

**ASSESSMENT TASK NOTIFICATION**

**YEAR 11**

|  |  |
| --- | --- |
| **Subject Area** | Software Engineering |
| **Year** | 12 |
| **Topic** | Programming for the Web and Secure Software Architecture |
| **Task Type** | Practical Programming Task |
| **Date of Issue** | 14/10/2024 |
| **Date Due** | 17/12/2025 3pm |
| **Weighting** | 20 |
| **Task Number** | 1 |

|  |
| --- |
| **Warning:**  Students should regularly perform a ‘GIT Commit’ with basic comments before pushing changes to their repository hosted on GitHub to document the development of their solutions.  Students who do not appropriately document their software development and produce a solution that does not align with their skill set, individual coding vernacular, and/or previous solutions will be asked to prove that their work is all their own work. Students who cannot prove that the work submitted is their own will be awarded a 0 mark and placed on the NESA malpractice register. |

|  |
| --- |
| **Outcomes Assessed:**  SE-12-03 - analyses how current hardware, software and emerging technologies influence the development of software engineering solutions  SE-12-04 - evaluates practices to safely and securely collect, use and store data  SE-12-05 - explains the social, ethical and legal implications of software engineering on the individual, society and the environment  SE-12-06 - justifies the selection and use of tools and resources to design, develop, manage and evaluate software  SE-12-07 - designs, develops and implements safe and secure programming solutions |

|  |
| --- |
| **Description of the task:**  Your team's Senior Software Engineer has tasked you with developing a simple in-house progressive web app that all software engineers in the team can use to maintain their developer logs/diaries. The Senior Software Engineer has given you the following requirements/specifications:   1. Security evident in all phases of the software development lifecycle 2. Ability for new team members to self-sign up 3. Two Factor Authentication (2FA) and session management 4. Data stored in a SQL Database (optional API interface) tables should be abstracted and joins used to relate queries. 5. Developer log entries are time/date stamped 6. Application meets minimum WC3 PWA standards, providing installation and online/offline functionality 7. Allow developers to search/filter entries by developer, date, project or log/diary contents 8. The app is modelled using: Level 0 data flow diagram, Structure chart & Data dictionary   Support resources:   * <https://github.com/TempeHS/Secure_Flask_PWA_Template> * <https://github.com/TempeHS/The_Unsecure_PWA> * <https://github.com/TempeHS/Flask_PWA_Programming_For_The_Web_Task_Source> |

**Example diary entry:**

**Developer**: Ben Jones

**Project**: SE2025

**Start Time**: 8:45am 09/October/2024

**End Time**: 10:15am 09/October/2024

**Diary entry:** 4:15pm 09/October/2024

**Time Worked**: 2.5 hours

**Repo**: <https://github.com/TempeHS/The_Unsecure_PWA-Source.git>

**Developer Notes**:

I Worked on the example API, implementing rate limiting to increase API security. The documentation was easy to follow: https://flask-limiter.readthedocs.io/en/stable/. I executed a range of torture tests using HTML requests from browsers and Postman to make sure it functioned to specification.

from flask\_limiter import Limiter

from flask\_limiter.util import get\_remote\_address

api = Flask(\_\_name\_\_)

cors = CORS(api)

api.config["CORS\_HEADERS"] = "Content-Type"

limiter = Limiter(

get\_remote\_address,

app=api,

default\_limits=["200 per day", "50 per hour"],

storage\_uri="memory://",

)

@api.route("/", methods=["GET"])

@limiter.limit("1/second", override\_defaults=False)

*Note: The Time worked should automatically round up to 15min increments for billing client.*

|  |  |
| --- | --- |
| **Steps** | **What I need to do/when I need to do it** |
| Apply the fundamental software development steps in relation to their project. | Gather a group of Junior Software Engineers (students in your class) to gather requirements for the project.  Define the specifications for the application. |
| Set up the development environment. | Install the required extensions and packages.  Create the required folder structure and files. |
| Database Design. | Develop a schema for your database, including field names, primary and foreign keys, and joins. |
| Designing Algorithms. | Develop a structured algorithm using pseudocode or flowcharts, including the use of subprograms, passing parameters and accessing files. Algorithms should include sequence, selection, iteration and sub-programs.  Develop SQL queries to access the database to insert and retrieve relevant data.  Create a Data Dictionary for your database schema. |
| Designing a user interface (UI). | Develop a storyboard for the website that considers accessibility and inclusivity and applies appropriate design, user interface (UI) and user experience (UX) principles.  Write privacy policy |
| Developing Solutions with Code with an Agile Approach. | Develop and construct your website using HTML and CSS.  Convert your algorithm into code using:   * Control structures * Data and Structures * Standard Modules * Functions (including parameter passing)   Database commands (inserting, retrieving and querying)  Document and implement a 2FA method  Document and implement session management  Document and implement appropriate data structures and file handling that support data storage.  Document and implement an appropriate hashing algorithm for securely storing passwords in the database.  Complete user accessibility testing using a range of appropriate scenarios. |
| Write a Security Summary Report. | Summarise what processes, technical solutions or design principles you applied at each step of the software development lifecycle.  Summarise how you have applied privacy by design in your solution.  Using a Threat Summarise how you have minimised vulnerabilities in user action controls. Including:   * broken authentication and session management * cross-site scripting (XSS) and cross-site request forgery (CSRF) * invalid forwarding and redirecting * race conditions |

