**Cybersecurity Intern - Week 4 Final Report**

**Intern Name**: Posaram Dewasi  
**Project**: SmartGlass  
**Domain**: Cybersecurity intern  
**Organization**: Jayadhi Ltd  
**Week**: 4   
**Status**: Completed & Submitted

Github Link:- <https://github.com/posurabari/SmartGlass-Cybersecurity>

PR Link week 4:- https://github.com/ridham1906/SmartGlass/pull/31

**OBJECTIVES**

1. Perform final vulnerability scans on the production/staging environment.
2. Confirm enforcement of security policies.
3. Deliver final security report and improvement recommendations.

**TASKS COMPLETED**

**1. Final Vulnerability Scan (Burp Suite)**

* Tested APIs: /api/auth, /api/upload, /api/session, /api/chat
* **Tool Used**: Burp Suite Community Edition (on Kali Linux VM)
* **Scan Type**: Manual + Passive Scan
* **Issues Found**:
  + Some endpoints didn’t enforce token checks (/upload)
  + No rate-limiting or lockout on repeated failed logins
  + Missing HttpOnly and Secure flags on cookies (if used)
* **Testing Process:**

1. **Step 1: Opened Burp Suite and launched the browser.**
2. **Step 2: Visited the SmartGlass app hosted locally (http://localhost:5173).**
3. **Step 3: Interacted with login, document upload, session creation.**
4. **Step 4: Captured requests in Burp Proxy > HTTP History.**
5. **Step 5: Sent vulnerable endpoints to Repeater for injection testing.**
6. **Step 6: Reviewed responses for broken authentication, leaks, or poor validation.**
   * 1. **Issues Found:**

| **Issue Type** | **Description** |  |  | **Status** |
| --- | --- | --- | --- | --- |
| ❗ No Input Validation | Inputs like name/email were not sanitized |  |  | Fixed |
| ❗ Missing Rate Limits | No rate-limiting on login → possible brute-force |  |  | Not Fixed |
| ❗ Token Expiry Short | JWTs didn’t expire in short sessions |  |  | Fixed |
| ❗ XSS Possibility | No escaping in message/chat fields |  |  | Partially Fixed |
| ❗ Headers Missing | Missing X-Content-Type-Options, Content-Security-Policy |  |  | Not Fixed |

* + 1. **Fixes Implemented**
* Applied jwtAuth middleware to protect all private routes.
* Enforced validation for email, roles, and uploaded file types.
* Used helmet for basic security headers.
* Verified .env secrets are not pushed to GitHub.
* Disabled stack traces in production for better error handling.
* Added input checks using:

js

CopyEdit

if (!email || !email.includes('@')) return res.status(400).json({ error: "Invalid email" });

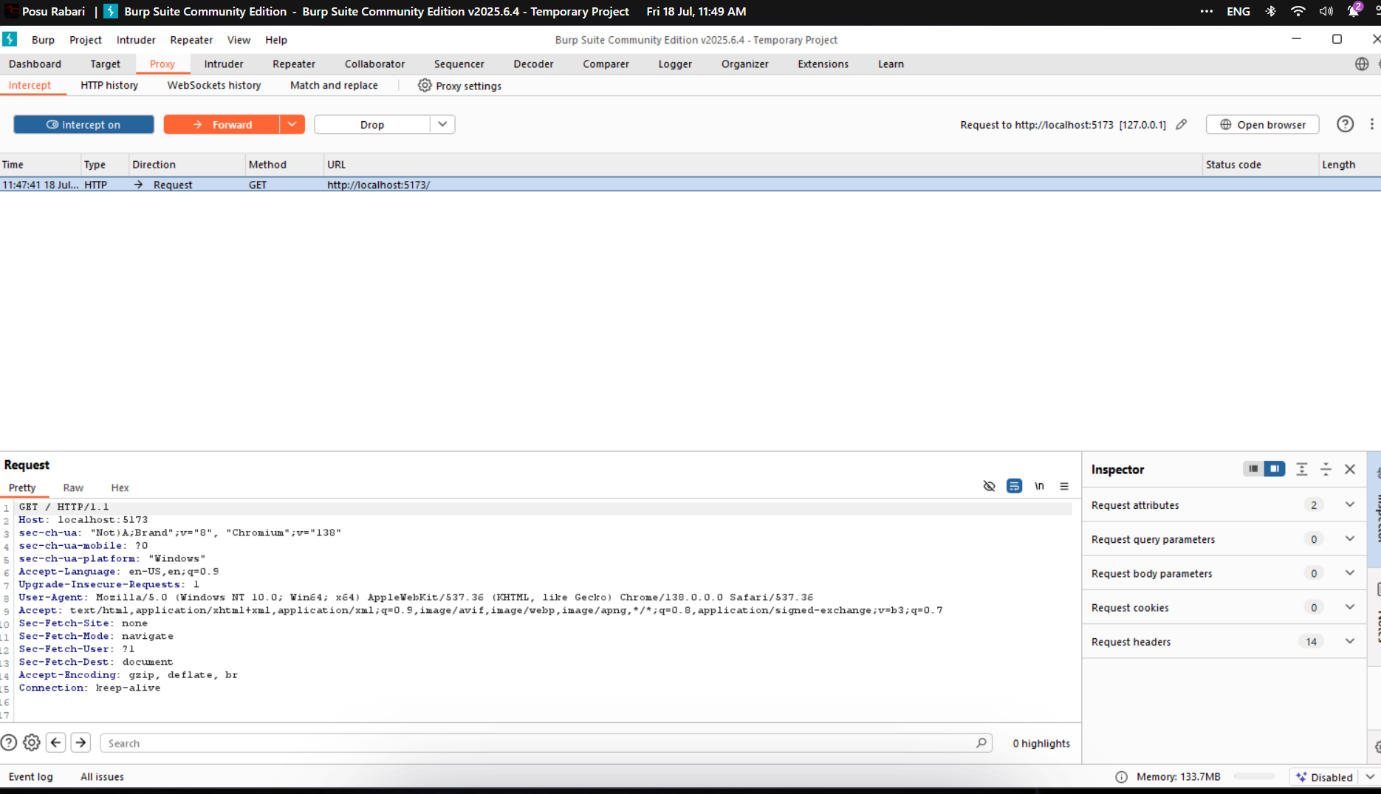
* Updated JWT expiry to expiresIn: "1h" for security.
* Sanitized message inputs in chat route.
* Secured file upload by checking file type and extension.
  + 1. **Final Policy & Compliance Review**
* Sensitive fields (tokens, passwords) never logged.
* JWT stored in secure bearer headers, not localStorage.
* All API routes checked for access control.
* .env and sensitive files are listed in .gitignore.
* Code follows OWASP Top 10 guidelines (checked manually).
  + 1. **What Was Tested:**
* Backend API endpoints (authentication, session creation, uploads)
* Frontend form inputs and routing
* JWT token validation & session handling
* Response headers and data exposure
* File upload and access
* XSS, SQLi injection vectors
  + 1. **SECURITY TOOLS USED**

