# Task for Cybersecurity Interns: Strengthening Security Measures for a Web Application

**Overview**

You will analyze a simple User Management System for vulnerabilities and apply basic security measures to strengthen it. The goal is to familiarize yourself with cybersecurity basics, using tools and techniques to identify and fix common vulnerabilities.

## Week 1: Security Assessment

### 1. Understand the Application

Set up the application:
Take any mock web base application from github for cyber security
testing npm install  npm start

- Explore the app at `http://localhost:3000`. Check the signup, login, and profile pages.

### 2. Perform Basic Vulnerability Assessment

- **Use simple tools** to identify vulnerabilities:
    - 0 **OWASP ZAP**: Automated scanner for web app vulnerabilities.
    - ○ **Browser Developer Tools**: Inspect elements and simulate XSS attacks by entering `<script>alert('XSS');</script>` in text fields.
    - ○ **Basic SQL Injection Test**: Test login with `admin' OR '1'='1` as the username and password fields.

**Focus Areas**:

- Check for:
    - 0 Cross-Site Scripting (XSS).
    - ○ Weak password storage.
    - ○ Simple security misconfigurations.

### 3. Document Findings

- Create a simple document listing:

    - ○ Vulnerabilities found.
    - ○ Areas of improvement.

# Week 2: Implementing Security Measures

### 1. Fix Vulnerabilities

- **Sanitize and Validate Inputs**:

Use the `validator` library to validate user inputs:
npm install validator

Sanitize inputs in your route handlers: const validator = require('validator');

```
if (!validator.isEmail(email)) {
  return res.status(400).send('Invalid email');
}
```

**Password Hashing**: Use `bcrypt` to hash passwords: npm install bcrypt const bcrypt = require('bcrypt');

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

### 2. Enhance Authentication

Add basic token-based authentication using `jsonwebtoken`:
npm install jsonwebtoken const jwt = require('jsonwebtoken');

```
const token = jwt.sign({ id: user._id }, 'your-secret-key'); res.send({
token });
```

### 3. Secure Data Transmission

Use `Helmet.js` to secure HTTP headers:

```
 npm install helmet  const helmet
= require('helmet');
app.use(helmet());
```

# Week 3: Advanced Security and Final Reporting

### 1. Basic Penetration Testing

- Use tools like **Nmap** or browser-based testing to simulate common attacks.

### 2. Set Up Basic Logging

Use the `winston` library for logging:

```
 npm install winston  const
 winston = require('winston');
const logger = winston.createLogger({  transports:
 [
   new winston.transports.Console(),  new
   winston.transports.File({ filename: 'security.log' })
 ]  });
logger.info('Application started');
```

### 3. Create a Simple Checklist

- Include best practices:
    - Validate all inputs.
    - Use HTTPS for data transmission.
    - Hash and salt passwords.

### 4. Final Submission

- **Recorded Video**: Explain what you did (record your screen).
- **GitHub Repository**: Upload your code, configurations, and README.
- **Report**: Summarize the tasks, results, and fixes.

## Submission Details

**Deadline**: 14th -August-2025

(You will have to submit all of these 3 tasks on your consoles collectively before deadline)

**Submission Format**:

- Video explanation.
- GitHub repository link.
- Report document.