# Home task #1 | .NET Basics | Data Processing

Your task is to create a basic data processing service which allows you to process files with payment transactions that different users save in the specific folder (A) on the disk (the path must be specified in the config). Users can save files at any time, and you have to process them immediately. A file can be either in TXT or CSV format (for CSV you need to skip the first line with headers) with the following content:

<first\_name: string>, <last\_name: string>, <address: string>, <payment: decimal>, <date: date>, <account\_number: long>, <service: string>

*Example (raw\_data.txt):*

*John, Doe, “Lviv, Kleparivska 35, 4”, 500.0, 2022-27-01, 1234567, Water*

*Mike, Wiksen, “Lviv, Kleparivska 40, 1”, 720.0, 2022-27-05, 7654321, Heat*

*Nick, Potter, “Lviv, Gorodotska 120, 3”, 880.0, 2022-25-03, “3334444”, Parking*

*Luke Pan,, “Lviv, Gorodotska 120, 5”, 40.0, 2022-12-07, 2222111, Gas*

### Process flow:

**1. Read.** The file can have tons of lines, so think about the efficiency of your solution. Keep in mind that immediate processing brings a lot of value to the customer (try to avoid delays between processing files).

**2. Validate.** Keep in mind that there can be other files in the folder (A). Ignore everything except TXT and CSV. File name does not matter. Every line in the file may have errors (missing values or invalid types) so ignore those lines. Count all invalid lines and files, and write them down in the “meta.log” file.

**3. Transform.** To process the file, you need to produce the result in a specified format.

*[{*

*"city": "string",*

*"services": [{*

*"name": "string",*

*"payers": [{*

*"name": "string",*

*"payment": "decimal",*

*"date": "date",*

*"account\_number": "long"*

*}],*

*"total": "decimal"*

*}],*

*"total": "decimal"*

*}]*

**4. Save.** When the file is processed, the service should save the results in a separate folder (B) (the path must be specified in the config) in a subfolder (C) with the current date (i.g. 09-21-2022). As a file name, you can use “output” + today’s current file number + “.json”

At the end of the day (midnight) the service should store in the subfolder (C) a file called “meta.log”. The file should have the following structure:

*parsed\_files: N*

*parsed\_lines: K*

*found\_errors: Z*

*invalid\_files: [path1, path2, path3]*

The service can be either a Windows service or a simple CLI tool with start/reset/stop commands. Try to extract business logic from the presentation layer. Do not start the service if the config file is not available or empty. A service should have graceful shutdown. Meaning, you need to handle all errors and show the information in the log (choose between console and file).

As a result, you will have the following structure on a disk:  
*folder\_a/*

*source1.txt*

*source2.csv*

*source3.csv*

*source4.txt*

*folder\_b/*

*09-21-2022/*

*meta.log*

*output1.json*

*output2.json*

*05-19-2022/*

*meta.log*

*output1.json*

*08-07-2022/*

*meta.log*

*output1.json*

### Acceptance Criteria:

1. Use appropriate data structures (List, Dictionary, your own classes)
2. Design your input / output models (do not just handle strings)
3. Try to apply patterns Factory and Strategy
4. Follow [SOLID](https://en.wikipedia.org/wiki/SOLID) principles whenever possible
5. You can use any 3rd-party tool to process the file
6. Prefer using LINQ over loops
7. Use concurrent processing whenever possible

Another skill you should practice is working with **Git and GitHub**. Implement the following git requirements while working on the task:

1. Make at least 3 commits
2. Push commits to the develop branch to your GitHub repository
3. When finished, create a pull request to the master branch
4. Try several git commands
   1. See commit log
   2. See diff between commits
   3. Make some code changes and see git status
   4. Perform reset --hard

<https://git-scm.com/docs>

<https://guides.github.com/introduction/flow/>