Assignment 1 – Part 2 & Part 3: Software Development & Showcase

Due date: Sunday of Week 13
Weight: 55%
Project: University Application (CLI and GUI)
Collaboration: Groups of 3-4. Group submission, individually assessed.

1- Project Abstract

A local university wants to develop a new interactive system – CLIUniApp - that offers access to two interactive subsystems for students and admins separately. CLIUniApp stores students' data into a local file `students.data`, with all CRUD operations interacting with this file.

Students accessing the student subsystem have the choice to login (if they are previously registered) or register. A registered student can enrol in a maximum of four (4) subjects, remove a subject, change password, and view enrolment.

Admins already exist in the system and do not need to register. An admin using the admin subsystem can remove a student from the `students.data`, partition students to PASS/FAIL categories, group students by grade, view all students, and clear all student data.

Develop the software using your preferred IDE in Java or Python with reference to the requirement analysis, modelling and design in Assessment 1 - Part 1, and submit it as a single .zip file per group. Requirements are further detailed in Section 2 - System Menu Requirements and Section 3 – Sample Model Classes. Additionally, it is important to check Section 4 - Sample I/O to understand the system menu options and potential outputs.

This project also includes a GUI challenge task that requires a standalone GUI application called GUIUniApp. Refer to Section 5 – Case Study GUI Implementation for more information.

2- System Menu Requirements

The software features four system menus: one for the university, one for students, one for admin, and one for subject enrollment.

The University System

The university menu allows users to navigate between the Admin subsystem and the Student subsystem with the following options:

(A) Admin
(S) Student
(X) Exit

The Student System

The student menu allows students to login or register with the following options:

(l) login
(r) register
(x) exit

- **Register**
  New students should be able to register. Their email and password should be validated against the defined patterns upon registration. The system then checks if the student already exists; only new students who do not exist in the database will be registered. Upon successful registration, the student's data should be stored in `students.data`.

- **Login:**
  A registered student should be able to login. Upon logging in, the system checks if the student exists and if so, reads the students' data from `students.data`. After logging in, a student is taken to the Subject Enrolment System.

The Subject Enrolment System

Logged-in students can access this system to perform the following actions:
(c) change: Change the password.
(e) enrol: Enrol in a subject. A student can enrol in a maximum of four (4) subjects.
(r) remove: Remove a subject from the enrolment list.
(s) show: Shows the enrolled subjects with their marks and grades.
(x) exit

The Admin System

Admins already exist in the system and can simply use the admin subsystem without logging in. The admin system supports the following actions:
(c) clear database: Clear all data on `students.data`.
(g) group students: Groups students by grade.
(p) partition students: Partition students to PASS/FAIL categories.
(r) remove student: Remove a student by ID.
(s) show: Show all students.
(x) exit

3- Sample Model Classes

The software should contain the following classes at a minimum: `Student`, `Subject`, and `Database`. You may add more classes based on your team's Part 1 design.

`Student`

The `Student` class has the following fields:

- id: a unique, randomly generated number formatted as a 6-digit value, ranging from 000001 to 999999.
- Other fields are name, email, password, and <List>subjects

Other requirements are:

- A student can only enrol in a maximum of four (4) subjects.
- A student can enrol/drop a subject at any time.

- Upon enrolling in a subject, a random mark between 25 and 100 is generated for this subject.
- Upon enrolling in a subject, the grade of that subject is calculated based on the mark.
- A student passes if the average mark of all four (4) subjects is greater or equal to 50.
- A student can change their password at any time.

`Subject`

The Subject class has the fields:

- id: a unique, randomly generated 3-digit number, ranging from 001 to 999.
- mark: randomly generated between 25 and 100.
- grade: determined based on the mark.

`Database`

The Database class should:

- check if `students.data` exists before using it and create the file if it doesn't
- read and write objects from and to `students.data`
- clear all objects from `students.data`

NOTE: All the above operations should interact with `students.data`.

Best Practices and Recommendations (Optional)

The software is best developed using Controller classes (`StudentController`, `SubjectController`, etc) to manage the data exchange between the Model classes (`Student`, `Subject`, `Database`) and the menu actions. The Controller classes contain system menus and use the Model classes to perform corresponding operations.

4- Sample I/O

The sample I/O is intended to help you understand the different application scenarios, demonstrating how the program should work and how the output should be formatted.

