────────────────────────────────────┐  │
│  │ MVC Portal (Dashboard)         │  │
│  │ http://localhost:5000         │  │
│  │ • Real-time Statistics         │  │
│  │ • Incident Tracking          │  │
│  │ • Charts & Visualization        │  │
│  └──────────────────────────────────────────────┘  │
│          ↓          │
│  ┌──────────────────────────────────────────────┐  │
│  │ Console App (Worker)          │  │
│  │ • Data Processing           │  │
│  │ • Cycle Management (10s intervals)  │  │
│  │ • SP Execution            │  │
│  └──────────────────────────────────────────────┘  │
│      ↓     ↓          │
│  ┌─────────────────┐  ┌──────────────────────┐   │
│  │ REST API    │  │ SQL Database    │   │
│  │ :5032      │  │ Electricity_FTA │   │
│  │ • Data Gen   │  │ • Tables     │   │
│  │ • Health    │  │ • SPs      │   │
│  │  Check     │  │ • Functions    │   │
│  └─────────────────┘  └──────────────────────┘   │
│                     │
└─────────────────────────────────────────────────────────┘
```

## Data Flow

1. **API generates test data** → Cutting_Down_A & Cutting_Down_B tables
2. **Console App (Worker)** reads new incidents
3. **SP_Create** processes incidents into processed tables
4. **SP_Close** closes resolved incidents
5. **Portal Dashboard** displays real-time statistics

---

## 📦 Prerequisites

### System Requirements

- **Operating System**: Windows 10/11 or Windows Server 2019+
- **RAM**: 4GB minimum (8GB recommended)
- **Disk Space**: 2GB

### Software Requirements

```
✓ SQL Server 2019 or later (with SQL Server Management Studio)
✓ .NET 6.0 SDK or later
✓ Visual Studio 2022 Community/Professional
✓ Git (optional)
```

### Installation Commands

#### 1. Install .NET SDK:

```powershell
# Check if installed
dotnet --version

# Download from: https://dotnet.microsoft.com/download
```

#### 2. Install SQL Server:

- Download from: https://www.microsoft.com/en-us/sql-server/sql-server-downloads
- Choose "Express" (free) or Developer Edition

#### 3. Install Visual Studio 2022:

- Download from: https://visualstudio.microsoft.com/downloads/
- Select workload: "ASP.NET and web development"

---

## 🔧 Installation Steps

### Step 1: Database Setup

#### 1.1 Create Database

```sql
-- Open SQL Server Management Studio (SSMS)
-- Connect to your server

-- Create the main database
CREATE DATABASE Electricity_FTA;

-- Create the source database (if not exists)
CREATE DATABASE Electricity_STA;
```

### 1.2 Execute SQL Scripts

Execute scripts in **exact order**:

```
📁 Database/
├── 00_Create_Tables.sql          ← Execute 1st
├── 01_Create_Functions.sql       ← Execute 2nd
├── 02_Create_StoredProcedures.sql ← Execute 3rd
└── 03_Sample_Data.sql            ← Execute 4th (optional)
```

### How to Execute:

```
1. Open SSMS
2. File → Open → Select SQL file
3. Press F5 or click "Execute"
4. Check for errors in Output
```

### 1.3 Verify Database

```sql
```

```
-- Check tables created
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_CATALOG = 'Electricity_FTA'
ORDER BY TABLE_NAME;

-- Check stored procedures
SELECT name FROM sys.objects
WHERE type = 'P' AND name LIKE 'SP_%'
ORDER BY name;

-- Check functions
SELECT name FROM sys.objects
WHERE type = 'FN' AND name LIKE 'FN_%'
ORDER BY name;
```

---

## Step 2: API Setup

### 2.1 Navigate to API Folder

powershell

```powershell
cd "path\to\ElectricityCuttingDownManagment.API"
```

### 2.2 Restore NuGet Packages

powershell

