Title: e-Portfolio Submission – https://helenhelene.github.io/eportfolio/

Final Reflection

(Word count: 1063)

Introduction

This reflection summarises my learning outcomes from the Secure Software Development (SSD) module. In addition to expanding my theoretical understanding of secure software, it strengthened my technical abilities and team-oriented skills. The module offered valuable insights into identifying and managing security risks, designing robust software, and working in diverse teams.

Summary of learning outcomes

A central aspect of the module involved a Team project – Design Document, where we designed a secure application for an online retailer. We produced a detailed document outlining core requirements, security concerns with mitigation strategies, relevant development methodologies, UML diagrams, and approaches to testing.

The second assignment, Individual Project – Output and Evidence of Testing and Demonstration, required implementing our prior design in Python. This entailed making appropriate security decisions, integrating the chosen libraries, and deploying effective testing techniques. Although the module did not provide an Integrated Development Environment (IDE), I self-learned Visual Studio Code (VS Code), which I had briefly explored during a previous module, enabling me to gain additional hands-on experience.

Beyond coding, the requirement to demonstrate our work pushed me to think carefully about how to present software to stakeholders, which is an aspect of development often overshadowed by purely technical tasks.

Finally, the third assignment comprised an e-portfolio submission (Appendix 1) and reflection documenting my journey through the MSc Computer Science. My Professional Profile (Appendix 2) originally contained personal life topics, but professor's feedback encouraged me to revise it to highlight professional attributes instead.

Throughout the module, I completed the majority of e-Portfolio tasks compiled in each unit (Appendix 3) and documented in the List of Artefacts (Appendix 4). These included UML and security discussions, various coding exercises that covered recursion, regular expressions, cryptography, and API development.

Several activities stood out. For instance, the first collaborative discussion - UML Flowchart (Appendix 5), helped me improve an Activity Diagram illustrating a secure login process. Professor's feedback prompted me to enhance both structure and clarity, which proved central to our Design Document and future work.

The Towers of Hanoi exercise in Unit 4 (Appendix 6) introduced me to a mathematical puzzle I had previously seen only as a toy. Implementing a recursive solution was both

stimulating and rewarding. I even acquired a miniature Hanoi Tower for my desk as a lasting reminder of the underlying logic.

Units 7 and 9 emphasised API development (Appendix 7). Configuring multiple terminals and using w3m within a Windows environment proved tricky, yet working through these problems enriched my problem-solving skills. These real-world scenarios mirrored the hurdles one might face in a professional setting.

In Unit 8 Cryptography Programming (Appendix 8), I explored Fernet encryption from the Python cryptography library. Although I had used cryptography tools in a practical sense before, this exercise afforded me a deeper understanding of how best to manage encryption libraries and methods for enhanced software security.

Evidence of Teamwork and Peer Interactions

Coordinating a multinational team with varied schedules and language backgrounds tested our flexibility. We held regular meetings and drafted a thorough team contract (Appendix 9) that summarised our responsibilities, ensuring that each member's skills were fully utilised. I took charge of documentation, compliance, and administrational tasks, while more coding-focused team members handled library selection and deep technical queries. Employing Trello (Appendix 10) helped us keep track of tasks and maintain clear accountability.

Even after finalising our team project, collective support continued. We shared insights, exchanged ideas, and clarified uncertainties throughout the preparation for Individual Assignment 2. A senior developer in my network offered invaluable help when I encountered setbacks with Python-based libraries. Without their support, relying solely on the coding activities in the module may not have been sufficient to successfully finalise Assignment 2. Whilst there is still room for refinement, seeing the software run reliably without errors confirmed the effort invested.

Learning and Changed Actions

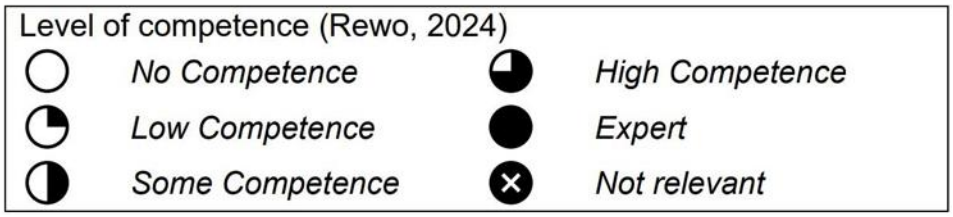
The scale of this module felt vast at first, particularly for a part-time student juggling assignments alongside professional obligations. Nonetheless, the complex workload propelled me to sharpen my time management strategies and acknowledge the need for external support on occasion. I learned to proactively clear hurdles related to IDEs, coding complexities, and international team interactions, all of which boosted my confidence and preparedness for future software projects.