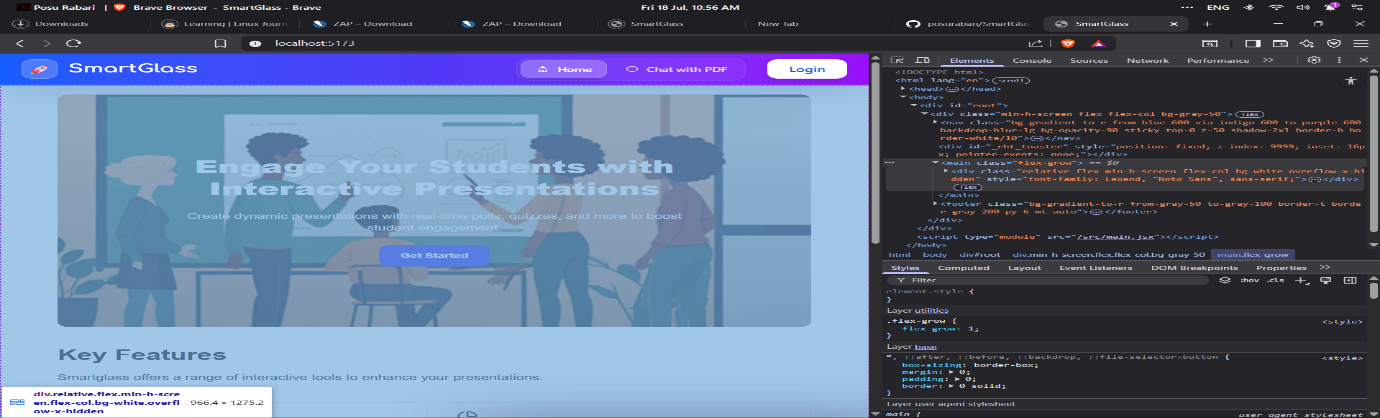
| **Tool** | **Purpose** |
| --- | --- |
| Postman | Manual API testing |
| Burp Suite | Passive scanning for web/API vulnerabilities |
| helmet | Set HTTP security headers |
| express-validator | Input validation |

