Back-end Homework

Pick one of this topics for the test:

- Restaurants
- Movies
- Pokemon
- Music

Implement an HTTP server API that fulfills next requirements:

Requirements:

1. Create a registration service that receives an email and a password.
    a. Validate email is a valid email address.
    b. Validate email is not already registered in the database.
    c. Validate password contains at least 10 characters, one lowercase letter, one uppercase letter and one of the following characters: !, @, #, ? or].
    d. If any of the above is not valid, send back a meaningful response.
2. Allow login into the server with an email and a password.
    a. Validate email is a valid email address
    b. Validate email is already registered in the database
    c. Validate password contains at least 10 characters, one lowercase letter, one uppercase letter and one of the following characters: !, @, #, ? or].
    d. Validate email and password matches for a previous registered user.
    e. If any of the above is not valid send back a meaningful response.
    f. If all of the above are valid send back a payload including some way for users to identify themselves for subsequent requests. That way to identify users should be invalid after 20 minutes and the user must login again to continue communication with the server.
3. Allow logged in users to do CRUD operations into a table/collection of the topic you picked above.
    a. Users should be able to create a new element that can only be retrieved by themselves (Private item).
    b. Users should be able to read all public elements in the table/collection.
    c. Users should be able to read all elements created by themselves.
    d. Users should be able to edit at least one field in one of their private items.
    e. Users should be able to delete their private items by id or all at once.
    f. Users should be able to like one of the public elements in the table/collection
    g. Users should be able to retrieve a list with all their liked public elements.
    h. Validate that users are trying to read, update or delete their own private elements, otherwise send a meaningful response.

Must have for this test:

1. A git repository with the code and a README.md explaining how to run the code in the reviewer computer with very clear steps.
2. Create the models for the selected topic with at least 5 meaningful fields.
3. Prefill the public elements with a list you built previously
4. Read requests must support pagination

Nice to have for this test:

1. Implement what you would consider a good architecture.
2. Implement requirements using well known standards.
3. Create a meaningful documentation for potential clients that would use the API
4. Unit testing
5. Add an endpoint that requires your server to retrieve a random number from a public API and send it back to the user.
6. **Bonus**: The API is deployed in a publicly accessible URL