Demonstrating the software to the professor presented another challenge. The mixture of anticipation and apprehension ensured I practised running the programme repeatedly, simplified processes, and rehearsed potential questions. Ultimately, the demonstration's positive reception reinforced my belief in the importance of careful presentation alongside technical competence.

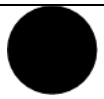
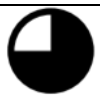
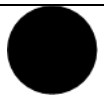
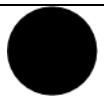
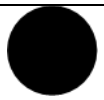
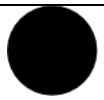
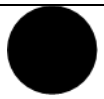
During the Cryptography Programming exercise in Unit 8 (Appendix 8), I initially planned to implement Fernet encryption for password handling in Assignment 2. After greater scrutiny, however, I realised it was not the most suitable approach for password management. This prompted additional research on password-hashing methods and a rewrite of my code, highlighting the importance of fully justifying technical decisions and applying critical thinking throughout a project.

Although my professional role in a management team rarely involves hands-on coding, the broad skills gained have enhanced my capacity to identify, select, and oversee the most suitable development tools for genuine business scenarios. Completing high-pressure group deliverables with tight deadlines and presenting my software to stakeholders proved both transformational and academically enriching.

Professional Skills Matrix



Skills	Competence	Achievement
Time Management	●	Met strict module deadlines whilst balancing professional duties.
Resilience	◕	Overcame challenges posed by cross-cultural teamwork, new tools, and adapted to coding hurdles.

Critical Thinking and Analysis		Evaluated security solutions and cryptographic methods.
Problem-Solving		Self-taught IDE usage to complete coding exercises; found workarounds for w3m on Windows.
Communication and Literacy skills		Refined UML diagrams and final demonstration delivery.
IT and Digital		Integrated libraries in Python, adopted best practices in secure design.
Interpersonal		Maintained positive team relationships and participated actively in group meetings.
Teamwork / Global Citizen and Leadership		Managed documentation while peers addressed technical complexities and contributed to the overall success of the team.
Emotional Intelligence		Harmonised differing perspectives in team discussions.
Critical Reflection		Remained mindful of strengths and weaknesses, diligently applying lessons to future projects.

Conclusion

In conclusion, the SSD module has been an invaluable experience, broadening my knowledge of secure software practices, highlighting thorough risk assessment, and refining my project management skills. Overcoming a demanding workload underlines the

importance of teamwork, self-directed learning, and continuous reflection. These talents are directly transferable to my professional environment, ensuring I can champion software security and manage complex technical projects more effectively.

References

Rewo. (2024) What is a skills matrix. Available from: https://www.rewo.io/skills-matrix-for-manufacturing/ [Accessed 14 January 2025].

