Assessment Brief 2025-26

Sheffield Hallam University

Module Leader: **Aliyuda Ali**		Level: **4**	
Module Name: **Software Development**		Module Code: **55-408813**	
Assignment Title: **Software Development - Coursework**			
Individual / ~~Group~~	Weighting: **100%**	Magnitude: **wordcount/length of 3000 words.**	
Submission date/time: **05/05/2026 at 16:00 (UTC+1)**	Blackboard submission: **Yes** Turnitin submission: **No**	Format: **Word, PDF, CSV, source code, digital media.**	
Planned feedback date:	Mode of feedback: **Written via Blackboard**		
In this assessment are students asked to consider:	Inclusivity and accessability		**Not applicable**
	Sustainability		**Not applicable**

<u>Module Learning Outcomes</u>

- Apply structured programming and problem-solving techniques using Python to design and implement small, secure software solutions.
- Apply appropriate testing, debugging, and software development lifecycle (SDLC) techniques to ensure program reliability and maintainability.
- Implement and explain encryption and decryption algorithms relevant to data security.
- Analyse and interpret algorithmic performance and scalability to evaluate program efficiency.

Indicative employability skills you will develop during this module:

- **Problem-solving**
- **Analytical thinking**
- **Ability to write good-quality code**
- **Debugging and Testing**
- **Communication and program documentation.**

Assessment Brief

This coursework guides students through a series of practical programming tasks that build and test console-based Python programs. Across various sections, students will apply key programming concepts, including file handling, loops, functions, conditionals, and data structures, to solve real-world problems. Students will design and implement a working system that reads and updates CSV files, performs Caesar cypher decryption, creates and runs unit tests to verify program correctness, and conducts algorithmic complexity analysis using Big-O notation.

1. This coursework is an individual piece of work and is marked at **100%.**
2. You are tasked to analyse algorithms, design, develop, and test a console application written in **Python** as detailed in the requirements. For this module, you are required to use PyCharm, which is an appropriate Python Integrated Development Environment (IDE). If you wish to use an alternative IDE, you **MUST** seek approval from the module tutor. Support and guidance will be provided for PyCharm during scheduled lab sessions.
3. You are **NOT ALLOWED** to add additional requirements or/and use advanced programming features.
4. Failure to adhere to the stated requirements may result in further academic review, which may include investigation. Therefore, you are advised to consult with the tutor for delivering nice-to-have features.

Coursework Structure

This coursework is structured in **SIX** (6) sections:

Section and Description	**Marks (%)**
Section A: Application Design	15
Section B: Implementation	30
Section C: Encryption	15
Section D: Unit Testing	10
Section E: Algorithm Complexity Analysis	10
Section F: Video Demonstration • Application Demonstration (10%) • Technical Demonstration (10%)	20
Total marks	**100**

All sections will be marked at 100%, which will then be translated into the mark proportion for each section.

Coursework Deliverables

The following table shows the deliverables for each section of the coursework

Section	**Deliverable**
A	A Word/PDF document
B	Python code
C	Python code and a CSV file
D	Python code and a Word/PDF document
E	A Word/PDF document
F	A Word/PDF file containing only the link to a Panopto video for Section F.

IMPORTANT NOTE

The teaching team reserves the right to request a live demonstration of the submitted work to verify its quality. Failure to abide by the demo request will result in an academic misconduct report.

Word Count and Written Submissions

The total word count for this coursework is **3,000 words (±10%)** and applies **only to written documentation**.

The following deliverables are excluded from the word count:

- Python source code files
- CSV data files
- Screenshots of program execution
- Test output logs

You are advised to distribute the written word count approximately as follows:

- Section A (System Design and Documentation): **1,800 words** (inclusive of words in the submission template).
- Section D (Unit Testing – written discussion only): **500 words**.
- Section E (Algorithm Complexity Analysis): **700 words**.

Please note that these figures are indicative and provided to help you balance your written work appropriately.

Section A (15 marks)

Developing a Console-Based Library Management System

This section requires you to study the case study and its requirements for developing **illustrations/sketches** of a Console-Based Library Management System. The illustrations should be comprehensively illustrated with sample data and explanations.

Background

You are part of a Sheffield Hallam University Library team tasked with developing a console-based Library Management System using Python. The system is intended to help librarians manage book inventories and manage borrowed books from students. The application should be user-friendly and allow for basic functionalities.

Your team has been assigned specific features to implement, including adding new books, borrowing books, returning books, and viewing the list of available books. The following section outlines the requirements of the application, and you, as the development team, must consider the design, functionality, and potential challenges.