The University System

```
University System: (A)dmin, (S)tudent, or X : A
        Admin System (c/g/p/r/s/x): x
University System: (A)dmin, (S)tudent, or X : S
        Student System (l/r/x): x
University System: (A)dmin, (S)tudent, or X : X
Thank You
```

The Student System – Register/Login

Upon logging in or regiatration, a student's credentials are validated against regular expressions (regex) defined as constants. Emails should end with the domain "@university.com", hence firstname.lastname@university.com is a valid email, while firstname.lastname@university is not.

A password should meet the following criteria: (i) It starts with an upper-case character, (ii) It contains at least five (5) letters, (iii) It is followed by three (3) or more digits.

Registered students are stored in `students.data`. When a student logs in, the system should read the data from the file and verify the credentials (as per Sample I/O).

```
University System: (A)dmin, (S)tudent, or X : S
        Student System (l/r/x): r
        Student Sign Up
        Email: johnsmith@university.com
        Password: helloworld123
        Incorrect email or password format
        Email: john.smith@university.com
        Password: Hello123
        Incorrect email or password format
        Email: john.smith@university.com
        Password: Helloworld123
        email and password formats acceptable
        Student John Smith already exists
        Student System (l/r/x): r
        Student Sign Up
        Email: alen.jones@university.com
        Password: Helloworld123
        email and password formats acceptable
        Name: Alen Jones
        Enrolling Student Alen Jones
        Student System (l/r/x):
```

```
Student System (l/r/x): l
Student Sign In
Email: alen.jones@university.com
Password: helloworld123
Incorrect email or password format
Email: alen.jones@university.com
Password: Helloworld222
email and password formats acceptable
Student does not exist
Student System (l/r/x): l
Student Sign In
Email: alen.jones@university.com
Password: Helloworld123
email and password formats acceptable
        Student Course Menu (c/e/r/s/x):
```

The Subject Enrolment System – Enrolment

The system should keep track of the enrolled subjects and store the enrolment data within the `Student` objects on `students.data`. Every time a student enrols in a new subject, the overall (average) mark should be re-calculated.

```
Student Course Menu (c/e/r/s/x): s
Showing 0 subjects
Student Course Menu (c/e/r/s/x): e
Enrolling in Subject-541
You are now enrolled in 1 out of 4 subjects
Student Course Menu (c/e/r/s/x): e
Enrolling in Subject-455
You are now enrolled in 2 out of 4 subjects
Student Course Menu (c/e/r/s/x): s
Showing 2 subjects
[ Subject::541 -- mark = 55 -- grade =   P ]
[ Subject::455 -- mark = 57 -- grade =   P ]
Student Course Menu (c/e/r/s/x): e
Enrolling in Subject-742
You are now enrolled in 3 out of 4 subjects
Student Course Menu (c/e/r/s/x): s
Showing 3 subjects
[ Subject::541 -- mark = 55 -- grade =   P ]
[ Subject::455 -- mark = 57 -- grade =   P ]
[ Subject::742 -- mark = 55 -- grade =   P ]
Student Course Menu (c/e/r/s/x): e
Enrolling in Subject-97
You are now enrolled in 4 out of 4 subjects
Student Course Menu (c/e/r/s/x): s
Showing 4 subjects
[ Subject::541 -- mark = 55 -- grade =   P ]
[ Subject::455 -- mark = 57 -- grade =   P ]
[ Subject::742 -- mark = 55 -- grade =   P ]
[ Subject::097 -- mark = 85 -- grade =  HD ]
Student Course Menu (c/e/r/s/x): e
Students are allowed to enrol in 4 subjects only
```

The Subject Enrolment System – Remove a Subject:

```
Student Course Menu (c/e/r/s/x): r
Remove Subject by ID: 541
Droping Subject-541
You are now enrolled in 3 out of 4 subjects
Student Course Menu (c/e/r/s/x): s
Showing 3 subjects
[ Subject::455 -- mark = 57 -- grade =   P ]
[ Subject::742 -- mark = 55 -- grade =   P ]
[ Subject::097 -- mark = 85 -- grade =  HD ]
Student Course Menu (c/e/r/s/x): e
Enrolling in Subject-764
You are now enrolled in 4 out of 4 subjects
Student Course Menu (c/e/r/s/x): e
Students are allowed to enrol in 4 subjects only
```