Appendix 1: e-Portfolio Submission

https://helenhelene.github.io/eportfolio/

Helen SIU

View My LinkedIn Profile

View the Project on GitHub
HelenHelene/eportfolio

This project is maintained by
HelenHelene

Hosted on GitHub Pages — Theme by orderedlist

E-Portfolio of

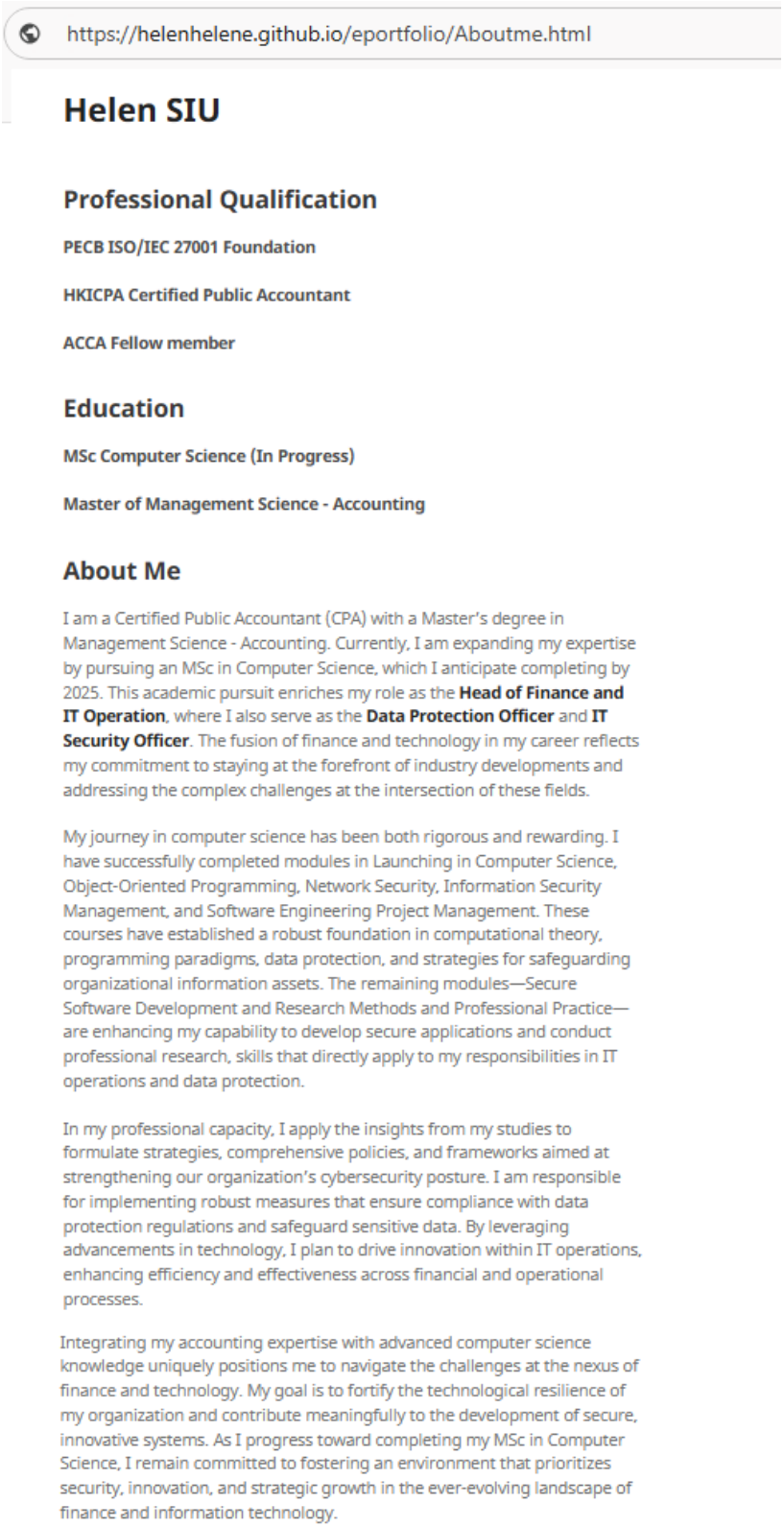
Helen SIU

Professional Profile

University of Essex Learning Experience

- Induction Module
- Module 1 Launching in Computer Science
- Module 2 Object Oriented Programming
- Module 3 Network Security
- Module 4 Information Security Management
- Module 5 Software Engineering Project Management
- Module 6 Secure Software Development
- Module 7 Research Methods and Professional Practice
- MSc Computing Project and Dissertation

Appendix 2: Professional Profile



Appendix 3: List of SSD Units

https://helenhelene.github.io/eportfolio/SSD/SSD_main.html

Module 6 Secure Software Development

In this module, we examine the security risks tied to programming languages through design and architecture strategies, programming paradigms, testing, and the role of operating systems and libraries in development. We also focus on distributed systems, APIs, and emerging trends, all within the framework of the secure software development life cycle.

Throughout the module, we will strengthen our understanding of abstraction, secure development methodologies, and the skills needed for effective analysis, design, construction, and testing. We will explore both classic and modern SDLC models (e.g., TOGAF, Agile), practice collaboration and conflict-resolution within a team, and reflect on our personal growth. By the end, we will be equipped to identify and manage security risks, select suitable development approaches, and build secure software.

There are three assignments in this module. In the first assignment, we participate in a team submission and individual peer assessment. For the team submission, we focus on designing a secure application for an online retailer and submitting a Design Document that outlines how the listed requirements will be met.

