&frankly Technical Assignment

Assignment A

Implement a simple swagger-based API (rest/json) backed by a very simple MySQL database that

a) Enables a user to authenticate himself with an email & password, and generates / returns a session token in return.

b) implements two basic functions: `addComment` and `getComments` that respectively i) adds a string to a table of comments or ii) gets the last 20 comments stored in the table. Both methods should accept a session token in its header to validate that the call should be allowed.

Together with the implementation, please provide a short summary of what considerations you have made/should be made with regards to security, how you would architect this API (if implemented with additional methods, used across customers' accounts) would scale to millions of requests, as well as what architectural patterns could be used in the implementation of the `addComment` and `getComments` functionality to make them extremely performant (low latency).

Or

Assignment B

Implement an API (preferably with a Swagger-definition) that allows export and import of a list of users on a standard format (e.g. csv) with basic user information (firstname, lastname, email (unique), role) from some type of data storage, preferably with a basic underlying object oriented library to handle User objects. It should be possible to export a list, make changes to the list (e.g. update names & roles, add & remove users), and import the list which would a) validate that the input file & data in it is correct and b) enact the changes that is represented in the list if valid or c) provides a list of errors found if not.

E.g. if an exported list looks like this:

```
Anders Andersson aa@aa.se

Bertil Bertilsson bb@bb.se
```

And it is changed to:

```
Anders Eriksson aa@aa.se

Christer Carlsson cc@cc.se
```

Then the lastname of Anders Andersson should be changed, Bertil should be removed and Christer be added.

Together with the implementation, please provide a short summary of possible errors/edge cases you have not yet implemented that may arise, and reason around how this type of function could be made robust in a setting where the lists could be very large/take a long time to process, requests may timeout etc.