Mandatory Assignment I – SD20w1

General Information

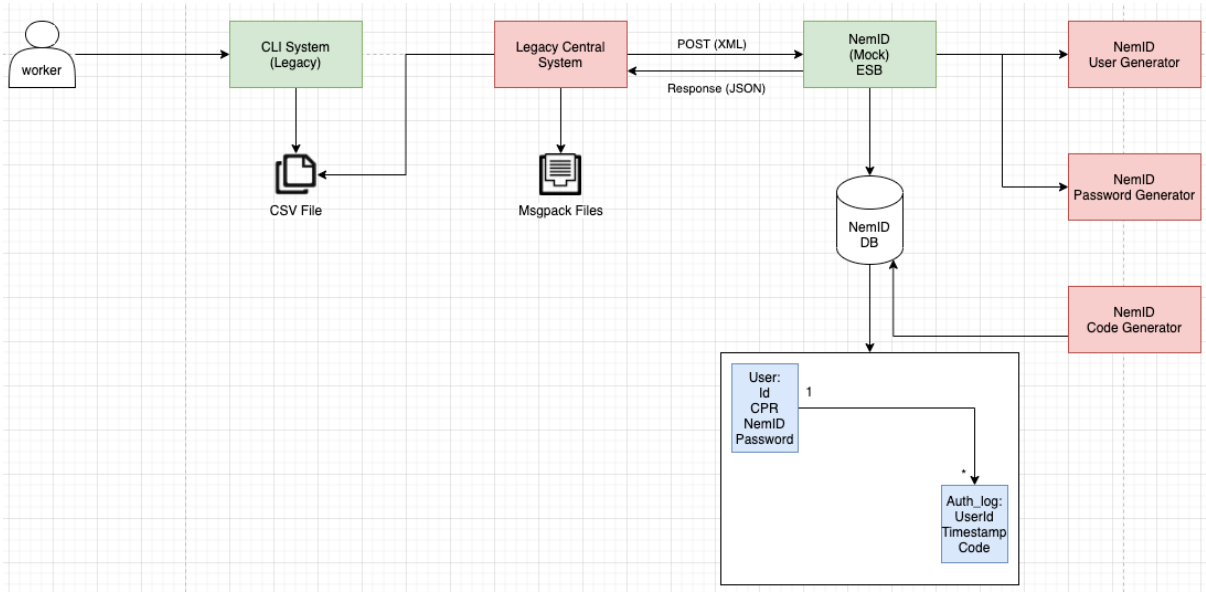
The time has come for you to apply the knowledge that was taught thus far and complete your first mandatory assignment. I hope that by now you have a group – if not please join/create one. (Fronter)

The purpose of this assignment is to showcase different scenarios and integration techniques in a way that contributes to the creation of a solution that incorporates file-based integration, shared databases as well as RPC (web services and RESTful APIs) integration patterns.

You can find the starting code base here:
https://bitbucket.org/_bogz_/si_mandatory_assignment_1/src/master/

It is expected of you to work together, learn something and of course have fun. Please be aware of all the rules and guidelines regarding working together in terms of COVID-19. While I cannot ensure that you will not be in close contact with each other (Less than 1 meter distance) outside of school, while working in a group in KEA you have to wear a mask, should you be in a less than 1 meter proximity to one of your group mates and under no circumstances you are to use another person's computer.

System Diagram:



The systems in green will be given by me. Your job is to develop the four systems (Legacy Central System, NemID User Generator, NemID Password Generator and NemID Code Generator)

A description of each system will be given and since you will also get the source code, it should be fairly simple to implement these systems based on the code snippets and example from your group's repository: https://bitbucket.org/_bogz_/sd20w1/src/master/

Task and System description:

1. Client CLI System (Will be provided):

- Takes input from keyboard (f_name, l_name, birth_date[DD-MM-YYYY], email, country, phone, address)
- Validates the input – Checks that the date is in the right format and that the other fields are not empty
- Stores it in a csv file called people.csv (Inside of the Central Legacy System folder)
- The system will always override the existing file upon new executions of the program