The second assignment is an Individual Project, which requires creating Python code based on the design from Assignment 1, along with accompanying test evidence and a live demonstration.

Lastly, we are expected to submit an e-Portfolio, collecting evidence of our work and submitting a reflective piece on personal development. This culminates in a final reflection that summarizes individual learning achievements and experiences.

Assignment 1: Development Team Project ***(🎓Pass with Distinction)***

Design Document

Assignment 2: Development Individual Project ***(Work in progress)***

Output and Evidence of Testing

Assignment 3: Individual Module e-Portfolio ***(Work in progress)***

Final Reflection

The units presented below serve as a compilation of evidence, showcasing the work accomplished in this module and documenting the learning journey.

Unit 1: Introduction to Secure Software Development

Unit 2: UML Modelling to Support Secure System Planning

Unit 3: Programming Languages: History, Concepts & Design

Unit 4: Exploring Programming Language Concepts

Unit 5: An Introduction to Testing

Unit 6: Using Linters to Support Python Testing

Unit 7: Introduction to Operating Systems

Unit 8: Cryptography and Its Use in Operating Systems

Unit 9: Developing an API for a Distributed Environment

Unit 10: From Distributed Computing to Microarchitectures

Unit 11: Future trends in Secure Software Development

Unit 12: The Great Tanenbaum-Torvalds Debate Revisited

You may also refer to the List of Artefacts for quick access to all artefacts.

Appendix 4: List of Artifacts

https://helenhelene.github.io/eportfolio/SSD/SSD_ArtefactsSummary.html

List of Artefacts for Each Unit

Unit(s)	Component	Artefacts
1 - 3	Collaborative discussion 1	UML Flowchart: Initial post, Peer Response 1, Peer Response 2, Summary post
2	Seminar Preparation	Scrum Security review
3	Team Activity	What is a Secure Programming Language?
3	Activity	Exploring Python tools and features (Not attempted)
3	Activity	The Producer-Consumer Mechanism
4	e-portfolio Component	Programming Language Concepts
4	Seminar Preparation	Programming exercises - recursion and regex
5	Activity	Equivalence Testing in Python
5	e-portfolio Component	Exploring the Cyclomatic Complexity's Relevance Today
6	Seminar Preparation	Exploring Linters to Support Testing in Python (Attempted but not finished)
7	e-portfolio Component	What is an Ontology?
7	Activity	Exploring a simple Python shell
7	Activity	Developing an API for a Distributed Environment
8 - 10	Collaborative Discussion 2	Cryptography Case Study - TrueCrypt : Initial post, Peer Response 1, Peer Response 2, Summary post
8	Seminar Preparation	Cryptography Programming Exercise
9	Seminar Preparation	API Demonstrations
10	e-portfolio Component	Faceted Data
11	Team Activity	Debate: Microservices and Microkernels
12	Seminar Preparation	Microservices and Microkernels Debate

These artefacts collectively document the learning journey and practical application of concepts throughout the SSD module.

Return to Module 6

Appendix 5: Collaborative Discussion 1 – Summary Post

Summary Post

◄ Initial Post

Display replies in nested form

Settings

Summary Post

by Oi Lam Siu - Friday, 8 November 2024, 6:12 AM

After reviewing the tutor and peer feedback for optimising the UML Activity Diagram and suggested security measures, I have created a new Activity Diagram (Figure 1) that implements several security measures to enhance the security of the website login process, aligned with the OWASP Top 10 Proactive Controls (OWASP, 2024).

