# Assessment title: Collaborative Software Project

## Assessment method: Product (Project/Assignment/Case Study) Unit information

Course name: **22603VIC Certificate IV in Cyber Security** Unit[s] of competency: **ICTPRG434 Automate processes**

## Assessment information for student

Task number: 3 of 3

Due date: See Gordon Online

Where assessment will take place: The Gordon, in-class

Time available to complete this assessment: 4 weeks

## Assessment requirements

In this assessment, you'll work in small groups to automate a real-world process. Your task is to select a case study, design an algorithm, and implement it using Bash, PowerShell, or Python. You'll create a functional program accompanied by thorough documentation that includes both development and user instructions. The final deliverable is a comprehensive report covering all stages of the project, testing your technical skills, problem-solving abilities, and teamwork.

Specific assessment requirements (if any)

N/A

### ThePerformanceandKnowledgeskills you must demonstrate

This assessment will display your knowledge of:

* Debugging Methods
* Principles of algorithms
* Scripting language syntax and techniques
* Script documentation methods
* Organisational guidelines and policies It will also test your ability to:
* Develop algorithmic solutions for a chosen case study• Create functional scripts to automate a selected process.
* Document the solution, catering to both internal and external stakeholders. This includes developer-level and user-level documentation.

### Your assessor will provide these resources

* Unit resources and materials
* Computer and internet access if necessary, and access to Gordon Online. You (the learner) will need to provide these resources
* Personal computer
* Storage device (USB drive/SSD)

### Plagiarism

The Gordon maintains its training integrity and has processes in place to identify plagiarism. Where a student uses material from other authors, the original source must be referenced.

Penalties apply for substantiated cases of plagiarism; penalties may include recording a Not Satisfactory result for the assessment and may result in a Fail being recorded against the specific unit linked to the plagiarism.

## Requirements of this assessment task

To successfully complete this assessment, you must collaborate in a team of 2-3 to select and work on a case study. This involves participating in brainstorming sessions to create and document solutions, designing an algorithm using sequence, selection, and iteration methods, and representing it through flowcharts or pseudocode. You'll then implement this design in Bash, PowerShell, or Python, ensuring thorough testing and debugging, possibly using platforms like GitHub. Your task also includes writing detailed developer-level documentation about your code and its architecture, along with user-friendly documentation for program operation. Finally, compile a comprehensive report with an executive summary, detailed project insights, and an explanation of your design and implementation process. The report should also include appendices for source code, extra documentation, and references.

How the task is to be submitted

Via Gordon Online

How feedback will be provided

Feedback will be provided via Gordon Online

# Project Outline

As part of this project, you and your team of 2-3 members will automate a real-world process, engaging with every phase of the software development lifecycle. You will:

1. **Select a Case Study**: Choose one from the list provided at the end of this document.
2. **Analysis and Design**: Analyse the problem in your selected case study. Collaborate with your team to design an algorithm that meets the identified needs, using flow-charts pseudocode, UML diagrams etc.
3. **Implementation**: Once you have designed your program, will must now implement your algorithm using Bash, PowerShell, or Python. The choice of language depends on your case study's specific requirements.
4. **Documentation**: After your project is implemented, tested and debugged, you must document the final product. Your submission must include detailed documentation covering:
   * Developer-level information explaining your program's design and algorithm.
   * User-level instructions for operating the program.
   * The rationale behind your design decisions and any design iterations.
   * Initial case study analysis, brainstorming sessions, and algorithmic representations (like flowcharts or UML diagrams).
   * Pseudocode and the transition to the final code.
   * Records of team meetings and discussions.

Each step of your project, from conception to final implementation, should be welldocumented and included in your final report.

## Part 1: Group Formation & Case Study Selection

**Task**: Form groups of 2-3 and select a case study from the provided list.

* Collaborate to choose a case study that interests all group members.
* Ensure you understand the requirements of the project, and the task that must be automated. See your trainer if you are unsure.
* Conduct an initial meeting and brainstorming session. Decide upon the most appropriate programming language to be used for the project and highlight the current process requirements and the benefits of automating it.
* Include any organisational guidelines and restrictions that the automation solution should follow.

## Part 2: Design the Software Solution

**Task**: Discuss and establish potential solutions, documenting your ideas and creating an algorithm.

* Hold regular team meetings to brainstorm and refine ideas.
* Document these discussions (meeting minutes, notes).
* Design an algorithm using sequence, selection, and iteration, ensuring it handles all user options and scenarios.
* Represent your algorithm as a flowchart and pseudocode. You can also use UML diagrams or other design tools if you wish.

