# SmartGlass Cybersecurity Internship Report (Week 1 to 3)

**Overview:-**

This report outlines the cybersecurity tasks completed from Week 1 to Week 3 for the **SmartGlass EdTech Project**, focused on backend security, secure coding, session/token protection, vulnerability identification, and best practices.

## Week 1 – Initial Security Risk Assessment & Setup

Risk assessments were performed on the backend codebase. Key vulnerabilities included unprotected routes, lack of input validation, and hardcoded secrets.  
Mitigations applied included JWT middleware enforcement and using environment variables for sensitive values.

// Before - Hardcoded JWT secret  
const JWT\_SECRET = '123456';

// After - Secured using .env  
const JWT\_SECRET = process.env.JWT\_SECRET;

// Before - No email validation  
const { email } = req.body;

// After - Email validation  
if (!email || !email.includes('@')) return res.status(400).json({ error: 'Invalid email' });

1. **Risk Assessment Performed**

| **Component** | **Risk Identified** |  |  |  |  |  |  |  | **Mitigation** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User Inputs | No validation on input fields |  |  |  |  |  |  |  | Implemented input validation logic |
| JWT Implementation | Secret key was exposed |  |  |  |  |  |  |  | Moved JWT\_SECRET to .env file |
| Public APIs | APIs not protected by authentication |  |  |  |  |  |  |  | Applied jwtAuth middleware |

**2. Secure Coding Best Practices Applied**

* Input validation (for email, role, file upload)
* try-catch blocks added for API stability
* Avoided exposing stack traces in responses

js

CopyEdit

// Example for email validation

if (!email || !email.includes("@")) {

return res.status(400).json({ error: "Invalid email format" });

}

**3. Files Worked On:**

* authController.js
* jwtAuth.js
* .env secured
* .gitignore updated

## Week 2 – Code Review & Secure Configuration

Audits were performed across controllers, models, and middleware. Sensitive keys were removed, token expiry was added, and proper error messages were implemented.

// Before - Unprotected upload route  
router.post('/upload', uploadFile);

// After - Protected with jwtAuth  
router.post('/upload', jwtAuth, uploadFile);

// JWT with expiry  
const token = jwt.sign({ id: user.\_id }, JWT\_SECRET, { expiresIn: '15m' });

**1. Code Audit Conducted**

| **Area Reviewed** | **Observation** | **Fix** |
| --- | --- | --- |
| Controllers | Unhandled promise rejections | Added try/catch |
| Sensitive Keys | Hardcoded in file | Moved to .env |
| Access Restrictions | Missing in routes | jwtAuth middleware enforced |

**2. Middleware Secured**

js

CopyEdit

// jwtAuth.js - Validates token before route access

const token = req.header("Authorization")?.replace("Bearer ", "");

if (!token) return res.status(401).json({ message: "Missing token" });

**✅3. MongoDB Protection**

* .env contains MONGO\_URI
* Used Mongoose schemas with validation rules

js

CopyEdit

email: {

type: String,

required: true,

unique: true

}

## Week 3 – Security Testing & Final Improvements

Final week focused on thorough API testing, vulnerability scanning, and documentation. Express-validator was added for input checking.  
File upload routes were restricted to allow only PDF and DOCX files.

// Input validation using express-validator  
body('email').isEmail().withMessage('Invalid email format');

// File upload validation  
if (!file.mimetype.includes('pdf') && !file.mimetype.includes('docx')) {  
 return res.status(400).json({ error: 'Only PDF/DOCX allowed' });  
}

**1. Security Testing Done**

* Tested /api/auth with Firebase token
* Verified JWT-based access
* Tested file upload with and without token (used Postman)

**2. Routes Audited for Missing jwtAuth**

| **Route** | **Status** | **Fix** |
| --- | --- | --- |
| /api/session | ✅ Protected | N/A |
| /api/upload | ❌ Missing jwtAuth | ✅ Added |
| /api/chat | ❌ Missing jwtAuth | ✅ Added |

**3. Postman Tests Conducted**

| **API** | **Method** | **Secure?** |
| --- | --- | --- |
| /api/auth | POST | ✅ |
| /api/upload | POST | ✅ |
| /api/session | POST | ✅ |
| /api/chat | POST | ✅ |

**4. Security Fixes Implemented**

* ✅ Added express-validator logic
* ✅ Added error responses
* ✅ Improved token expiry handling

**Secure Coding Checklist Applied**

* Input Validation ✅
* Token Verification ✅
* Session Control ✅
* Authentication Required on Protected Routes ✅
* Error Handling with Messages ✅
* .env configured for Firebase/Mongo ✅
* .gitignore excludes secrets ✅

**📁 Key Files Modified**

| **File** | **Purpose** |
| --- | --- |
| authController.js | Firebase token login |
| jwtAuth.js | JWT middleware |
| User.js, Session.js, Chat.js | Mongoose schema validation |
| server.js | Applied route protections |
| .env, .gitignore | Secured credentials |