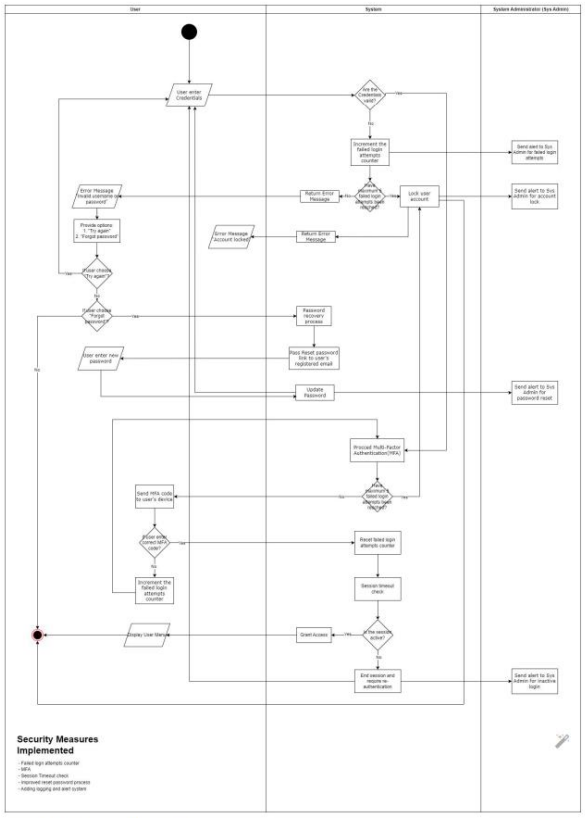


Figure 1: Activity Diagram for Website Login

Below are the details of the security measures implemented:

Failed Login Attempts Counter

The system increments a counter each time a user fails to log in successfully. After a maximum of 5 failed attempts, the account is locked.

Aligned with OWASP Proactive Control C7: Secure Digital Identities (OWASP, 2024), which emphasises protecting user identities by implementing strong authentication mechanisms, including account lockout policies to prevent unauthorised access through brute force attacks.

Multi-Factor Authentication (MFA)

Users must enter a correct MFA code sent to their registered device after providing valid credentials.

Aligned with OWASP Proactive Control C7: Secure Digital Identities (OWASP, 2024), adding an additional layer of security beyond passwords, reducing the risk of unauthorised access even if credentials are compromised.

Session Timeout Check

The system checks if the session is active and ends the session, requiring re-authentication after inactivity or a predefined time.

Aligned with OWASP Proactive Control C1: Implement Access Control (OWASP, 2024), ensuring that user access is appropriately managed and that sessions are securely maintained and terminated to prevent unauthorised use.

Improved Reset Password Process

Password reset links are sent to the user's registered email without revealing whether the account exists, accompanied by alerts to the system administrator.

Aligned with OWASP Proactive Control C7: Secure Digital Identities (OWASP, 2024), securing the process of handling digital identities, including password resets, and preventing account enumeration by not disclosing account validity.

Adding Logging and Alert Systems

Alerts are sent to system administrators for failed login attempts, account locks, password resets, and inactive logins.

Aligned with OWASP Proactive Control C9: Implement Security Logging and Monitoring (OWASP, 2024), enabling detection and response to suspicious activities, facilitating timely intervention and incident response.

By aligning the security measures in the activity diagram with the OWASP Top 10 Proactive Controls 2024, particularly focusing on C1: Implement Access Control, C7: Secure Digital Identities, and C9: Implement Security Logging and Monitoring, the risks associated with A07: Identification and Authentication Failures (OWASP, 2021) can be effectively mitigated.

References

OWASP. (2021) OWASP Top 10 – A07:2021 – Identification and Authentication Failures. Available from: OWASP [Accessed 26 October 2024]

OWASP. (2024) OWASP Top 10 Proactive Controls – Top 10 2024. Available from: OWASP Proactive Controls [Accessed 7 November 2024]

Bibliography

Gedam, N. & Meshram, B. (2023) Proposed Secure Activity Diagram for Software Development. International Journal of Advanced Computer Science and Applications, 14(6), 671-680.

Soni, R. (2020) Login Security: 7 Best Practice to Keep Your Online Accounts Secure. Available from: **https://www.loginradius.com/blog/identity/login-security/** [Accessed 8 November 2024]

Tan, T.G. et al. (2020) Securing Password Authentication for Web-based Applications. DOI: 10.48550/arXiv.2011.06257

Maximum rating: -

Permalink Edit Delete Reply

Appendix 6: Programming exercises – recursion and regex

https://helenhelene.github.io/eportfolio/SSD/SSD_Unit04_Seminar.html

Programming language concepts: Programming exercises – recursion and regex

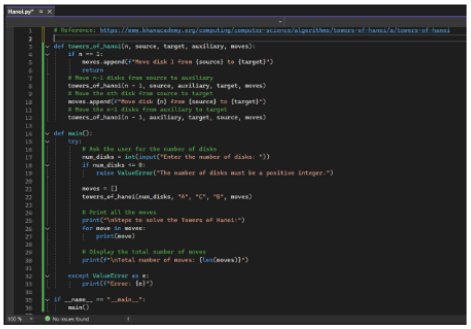
Requirement

This week there are two programming exercises that will help you understand two valuable language concepts – **recursion and regex**.