## Additional Summary Report Guidance

**Diagrams:**

You need to provide the following models of you app, make sure your model aligns to the style, symbols and connections defined in the course specifications:

* Level 0 data flow diagram
* Structure chart (make sure flags are correct
* Data dictionary (focusing on SQL database)

Each model should be exported to A4 size and be laid out, so it is clear and easy to read.

**Security Summary Report:**

You need to write a 3-5 page report, do not go over 5 pages.

Summarise what processes, technical solutions or design principles you applied at each step of the software development lifecycle.  ->

Use the SDLC as sub headings

* For each phase identify and describe 2-3 processes that you applied to improve the secure architecture of your app
* Explain how these processes made that phase of the SDLC improved the security of your application.
* Summarise how you have applied privacy by design in your solution.
* Answer the question referring to SDLC

Using a Threat/Summary structure, state how you have minimised vulnerabilities in user action controls. Including: broken authentication and session management, Cross-site scripting (XSS), cross-site request forgery (CSRF), invalid forwarding and redirecting and race conditions,

* Use a threat/countermeasure approach so for each of the vulnerabilities listed you:
* State the threat and briefly describe it in your words
* List and explain the countermeasures you HAVE implemented
* Explain how the countermeasure collectively will minimise the vulnerability.
* Read more about threat/countermeasure but not the table provided is far to simple, I want detail and knowledge

**General:**

* Be succinct and clear no rewording questions or prolonged lofty sentences
* Give small code snippets only if it enhances your argument or simply say "As applied in main.py lines 5-9" (you can even hyperlink from the repository to those lines in your report)
* You are trying to show me you know the course specifications for models, and you understand the vulnerabilities and how to counter measure them
* Put the 3 models and 5-page report in clean simple PDF with a clean simple cover page and table of contents.
* Do not write a summary, it won't be read

|  |  |  |
| --- | --- | --- |
| **Marking rubric** | |  |
| **Grade** | **Content of Task** | **Result** |
| **Grade**  **A** | The student demonstrates an extensive understanding of the steps used by programmers when designing private and secure software by design.  The student develops highly effective algorithms to demonstrate the logic required for a software solution.  The student develops highly effective software solutions using HTML, CSS and the JavaScript programming language and Node.js and/or Python Flask frameworks.  The student demonstrates an extensive understanding of designing and creating user interfaces and user experiences to meet accessibility requirements.  The student develops an appropriate database schema and applies highly effective and secure SQL queries to access the database.  The student implements an Authentication API.  The student demonstrates an extensive understanding and application of testing and debugging of the software solution. |  |
| **Grade**  **B** | The student demonstrates a thorough understanding of the steps used by programmers when designing secure software and addressing privacy issues.  The student develops effective algorithms to demonstrate the logic required for a software solution.  The student develops effective software solutions using HTML, CSS and the JavaScript programming language and Node.js and/or Python Flask frameworks.  The student demonstrates a thorough understanding of designing and creating user interface and user experience to meet accessibility requirements.  The student develops an appropriate database schema and effective and secure SQL queries to access the database.  The student demonstrates a thorough understanding and application of testing and debugging of the software solution. |  |
| **Grade**  **C** | The student demonstrates a sound understanding of the steps used by programmers when designing software with evidence of privacy and security.  The student develops sound algorithms to demonstrate the logic required for a software solution.  The student develops sound software solutions using HTML, CSS and the JavaScript programming language and Node.js and/or Python Flask frameworks.  The student demonstrates a sound understanding of designing and creating user interface and user experience to meet accessibility requirements.  The student develops a schema that demonstrates a sound understanding of database construction and SQL queries.  The student demonstrates a sound ability to test and debug a software solution. |  |
| **Grade**  **D** | The student demonstrates a basic understanding of the steps used by programmers when designing software.  The student develops basic algorithms to demonstrate the logic required for a software solution.  The student develops a basic software solutions using HTML, CSS and the JavaScript programming language and Node.js and/or Python Flask frameworks.  The student demonstrates a basic understanding of designing and creating user interface and user experience to meet accessibility requirements.  The student develops a schema and queries that demonstrates a basic understanding of databases.  The student demonstrates a basic ability to test and debug a software solution. |  |
| **Grade**  **E** | The student identifies the requirements for the documentation and production of a software solution. |  |