**Lab3**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

We are going to extend the past app capabilities,

1 - Use database instead of files.

2- Create two models (user, todo)

User : {

username: String, required,unique

password : String, required,

firstName: String,required, min length 3, max length 15

age: Number, min 13

}

Todo {

userId: the ObjectId of the user how put ref in schema search about it

title: String, required, min 5, max 20, indexed

status: String, optional, default is “to-do”

createdAt: Date,

}

**search for validation schema**

3 - implement the following end points

|  |  |  |
| --- | --- | --- |
| HTTP Method | route | Description |
| post | /users/register | - Register a user with the following **required** attributes Username,password , firstname  Notes:  - Return ({message:”user was registered successfully”}) if success  - Handle validation errors returned from mongo |
| Post | /users/login | Return ({message: "logged in successfully" ,username ,todos } )  If the the authentication failed  Return ({error:”invalid credentials” }) with 401 status code |
| GET | /users | Return the first name of registered users |
| DELETE | /users | Delete the user |
| PATCH | /users | - Return ({message:”user was edited successfully”, user: theUserAfterEdit”}) if success  - Handle validation errors returned from mongo |
| POST | /toods | Create new todo  ({username, title,tags})  Return the new todo to the user |
| GET | /todos/:userId | Return the todos of specific user |
| PATCH | /todos/:id | Edit todo |
| DELETE | /todos/:id | Delete todo |