2. Legacy Central System (To be developed):

- The system must read the people.csv file
- For each person that is found in the file it will:
    - Generate a CPR similarly to how a normal CPR looks: ddMMyyy-[random-4-digits]
    - Build an xml body that contains the first name, last name and CPR number

```xml
<?xml version="1.0"?>
<Person>
    <FirstName>Tanmay</FirstName>
    <LastName>Patil</LastName>
    <CprNumber>1234567890</CprNumber>
    <Email>tanmaypatil@xyz.com</Email>
</Person>
```

    - Send a POST request to http://localhost:8080/nemID with the XML as a body
    - The NemID system will return a JSON body:

```json
{
  "nemID": "some 9 digit nemID"
}
```

    - An msgpack file will be created with the name [CPR].msgpack which will contain f_name, l_name, birth_date[DD-MM-YYYY], email, country, phone, address, CPR and NemID number. I suggest you make a JSON object and then serialize it.

3. NemID ESB (Will be provided):

- Has 3 endpoints:
    - http://localhost:8080/nemID
    - http://localhost:8080/generate-4-digit-code
    - http://localhost:8080/change-nemid-password
- Receive a POST request to endpoint http://localhost:8080/nemID with an XML body of format

```xml
<?xml version="1.0"?>
<Person>
    <FirstName>Tanmay</FirstName>
    <LastName>Patil</LastName>
    <CprNumber>1234567890</CprNumber>
    <Email>tanmaypatil@xyz.com</Email>
</Person>
```

    - Make a JSON message of format:

```json
{
  "cpr": "some 10 digit CPR",
  "email": "some@email.com"
}
```

    - Send a POST request to http://localhost:8088/generate-nemID with the aforementioned message as body

- Send a POST request to http://localhost:8089/generate-password-nemID with the following body:

```
{
  "nemId": "random_5_digit_number-Last_4_digits_of_CPR",
  "cpr": "cpr_number"
}
```

- Based on the response, it will save the new user in a database with two tables:
  - User:
    - ID – primary key
    - CPR – TEXT NOT NULL
    - NemID – TEXT NOT NULL
    - Password – TEXT NOT NULL
  - Auth_log:
    - Id – primary key
    - UserId – Foreign Key from USER table
    - Auth Code
    - Timestamp – When the request was made

- The other endpoint will simply check if the credentials match and make a POST request to http://localhost:8090/nemid-auth with a JSON body of format:

```
{
  "nemIdCode": "code of 4 digits",
  "nemId": "generated 9 digit nemID"
}
```

- This will return a JSON body:

```
{
  "generatedCode": "random 6 digits code"
}
```

- Whenever a new code is generated, an entry in the database is made in the table auth_log. Keep in mind that the timestamp column does need to be provided…

- Receive a PATCH request to http://localhost:8080/change-nemid-password with body

```
{
  "cpr":"cpr-number",
  "newPassword": "some 4 digit code"
}
```

- The Mock ESB will simply update the password from the database.
- It will send a status code 204 and no body.

4. NemID User Generator:

- Will receive a POST request to http://localhost:8088/generate-nemID with body:

```
{
  "cpr": "some 10 digit CPR",
  "email": "some@email.com"
}
```

- Will return a JSON response (status 201):

```
{
  "nemId": "random_5_digit_number-Last_4_digits_of_CPR"
}
```

5. NemID Password Generator:

- Will receive a POST request to http://localhost:8089/generate-password-nemID with body:

```
{
  "nemId": "random_5_digit_number-Last_4_digits_of_CPR",
  "cpr": "cpr_number"
}
```

- Will send a JSON response (status 200):

```
{
  "nemIdPassword": "first 2 digits of nemId and last 2 digits of the cpr"
}
```

6. NemID Code Generator:

- Will receive a POST request at http://localhost:8090/nemid-auth with JSON body

```
{
  "nemIdCode": "code of 4 digits",
  "nemId": "generated 9 digit nemID"
}
```

- Check against the data from the database. If it matches this will return a JSON body with status code 200. Otherwise it will return a 403 (forbidden):

```
{
  "generatedCode": "random 6 digits code"
}
```

Deadline and submission:

The deadline is Friday, 2nd of October at 11:59 PM. The deliverables for this assignment are a document that contains a link to your repository, as well as who implemented what part

of the system. Feel free to leave any additional feedback that you might have. For example, what went well and/or What could have been done better.

You will submit your assignment through fronter. Make sure that the repository is public. During week 41 we will look at the mandatories and reflect upon them. Some questions might occur from my side as well ☺.