General Requirements

1. Dataset

   Study the datasets provided in this coursework (**Appendix A**) to support your application design. There are four (4) CSV files (**borrowers.csv, inventories.csv, loaned.csv, staff.csv**) provided for the application. **No additional files and fields can be added** to the datasets.

2. Users
   The console application should have two (2) user types:

   - Librarian

- Supervisor

Users can access the application functionalities based on their user types. The detailed requirements for each user are explained in the subsequent sections.

3. Login

   Both librarian and supervisor users with **<u>active status</u>** (from the dataset) should be able to log in to the application using the common login page. Users can attempt to log in **<u>three times for invalid username and password combinations</u>**, otherwise, their account will be blocked.

4. Error and Exception Handling and User-Friendliness

   The application should support exception handling and user-friendliness, such as (but not limited to):
   - invalid input
   - wrong password
   - maximum number of attempts for login
   - non-active status for login. The user will receive a message cannot log in due to their account status.
   - list of menu items/sub-menu items for users to select from.
   - enable users to go back to the main menu if necessary.

Librarian Requirements

1. Manage Borrowing Books

   All librarians can access this menu to manage borrowing books for the following sub-menu items:
   - ***Loan Books***. The librarian can search the borrower's ID and input the book's ID to borrow a book. Upon successful recording of the loan book, appropriate CSV files should be updated. The borrowing period for books is 14 days.
   - ***Return books***. The librarian should be able to input the book's ID to return the book. Upon returning a book, appropriate CSV files should be updated.
   - ***Extend loan period***. The librarian can input the book's ID to extend the borrowing period **<u>for books on loan</u>**. The extension period is 2 weeks (14 days). The extension period should be calculated automatically from the due date.

2. Search and View Books

   Librarians can search for books by the following fields to view the book information:

   - Book's ID
   - Book's title

   If the book is found, display the book information on the console.

Supervisor Requirements

1. Manage Book Inventories

   The supervisor user should be able to manage book inventories for the following sub-menu items:

   - Add new books. Only supervisors can add new books. Use the fields from the dataset to add new books.

- Update books' status. There are three statuses for books:
    - ***Available*** – The book is currently available and should be on the shelf at the library.
    - ***On loan*** - The book is not available for borrowing because it is on loan.
    - ***Deleted*** - The book is not available for borrowing because it has been marked for withdrawal from the application.

2. Manage Account
   The supervisor can change librarians' and other supervisors' status from "blocked" or "inactive" to "active".

The deliverable for Section A is **a Word/PDF document**. **You are strictly required to use the submission template provided on the Blackboard for this deliverable.**

Section B (30 marks)

Implementation of the Case Study

Using **Python**, develop **a console application** for the **Console-Based Library Management System** based on the requirements outlined in Section A. **The primary deliverable for this section is a single Python program file titled main.py.** However, students who are familiar with modular programming may, if they wish, submit multiple Python files, provided that **main.py** is clearly identified as the system's entry point, and the program executes correctly from this file. No written report is required for this section beyond brief in-code comments.

Section C (15 marks)

Decryption

Write a decryption program that implements the Caesar cipher. The program should read the CSV file (message.csv) provided in Appendix B, display its content before decryption, prompt the user for the shift value, decrypt using a while loop, and write the result to **decrypted_file.csv**. The two special characters present in the file should be left unchanged. Only alphabetic characters are shifted; case is preserved.

Your decryption program should have the following requirements:

- Read the CSV file and display the content on the console.
- Prompt the user to input the shift value. **The shift value is 11**. Exception handling should be in place for invalid input.
- A variable that has the shift alpha value should be used in the program.
- The encrypted message has whitespace characters. Whitespace and special characters should be preserved and not shifted.

- The decryption program should use a while loop.
- There are two special characters in the message that should not be decrypted. Display them as they are in the decrypted file.

The deliverables for this section are **a Python program file and a CSV file**. No written report is required for this section beyond brief in-code comments.

Section D (10 marks)

