|  |
| --- |
| Internship |
|  |
|  |
| Exercise |
|  |
|  |
| Prepared by: |
| Arvid Böttiger, Engineering Manager (AI), A-SQUAD CPH |

Table of Content

|  |
| --- |
|  |
| [Scope 3](#_Toc51149919)  [Rules 3](#_Toc51149920)  [Task requirements 3](#_Toc51149921)  [Setting 3](#_Toc51149922)  [Task 1 – Design and interfaces 4](#_Toc51149923)  [Task 2 – The Reader 4](#_Toc51149924)  [Task 3 – The analyzer 4](#_Toc51149925)  [3.1 Add and reset the analyzer 4](#_Toc51149926)  [3.2 Number of unique cars 4](#_Toc51149927)  [3.3 Identify country of origin 4](#_Toc51149928)  [Task 4 – The Writer 5](#_Toc51149929)  [Optional 6](#_Toc51149930)  [Task 5 – Concurrency 6](#_Toc51149931)  [Additional Bonus Tasks: 6](#_Toc51149932) |

# Scope

This test serves as the foundation of a good and result-driven interviewing process, so that we can onboard the right tech-talent as quickly as possible. Key points that are especially important during the assessment are:

* Clean and orderly code
* Documentation of your thoughts and choices

# Rules

The solution should be written in C# or another major general-purpose programming language. We expect you to dedicate a maximum of 4 consecutive hours for the test. If you do not manage to finalize the test within this timeframe, please prepare some well-documented thoughts on how you would have solved the remaining work. Upon completion, please reply to this e-mail with the submission. During the completion of the test, do not hesitate to contact us if you have any questions.

## Task requirements

* Documentation on how to run your solution
* The solution needs to be written with only stdlib of your language of choice

# Setting

The large bridge connecting Denmark and Sweden - “Øresundsbroen” - has installed Automatic number-plate recognition (ANPR) on all road leading towards and away from the bridge. The number plates are photographed, recognized and logged on a local computer at every road. Every day the log files are transmitted to a server as a text file to be analyzed. The log files are used for both billing cars crossing the bridge as well as general statistics about the traffic on the bridge.

The format of the text files is very simple and contains one number plate pr line.

It is your responsibility to read these text files and process these files accordingly.

* Read all the number plates from a file
* Add the number plates to an analysis service
* Use the analysis service to identify how many cars from various countries are driving though
* Write the number plates of various subsets to a file

# Task 1 – Design and interfaces

Design a set of types and interfaces that represent the components used in the system described above. You should keep the analysis service as a separate module/component. A logical start would be to define the following components

* **A reader:** Which can read a text file and add the entries to an analyzer one by one.
* **An analyzer:** This should only accept on number plate at the time. And only return on number plate at the time when queried
* **A writer:** Which requests number plates from the analyzer and writes them to a text file. One plate pr line.

# Task 2 – The Reader

The file “plates.txt” contains all the plates which was scanned by a camera on the bridge on a given day. The file contains one number plate pr line.

* Implement a reader which read the content of the file into memory.
* Print out the number of number plates in the file

# Task 3 – The analyzer

The analyzer processes the data it receives. It can calculate and outputs various metrics and summaries.

## 3.1 Add and reset the analyzer

Implement at simple analyzer which is a separate component of your program. You should be able to

* Add a single number plates to the analyzer, one at the time
* Be able to reset the state of the analyzer

## 3.2 Number of unique cars

During a day, some cars will cross the bridge a single time, and some will cross it multiple times.

* Create a function/method to request the number of cars provided to the analyzer since last reset .

## 3.3 Identify country of origin

Cars from different countries are billed using different systems. In order to accommodate this, you need to be able to extract all cars from a specified country.

1. **Swedish** number plates have tree letters (a-z) followed by three numbers. For instance: “ABC 123” or “UGH 342”.
2. **Danish** number plates have two letters (a-z), followed by two number (0-9) followed by three numbers (0-9). For instance: “RX 54 383” or “AY 82 283”.

Most cars crossing the bridge is from either Denmark or Sweden. However, there will also be cars other countries. In addition, sometimes the cameras and image-to-text algorithms fails and will misread a plate and write some malformed data to the log file.

* Create a new call to the analyzer, called “GetPlates” which accepts a country as argument.
* Limit the “GetPlates” function to return at most 100 plates when called

If there are more than 100 plates from the specified country, you need a way to retrieve them as well. The analyzers GetPlate function should have an offset-argument which allows to call it multiple times. If there are no more plates it should return an empty set.

* Add a second “offset” argument to “GetPlates”, which skips the first “offset” cars before returning
  + **Example:** There are 500 cars in the analyzer
    - When **offset=0** GetPlates returns car 0 to 99
    - When **offset=100** GetPlates returns car 100 to 199
    - When **offset=480** GetPlates returns car 480 to 499

Some of the entries in the file will come from foreign cars with different number plates and some will be malformed due to errors in the image-to-text algorithm in the camera.

# Task 4 – The Writer

The billing department wants a list of all unique Swedish which have driven over the bridge one day.

The software department which implements the image-to-text analysis in the cameras wants a list of all unique (valid and invalid) Swedish number plates in order to diagnose and fix potential problems in their software.

Create a writer which can get data from the analyzer and write it to an output file. An output file should have the following format:

1. The first line should include the number of entries in the file
2. The following lines should include a list of plates, one plate pr line.

Use it to create the following two text files:

* One output file with all unique and valid Swedish number plates
* One output file with all unique number plates in the analyzer – valid and invalid.

# Optional

**If you made it this far feel free to spend the rest of the time polishing/optimizing your existing code. You are also welcome to continue to Task 5 and 6.**

# Task 5 – Testing

In order to verify your solution, you will need to write a number of unit tests

* Create a small dataset suitable for testing your solution, and all of it’s edge cases
* Write a set of test-cases which tests the integrity of your solution
  + For this you are allowed to use any unit testing framework, but a simple method/function which throws an exception is also fine.

# Task 6 – Concurrency

In the folder “cameras” there are 500 text files with number plates. Simulate the scenario where the cameras are added data to the analyzer live by creating 500 readers which reads one file each and post the plates to the analyzer concurrently.

# Additional Bonus Tasks:

* Instead of having a single analyzer service, create 10 analyzer services and distribute the data among them
* Use one process for the reader/writer and one for the analysis service
* You have a container based deployment strategy for the service
* Documentation on design choices and alternatives
* Scaling strategy

At the technical interview you may be asked to demo your solution on your own computer.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |
|  | Milestone Systems is a leading provider of open platform video management software; technology that helps the world see how to ensure safety, protect assets and increase business efficiency. Mile-stone enables an open platform community that drives collaboration and innovation in the develop-ment and use of network video technology, with reliable and scalable solutions that are proven in more than 500,000 sites worldwide. Founded in 1998, Milestone is a stand-alone company in the Canon Group. |