German International University
Faculty of Informatics and Computer Science
Dr. Nada Sharaf
Eng. Donia Ali
AL. Amany Hussein
Eng. Hania Ashraf
Eng. Omar Ashraf

Software Project I, Winter 2024
Individual Evaluation solution V1

Duration: 1 hour

Name: **Tutorial:** **Appl. Number:**

Check last pages for authentication and authorization middleware and All needed database Schemas

Blogging platform system

In our system we have two types of users; Admin and user (Normal).
Each user/admin can create, update and delete a blog, and each user can create many blogs.
Any user/admin can delete or update his/her own blog.
The Admin only can delete a user

Backend

Exercise V1-1

You are required to implement the following endpoints:

assume express is imported and all needed imports like project
express= require('express');
app=express();
//mongoose connection

- Post a blog: This endpoint **creates** a blog by the user ,
  API endpoint: /blogs/
- Update a blog: This endpoint **updates** an existing blog for the current user,
  with body {title,content}
  API endpoint: /blogs/:blogId
- Delete a blog: This endpoint **Deletes** an existing blog for the current user,
  API endpoint: /blogs/:blogId
- Get all blogs: This endpoint **Gets** all the blogs,
  API endpoint: /blogs/
- Delete a user: This endpoint **Deletes** an existing user by the admin,
  API endpoint: /users/:userId
  Hint:findByIdAndDelete

App.js

```
const blogRouter = require("./Routes/blogs");
const userRouter = require("./Routes/users");
const authRouter = require("./Routes/auth");// login and register
const authenticationMiddleware = require("./Middleware/authenticationMiddleware");


app.use("/api/v1", authRouter);

_______________________________________// compelete with suitable code
app.use("/api/v1/blogs", blogRouter);
app.use("/api/v1/users", userRouter)
```

Controller

- **blogController:**
  const blogModel = require("../Models/blogModel");

```
const blogController = {
  getAllBlogs: async (req, res) => {
       try {

      const blogs = await blogModel.find();
      return res.status(200).json(blogs);


    } catch (e) {
      return res.status(500).json({ message: e.message });
    }
  },

  createBlog: async (req, res) => {
    try {

    const blog = new blogModel({
      title: req.body.title ,
      content: req.body.contend ,
      userId: req.user.id // req.user.userId ,
    });
    try {
      const newBlog = await blog.save();

      const user = await userModel.findById(req.user.id);

       const updateUser =userModel.findByIdAndUpdate(
        req.user.id ,
        blogPosts:user.blogPosts.push(newBlog._id) ,
        { new: true }
      );

      return res.status(201).json(newBlog);//'any msg'


    } catch (e) {
      return res.status(400).json({ message: e.message });
    }
  },

  updateBlog: async (req, res) => {
    try {

    const currentBlog=await blogModel.findById(req.params.blogId)

    if(!currentBlog.userId==req.user.id)
    {
    return res.status(404).send('enta meen')
    }

   const blog = await blogModel.findByIdAndUpdate(
        req.params.blogId,
        req.body,
        { new: true }
      );
      return res
        .status(200)
        .json({ blog, msg: "Blog updated successfully" });




    } catch (error) {
      return res.status(500).json({ message: error.message });
    }
  },

  deleteBlog: async (req, res) => {
    try {
      const currentBlog=await blogModel.findById(req.params.blogId)


        if(!currentBlog.userId==req.user.id)
        {
        return res.status(404).send('enta meen')
        }


        const blog = await blogModel.findByIdAndDelete(req.params.blogId);
        return res
        .status(200)
        .json({ blog, msg: "blog deleted successfully" });






    } catch (error) {
      return res.status(500).json({ message: error.message });
    }
  },
};
module.exports = blogController;
```

- **userController**
  const userModel = require(“../Models/userModel“);

```
const userController = {
   deleteUser: async (req, res) => {
    try {

     const user = await userModel.findByIdAndDelete(req.params.userId);
        return res
        .status(200)
        .json({ user, msg: "user deleted successfully" });




    } catch (error) {
      return res.status(500).json({ message: error.message });
    }
  }
 };
module.exports = userController;
```