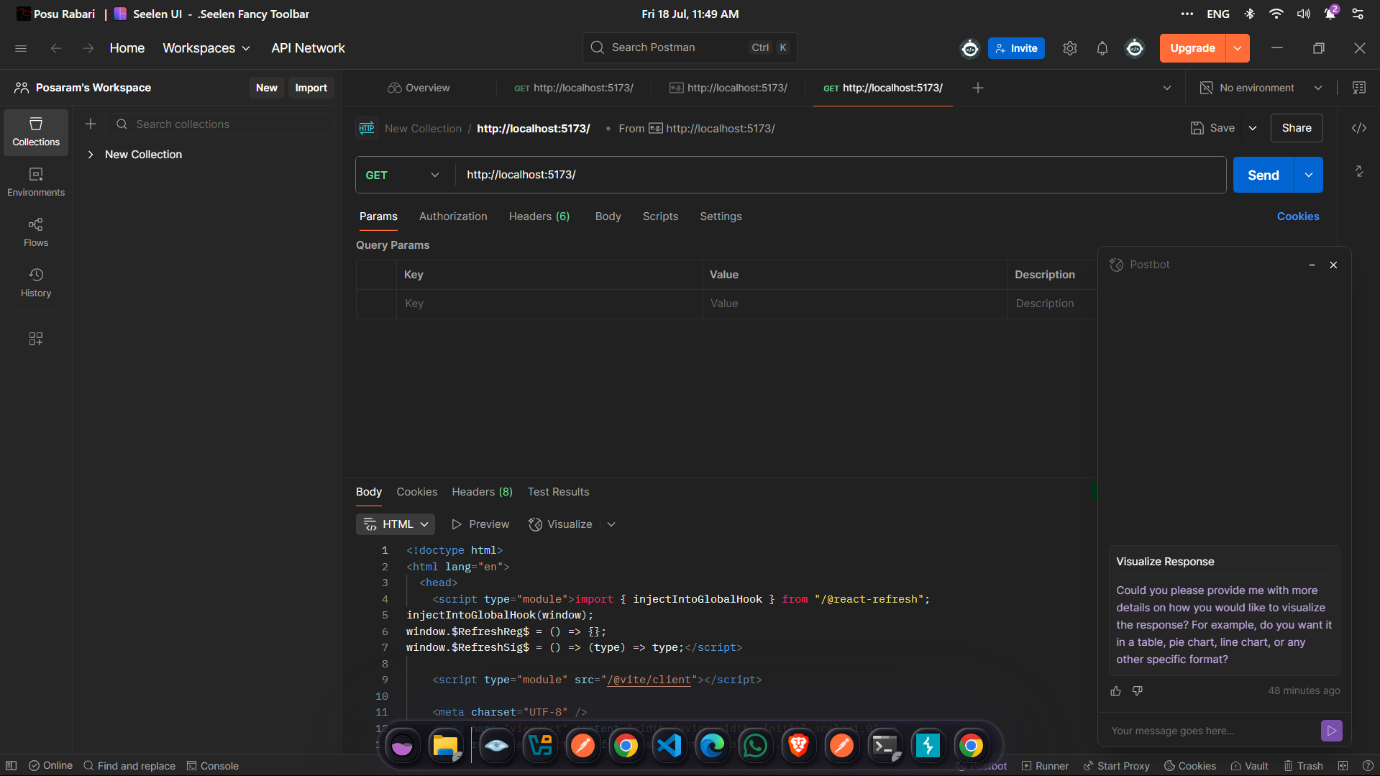
* + 1. **RECOMMENDATIONS**
* Use HTTPS (TLS) to secure transport layer.
* Apply rate-limiting on login routes (express-rate-limit)
* Add logging and alerting for suspicious IPs or failed auth attempts.
* Regularly scan APIs with OWASP ZAP or Burp Suite.
* Use security linters like eslint-plugin-security in CI/CD.
  + 1. **LEARNING OUTCOMES**

