Week 2: Implementing Security Measures – Report

Student Name: Iqra Azam
Internship: Developers Hub – Cybersecurity
Task: Week 2 – User Management System Security

1. Overview

In Week 2, we secured the **User Management System** by implementing **basic cybersecurity measures**.
Goals:

1. Validate user inputs (email, password)
2. Hash passwords for safe storage
3. Implement token-based authentication
4. Secure HTTP headers

Tools / Libraries Used:

- Node.js / Express
- Validator
- Bcrypt
- JSON Web Token (JWT)
- Helmet

2. Backend Setup

1. Created a `server.js` file inside the backend folder.
2. Installed required libraries:

```
npm install express body-parser validator bcrypt jsonwebtoken helmet
```

3. Started the server:

```
node server.js
```

- **Port:** 3000
- **Test URL:** `http://localhost:3000`

- **Confirmation:** "Backend running"

3. Security Measures Implemented

Step 1: Input Validation (Validator)

Purpose: Protect the system from invalid or malicious inputs

Implementation:

```
app.post('/signup', async (req, res) => {
    const { email, password } = req.body;
    if (!validator.isEmail(email)) return res.status(400).send('Invalid email');
    if (!validator.isStrongPassword(password, { minLength: 6 })) return res.status(400).send('Weak password');
});
```

Testing:

Test Case	Input	Output
Invalid email	notanemail	Invalid email
Weak password	123	Weak password
Strong password	StrongPass123!	User registered. Hashed password: <hashed>

Step 2: Password Hashing (Bcrypt)

Purpose: Ensure passwords are not stored in plain text

Implementation:

```
const hashedPassword = await bcrypt.hash(password, 10);
```

Result: Passwords are hashed before storage for security

Step 3: Token Authentication (JWT)

Purpose: Provide secure access after login

Implementation:

```
app.post('/login', (req, res) => {
    const userId = 123; // mock user
    const token = jwt.sign({ id: userId }, 'your-secret-key', { expiresIn:
'1h' });
    res.send({ token });
});
```

Result: After login, user receives a unique token

Step 4: Secure HTTP Headers (Helmet)

Purpose: Increase security between server and browser

Implementation:

```
app.use(helmet());
```

Result: Helmet automatically protects the server from common attacks like XSS and clickjacking

4. Testing Summary

Testing using Postman / API requests:

Route	Input	Expected Output	Result
`/signup`	Invalid email	`Invalid email`	Pass
`/signup`	Weak password	`Weak password`	Pass
`/signup`	Strong password	`User registered. Hashed password: ...`	Pass
`/login`	Correct email/password	JWT token	Pass

5. Observations

- Backend server runs successfully on localhost:3000
- Validator ensures only valid email and strong password are accepted
- Bcrypt hashes passwords for secure storage
- JWT provides token-based authentication
- Helmet adds basic HTTP security headers

6. GitHub Repository Used

Mock User Management System:
https://github.com/ashiktr/reactjs_user_management

- Used for testing input validation, password hashing, and authentication
- Backend folder created inside the repository for implementing security measures

7. Conclusion

In Week 2, we successfully:

1. Validated user inputs
2. Secured passwords using hashing
3. Added token-based authentication
4. Secured HTTP headers

All security measures were successfully implemented and tested.