**Project Title: StationEasyPrice - CSV Data Processing Application**

**Table of Contents**

1. **Introduction**
2. **Project Overview**
3. **Technologies Used**
4. **Project Features**
5. **Code Structure**
6. **Error Handling and Logging**
7. **CSV File Format**
8. **How to Run the Project**
9. **Conclusion**
10. **Appendix**

**1. Introduction**

The ***StationEasyPrice*** project is a .NET-based application designed to read and process station data from CSV files. The core functionality of the application involves parsing station data (including pricing, coordinates, and more) from a CSV file, applying custom logic to handle potential errors, and converting data into appropriate types for further use.

This project was developed using C# and leverages the CsvHelper library for CSV reading and conversion.

**2. Project Overview**

* **Goal:** To process station data from CSV files, including station details, pricing, and geographical information, and convert them into usable objects.
* **Main Features:**
  + Custom converters to safely handle missing or invalid data (e.g., empty latitude/longitude).
  + Error handling to ensure smooth data processing without crashes.
  + Efficient parsing and mapping of CSV data into strongly typed objects (StationData).

**3. Technologies Used**

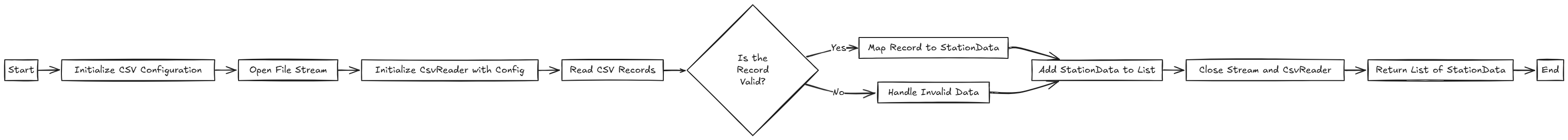
* **Programming Language:** C#
* **Libraries/Frameworks:**
  + CsvHelper (for CSV file parsing)
* **Development Environment:** Visual Studio, .NET Core SDK
* **Error Logging and Handling:** Console-based logging for error identification
* **Version Control:** GitHub (for source code management)

**4. Project Features**

1. **Custom Type Converters:**
   * **SafeDoubleConverter**: Ensures that invalid or empty double values (e.g., for latitude and longitude) are converted to 0.0 rather than throwing an exception.
2. **Error Handling:**
   * **Missing Field Handling:** Configured CsvConfiguration to avoid errors from missing fields using MissingFieldFound = null.
   * **Bad Data Handling:** Invalid or empty fields are handled gracefully without causing application crashes.
3. **CSV Parsing and Data Mapping:**
   * The CSV file is parsed into a list of StationData objects.
   * The StationData class maps CSV columns to properties like station name, latitude, longitude, and pricing information.

**5. Code Structure**

* **CsvService.cs**:
  + Main service responsible for reading and processing the CSV file.
  + Uses CsvReader from CsvHelper to convert CSV rows into StationData objects.
  + Includes logic for custom converters to handle data issues like missing or invalid numerical values.
* **StationData.cs**:
  + Represents the data model for station information.
  + Contains properties like STATION\_ID, SITE\_NAME, LATITUDE, LONGITUDE, etc.
* **SafeDoubleConverter.cs**:
  + Custom converter to safely handle conversion of invalid or missing double values.



**6. Error Handling and Logging**

The application ensures that errors encountered during CSV parsing (such as missing or invalid data) are handled gracefully using the following strategies:

* **Missing Field Handling:** Uses the MissingFieldFound = null configuration in CsvHelper to prevent errors if a column is missing.
* **Invalid Data Handling:** Custom type converters, such as SafeDoubleConverter, default invalid or empty data to 0.0 without throwing exceptions.

**7. CSV File Format**

The CSV file should adhere to the following structure:

STATION\_ID,SITE\_NAME,ZDALY\_GAS\_BRAND,ADDRESS,CITY,STATE,ZIP,COUNTY\_NAME,PRICING\_ZONE,CLUSTER\_MEDIAN\_PRICE,CLIENT\_MARKET\_PRICE,LATITUDE,LONGITUDE

9005,"(T1 Vía Rápida Poniente esq. Pacífico)",Petro 7,"Avenida Via Rapida Poniente No. 13002",Tijuana,Baja California,22320,"ZM Tijuana",,,"7.46","7.33","31.79353","-116.5928"

...