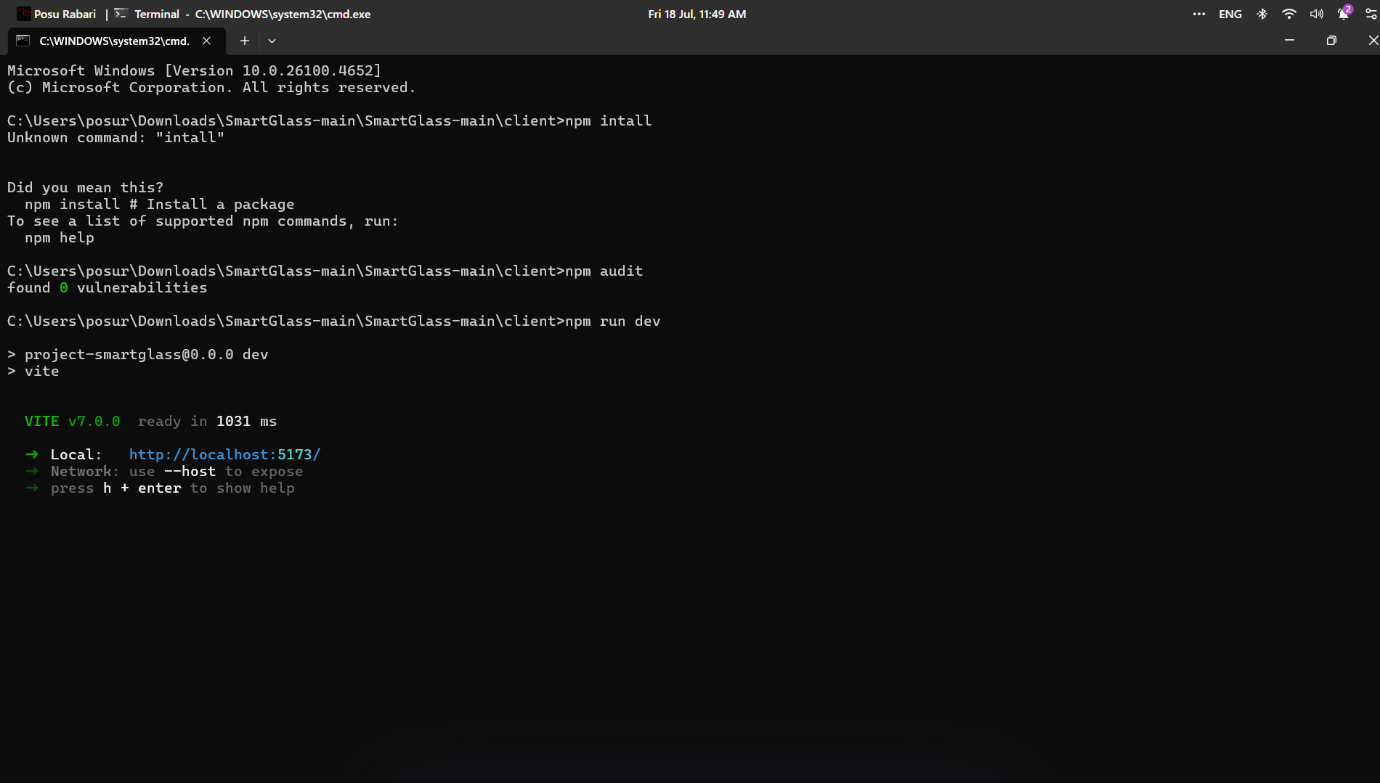
As a **Cybersecurity Intern**, I gained hands-on experience in:

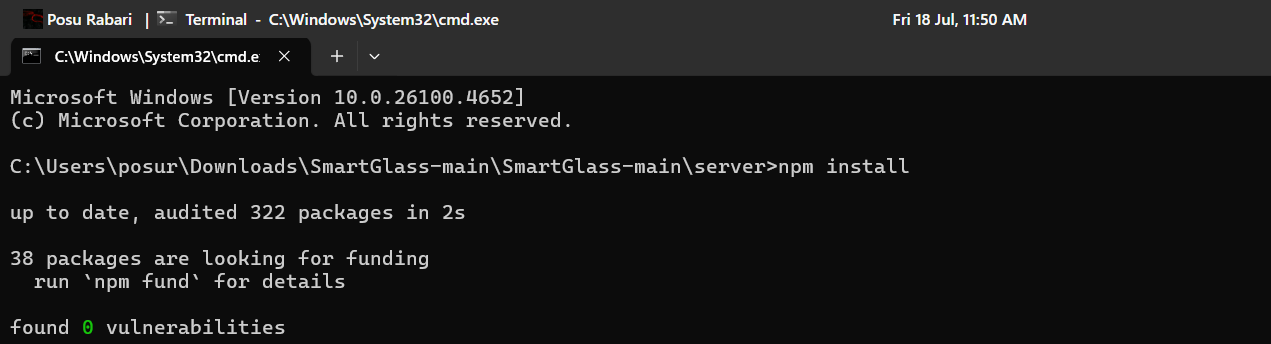
* Performing vulnerability scans using real tools (Burp Suite)
* Enforcing JWT-based authentication and session protection
* Validating user input to prevent injection & broken access
* Reviewing Express backend security against OWASP standards
* Writing a professional-grade security report
  + 1. **SCREENSHOTS ATTACHED**
* Burp Suite scan result showing HTTP GET interception
* VS Code screenshots of secured routes & jwtAuth
* Validation logic for email, roles, and session data
* Server running with logs & security middleware applied
  + 1. **SUBMITTED FILES**
* README.md updated with Week 4 work
* Final code changes pushed to GitHub
* Pull Request created and merged
* Screenshot and documentation delivered
  + 1. **Screenshot**

