RICHARD JOHNSON
ANGLICAN COLLEGE
93 Hyatts Road Oakhurst NSW 2761 | 02 9677 2455

Assessment Notification

Year: 12 Subject: Software Engineering Task 2

Task title	**Secure Software Architecture**		
Date issued	Friday, 28 February 2025	Weighting	20%
Due date	Friday, 21 March 2025	Total marks	50

Outcomes:

- **SE-12-01** justifies methods used to plan, develop and engineer software solutions
- **SE-12-02** applies structural elements to develop programming code
- **SE-12-06** justifies the selection and use of tools and resources to design, develop, manage and evaluate software
- **SE-12-09** applies methods to manage and document the development of a software project

Context:

Securing a Progressive Web App

Your client, "The Unsecure PWA Company," has engaged you as a software engineering security specialist to provide expert advice on the security and privacy of their Progressive Web App (PWA). The application is currently in the testing and debugging phase of the software development lifecycle, with intentional vulnerabilities introduced for assessment and remediation.

Your task is to identify these security issues, implement fixes, document your process, and present your solutions to the client. This assessment is divided into four distinct components, each with specific requirements and deliverables: a Security Report, Secure Code Submission, a Client Presentation, and an Online Quiz.

The repository that you will need to fork are at:
https://github.com/Mr-Zamora/12SE_Task2

Description of the task:

Component A: Security Report (25 marks)

Purpose: Document the vulnerabilities you identify in the PWA, explain your testing methods, detail the fixes you applied, and provide recommendations for future security.

Requirements:

A1. Introduction (5 marks):

- Describe the PWA and the scope of your security assessment.
- Outline the testing methods used (e.g., blackbox, whitebox, greybox testing).

A2. Findings (10 marks):

- Identify and list the vulnerabilities you discovered (e.g., SQL Injection, XSS).
- Describe how each could be exploited and its potential impact on the PWA.
- Include evidence such as screenshots from testing tools (e.g., Google Lighthouse, Postman).

A3. Fixes (10 marks):

- Refer to your submission for **Component B** and explain how you resolved each vulnerability.
- Provide **evidence of fixes** (e.g., before-and-after screenshots, code snippets showing input validation or password hashing).

Deliverable:

- Submit a professionally formatted document (PDF or Word) uploaded to Canvas.
  Table of Contents, Headings, Sub-headings and Bibliography. Use Tables for clarity.

Component B: Secure Code Submission (15 marks)

Purpose: Demonstrate your ability to apply secure coding practices by submitting the updated PWA code with vulnerabilities fixed.

B1: Code Fixes for Identified Vulnerabilities (5 marks):

1. **Correcting the Vulnerabilities**
   - Locate each vulnerability (no matter how many you found) and fix it in the code.
   - Examples of fixes could involve parameterised queries for potential injections, stricter validation, etc.
   - Ensure no remaining instances of the same vulnerability persist.
2. **In-Line Comments or Docstrings**
   - Clearly comment (in code) how you resolved each issue.
   - Provide just enough detail that a reviewer understands what changed and why.

B2: Implement Secure Coding Practices (5 marks):

1. **Security Measures**

Depending on the vulnerabilities discovered, strengthen the PWA by applying proper security measures in the code to resolve each vulnerability you've discovered.

B3: Test and Verify Your Fixes (5 marks)

Re-Test Your Previously Discovered Vulnerabilities

- Run the same tests that exploited the PWA before (e.g., same payloads or steps). Confirm that they now fail to produce the old exploit results.

Show Evidence

- Screenshots **(Component A, A3)** and short descriptions in code comments **(Component B, B1)** that indicate the exploit is no longer successful.

Check for New Issues

- Make a quick pass to confirm you didn't introduce a new vulnerability while fixing the old ones. If you spot anything suspicious, address it before submitting.

Deliverables:

- **Zip File** and **Private GitHub Repository**:
  - All your fixes are to be documented in (**Component A, A3)**.
  - Upload a zip file to Canvas containing:
    - The updated PWA codebase.
    - Screenshots or logs showing the fixes in action (e.g., a failed SQL Injection attempt after your fix).
  - Share a link to a private GitHub repository with your teacher, including:
    - The complete updated codebase.
    - A README file outlining the fixes and instructions to run the PWA.