## Part 3: Implement the Solution

**Task**: Translate your algorithm into a working program using your chosen programming language.

Code must follow the designed algorithm accurately and be as user-friendly as possible. It should be a **complete program**, and not just a single function.

* Perform thorough testing to ensure all scenarios are handled.
* Document and correct any errors or unhandled situations. If necessary, get a student from another group to test-drive your program and see if they can break it!
* It is recommended that you use [GitHub,](https://github.com/) [GitLab,](https://about.gitlab.com/) or another platform to allow you to work on source code collaboratively. Services like [Trello](https://trello.com/) allow you to organise and prioritise tasks in a clear, visual format.

## Part 4: Create Documentation

**Task**: Write detailed documentation for future developers who might work on your project, as well as documentation aimed at non-technical users to help them understand and use your program.

For developer-level documentation:

* Document the purpose, architecture, and flow of the program.
* Include detailed comments within the source code.
* Explain complex or critical sections of the code for clarity.

For user-level documentation:

* Create clear, concise, and informative user guides or help files.
* Include step-by-step instructions for using the program.
* Provide troubleshooting tips and FAQs.

The developer-level documentation is particularly important, as it will show your trainer that you understand the code and algorithms that you have implemented.

**Part 5: Compile the Final Report**

**Task**: Compile your work into a final report.

The final report should encapsulate all aspects of your work, presented in a clear, professional, and well-organised manner. **Use the provided templates** to ensure all aspects of your project are fully covered.

To do this, you may use regular word-processing software such as Microsoft Word, or you can alternatively [use markdown.](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet) Markdown is a particularly good option, due to its ability to embed flowcharts and automatically colourise program source-code. Some good applications for editing markdown are [MarkText](https://github.com/marktext/marktext) and [Typora.](https://typora.io/)

## Part 6: Final Submission

**Task**: Submit your work to Gordon onlineYour final submission should consist of:

* Your final report.
* Source code files for your project.
* User manuals, presentations, or other documentation, if this was not compiled into the report itself.

Each member of your group should submit a separate copy of the work.

# Project Case Studies

You are to choose **one** of the following projects:

## Hosts-Based Domain Blocker

To enhance network security, you are tasked with developing a program that blocks access to known ad and malware domains. You must create a program that downloads updated lists of these domains, filters them, and adds them to the hosts file of the current computer. The program must:

1. Accept options that allow the user to choose the type of content they would like to block, eg. advertisements, malware, tracking.
2. Read from a configuration file the URLs of domain block-lists to be downloaded and parsed. See sites such as [https://oisd.nl/,](https://oisd.nl/) [Developer Dan's Hosts,](https://www.github.developerdan.com/hosts/) and <https://firebog.net/>for lists of these domains.
3. Download, parse and merge the hosts files into a list of unique domains. You may need to use regular expressions to do this.
4. Back up the original hosts file. This is **extremely** important, and the user needs to have the option to roll back the changes if they wish. This means preserving the *original* hosts file and the *last used* hosts file.
5. Add the blocked domains to the existing hosts file. This must not **replace** the existing file but **append** to it.

This program must be cross-platform and be able to run on any version of Windows or Linux. The program must be able to identify the operating system the computer is currently running and determine the correct hosts path from that.

Page

## Assessment criteria

The list below contains the criteria against which you will be assessed, for this assessment. You must satisfactorily address every criterion listed to pass the assessment.

### **Criteria**

1. Shows understanding in pinpointing a process that requires automation within the chosen case study.
2. Follows organisational guidelines and instructions in the automation process.
3. Demonstrates active participation in team discussions to establish potential solutions and process requirements.
4. Develops an algorithm that accounts for all scenarios, is guaranteed to terminate, and is accurately described using sequence, selection, and iteration.
5. Effectively creates and amends an abstract design for the solution.
6. Successfully translates the abstract design into a working script using the chosen programming language, following organisational guidelines.
7. Produces comprehensive internal documentation that aligns with organisational policies.
8. Shows proficiency in checking, identifying, and amending both syntax and semantic errors in the script.
9. Collaborates with team members to identify and resolve problems in the script and code.
10. Creates detailed documentation for both users and developers and finalises it effectively.
11. Actively seeks feedback and achieves sign-off from the required personnel.

Page