Secure Software Development

Team Meetings for Team 4

Notes:

Date	24 Aug 2021	**Meeting ID**	3
Attendees	Andrey Smirnov; Taylor Edgell; Michael Justus;		

Agenda

- Discuss and feedback on UML solution diagrams.
  - Naming conventions.
  - Actions.
  - Activity diagrams and swim lanes.
  - Roles and permissions.

Minutes of Meeting

UML Solution Design

- We discussed the design thinking about Standard Report, Safety**,** and authors of content.
- The generation of Use Case diagrams was accepted by all members and agreed that the system is small, therefore not much detail can be shown on this diagram.

- **Security Tokens.**
  - We discussed the role of security tokens in the solution design.
  - It was considered that users are created by an Admin user.
  - In terms of UML diagram, use of Call Activity nodes can be leveraged to show additional details.
  - A database of dummy passwords will be created. These passwords will be stored as encrypted text.

- ***Design Changes: Additions***
    - *Authentication Class.* The team considered the use of an authentication calls to provide a single point of access control for code developed in Part 2. Any assumptions about the authentication of a security token will be handled by the Authentication Class.

    - *Security Levels.* We will link security to Safety Entry class

- ***Design Changes: Removals.***
    - The Creator class will be removed and replaced with an authentication link between Users and Safety Entry.

- **Activity Diagram.**
    - The team discussed labelling Decision Nodes with the type of check performed. One approach was to label the Decision Node, while an alternate approach is to create an activity node *prior* to the Decision Node that performs the check (the Decision Node, therefore, surfaces the choice made by the Activity Node).

- **Part 2: API Design.**
    - In terms on delivery of an API, the module dealing with development of an API only shows a bare-bones technique, therefore our API implementation very likely will not be considerably complex beyond what the module introduces.

- **Part 2: Class Design.**
    - Static classes will not be leveraged in Python for Part 2.
    - Dependency injection is not required for Part 2 implementation.

- **Part 2: Data**
    - Data will be stored securely, whether in a database or a text file.

Other Notes

1. Another discussion will be scheduled around security vulnerabilities prior to Part 2.

2. Knowledge of secure software shall be demonstrated through our tests and learned outcomes. We will document what we knew before the tests, what was identified during testing and recommendations made as a result. The idea expressed was to demonstrate the steps taken to mitigate vulnerabilities in a monolith.