Unit Testing

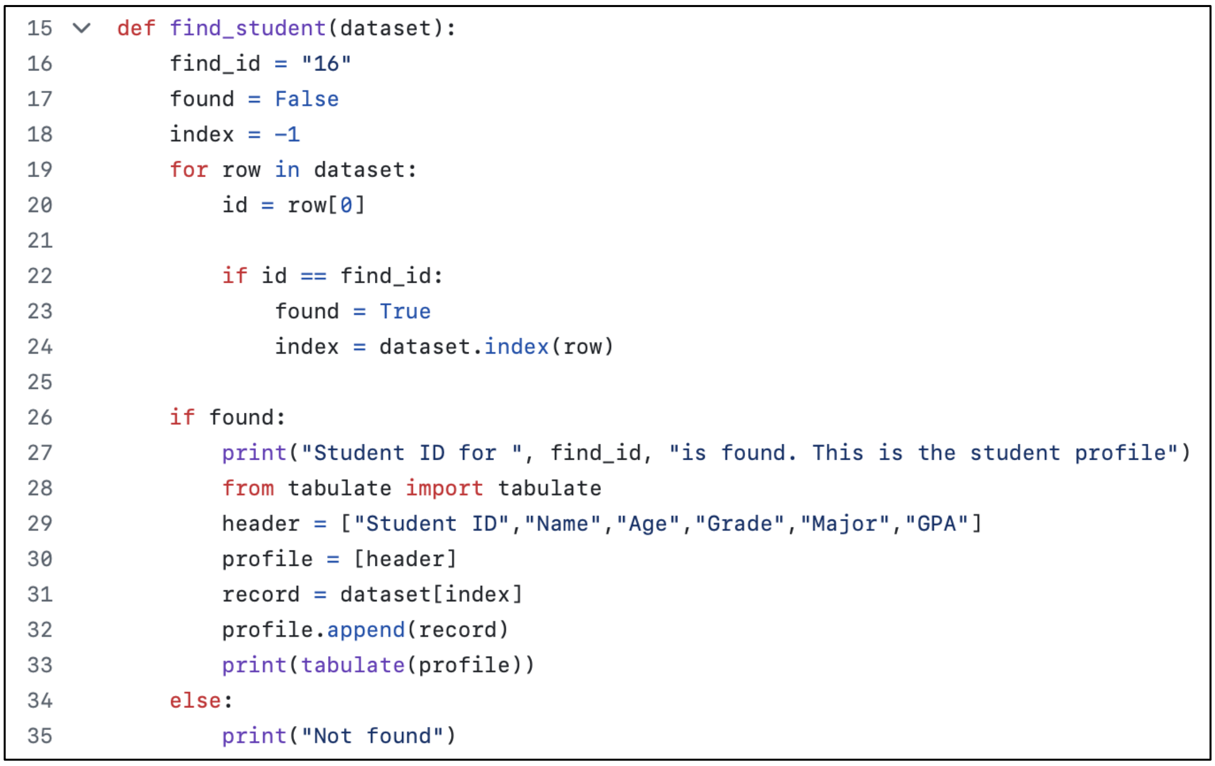
Pick **two** functionalities from **Sections A or C** for unit testing. Document your unit testing using **a unit test table as well as the code snippets**. Provide at least 3 test cases for each functionality to attain maximum marks.

The deliverables for this section are **a Word/PDF document and a Python program**.

Section E (10 marks)

Algorithm Complexity Analysis

Study the code snippet below. Analyse the code from **lines 16 to 35** for the algorithm complexity. If the dataset size is 20, what is the value of T(n) for the code?



```
15  def find_student(dataset):
16      find_id = "16"
17      found = False
18      index = -1
19      for row in dataset:
20          id = row[0]
21
22          if id == find_id:
23              found = True
24              index = dataset.index(row)
25
26      if found:
27          print("Student ID for ", find_id, "is found. This is the student profile")
28          from tabulate import tabulate
29          header = ["Student ID","Name","Age","Grade","Major","GPA"]
30          profile = [header]
31          record = dataset[index]
32          profile.append(record)
33          print(tabulate(profile))
34      else:
35          print("Not found")
```

The deliverable for Section E is **a Word/PDF document.** You are not required to understand or use the ***tabulate*** library in the above code. Focus your analysis on loop structure, control flow, and algorithm behaviour rather than third-party libraries.

Section F (20 marks)

Video Demonstration

A video recording (**one video only**) featuring a screencast and the audio from the presenter via Panopto (accessible through Blackboard), demonstrating **Sections B and C**, should be submitted. The duration of the video recording **should not exceed 10 minutes.** Use the following guidance to prepare the video demonstration:

Section B

- Run the application for all functionalities based on the user types. Use simple sample datasets.
- Demonstrate exception handlings in your application.
- Explain crucial source codes such as authentication, validation, insert records, update records and delete records.
- Demonstrate at least one (1) process for file handling in your application.
- Plan and organise the flow of your presentation to make it easy to follow and engaging.

Section C

- Run the application to demonstrate that the file is read and the original content is displayed on the console.
- Input **11** as the value for shift for decryption.
- Open the decrypted file as evidence that the decryption is successful.

