Here's a detailed documentation for your **Carbon Footprint Monitoring Tool** implementation. It includes an overview, explanation of the code with screenshots (conceptually), and examples with input/output.

Carbon Footprint Monitoring Tool Documentation

Overview

This program fetches global CO_2 emission data from an online CSV file, processes the data, and provides insights into CO_2 emissions for a user-specified country. The application generates a report with key statistics such as average emissions, maximum and minimum emission years, and a list of yearly emissions for the chosen country.

Features

1. **Download Dataset**: Automatically fetches the latest CO_2 emission dataset from an online source.
2. **Parse and Analyze Data**: Extracts data for all countries and organizes it by year.
3. **Country Selection**: Lists all available countries for user selection.
4. **Generate Statistics**:
    - Average emissions.
    - Maximum and minimum emissions with corresponding years.
5. **Report Generation**: Saves the analysis in a text file for easy sharing.

Code Explanation

1. Including Libraries

The code imports several standard libraries and an external library for handling HTTP requests (`libcurl`).

#include <iostream>

```
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <map>
#include <set>
#include <cstdlib>
#include <iomanip>
#include <curl/curl.h>
```

2. Data Structures

- **`EmissionData`**: A structure to store a year and its corresponding CO_2 emissions for a country.

```
struct EmissionData {
   std::string year;
   double emissions;
};
```

- **Data Storage**: A `std::map` is used to store emission data for each country, where the key is the country name, and the value is a vector of `EmissionData`.

---

3. Fetching the Dataset

The function `downloadDataset` downloads the CSV file from the specified URL.

```
bool downloadDataset(const std::string& url, const std::string& filename);
```

Working Example

- **Input**: A dataset URL (e.g., `https://raw.githubusercontent.com/...`) and the local filename (`owid-co2-data.csv`).
- **Output**: Saves the file locally.
- **Error Handling**: Reports failure if cURL is not initialized or file writing fails.

---

4. Parsing the Dataset

The `parseCSV` function reads the downloaded file and extracts data into a `std::map`.

```
void parseCSV(const std::string& filename, std::map<std::string, std::vector<EmissionData>>&
data);
```

Key Operations

- **Skip Header**: Skips the first line of the CSV file.
- **Tokenization**: Splits each line by commas and extracts `country`, `year`, and `CO2`.
- **Data Validation**: Ensures the extracted fields are non-empty and converts `CO2` to a floating-point number.

---

5. Displaying Countries

Lists all countries with available data.

```
void displayCountries(const std::set<std::string>& countries);
```

---

6. Analyzing Data

The `analyzeCountry` function calculates:

1. Total emissions.
2. Average emissions.
3. Year of maximum and minimum emissions.

Core Logic

Iterates over the data of a chosen country:

```
for (const auto& record : emissions) {
    total += record.emissions;
    if (record.emissions > max_emission) {
        max_emission = record.emissions;
        max_year = record.year;
    }
    if (record.emissions < min_emission) {
        min_emission = record.emissions;
        min_year = record.year;
    }
```

}

---

7. Report Generation

Writes the analysis and statistics into a text file using `std::ofstream`.

void generateReport(const std::string& country, const std::vector<EmissionData>& emissions, double average, double max_emission, const std::string& max_year, double min_emission, const std::string& min_year);

Example Workflow

Input

1. User chooses "India" as the country to analyze.

Data extracted for India from the dataset:
 Year: 2000, Emissions: 1000.5 Mt
Year: 2010, Emissions: 1200.3 Mt
Year: 2020, Emissions: 1100.1 Mt

2.

Output

Console Output

```
CO2 Emissions for India:
Year     Emissions (Mt)
2000      1000.5
2010      1200.3
2020      1100.1

Statistics for India:
- Average Emissions: 1100.3 Mt
- Highest Emissions: 1200.3 Mt in 2010
- Lowest Emissions: 1000.5 Mt in 2000

Report saved to report.txt
```

Report File (report.txt)

```
CO2 Emissions Report for India
---------------------------------------
Year     Emissions (Mt)
2000      1000.5
2010      1200.3
2020      1100.1

Statistics for India:
- Average Emissions: 1100.3 Mt
- Highest Emissions: 1200.3 Mt in 2010
- Lowest Emissions: 1000.5 Mt in 2000
```

Visualization

To enhance understanding, you can generate visualizations using Python or Excel by importing the report data.

Example Graphs

1. **Line Chart**: CO_2 Emissions Over Years.
2. **Bar Chart**: Emissions by Year.
3. **Summary Table**: Max, Min, and Average Emissions.

Conclusion

This program is a robust and extensible CO_2 emissions analyzer. It demonstrates:

- Efficient use of C++ libraries for file I/O, parsing, and HTTP requests.
- Proper error handling and validation.
- User-friendly interaction and informative outputs.

Further enhancements:

1. Add a graphical interface for visualizations.
2. Integrate a database for real-time queries.
3. Expand functionality for multi-country comparisons.