Create a Console application which is capable for the followings:

1. As a Step 0 check whether the application has its own data files or not.
2. If not, first transform the attached airpots.dat into a handful of .NET classes: Airport, City, Country (See **Appendix** for details)
3. Use Regex to validate the data rows and ignore the invalid ones (No IATA/ICAO code, etc.)
4. Create a log file (using a popular logging framework like Log4Net or NLog) which contains all the wrong rows and a summary row with the number of rows ignored.
5. Use the timezoneinfo.json file to map time zone information to airports and cities (see the TimeZoneInfo class)
6. Every airport should have a FullName property where the value is: name + the “ Airport” string literal
7. Find the two and three letter ISO codes of countries by their English names (See the RegionInfo class)
8. Finally serialize them into JSON files (See **Appendix** for the desired formats)
9. Once your data model is ready and have been successfully serialized to disk (or have been deserialized back from disk!), implement the following queries:
   1. List all the countries by name in an ascending order, and display the number of airports they have.   
      i.e. “Hungary: 5”
   2. Find the city which has got the most airports. If there are more than one cities with the same amount, display all of them.
   3. Find out which airport is at the following GPS coordinate, and display its Name and City:   
      35.764722, 140.386389, 141
   4. Let the user enter an IATA code (validate that the string entered is correct) and display the following information about the given airport: Name, City, Country, the local time and current GMT offset.   
      i.e. “Ferihegy – Budapest, Hungary – Local time: 11:45:00 (GMT+2:00)”

Optional tasks:

1. Hide your logger behind a singleton.
2. Localize the application (introduce at least 2 languages: English and Hungarian).
3. Create a user interface to be able to manage the data: add new entities and modify existing ones.

Remarks:

1. Manage your dependencies with NuGet package manager
2. Use LINQ wherever you can
3. Follow the SOLID principles, for your data model you should create some kind of a DataContext class with generic Airports, Cities, Counties properties and separate methods and classes for all the operations. (i.e. loading/saving data, get query results, etc.)
4. Your Program Main method is allowed to contain UI logic only! (Console reads and writes, etc.)
5. Use meaningful naming
6. Handle exceptions and log all errors occurred during runtime

Appendix

# Contents of airports.dat

* ID
* Airport name
* City name
* Country name
* IATA code
* ICAO code
* GPS location: Longitude, Latitude, Altitude
* *Not defined*
* *Not defined*

# Content definition for airports.json

{

"id": 1,  
 "iataCode": "GKA",  
 "icaoCode": "AYGA",  
 "name": "Goroka",  
 "fullName": "Goroka Airport",  
 "cityId": 1,  
 "countryId": 1,  
 "timeZoneName": "West Pacific Standard Time",  
 "location": {  
 "longitude": 145.391881,  
 "latitude": -6.081689,  
 "altitude": 5282.0  
 }  
 }

# Content definition for cities.json

{  
 "id": 1,  
 "name": "Goroka",  
 "countryId": 1,  
 "timeZoneName": "West Pacific Standard Time"  
 }

# Content definition for countries.json

{  
 "id": 5,  
 "name": "Algeria",  
 "twoLetterISOCode": "DZ",  
 "threeLetterISOCode": "DZA"  
 }