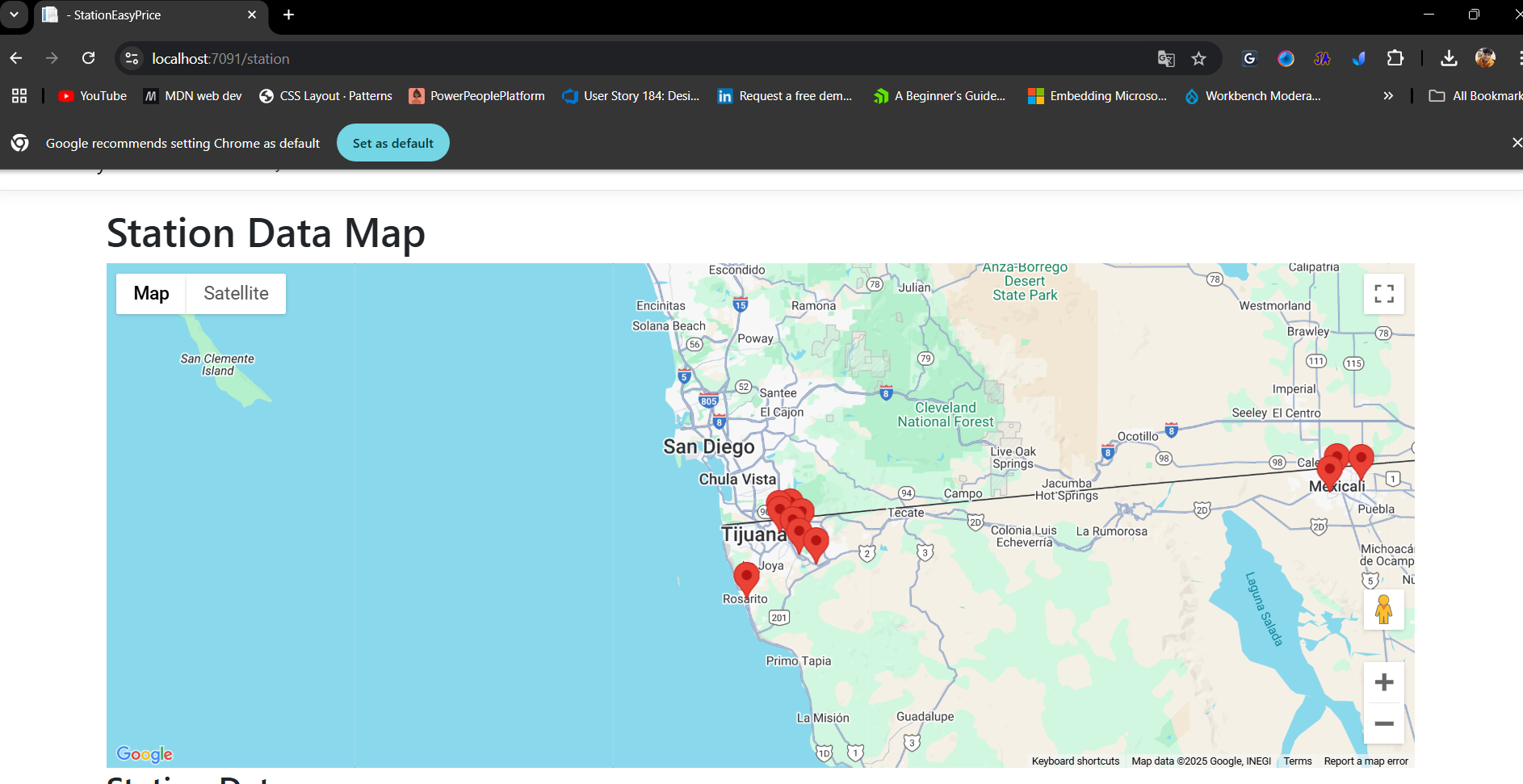
* **STATION\_ID**: Unique station identifier
* **LATITUDE**: Latitude of the station
* **LONGITUDE**: Longitude of the station
* **PRICING\_ZONE**: Pricing zone (optional)
* **CLIENT\_MARKET\_PRICE**: Price data for the station (optional)

