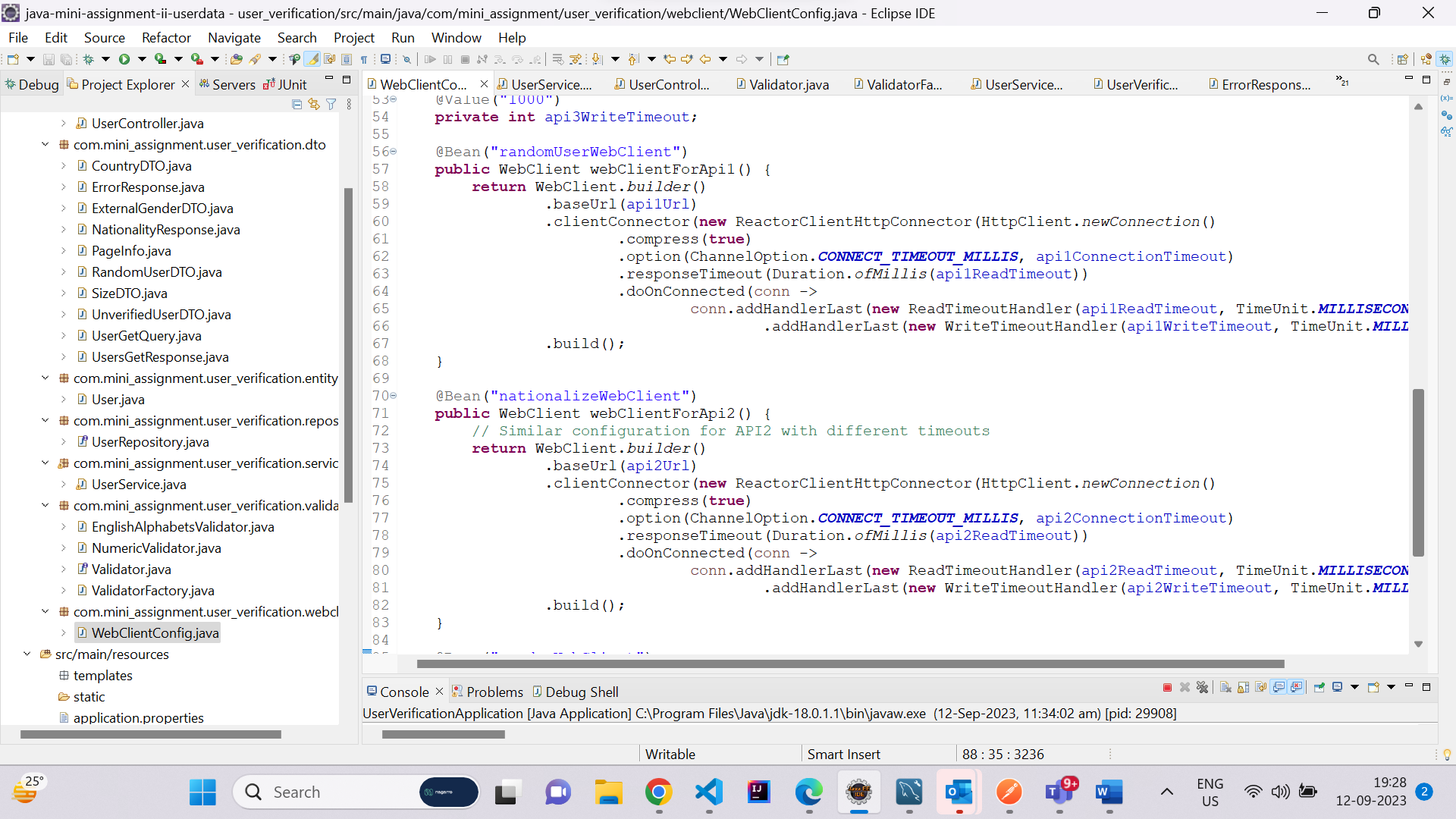
**Java Mini Assignment 2 Documentation –**

* Steps given were followed as given in the pdf.
* As mentioned 3 webClient beans were created to call 3 external APIs shown below –



A screenshot of a computer

Description automatically generated

* For POST request as mentioned in the pdf –
  + A service was created which uses webclient bean to call the random user api and return a randomUser or a list of randomUsers based on the payload { size: }.
  + A service to get nationality was created to retrieve the gender from external api using webclient bean as mentioned in the pdf. (async using completable future and executorService).
  + A service to get gender was created to retrieve the gender from external api using webclient bean as mentioned in the pdf. (async using completable future and executorService).
  + A service was created to verifyUser which matches the nationality and gender with the random user as mentioned.
  + A service was created to save a user or a list of users in the database.
* Working POST API is shown below

A screenshot of a computer

Description automatically generated

A post request with payload {size: 2} is shown below using postman.