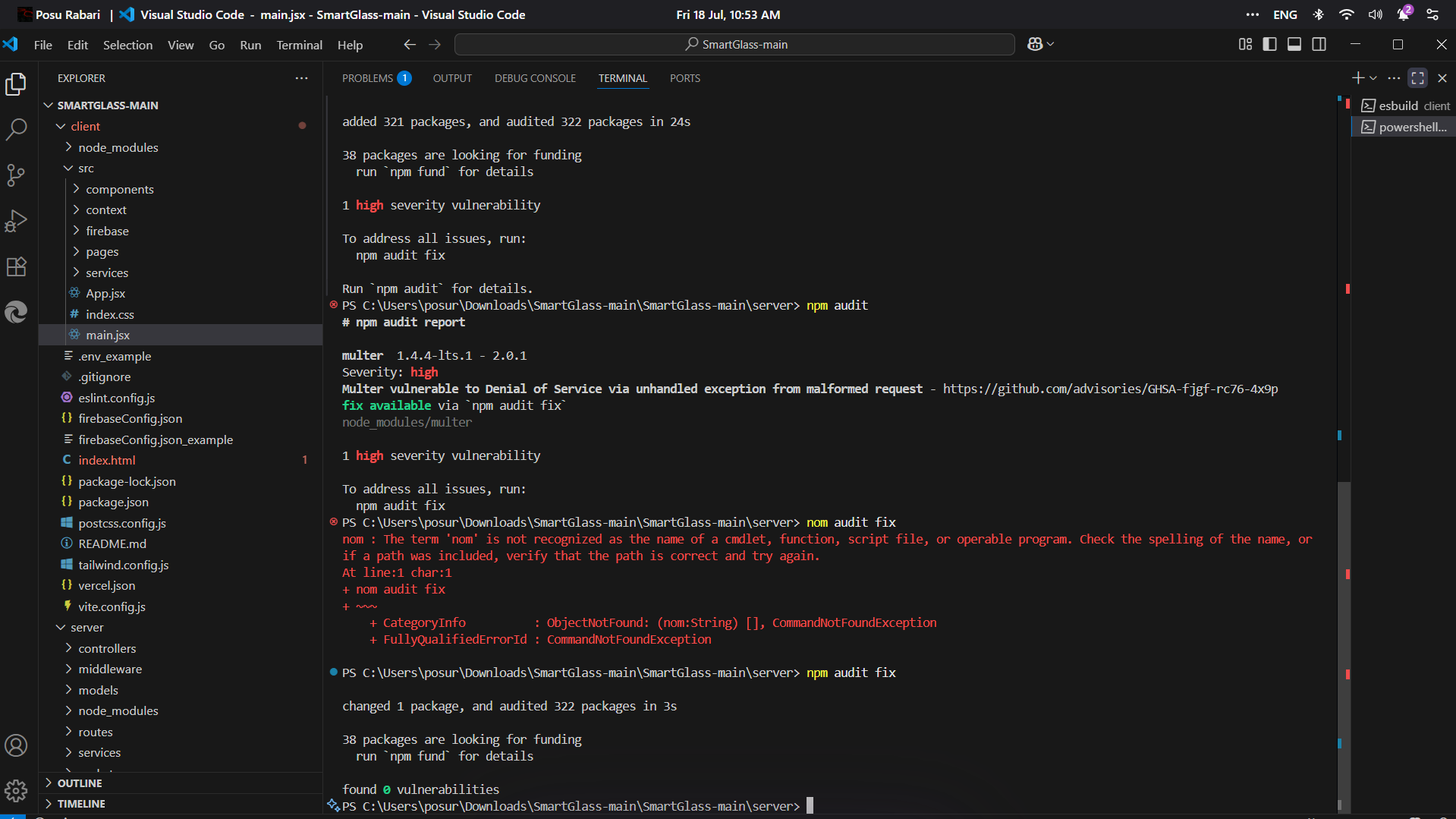
****

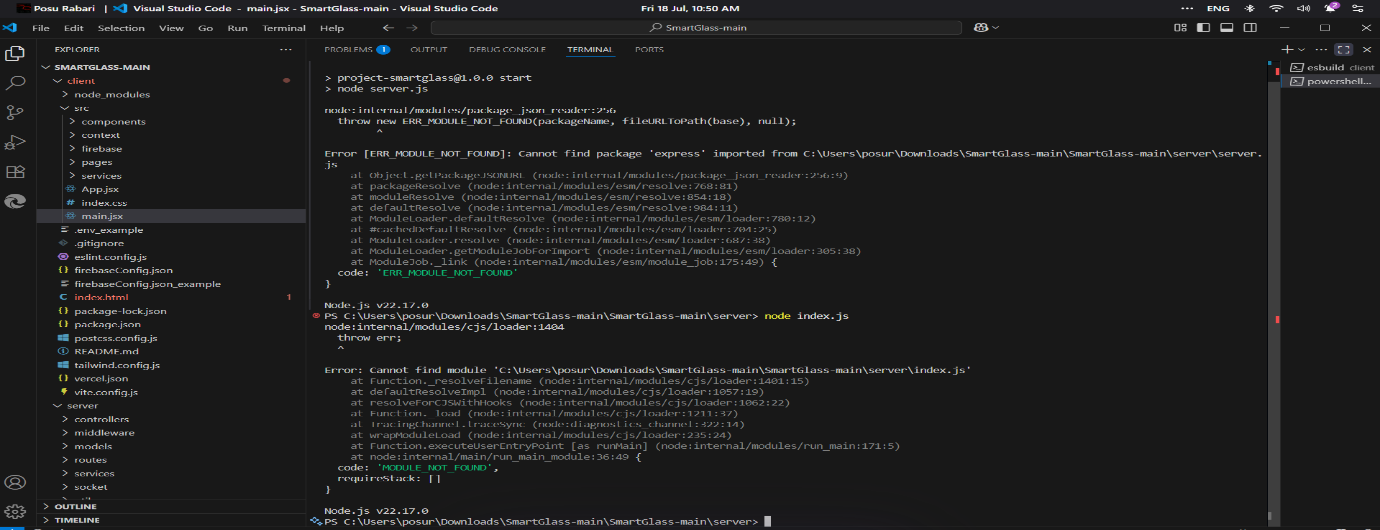
