## 1. Declaration

I, [yuanjili, declare that this assignment, titled Basic Web App Debug and Extend (Version 2) is my own original work and has not been copied from any other source except where explicitly acknowledged. I have not engaged in plagiarism, collusion, or any other form of academic misconduct in the preparation and submission of this assignment. All sources of information and data used in this assignment have been properly cited and referenced in accordance with the prescribed guidelines. I have not used unauthorized assistance in the preparation of this assignment and have not allowed any other student to copy my work. I am aware that any breach of academic integrity may result in disciplinary action as per the [policies of Monash University](https://www.monash.edu/learning-teaching/priorities-and-programs/assessment-and-academic-integrity/assessments-and-integrity-policy-and-procedure), which may include failing this assignment or the course, and further academic penalties.

Signature: \_\_\_\_\_Yuanji Li\_\_\_\_\_\_ Date: \_\_\_\_\_\_12/09/2025\_\_\_\_\_\_\_\_\_\_\_\_\_

## 2. Github Check

Enter your Github details here.

|  |  |
| --- | --- |
| Github Username  *Enter your username here* | apr1ee |
| **Repository Shared?**  *Have you started and shared your assignment repository with your tutor yet?* | https://github.com/apr1ee/FIT5032\_Assignment |

## 3. Self-Evaluation

Rate your performance for each criteria. Put a ✅(tick) in the box where you think your work belongs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Exceeds Expectations** | **Meets Expectations** | **Needs Improvement** | **Fail to meet expectations** |
| BR (C.1): Authentication | ✅ |  |  |  |
| BR (C.2): Role-based authentication | ✅ |  |  |  |
| BR (C.3): Rating | ✅ |  |  |  |
| BR (C.4): Security | ✅ |  |  |  |

## 4. Screen Recording of BRs

Create a 3 minute video showing your basic web application in action! Upload this video to your Google Drive and put the link here (ensuring that you have updated the access list so its not private).

|  |
| --- |
| https://drive.google.com/file/d/13RKgqcfrNwuiAPQP7rec6slYxUV42DtO/view?usp=drive\_link |

## 5. Reflections: Implementation of C.4 Security

If you have implemented BR C.4, in less than 200 words describe the approach that you have taken to implementing Security in your application. What security flaws were you trying to prevent and what security measures have you implemented to fix those flaws? How do you know that these measures will help prevent those issues from happening? Optionally you can cite external sources to provide evidence for your claim.

|  |
| --- |
| I treated C.4 Security as defence-in-depth for a client-only prototype. The main threats were XSS (malicious HTML/JS injected via journal text or tags), unsafe inputs, tab-nabbing, and click-jacking. Mitigations: (1) **Client-side validation** on all forms—required fields, email format, password length, journal text length (5–1000), mood range (1–5), tag whitelist (reject < >). (2) **Output encoding** before storage/rendering: user text is passed through an escapeHTML() utility and rendered only with Vue interpolation ({{ }}), never v-html. (3) **Content Security Policy** in index.html (default-src 'self', restricted script-src, frame-ancestors 'self') to block inline/third-party script execution and framing. (4) **Safe links** use rel="noopener noreferrer" to prevent tab-nabbing. (5) **Credential handling (demo)**: passwords are stored as SHA-256 hashes via Web Crypto (with legacy plaintext auto-migrated); real systems would add salted hashes server-side and HTTPS. |

## 6. Reflections: Challenges

What has been the most challenging part of this assignment for you? How has this stretched you as a programmer?

|  |
| --- |
| The most challenging part of this assignment was implementing authentication and role-based access. Since the project is front-end only, I had to simulate account creation, login, and session management securely in the browser using LocalStorage. Designing this in a way that feels realistic while still meeting security requirements (hashing passwords, preventing XSS, adding CSP) pushed me to think more carefully about real-world web security. It stretched me as a programmer because I had to combine Vue reactive programming with practical concerns like validation, routing guards, and safe storage. I learned to look beyond “making it work” and to consider how users and attackers might interact with the system. This has deepened my appreciation for defence-in-depth and secure coding practices. |