Component C: Online Quiz (10 marks)

Purpose: Assess your understanding of the security concepts and practices applied in this assessment.

Requirements:

- Complete an online quiz in class on the submission due date.
- The quiz will include questions on:
  - Identifying vulnerabilities and their impacts from https://www.hacksplaining.com/lessons
  - Secure coding techniques (e.g., input validation, password hashing).

Deliverable:

- No separate submission is required; the online quiz will be completed in, Friday, 21 March, Period 1 (30mins)

Submission Checklist

To ensure you meet all requirements, use this checklist before the due date:

- **Component A**: Security Report (Word) uploaded to Canvas.
- **Component B**:
    - Zip file with updated code and evidence uploaded to Canvas.
    - Private GitHub repository link shared with your teacher.
      Include this on your first page on Component A.
- **Component C**: Online quiz completed in class. Friday, 21 March, Period 1 (30mins)

Marking criteria: Refer to the marking scaffold below. All written work must be the student's original creation. Failure to cite sources, including AI-generated content, will be considered academic malpractice and may result in penalties such as reduced marks, a zero grade, or further disciplinary action per the school's academic integrity policy.

Feedback provided: Throughout the unit, you will receive both written and verbal feedback. During discussions, the teacher will highlight your strengths and areas that require further development.

Submission Guidelines:

1. Submit a **single Word document** containing all required written tasks, a **zip file** containing all relevant Python code and project files. **Include a link your GitHub private repo in your Word document on the first page.**
2. Submit all folders, files and resources in the Canvas submission
3. Submit all code in your GitHub private repo.
4. **Any use of AI must be clearly and explicitly acknowledged and cited in the Bibliography**.

Faculty Leader approval:

Students are reminded of the rules and requirements relating to completion, submission and absences for assessment tasks.	_____ **Faculty Leader approval**

Year 12 Software Engineering: Secure Software Architecture – Task 2 Marking Scaffold

CRITERIA	WORK ASPECT	RATINGS					
SE-12-09 – Applies methods to manage and document the development of a software project **SE-12-01** justifies methods used to plan, develop and engineer software solutions **SE-12-02** applies structural elements to develop programming code **SE-12-06** justifies the selection and use of tools and resources to design, develop,	**COMPONENT A: Security Report (25 marks)**						
		EXTENSIVE	**THOROUGH**	**SOUND**	**BASIC**	**ELEMENTARY**	**NON-SUBMSSION**
	A1. Introduction (5 marks)	**5**	**4**	**3**	**2**	**1**	**0**
		Provides a comprehensive overview of the PWA's purpose and scope, with well-reasoned justification of the chosen testing methods (e.g., blackbox, whitebox, greybox). Clearly demonstrates how planning and documentation methods underpin the security approach.	Gives a clear introduction with mostly strong justification of testing methods. Some minor detail or rationale for the approach could be expanded.	Offers a moderate explanation of the PWA's scope and testing plan. May not fully connect the methods used to broader project documentation.	Provides a brief or patchy introduction with limited justification for testing methods. Documentation or planning references are weak.	Introduction is unclear or missing. Minimal or no mention of how methods or documentation processes were justified.	Not attempted
	A2. Findings (10 marks)	**10 – 9**	**8 – 7**	**6 – 5**	**4 – 3**	**2 – 1**	**0**
		Discovers a broad range of critical vulnerabilities (e.g., SQL Injection, XSS) and articulates testing methods clearly. Supports findings with strong evidence (screenshots/logs). Thoroughly explains potential impact on the PWA, referencing selected tools and how they assisted in the process.	Identifies most relevant vulnerabilities with good discussion of their impact. Provides solid evidence from tools such as Postman or Lighthouse. Minor gaps in explaining how the chosen tools and methods aided discovery.	Notes some vulnerabilities but discusses them in limited detail. Evidence may be present but partially explained. Rationale for tool selection or documentation is average.	Lists few vulnerabilities or provides minimal evidence. Discussion of impacts or tool usage is superficial. Documentation is inconsistent or unclear.	ails to identify vulnerabilities or present meaningful evidence. Limited to no justification of tool choice or testing approach. Documentation is absent or highly fragmented.	Not attempted