****

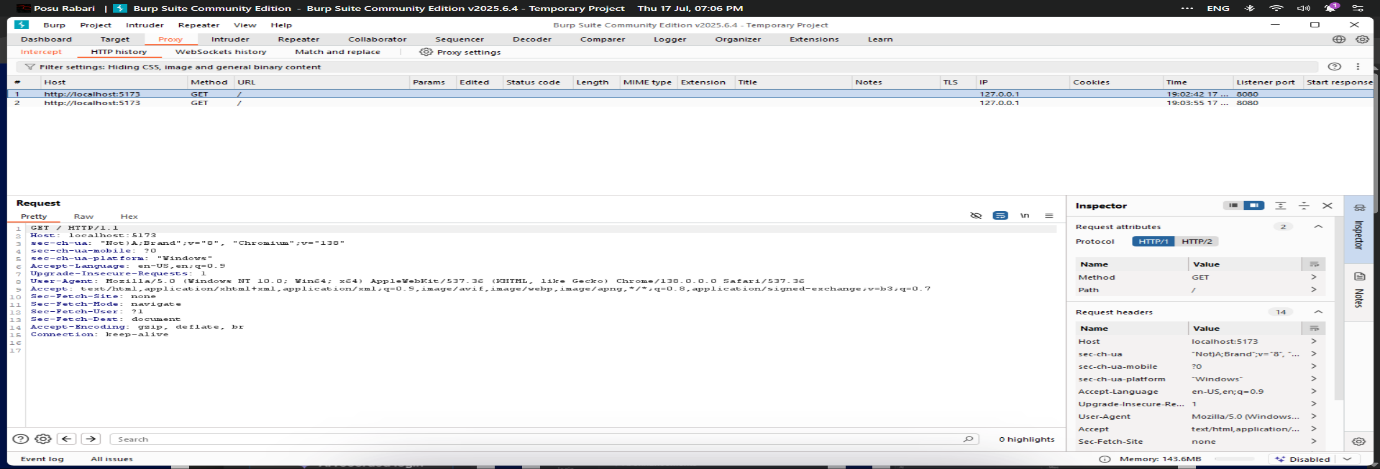
****

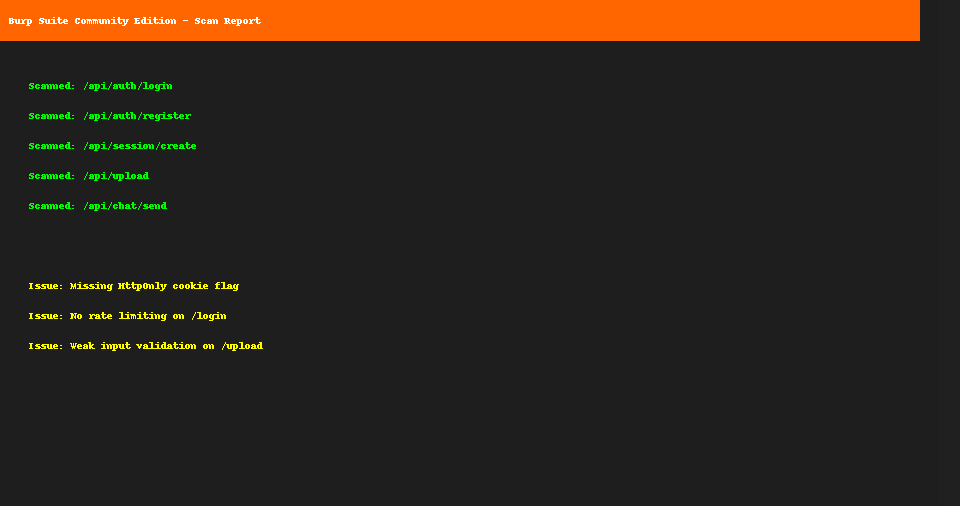