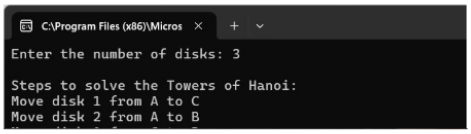
Recursion

One of the classic programming problems that is often solved by recursion is the towers of Hanoi problem. A good explanation and walkthrough are provided by Cormen & Balkcom (n.d.). (the code they used for their visual example is provided on their website as well).

Read the explanation, study the code and then create your own version using Python. Create a version that asks for the number of disks and then executes the moves, and then finally displays the number of moves executed.



```
# Reference: https://www.khanacademy.org/computing/computer-science/algorithms/towers-of-hanoi/a/towers-of-hanoi

def towers_of_hanoi(n, source, target, auxiliary, moves):
    if n == 1:
        moves.append(f"Move disk 1 from {source} to {target}")
        return
    # Move n-1 disks from source to auxiliary
    towers_of_hanoi(n - 1, source, auxiliary, target, moves)
    # Move the nth disk from source to target
    moves.append(f"Move disk {n} from {source} to {target}")
    # Move the n-1 disks from auxiliary to target
    towers_of_hanoi(n - 1, auxiliary, target, source, moves)

def main():
    try:
        # Ask the user for the number of disks
        num_disks = int(input("Enter the number of disks: "))
        if num_disks <= 0:
            raise ValueError("The number of disks must be a positive integer.")

        moves = []
        towers_of_hanoi(num_disks, "A", "C", "B", moves)

        # Print all the moves
        print("\nSteps to solve the Towers of Hanoi:")
        for move in moves:
            print(move)

        # Display the total number of moves
        print(f"\nTotal number of moves: {len(moves)}")

    except ValueError as e:
        print(f"Error: {e}")

if __name__ == "__main__":
    main()
```



```
C:\Program Files (x86)\Micros
Enter the number of disks: 3

Steps to solve the Towers of Hanoi:
Move disk 1 from A to C
Move disk 2 from A to B
```

Regex

The second language concept we will look at is regular expressions (regex). We have already presented some studies on their use, and potential problems, above. The lecturecast also contains a useful link to a tutorial on creating regex. Re-read the provided links and tutorial (Jaiswal, 2020) and then attempt the problem presented below:

- The UK postcode system consists of a string that contains a number of characters and numbers – a typical example is ST7 9HV (this is not valid – see below for why). The rules for the pattern are available from idealpostcodes (2020).

Create a python program that implements a regex that complies with the rules provided above – test it against the examples provided.

- Examples:
  M1 1AA
  M60 1NW
  CR2 6XH
  DN55 1PT
  W1A 1HQ
  EC1A 1BB

According to the rules provided by IdealPostcodes (2020) and commonly accepted patterns for UK postcodes:

- Outward Code: 1–4 characters (letters and numbers). Examples: M1, DN55, EC1A.
- Inward Code: A single digit followed by two uppercase letters. Examples: 1AA, 6XH, 1HQ.

General Regex Pattern for UK Postcodes:

```
^[A-Z]{1,2}[0-9][0-9A-Z]?\s[0-9][A-Z]{2}$
```

Explanation:
`^[A-Z]{1,2}` : The outward code starts with 1–2 uppercase letters.
`[0-9]` : Followed by one digit.
`[0-9A-Z]?` : Optionally followed by another digit or letter.
`\s` : There is a mandatory space separating the outward and inward codes.
`[0-9]` : The inward code starts with a single digit.
`[A-Z]{2}$` : Ends with two uppercase letters.

Appendix 7: Developing an API for a Distributed Environment

https://helenhelene.github.io/eportfolio/SSD/SSD_Unit07_Activity2.html

Developing an API for a Distributed Environment

Requirement

In this session, you will create a RESTful API which can be used to create and delete user records. Responses to the questions should be recorded in your e-portfolio.

You are advised to use these techniques to create an API for your submission in Unit 11.

