Лабораторна робота №4

API Authentication and Security

Завдання 1. При реєстрації та при оновленні користувача пароль потрібно зберігати в зашифрованому вигляді

```
const mongoose = require("mongoose");
const bcrypt = require("bcrypt");

const userSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true,
        trim: true
    },
    age: {
        type: Number,
        default: 0,
        validate(value) {
            if (value < 0) {
                throw new Error("Age must be a positive number");
            }
        }
    },
    email: {
        type: String,
        required: true,
        lowercase: true,
        unique: true,
        validate: {
            validator: function(value) {
                const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
                return emailRegex.test(value);
            },
            message: props => `${props.value} is not a valid email!`
        }
    },
    password: {
        type: String,
        required: true,
        minlength: 7,
        trim: true,
        validate: {
            validator: function(value) {
                if (value.toLowerCase().includes("password")) {
                    throw new Error("Password cannot contain the word 'pass-
word'");
                }
            },
            message: props => `Password validation failed`
        }
    }
});

userSchema.pre('save', async function(next) {
    const user = this;
    if (user.isModified('password')) {
        user.password = await bcrypt.hash(user.password, 8);
```

```
    }
    next();
});

const User = mongoose.model('User', userSchema);

module.exports = User;
```

Завдання 2. Створити запит /users/login на вхід.

- Здійснити пошук користувача по email та верифікацію паролю.
- Якщо успішна автентифікація, потрібно залогінитись: згенерувати token, зберегти його в БД, а також відправити в клієнт.
- Клієнт повинен зберегти token в змінній оточення authToken.

```
router.post("/login", async (req, res) => {
    try {
        const { email, password } = req.body;

        const user = await User.findOne({ email });
        if (!user) {
            return res.status(404).send("User not found");
        }

        const isPasswordMatch = await bcrypt.compare(password, user.password);
        if (!isPasswordMatch) {
            return res.status(401).send("Invalid email or password");
        }

        const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, { ex-
piresIn: "1h" });

        user.tokens.push({ token });
        await user.save();

        res.json({ authToken: token });
    } catch (error) {
        res.status(500).send(error.message);
    }
});
```

```
router.post('/logout', auth, async (req, res) => {
    try {
        req.user.tokens = req.user.tokens.filter(token => token.token !==
req.token);
        await req.user.save();
        res.send('Logged out successfully');
    } catch (error) {
        res.status(500).send(error.message);
    }
});
```

```
router.post('/logoutAll', auth, async (req, res) => {
    try {
        req.user.tokens = [];
        await req.user.save();
        res.send('Logged out from all devices successfully');
    } catch (error) {
        res.status(500).send(error.message);
    }
});
```

Завдання 3. Примінити авторизацію:

- В запиті відправити authToken в заголовку Authorization
- Сервер отримує токен, верифікує його і авторизує користувача (або ні в іншому випадку)

```
const jwt = require('jsonwebtoken');

function verifyToken(req, res, next) {
    const authToken = req.headers.authorization;
    if (!authToken) {
        return res.status(401).send('Unauthorized: No token provided');
    }

    const token = authToken.split(' ')[1];
    jwt.verify(token, process.env.JWT_SECRET, (err, decoded) => {
        if (err) {
            return res.status(401).send('Unauthorized: Invalid token');
        }
        req.userId = decoded.id;
        next();
    });
}

module.exports = verifyToken;
```

Завдання 4. Створити запити /users/logout, /users/logoutAll. Вимагає авторизації. Видаляє запис про токен із БД.

```
        router.post('/logout', auth, async (req, res) => {
    try {
        req.user.tokens = req.user.tokens.filter(token => token.token !==
req.token);
        await req.user.save();
        res.send('Logged out successfully');
    } catch (error) {
        res.status(500).send(error.message);
    }
});

router.post('/logoutAll', auth, async (req, res) => {
    try {
        req.user.tokens = [];
```

```
        await req.user.save();
        res.send('Logged out from all devices successfully');
    } catch (error) {
        res.status(500).send(error.message);
    }
});
```

Висновок: під час виконання даної лабораторної роботи, я вдосконалив свої знання з Node.

GitHub: https://github.com/AdamLahovskyi/NodeJS