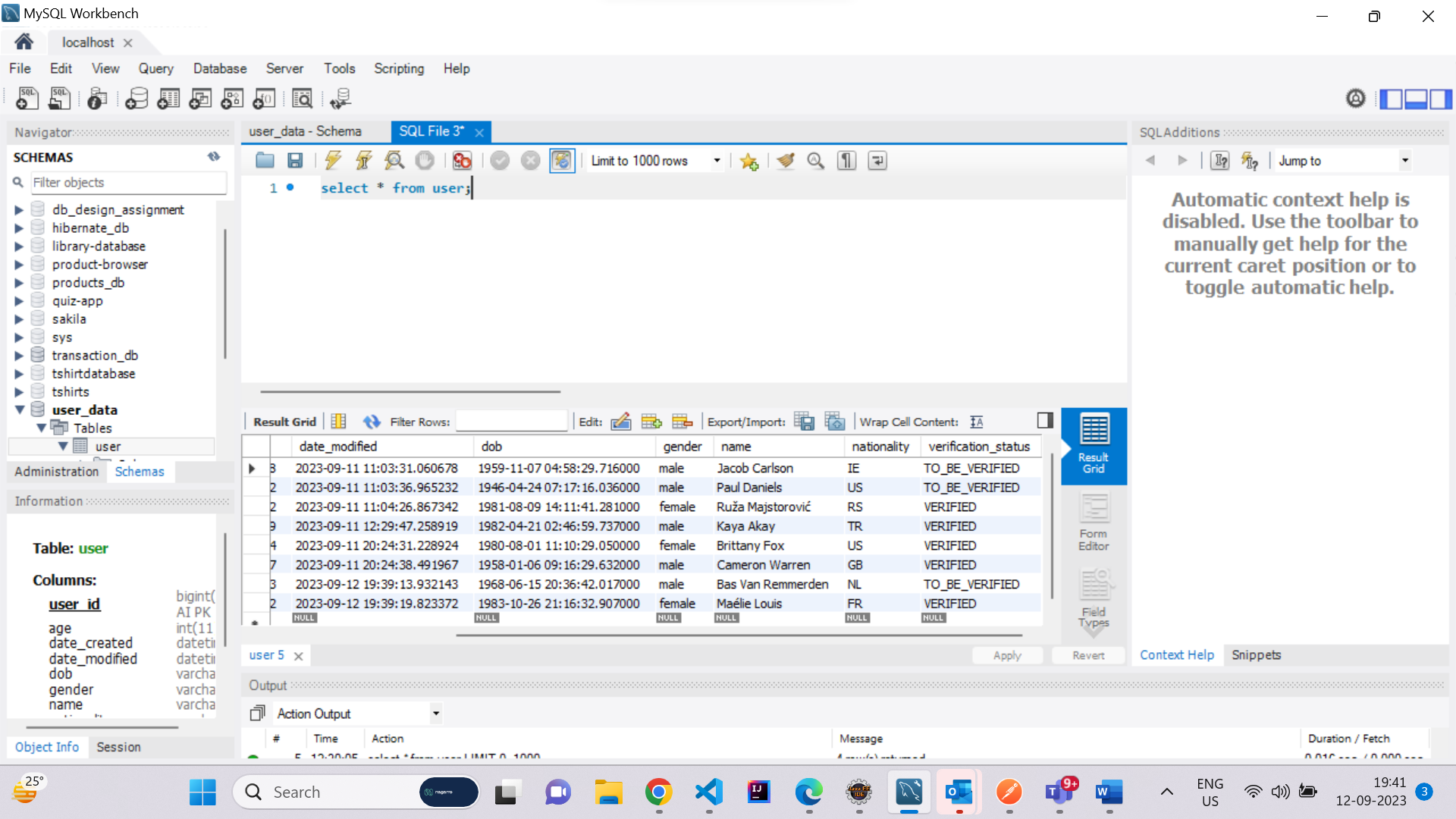
The response is given below –

A screenshot of a computer

Description automatically generated

2 Users are saved in the database with verification status.

You can see the database below –



* FOR GET REQUEST

2 Validators were created using Factory Pattern and implemented in the query parameters for the get request.

Singleton pattern was also implemented in the project.