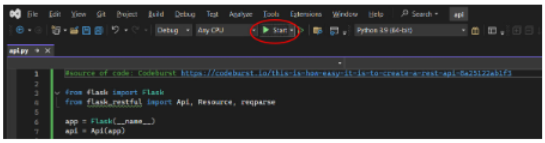
Create a file named api.py and copy the following code into it:

```python
#source of code: Codeburst https://codeburst.io/this-is-how-easy-it-is-to-create-a-
from flask import Flask
from flask_restful import Api, Resource, reqparse

app = Flask(__name__)
api = Api(app)

users = [
    {
        "name": "James",
        "age": 30,
        "occupation": "Network Engineer"
    },
    {
        "name": "Ann",
        "age": 32,
        "occupation": "Doctor"
    },
    {
        "name": "Jason",
        "age": 22,
        "occupation": "Web Developer"
    }
]

class User(Resource):
    def get(self, name):
        for user in users:
            if(name == user["name"]):
                return user, 200
        return "User not found", 404

    def post(self, name):
        parser = reqparse.RequestParser()
        parser.add_argument("age")
        parser.add_argument("occupation")
        args = parser.parse_args()

        for user in users:
            if(name == user["name"]):
                return "User with name {} already exists".format(name), 400

        user = {
            "name": name,
            "age": args["age"],
            "occupation": args["occupation"]
        }
        users.append(user)
        return user, 201
```

Question 1

Run the API.py code. Take a screenshot of the terminal output. What command did you use to compile and run the code?

I clicked the Start button (or press F5) in **Visual Studio** to compile and run the code.



Below is the output:

This output shows that the API is running on http://127.0.0.1:5000.

Question 2

Run the following command at the terminal prompt, what happens when this command is run, and why?

```
w3m http://127.0.0.1:5000/user/Ann
```

I am using Windows and Visual Studio (VS). For convenience, I run the Flask app in the VS terminal and use the Windows Command Prompt with curl to execute the HTTP commands.

Appendix 8: Cryptography Programming exercise

https://helenhelene.github.io/eportfolio/SSD/SSD_Unit08_Seminar.html

Cryptography Programming Exercise

Requirement

Read the Cryptography with Python blog at tutorialspoint.com. Select one of the methods described / examples given and create a python program that can take a short piece of text and encrypt it.

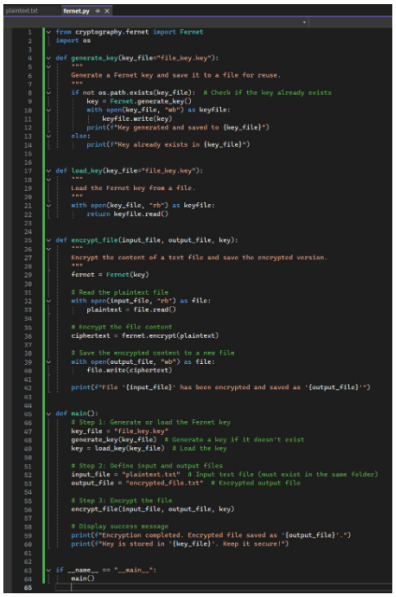
Select one of the methods described / examples given.

Before selecting a method and creating the Python program, I created a table to review all the methods described on Tutorialspoint.com.

Method	Type	Key Management	Use Case	GDPR Compliance	Performance	Weakness	Ease of Implementation
Reverse Cipher	Substitution Cipher	No key required	Simple encryption for educational purposes	No	High (Fast)	Extremely weak; easily reversible (not secure)	Very Easy
Caesar Cipher	Substitution Cipher	Single key (shared)	Encoding small text, educational purposes	No	High (Fast)	Vulnerable to brute force (only 25 possible keys)	Very Easy
ROT13 Algorithm	Substitution Cipher	No key required (fixed shift of 13)	Simple encoding, educational purposes	No	High (Fast)	Fixed shift makes it easily reversible; not secure	Very Easy
Transposition Cipher	Rearrangement Cipher	Single key (shared)	Securing small text by rearranging characters	No	Moderate	Vulnerable to frequency analysis; not suitable for modern secure communication	Moderate
Base64 Encoding	Encoding (Not Encryption)	No key required	Encoding binary data into text for transmission	No	High (Fast)	Not encryption; easily decoded by anyone	Easy

Create a python program that can take a text file and output an encrypted version as a file in your folder on the system .