The deliverable for this Section is a Word/PDF file titled **"Panopto_Recording"** that contains only the link to your Panopto video. Please ensure your Panopto video is accessible to the assessor by making the accessibility public. Consult with your tutor for help. **Students are strongly advised to verify video accessibility before submission.** An inaccessible Panopto video by the assessor will result in **zero marks**.

Submission Process

Your assignment should be submitted electronically through the submission point on the module Blackboard site. You should upload a single zipped folder with the following naming convention: **SDCoursework_StudentID.zip,** where "StudentID" is your student number**.** For example, **SDCoursework_232208.zip**. Your .zip folder must contain all the required deliverable files for all the sections. Submissions made after the deadline are subject to the usual University regulations.

Make sure that you upload the correct file by checking once you have submitted (i.e., preview your submission in Blackboard and check it and/or download it again from Blackboard). **Mistakes discovered**

after the deadline will not be corrected; it is your responsibility to ensure that you submit the correct files by the deadline.

If we suspect collusion, collaboration, or plagiarism, you may be asked to provide a walkthrough of your code, during which you will need to discuss all aspects of the work you submitted, with your grade being subject to a successful walkthrough and discussions of your work.

Artificial Intelligence and Academic Integrity – AI&AI

It is important that if you use AI tools to generate an assignment that you do not submit it as if it were your own work. You must check the AI Transparency Scale (AITS) to determine what level of AI use is allowed.

Our regulations state:

Contract cheating/concerns over authorship: This form of misconduct involves another person (or artificial intelligence) creating the assignment which you then submit as your own. Examples of this sort of misconduct include: buying an assignment from an 'essay mill'/professional writer; submitting an assignment which you have downloaded from a file-sharing site; acquiring an essay from another student or family member and submitting it as your own; attempting to pass off work created by artificial intelligence as your own. These activities show a clear intention to deceive the marker and are treated as misconduct.

Further guidance is available here: https://blogs.shu.ac.uk/assessment4students/preparing-to-submit-work/#AI

AI Transparency Scale (AITS)

For this assessment, the permitted use of Artificial Intelligence is highlighted in the table below. All students are required to include a transparency declaration statement, which can be added as an appendix to the assessment (not included in the word count)

SHU AI Transparency Scale (AITS)				
AITS	**Descriptor**	**Transparency Statement**	**AI Contributions**	**Human Contribution**
1	No AI	Artificial Intelligence (AI) has not been used for any part of the activity.	AI is not used for any part of the activity.	All aspects of the activity are human generated, created, edited, and developed.
2	AI for Shaping	AI has been used to shape the initial and/or final parts of the activity.	AI is used for shaping parts of the activity. This includes initial outlining, concept development, prompting thinking, and/or improving structure/quality of the final output.	Most of the activity is human developed/generated. AI ideas and suggestions are refined and reviewed. AI outputs are used for discrete and specific goals/outcomes.
3	AI for Developing	AI has been directed for enhanced development of concepts and outputs.	AI is used to undertake detailed development of many or most aspects of an activity and outputs of that activity.	The human takes a significant role in the enhancement, refinement, and critical review of AI generated

				elements, combining or curating for any outputs.
4	AI for Enhancing	AI has been implemented for all elements of the task.	AI is used extensively throughout the task to achieve goals and outcomes.	The human directs the use of AI for effective outcomes within an activity. Critical thinking is evidenced for any outputs.
5	AI for Innovating	AI has been used for all elements of a task or piece of work, and it has been used in new, creative, and innovative ways through advanced techniques.	AI is implemented in an advanced and innovative way throughout all aspects of the activity.	AI is used creatively and critically by the human. The human uses AI as a co-creator with a critical thinking approach to generating novel AI activities and outputs.

Transparency declaration statements

Your statement should confirm that no AI was used. Please include this statement as a **block comment** right at the top of your program code (for any code file you included in your submission) and as **an appendix** in any Word/PDF file you included in your submission. **Failure to comply with this requirement may be considered a breach of academic integrity under our Academic Conduct Policy.**

Assessment Criteria

The marking of this task is done based on your ability to successfully choose and apply appropriate programming structures to the complete coding task. You will need to fully implement the required functionality to obtain the highest marks; marks will still be awarded for partial solutions, but they will be subject to the level of completion.

Refer to the assessment criteria file.

University Grade Descriptor (UGD)

Level 4 University Grade Descriptor link here. Level 4 Generic Grade Descriptors | Sheffield Hallam University

Appendix A

Datasets (**borrowers.csv, inventories.csv, loaned.csv, staff.csv**) are available on Blackboard in the **"Assessment => Datasets"** folder.

Appendix B

Message.csv – Available on Blackboard in the **"Assessment => Datasets"** folder.