1. Introduction: AOE Backend Challenge

This challenge is intended to get an idea of the knowledge and experience you have as a developer. We would like to give you the opportunity to showcase your knowledge and skills based on different requirements. Please read through this document carefully before starting and have fun solving our backend challenge!

Please implement the exercise using one of: java, kotlin, scala, go, php

1.1. Challenge Overview

Our coding challenge is divided into 3 tasks which will be explained in more detail in chapter 2. To give you an first impression, here are the tasks that will be required to be implemented:

- String *encryption*
- Domain modelling (with filter logic)
- Webservice combining the domain model, filter logic and encryption

Depending on your knowledge, experience and your available time you could consider skipping the third challenge, i.e. if you apply as a working student and haven't touched any webservice yet.

Like with everything: Don't stress yourself too much. If you notice, that you spend too much time or can't get a grip around a task, there's no shame in skipping an exercise. Just let us know in the *readme* why you might have decided not to implement it.

1.2. Evaluation Criteria

As this challenge is used to assess your experience and skill level, we want to be transparent about our evaluation and give you some hints about what is important to us. The following points in no particular order are some of the criteria we will analyse:

- *Understandable* code (naming, structure, readability)
- Project structure (package structure, reusability)
- *Meaningful* unit test coverage

1.3. Submission

Please provide your implementation via github (private repository, shared with: *code-review-backend@aoe.com*).

Ensure to include the following things in your submission:

- Source code
- Readme
    - Short explanation of how to run your code
    - Optional: Notes on specific tasks if you want to clarify or explain something

Please provide all of your text documents as either *txt*, *md* or *pdf* files.

2. Coding Challenge

Since the number of superheroes in our world is growing bigger and bigger, a company called *"DeeSee Comics"* has instructed us to build a superhero distribution platform to keep track of their employed superheroes. The first feature of the new platform will be a webservice that provides a way to store and retrieve superheroes. Superheroes can be filtered by their powers and their true identity can be encrypted to further provide protection for the ones enrolled in the platform.

***Note:** If you decide **not** to implement the webservice (**2.3**), please provide an executable code fragment, in which you showcase your implementation of **2.1** and **2.2**. Therefor we recommend to just create a few superheroes, filter them by their superpowers and print the resulting superheroes with encrypted identities.*

2.1. Identity Encryption

Beginning with the most critical requirement: *DeeSee* requires the encryption of the true identity of their superheroes. Therefore it is essential to provide a function that takes an identity and returns an encrypted identity. Contrary to our recommendation, DeeSee insists on using a proprietary encryption called the *"DeeSee Chiffre"*.

To encrypt a string using the "DeeSee Chiffre", you will need to shift each letter of the identity by a key ***n***. For the sake of this exercise, you can assume that identities only contain lowercase characters and spaces. The following example might help you understand the algorithm:

Input	Key	Output
clark	5	hqfwp
cherry blossom	3	fkhuub eorvvrp

***Note**: Please make sure to implement the algorithm by yourself without using available solutions from the internet / libraries.*

2.2. Domain Model

Before continuing to the webservice, we will need a domain model to efficiently work with our stored superheroes. The following json contains an example of a superhero which can be used to infer the domain model.

```
{
    "name": "superman",
    "identity": {
      "firstName": "clark",
      "lastName": "kent",
    },
    "birthday": "1977-04-18",
    "superpowers": ["flight", "strength", "invulnerability"]
}
```

To keep the system as simple as possible, *DeeSee* has agreed to **only** accept superheroes with the following superpowers: *"strength", "speed", "flight", "invulnerability", "healing"*

2.3. Superhero Webservice

For the core of the new distribution platform, we will need a webservice that will handle all the superheroes via an RESTful API. The microservice needs to provide possibilities to allow *DeeSee* to store and retrieve their superheroes.

It is up to you whether you implement the webservice by hand or use a framework of your choice.

***Hint**: Using a database for storing the data is **not** required **nor** encouraged. Keep it as simple as possible for this use-case. Data storage is not part of the evaluation.*

***Hint**: Test data should be loaded on startup. Therefor we have provided you a json file containing some sample data you can use. How you use the json is up to you. Chose the easiest solution and avoid wasting time for this as it not part of the evaluation either.*

2.3.1. Superhero Retrieval

To allow *DeeSee* to quickly find matching superheroes for their crime prevention system, the webservice is required to provide an interface to quickly retrieve superheroes using *HTTP*.

Based on DeeSee's requirements, the service must be able to provide all superheroes at once as well as provide all matching superheroes given a required superpower. In addition, our customer wants to have the possibility to tell the service to have all returned superheroes' identity encrypted (using **2.1**).

Instead of returning a superhero's identity as the composed object described in **2.2**, DeeSee has requested to return the identity as a string in the form of "$firstName $lastName".

The following list shows some usecases that should be supported:

- Retrieve all superheroes
- Retrieve all superheroes with encrypted identities
- Retrieve superheroes that match given superpower(s)
- Retrieve superheroes that match given superpower(s) with encrypted identities