```powershell
dotnet restore
```

### 2.3 Update Configuration

Edit `appsettings.json`:

json

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=HABIBA\\SQLEXPRESS;Database=Electricity_FTA;Integrated Security=True;TrustServer
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information"
    }
  }
}
```

**Replace:**

- `HABIBA\SQLEXPRESS` → Your SQL Server name
- `Electricity_FTA` → Your database name

### 2.4 Run API

```powershell
dotnet run
```

**Expected Output:**

```
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5032
      Press Ctrl+C to stop
```

### 2.5 Test API

Open browser and go to:

```
http://localhost:5032/api/CuttingDownA/health
```

Should return:

```json
```

```
{
  "status": "Healthy",
  "service": "Cutting Down A API (Cabins)",
  "timestamp": "2025-11-05T10:30:00.000Z"
}
```

---

## Step 3: Console App Setup

### 3.1 Navigate to Console App Folder

```powershell
cd "path\to\ElectricityCuttingConsoleApp"
```

### 3.2 Restore Packages

```powershell
dotnet restore
```

### 3.3 Update Configuration

Edit `Program.cs`:

```csharp
private static string _connectionString =
    "Data Source=HABIBA\\SQLEXPRESS;Initial Catalog=Electricity_FTA;Integrated Security=True;Trust Server Certificat
private static string _apiUrl = "http://localhost:5032";
```

### 3.4 Run Console App

```powershell
dotnet run
```

**Expected Output:**

```
=== Electricity Incident Worker ===

Testing API connection...
✓ API connection successful

──── Cycle #1 - 2025-11-05 10:30:45 ────

✓ Building hierarchy - Records: 0
✓ [Source A (Cabins)] Success - OK
✓ [Source B (Cables)] Success - OK
✓ Creating incidents - Records: 9
✓ Closing incidents - Records: 0
✅ Cycle completed successfully

⏳ Waiting 10 seconds...
```

---

## Step 4: Portal Setup

### 4.1 Navigate to Portal Folder

```powershell
cd "path\to\ElectricityCuttingDownManagment.Portal"
```

### 4.2 Restore Packages

```powershell
dotnet restore
```

### 4.3 Update Configuration

Edit `appsettings.json`:

```json
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=HABIBA\\SQLEXPRESS;Database=Electricity_FTA;Integrated Security=True;TrustServer
  }
}
```

### 4.4 Run Portal

```powershell
dotnet run
```

**Expected Output:**

```
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
```

### 4.5 Access Portal

Open browser:

```
http://localhost:5000/Dashboard
```

---

## ⚙️ Configuration

### Connection Strings

**For Local SQL Server:**

```
Server=HABIBA\SQLEXPRESS;Database=Electricity_FTA;Integrated Security=True;TrustServerCertificate=True;
```

**For Remote SQL Server:**

```
Server=192.168.1.100,1433;Database=Electricity_FTA;User Id=sa;Password=YourPassword;TrustServerCertificate=True;
```

### API Ports

- **Default**: 5032
- **To change**: Edit `launchSettings.json`

```json
```

```
"profiles": {
  "http": {
    "applicationUrl": "http://localhost:5032"
  }
}
```

### Console App Settings

Edit in `Program.cs`:

```csharp
// Cycle interval (in seconds)
const int cycleIntervalSeconds = 10;

// Number of test data to generate
const int testDataCount = 10;
```

---

## 🚀 Running the System

### Complete Startup Sequence

#### Terminal 1: Start SQL Server

```powershell
# Already running as service (no action needed)
# Or start manually if installed as application
```

#### Terminal 2: Start API

```powershell
cd "path\to\API"
dotnet run
```

#### Terminal 3: Start Console App

```powershell
```

```
cd "path\to\ConsoleApp"
dotnet run
```

**Terminal 4: Start Portal**

```powershell
cd "path\to\Portal"
dotnet run
```

**Browser: Open Portal**

```
http://localhost:5000/Dashboard
```

### Logs Location

- **API**: Console output
- **Console App**: Console output
- **Portal**: Console output
- **Database**: SQL Server Logs (`%ProgramFiles%\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Log\`)