manage and evaluate software **SE-12-09** applies methods to manage and document the development of a software project	**A3. Fixes** (10 marks)	**10 – 9** Clearly explains how each identified vulnerability was resolved, referencing code snippets, before/after comparisons, and effective tool usage (e.g., for retesting). Demonstrates secure coding principles comprehensively.	**8 – 7** Addresses most vulnerabilities with good clarity, including evidence of the fixes. Some minor areas may need more detail or completeness.	**6 – 5** Offers partial fixes with moderate explanation. Some vulnerabilities may be insufficiently documented or justified in terms of code changes.	**4 – 3** Fixes are simplistic or poorly explained. May lack consistent references to how tools/resources were used to verify or implement solutions.	**2 -1** Minimal or no effective fixes presented. Code snippets or rationale are absent, unclear, or fail to address original issues.	**0** Not attempted
SE-12-02 – Applies structural elements to develop programming code **SE-12-06 –** Justifies the selection and use of tools and resources to design, develop, manage, and evaluate software	**Component B: Secure Code Submission** (15 marks)						
	B1: Code Fixes for Identified Vulnerabilities (5 marks)	**5** All identified vulnerabilities are thoroughly fixed in the code. Clear in-line comments or docstrings justify changes, showcasing advanced security and coding standards.	**4** Most vulnerabilities are adequately resolved. Comments explain how the fixes strengthen code security. Minor omissions or small residual issues.	**3** Fixes applied to key issues but with limited explanation or incomplete coverage. Some vulnerabilities may remain.	**2** Inconsistently fixes vulnerabilities. In-line commentary is superficial or unclear.	**1** Few or no fixes provided. Code changes lack explanation, leaving vulnerabilities largely unresolved.	**0** Not attempted
	B2: Implement Secure Coding Practices (5 marks)	**5** Code demonstrates robust security measures (e.g., parameterised queries, data validation, password hashing) integrated throughout. Justification of security decisions is clear and well-documented.	**4** Implements a range of secure coding practices with minor gaps. Explanations of decisions are mostly sound.	**3** Applies some secure techniques but coverage is patchy. Details on rationale are moderate or unclear.	**2** Security practices are minimal, with inadequate or inconsistent integration in the code. Little explanation provided.	**1** Shows negligible secure coding efforts. Code remains vulnerable, lacking documentation or justification.	**0** Not attempted

	B3 Test & Verify (5 marks)	**5**	**4**	**3**	**2**	**1**	**0**
		Retesting shows all previous exploits fail. Comprehensive documentation (screenshots/logs) confirms fixes. No new vulnerabilities introduced. Clear alignment with project documentation	Most exploits are successfully mitigated and retested with good evidence. Minor gaps in either documentation or final verification.	Some retesting provided but partially documented. Certain exploits may still function or lack clear verification.	Retesting is minimal or poorly documented, leaving doubt about fixes' effectiveness. Project documentation is inconsistent.	No meaningful retesting or verification of solutions. Little or no evidence that the solutions work.	Not attempted
SE-12-06 – Justifies the selection and use of tools and resources to design, develop, manage, and evaluate software	**PART C: Online Quiz** (10 marks)						
	(10 marks)	**10 – 9**	**8 – 7**	**6 – 5**	**4 – 3**	**2 – 1**	**0**
		Demonstrates outstanding knowledge of vulnerabilities (e.g., from Hacksplaining) and secure coding practices (validation, hashing, etc). Answers are accurate and well-reasoned.	Demonstrates outstanding knowledge of vulnerabilities (e.g., from Hacksplaining) and secure coding practices (validation, hashing, etc). Answers are accurate and well-reasoned.	Displays moderate grasp of security concepts. Some inaccuracies or gaps.	Partial understanding of security concepts. Several misconceptions or incomplete answers.	Shows little to no familiarity with fundamental security principles. Responses are largely incorrect or missing	Not attempted