**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”

tags:[String], optional, max length for each tag is 10

createdAt: Date,

}

**make controllers to all crud operations**

3 - implement the following functions in controllers files and call it in index file

|  |  |
| --- | --- |
| Functions | Description |
| 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 |
| login() | Return ({message: "logged in successfully" ,username ,todos } )  If the the authentication failed  Return ({error:”invalid credentials” }) with 401 status code |
| getAllUsers() | Return the first name of registered users |
| deleteuser(email) | Delete the user |
| editUser(email, editvalue) | - Return ({message:”user was edited successfully”, user: theUserAfterEdit”}) if success  - Handle validation errors returned from mongo |
| createTode(data) | Create new todo  ({username, title,tags})  Return the new todo to the user |
| getTodos(userid) | Return the todos of specific user |
| editTode(todoid,editdata) | Edit todo |
| deleteTodo(todoid) | Delete todo |