---

## ✅ Testing

### 1. Database Testing

```sql
-- Count total incidents
SELECT COUNT(*) FROM Cutting_Down_Header;

-- Check recent incidents
SELECT TOP 5 * FROM Cutting_Down_Header
ORDER BY ActualCreateDate DESC;

-- Verify stored procedures
EXEC SP_BuildHierarchy;
EXEC SP_Create;
EXEC SP_Close;
```

## 2. API Testing

### Using PowerShell:

```powershell
# Test health
Invoke-WebRequest -Uri "http://localhost:5032/api/CuttingDownA/health"

# Generate test data
Invoke-WebRequest -Uri "http://localhost:5032/api/CuttingDownA/generate-test-data?count=5" -Method Post
```

### Using cURL:

```bash
# Test health
curl http://localhost:5032/api/CuttingDownA/health

# Generate test data
curl -X POST "http://localhost:5032/api/CuttingDownA/generate-test-data?count=5"
```

## 3. Console App Testing

Verify:

- ✓ API connection successful
- ✓ Cycles running every 10 seconds
- ✓ Records increasing in database
- ✓ No errors in output

## 4. Portal Testing

Open http://localhost:5000/Dashboard and verify:

- ✓ Statistics updating
- ✓ Charts displaying data
- ✓ Recent incidents table populated
- ✓ Status colors correct (Green=Resolved, Red=Active, Yellow=Pending)

---

## 🐛 Troubleshooting

### Issue: "Connection refused" to database

**Solution:**

```powershell
# Check SQL Server is running
Get-Service | Where-Object {$_.Name -like "*SQL*"} | Select-Object Status, Name

# If not running, start it
Start-Service -Name MSSQLSERVER

# For SQL Express
Start-Service -Name MSSQL$SQLEXPRESS
```

### Issue: API returns 500 error

**Solution:**

```powershell
# Check logs
dotnet run --verbose

# Verify connection string in appsettings.json
# Verify database exists

# Restart API
```

### Issue: Console App can't connect to API

**Solution:**

```powershell
# Test API is running
Invoke-WebRequest http://localhost:5032/api/CuttingDownA/health

# If fails, ensure API is running
# Check firewall allows port 5032
```

### Issue: Portal shows no data

**Solution:**

1. Ensure Console App is running (generating data)
2. Wait 10+ seconds for first cycle
3. Refresh browser (Ctrl+F5)
4. Check browser console for JS errors (F12)

### Issue: "Stored procedure not found"

**Solution:**

```sql
-- Verify SP exists
SELECT name FROM sys.objects WHERE type = 'P' AND name = 'SP_Create'

-- If not found, re-run the SQL script
-- Ensure you're connected to correct database
```

### Issue: Database date/time incorrect

**Solution:**

```powershell
# Check system date
Get-Date

# Set correct timezone
Set-TimeZone -Id "Egypt Standard Time"

# Or update system time through Settings
```

---

## 📞 Support & Debugging

### Check System Health

```powershell
```

```
# Check all services
$services = @("MSSQLSERVER", "MSSQL`$SQLEXPRESS")
$services | ForEach-Object {
  Get-Service -Name $_ -ErrorAction SilentlyContinue |
  Select-Object -Property Name, Status
}

# Test database connection
sqlcmd -S HABIBA\SQLEXPRESS -Q "SELECT @@VERSION"
```

### Enable Verbose Logging

#### In appsettings.json:

```json
"Logging": {
  "LogLevel": {
    "Default": "Debug",
    "Microsoft": "Information"
  }
}
```

---

## 🎓 Additional Resources

- **SQL Server Documentation**: https://docs.microsoft.com/en-us/sql/
- **.NET Documentation**: https://docs.microsoft.com/en-us/dotnet/
- **ASP.NET Core Documentation**: https://docs.microsoft.com/en-us/aspnet/core/

---

**Last Updated**: November 5, 2025
**Version**: 1.0.0
**Status**: Production Ready ✅