**8. How to Run the Project**

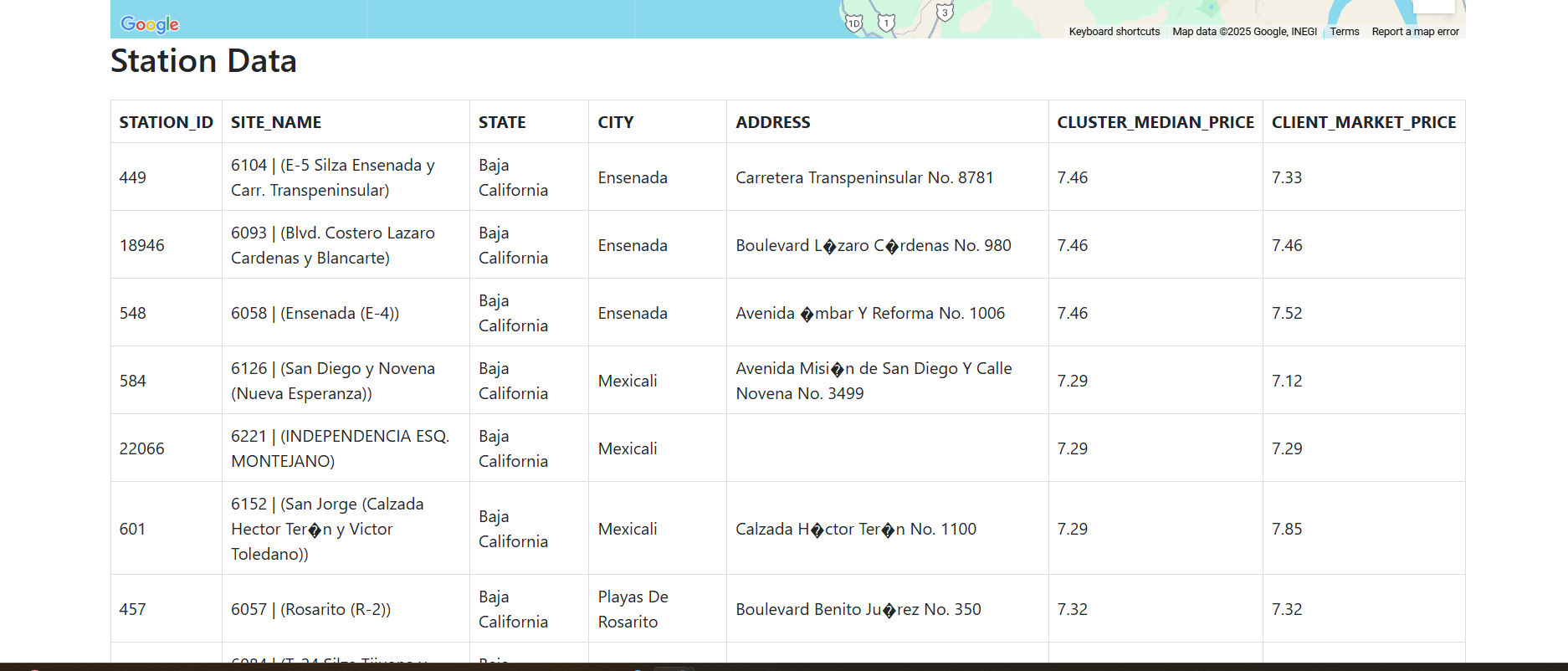
1. **Clone the repository**:
   * Use the command: git clone https://github.com/raguram147/StationEasyPrice.git
2. **Navigate to the project folder**:
   * cd StationEasyPrice
3. **Restore the required dependencies**:
   * Run: dotnet restore
4. **Set the correct CSV file path** in CsvService.cs:

private string \_csvFilePath = "path\_to\_your\_csv\_file.csv";

1. **Run the project**:
   * Use the command: dotnet run
2. **Results:**
   * **Map with data:**

****

* **Stations Data table:**

****

* **High Charts that reflect the Price comparison:**

****

**9. Conclusion**

The ***StationEasyPrice*** project successfully demonstrates the ability to handle CSV files with potential missing or invalid data. By implementing custom converters and error handling, the application can safely parse and process station data. This project is a useful tool for handling CSV-based data processing tasks, especially in cases where data integrity cannot be guaranteed.

**10. Appendix**

* **Sample Data:** A sample CSV file showcasing the expected structure can be found in the project's root directory.
* **Known Issues:** Currently, the application does not support extremely large CSV files (over 1GB). Future improvements can include support for handling large files efficiently.