Routes

```
//assume imports are done
const blogController = require("../controller/blogController");
const userController = require("../controller/userController");
const authorizationMiddleware = require("../Middleware/authorizationMiddleware")
// * Get all Blogs
router.get("/", authorizationMiddleware['admin','user']
,blogController.getAllBlogs );

// * Create a blog
router.post("/",authorizationMiddleware['admin','user']
,blogController.createBlog);

// * Update a blog
router.put("/:blogId",authorizationMiddleware['admin','user']
,blogController.updateBlog);

// * Delete a blog
router.delete("/:blogId",authorizationMiddleware['admin','user']
,blogController.deleteBlog);

// * Delete one user
router.delete("/:userId",authorizationMiddleware['admin']
,blogController.deleteUser);

module.exports=router
```

Frontend

Exercise V1-2

Given the following React code
Imagine you have a page contains all blogs where each blog is displayed as a card. So you have to complete the following code using concepts of state, props and connection between backend with frontend

BlogsPage

```
import BlogCard from "../components/blogCard";
import axios from "axios";
let backend_url = "http://localhost:3000/api/v1";

export default function BlogsPage() {
   const [blogs,setBlogs]=useState([])   // insert your code here
  const [cookies, removeCookies] = useCookies([]);
  useEffect(() => {
    async function fetchData() {
      try {
        if (!cookies.token) {
          navigate("/login");
        }

        const response = await axios.get(`${backend_url}/blogs/`
        ,withCredentials:true);// insert your code

        setBlogs(response.data) ; // insert your code
      } catch (error) {
        console.log(error);
      }
    }
    fetchData();
  }, [cookies]);

  return (
    <>
      <div >
          {blogs.map((blog) => (// insert your code
            <div style={{ margin: "20px" }}>
              <BlogCard blog={blog} />// insert your code
            </div>
          ))}

        </div>


    </>
  );
}
```

BlogCard

```
import Card from "react-bootstrap/Card";
export default function BlogCard({ blog }) {// insert your code
  return (                             //props
    <>
      <Card style={{ width: "18rem" }}>
        <Card.Img variant="top" />
        <Card.Body>      //props.blog.title
          <Card.Title>{blog.title}</Card.Title>// insert your code
          <Card.Text>
           {blog.content}// insert your code //props.blog.content
          </Card.Text>
        </Card.Body>
      </Card>
    </>
  );
}
```

Database Schemas

```
const userSchema = new mongoose.Schema({
  username: String,
  email: String,
  password: String,
  blogPosts: [{ type: mongoose.Schema.Types.ObjectId, ref: 'BlogPost' }],
});
module.exports=mongoose.model('userSchema',userSchema)


const blogPostSchema = new mongoose.Schema({
  title: String,
  content: String,
  createdAt: { type: Date, default: Date.now },
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'userSchema' },
});
module.exports=mongoose.model('blogPostSchema',blogPostSchema)
```

Authentication and Authorization MiddleWare

```
const jwt = require("jsonwebtoken");
const secretKey = "s1234rf,.lp";

module.exports = function authenticationMiddleware(req, res, next) {
  const cookie = req.cookies;
    if (!cookie) {
    return res.status(401).json({ message: "No Cookie provided" });
  }
  const token = cookie.token;
  if (!token) {
    return res.status(405).json({ message: "No token provided" });
  }

  jwt.verify(token, secretKey, (error, decoded) => {
    if (error) {
      return res.status(403).json({ message: "Invalid token" });
    }

    // Attach the decoded user ID to the request object for further use
    // console.log(decoded.user)
    req.user = decoded.user;
    next();
  });
};

module.exports= function authorizationMiddleware(roles) {
  return (req, res, next) => {
    console.log('req:',req)
    const userRole = req.user.role;
    if (!roles.includes(userRole))
      return res.status(403).json("unauthorized access");
    next();
  };
}
```

Scratch Paper

Scratch Paper