

**National College of Ireland**

**Project Submission Sheet**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name:** | David O Connor | | |
| **Student ID:** | x23153784 | | |
| **Programme:** | Computing | **Year:** | 4 |
| **Module:** | Secure Programming | | |
| **Lecturer:** | Eugune McLaughlin | | |
| **Submission Due Date:** | 31/10/25 | | |
| **Project Title:** | Secure Programming CA 1 | | |
| **Word Count:** | ……………………………………………………………………………………………………………… | | |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

|  |  |
| --- | --- |
| **Signature:** | David O’Connor |
| **Date:** | 28/10/25 |

**PLEASE READ THE FOLLOWING INSTRUCTIONS:**

1. Please attach a completed copy of this sheet to each project (including multiple copies).

2. Projects should be submitted to your Programme Coordinator.

3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.

4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**

5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

**6. Please check that you read AI and Academic Integrity Acknowledgement Supplments in this document**

|  |  |
| --- | --- |
| **Office Use Only** | |
| Signature: | David O Connor |
| Date: | 28/10/25 |
| Penalty Applied (if applicable): |  |

# AI Acknowledgement Supplement

# [Secure Programming]

# [Secure Programming CA 1]

|  |  |  |
| --- | --- | --- |
| **Your Name/Student Number** | **Course** | **Date** |
| **x23153784** | Computing | 28/10/2025 |

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click [here](https://libguides.ncirl.ie/useofaiinteachingandlearning/studentguide).

# AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

|  |  |  |
| --- | --- | --- |
| **Tool Name** | **Brief Description** | **Link to tool** |
|  |  |  |
|  |  |  |

# Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used**.

|  |  |
| --- | --- |
| **[Insert Tool Name]** | |
| [Insert Description of use] | |
| [Insert Sample prompt] | [Insert Sample response] |

# Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

# Additional Evidence:

[Place evidence here]

# Additional Evidence:

[Place evidence here]

Secure Programming CA 1 Auth System Class

I have been tasked to secure an Auth System with best practices and I must explain the faults within the code by classifying them where they fall short under principles such as the CIA Triad.

# PRIVATE MAP<STRING, User> users = new HashMap

To start with the faults within the code let's go to how the accounts are stored.

The accounts are stored in memory, which breaks all 3 pillars of the CIA triad.

Starting with Confidentiality, accounts are easily accessible and visible to any attacker that is sufficient in understanding memory which breaks down the trust of the end user who believes that their account is stored securely.

Next is integrity, due to the accounts being stored in memory and are stored as strings. When a user goes to log in, an attacker will be able to intervene and steal their details and impose as the end user.

Next pillar that it breaks is the availability, when an account of an end user is stored. The storage should be persistent and constantly available to the end user when they wish to come back and login because of the accounts being stored in a temporary memory, when their session ends so does their account.

Well, how about this line code in defense in depth? There is no defense in depth for this; we are not storing the password in a secure database; there is no form of authorization or authentication when using this code to confirm a user. There are multiple factors why this code does not fit under the defense in depth. Another example would be that there is no form of access privilege put in place because the accounts are stored in memory, so all it would take to remove the account is someone with knowledge in memory analysis.

# USER CLASS

The next segment of code I will cover is the USER CLASS.

This code segment does not abide by the confidentiality of the Triad because the password is stored in plain text. There is no encryption for the password to cover what it may be.

Integrity, it also breaks this pillar in turn with the Confidentiality because the account is esstentially exposed to any attacker who are sufficient in memory analysis, it would take the attacker little effort to hack an users account and change their details such as password and username.

Availability; user class sets the format of a user. In terms of breaking availability, it can be awkward to distinguish if it does break it. In my opinion, because this user class isn’t a form of storing the class, I would say that it does not break the availability, but I can imagine in the eyes of a different cyber security that it would in some shape or form.

Defense in depth? Yes, I believe that it would break defense in depth. There is a lack of control to protect anything within it. The password is stored in plain text; the login Attempts counter is available to anyone the same as the password. There is no form of encryption to protect the user.

# Register Method

# With confidentiality, this Boolean method would break that because the user and password are both stored within plain text. There is no encryption for the password in place.

With Integrity, I would say it is quite loose. Lack of controls in place to prevent someone from getting into your session because there is no session id or event happening preventing someone from doing that. Where this could benefit and not break the integrity of this is creating an encrypted session token attached to your device during this session preventing the possibility of a intervention of details during the initial sign up of your account. However, the main thing that does well here is that if you try to enter a username already on the system, it will prevent you from doing that.

Availability, I would say in its simplest form that it does uphold availability in the sense that the user will always be able to use the account that they made the name and password meaning that the access to the account is always there.

Defense in depth, well no rolling back to the integrity there is no encrypted session token made specifically for your device, so your use of the method is unsafe. However, it does contain a nice method to prevent collisions of usernames which can be considered defense, but it doesn’t account for if someone was to leave their username blank or the possibility that the username is null which can disrupt the system and create vulnerabilities.

# Login Method

Confidentiality, I would say that it is another broken rule here, and that is because of timing attacks in this case and plainly stored users. Since the system will take different times between an actual account and a non-existent account. On top of that, there is a leak of session token information exposed to anyone who knows how to check the code or the console.

Integrity, there are no session tokens meaning that any attacker can take advantage of that, and this compromises session integrity.

Availability, there are no lock enforcements the methods are there, but nothing is happening within the code. The counter does not go up if a user fails; it does not get reset if a user succeeds in a login.

Defense in depth, although this is an application to be run on your device as a small project in a real-world perspective, we would require more than one factor to determine the user who is logging into the system; it would need multifactor authentication. Coming back to the point of the constant-time comparison, not being in place does put all users at risk because of time-based attacks. Another issue is the sessions not being encrypted; they are exposed and easily exploited and abused by attackers who are aware of holes in the fortress.

# 