****

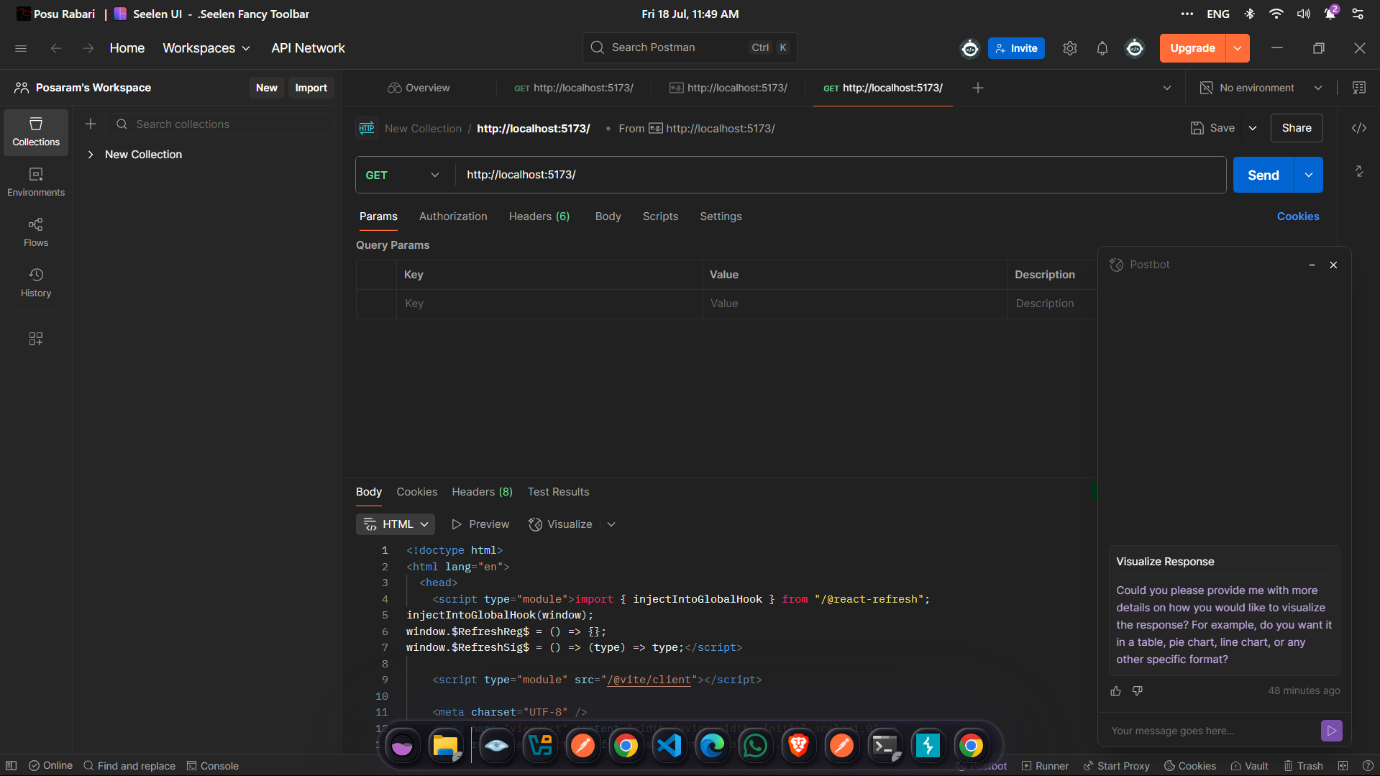
****

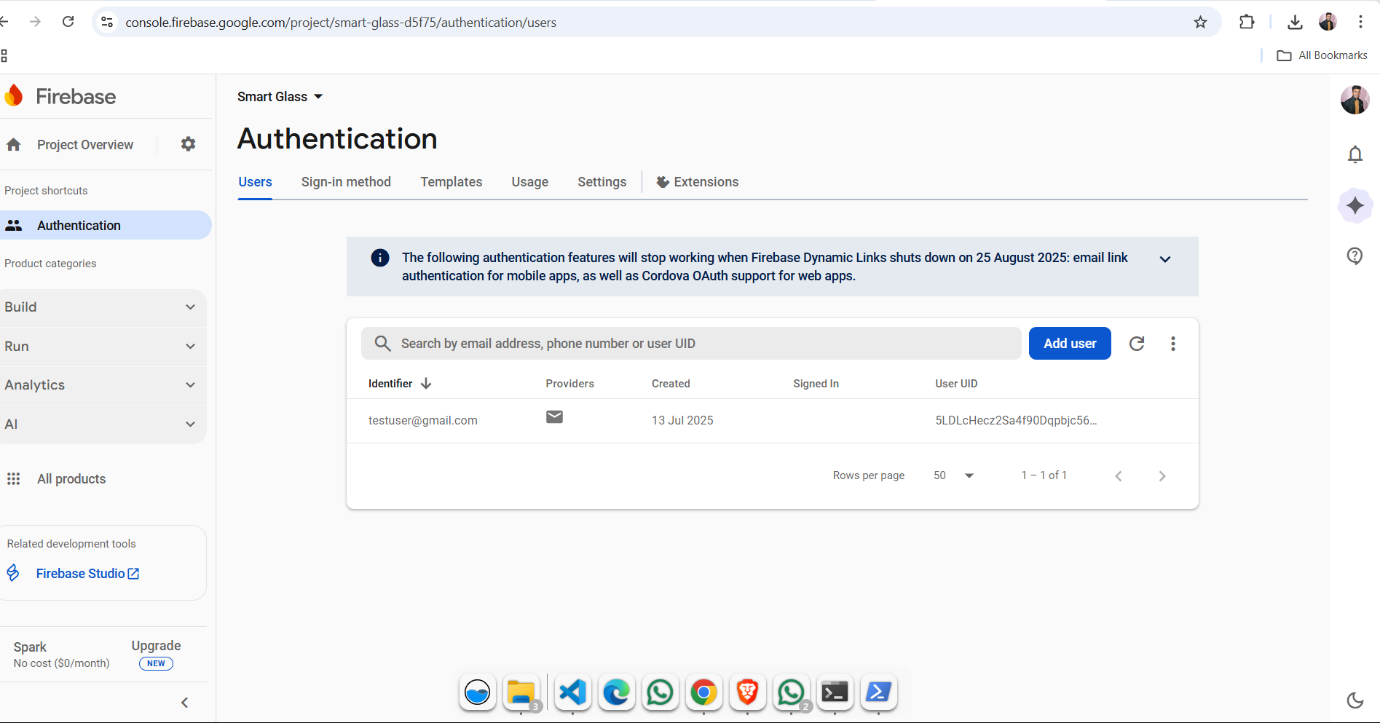
