National University of Computer and Emerging Sciences

Lab Manual

for

Web Engineering (SL3003)

Course Instructor	Mr. Ayaz Gillani
Lab Instructor	Ahmad Jawad Mustasim
Lab Demonstrator	Mohid Jillani
Section	6A
Semester	Spring 2025

Department of Software Engineering

FAST-NU, Lahore, Pakistan

Lab 14 – Session handling and middleware etc.

Objectives

- This lab is second part of lab 13
- Storing password in hash, session handling through middleware

Doraemon Gadget Secured Center

Use following tools/soft wares:

- VS Code
- Postman
- MongoDB Compass

Setup your project:

- Firstly, setup node project: **npm i init –y**
- Now install express: **npm install express**
- Now install mongoose for mongodb: **npm install mongoose**
- For automatic restart of server after you save something after updating your code, you can install nodemon: **npm install --save-dev nodemon**
- Dependency for encryption and decryption of password: **npm install bcrypt**
- For token based authentication: **npm install jsonwebtoken**

File structure of your project should be:

- Same as lab 13, just add folder for middleware.

Lab Tasks

Note:

- Test all your APIs on POSTMAN and attach screenshots also.

1. Copy code of your previous lab and implement following modifications
2. Setup an Express server with express.json() and connect it to MongoDB using Compass.
3. Change character and gadget objects into mongoDB collections.
4. Add password attribute in character collection
5. In register a new character now password field will also be required. And it should be stored in hash using bcrypt.
6. Implement a route POST /login to login a registered character.
7. Implement middleware for user authentication using JWT.
8. Modify route POST /gadgets (gadget should only be added when robot type character is logged-in)
9. No need to modify route GET /characters that returns all characters. (anyone can access characters even though not logged-in)
10. Modify route GET /gadgets that returns all gadgets. (any logged-in character can access gadgets but can't be accessed if logged-out)
11. Modify route GET /gadgets/:id that returns a single gadget. (any logged-in character can access gadgets but can't be accessed if logged-out)
12. Modify route GET /character-gadgets/:id that returns all gadgets by a specific character. In case of human character return error response that humans don't own gadgets. (any logged-in character can access gadgets but can't be accessed if logged-out)
13. Modify route PATCH /gadgets/:id to update a gadget (only robots

can update their gadgets)

14. Modify route DELETE /gadgets/:id to delete a gadget (only robots can update their gadgets)

15. Modify GET /gadgets?name=bamboo to filter gadgets by name using req.query. (any logged-in character can access gadgets but can't be accessed if logged-out)

16. Use appropriate response types with correct codes

Code	Type	Meaning	When to Use
`200`	✓ Success	OK – Request succeeded	GET, PUT, PATCH, DELETE operations when everything is fine
`201`	✓ Created	Resource created	POST request when a new user or gadget is successfully created
`204`	✓ No Content	Success, no data to return	DELETE request when item is successfully removed
`400`	✕ Client Error	Bad Request	Invalid input, missing required fields
`401`	🔐 Unauthorized	Authentication required	Used when user is not logged in
`403`	⛔ Forbidden	Access denied	e.g. normal user tries to access admin functionalities
`404`	? Not Found	Resource not found	e.g. user ID doesn't exist
`500`	✸ Server Error	Internal server error	Uncaught error, DB failure, server crash