Create *fernet.py* using Fernet encryption (from the cryptography module) to take a plaintext file, encrypt its contents, and save the encrypted version as a new file in the same folder.



```python
from cryptography.fernet import Fernet
import os

def generate_key(key_file="file_key.key"):
    """
    Generate a Fernet key and save it to a file for reuse.
    """
    if not os.path.exists(key_file):  # Check if the key already exists
        key = Fernet.generate_key()
        with open(key_file, "wb") as keyFile:
            keyFile.write(key)
        print(f"Key generated and saved to {key_file}")
    else:
        print(f"Key already exists in {key_file}")

def load_key(key_file="file_key.key"):
    """
    Load the Fernet key from a file.
    """
    with open(key_file, "rb") as keyfile:
        return keyfile.read()

def encrypt_file(input_file, output_file, key):
    """
    Encrypt the content of a text file and save the encrypted version.
    """
    fernet = Fernet(key)

    # Read the plaintext file
    with open(input_file, "rb") as file:
        plaintext = file.read()

    # Encrypt the file content
    ciphertext = fernet.encrypt(plaintext)

    # Save the encrypted content to a new file
    with open(output_file, "wb") as file:
        file.write(ciphertext)

    print(f"File '{input_file}' has been encrypted and saved as '{output_file}'")

def main():
    # Step 1: Generate or load the Fernet key
    key_file = "file_key.key"
    generate_key(key_file)  # Generate a key if it doesn't exist
    key = load_key(key_file)  # Load the key

    # Step 2: Define input and output files
    input_file = "plaintext.txt"  # Input text file (must exist in the same folder)
    output_file = "encrypted_file.txt"  # Encrypted output file

    # Step 3: Encrypt the file
    encrypt_file(input_file, output_file, key)

    # Display success message
    print(f"Encryption completed. Encrypted file saved as '{output_file}'.")
    print(f"Key is stored in '{key_file}'. Keep it secure!")

if __name__ == "__main__":
    main()
```

Create a plaintext file named *plaintext.txt* in the same folder as Input File. Add text "This is a test file for encryption." to it.

Appendix 9: Team Contract and Meeting Recordings

https://helenhelene.github.io/eportfolio/SSD/SSD_A1_MoM.html

Team Contract

28 October 2024

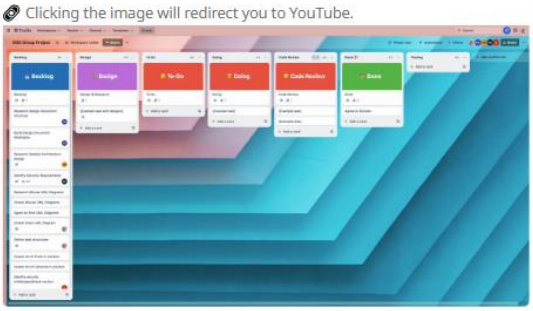
Group - GAZHA

Meeting Recordings

2 November 2024

First meeting for member introductions, project outline, and strategy.

Clicking the image will redirect you to YouTube.



SSD Group Project: Meeting #1

17 November 2024

Regular meeting to provide updates and follow up on missing tasks.

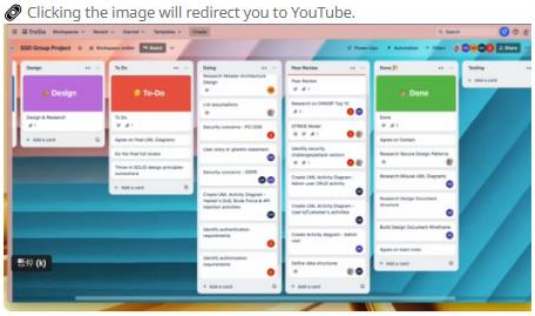
Clicking the image will redirect you to YouTube.

SSD Group Project: Meeting #2

24 November 2024

Regular meeting to provide updates and follow up on missing tasks.

Clicking the image will redirect you to YouTube.



SSD Group Project: Meeting #3

26 November 2024

We have scheduled a meeting with the professor to discuss the design document and clarify any questions.

28 November 2024

A small group meeting is scheduled to follow up on the points discussed during the meeting with the professor.

1 December 2024

Final meeting to agree on the design document and submission arrangement.

Return to Assignment 1 - Design Document
Return to Assignment 3 - e-Portfolio
Return to Module 6

Appendix 10: Trello Dashboard