****

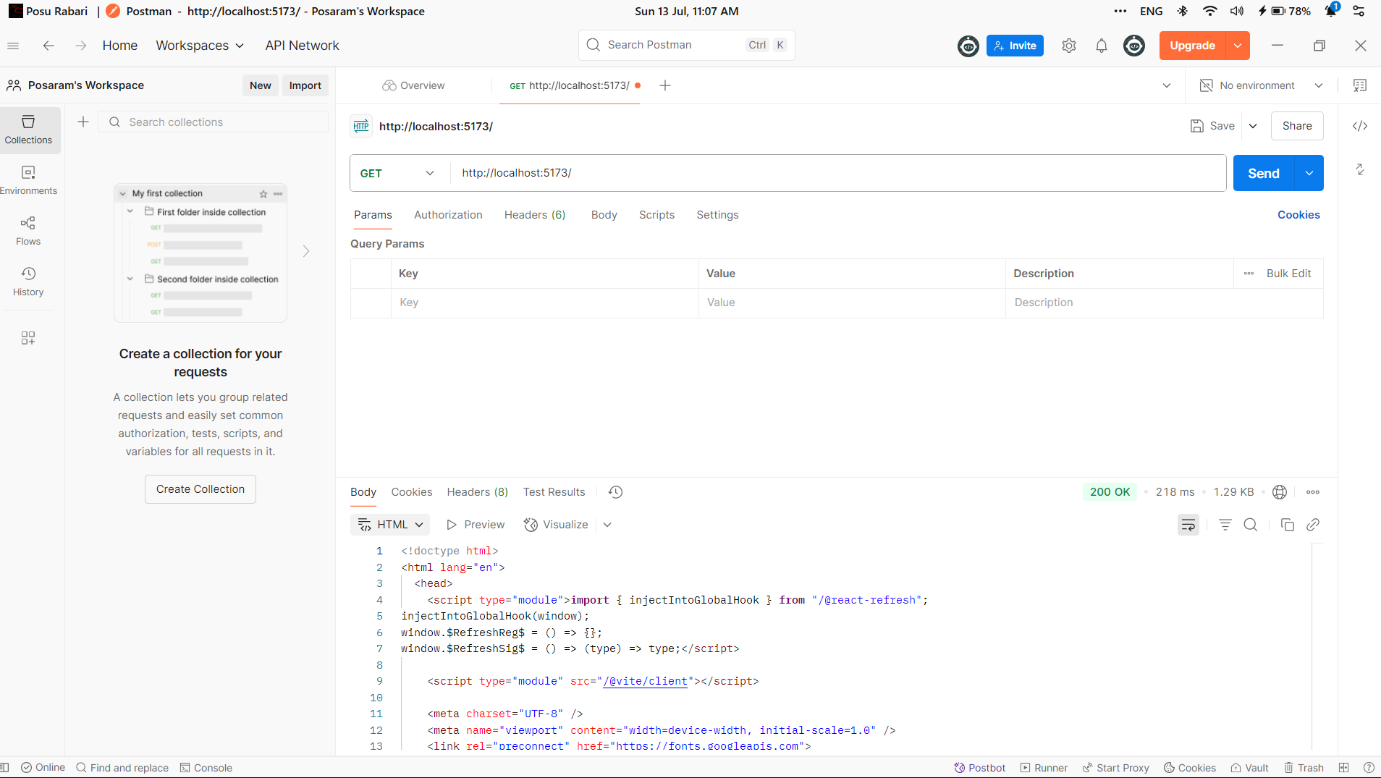
****

****

****

****

****

****