One instance of Singleton pattern is shown below –

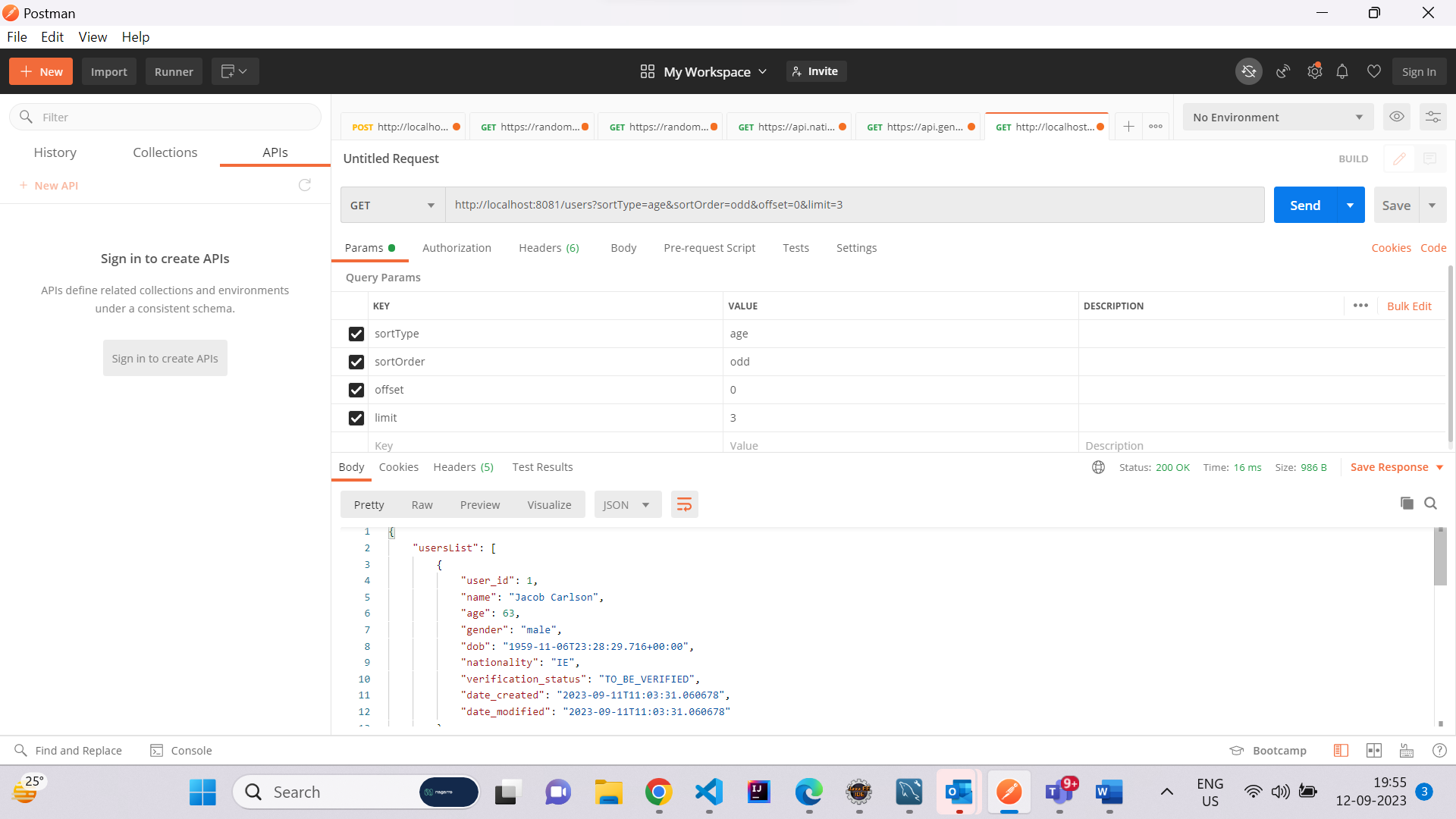
A screenshot of a computer

Description automatically generated  
(we use getInstance method to get the instance of the class so it is not created/instantiated again)

Similarly Design patterns were followed in the project.

* 4 Query Parameters were taken as ModelAttribute In the Controller.
* Validated using the validators mentioned in the pdf.
* Comparators were created for sorting according to the conditions given in the pdf.
* PageRequest was used to handle the the limit and offset. Items per page and if it has next page or previous page was handled.

BELOW SHOWN is GET REQUEST –



Get request was sent using postman with the given params as shown

Here is the response –

{

"usersList": [

{

"user\_id": 1,

"name": "Jacob Carlson",

"age": 63,

"gender": "male",

"dob": "1959-11-06T23:28:29.716+00:00",

"nationality": "IE",

"verification\_status": "TO\_BE\_VERIFIED",

"date\_created": "2023-09-11T11:03:31.060678",

"date\_modified": "2023-09-11T11:03:31.060678"

},

{

"user\_id": 2,

"name": "Paul Daniels",

"age": 77,

"gender": "male",

"dob": "1946-04-24T01:47:16.036+00:00",

"nationality": "US",

"verification\_status": "TO\_BE\_VERIFIED",

"date\_created": "2023-09-11T11:03:36.965232",

"date\_modified": "2023-09-11T11:03:36.965232"

},

{

"user\_id": 3,

"name": "Ruža Majstorović",

"age": 42,

"gender": "female",

"dob": "1981-08-09T08:41:41.281+00:00",

"nationality": "RS",

"verification\_status": "VERIFIED",

"date\_created": "2023-09-11T11:04:26.867342",

"date\_modified": "2023-09-11T11:04:26.867342"

}

],

"pageInfo": {

"hasNextPage": true,

"hasPreviousPage": false,

"total": 8

}

}

Person with even age is at the last. Now if I change the param to even. –

A screenshot of a computer

Description automatically generated

You can see the first user is now with even age then others are shown.

In page info it tells us the details mentioned in the pdf.

All details were successfully completed.