The Subject Enrolment System – Change Password

You may reuse the same password.

```
Student Course Menu (c/e/r/s/x): c
Updating Password
New Password: Helloworld123
Confirm Password: Helloworld123
Student Course Menu (c/e/r/s/x): c
Updating Password
New Password: Newworld123
Confirm Password: newworld123
Password does not match - try again
Confirm Password: Newworld123
Student Course Menu (c/e/r/s/x): x
```

The Admin System – Listing, Grouping, and Partitioning Students

```
University System: (A)dmin, (S)tudent, or X : A
        Admin System (c/g/p/r/s/x): s
        Student List
        John Smith :: 673358 --> Email: john.smith@university.com
        Alen Jones :: 762740 --> Email: alen.jones@university.com
        Admin System (c/g/p/r/s/x): g
        Grade Grouping
        P  --> [Alen Jones :: 762740 --> GRADE:  P - MARK: 63.50]
        C  --> [John Smith :: 673358 --> GRADE:  C - MARK: 68.25]
        Admin System (c/g/p/r/s/x): p
        PASS/FAIL Partition
        FAIL --> []
        PASS --> [John Smith :: 673358 --> GRADE:  C - MARK: 68.25, Alen Jones :: 762740 --> GRADE:  P - MARK: 63.50]
```

The Admin System – Remove a Student and Removing All Students

The system should read the `Student` objects from `students.data` when listing, grouping, partitioning students, and remove the `Student` object(s) from `students.data` when removing a student or all students.

```
        Admin System (c/g/p/r/s/x): r
        Remove by ID: 762740
        Removing Student 762740 Account
        Admin System (c/g/p/r/s/x): r
        Remove by ID: 777888
        Student 777888 does not exist
        Admin System (c/g/p/r/s/x): x
University System: (A)dmin, (S)tudent, or X : S
        Student System (l/r/x): l
        Student Sign In
        Email: alen.jones@university.com
        Password: Newworld123
        email and password formats acceptable
        Student does not exist
        Student System (l/r/x): x
University System: (A)dmin, (S)tudent, or X : A
        Admin System (c/g/p/r/s/x): c
        Clearing students database
        Are you sure you want to clear the database (Y)ES/(N)O: N
        Admin System (c/g/p/r/s/x): s
        Student List
        John Smith :: 673358 --> Email: john.smith@university.com
        Admin System (c/g/p/r/s/x): c
        Clearing students database
        Are you sure you want to clear the database (Y)ES/(N)O: Y
        Students data cleared
        Admin System (c/g/p/r/s/x): s
        Student List
                < Nothing to Display >
        Admin System (c/g/p/r/s/x): g
        Grade Grouping
                < Nothing to Display >
        Admin System (c/g/p/r/s/x): p
        PASS/FAIL Partition
        FAIL --> []
        PASS --> []
        Admin System (c/g/p/r/s/x):
```

```
University System: (A)dmin, (S)tudent, or X : S
        Student System (l/r/x): l
        Student Sign In
        Email: john.smith@university.com
        Password: Helloworld123
        email and password formats acceptable
        Student does not exist
        Student System (l/r/x): x
University System: (A)dmin, (S)tudent, or X : X
Thank You
```

5- Case Study GUI Implementation

GUIUniApp is a challenge task of Assessment 1 - Part 2 and is designed only for registered students. The “login window” is the main window of GUIUniApp. A registered student should be able to log

into GUIUniApp and access the "enrolment window" to enrol in a maximum of four (4) subjects. Each time a student is enrolled in a subject, the subject is added to their enrolment list. GUIUniApp should handle possible exceptions for empty login fields and for attempts to enrol in more than four (4) subjects. More specifically, GUIUniApp supports the following:

- Access management: Login for students (No admin options).
- Enrolment management: A logged-in student can enrol in a maximum of four (4) subjects. New subjects should be added to the student's enrolment list.
- Exception handling. Exceptions such as empty login fields, incorrect student credentials, incorrect email format should be handled.
- Data source management: Use `Student` objects stored in `students.data` as registered students to log into the GUIUniApp.

The GUIUniApp should have at least 4 windows as indicated in the marking scheme.

6- Marking Scheme

Total Assessment 1 - Part 2 mark is 50/70. The program must function correctly to receive marks for functionality, accuracy, design, and output that aligns with the sample I/O.

Your software (should it work) will be marked according to the following marking schema:

a. The University System (4 Marks)

Entity	Criteria	Mark
Student	Can go to student menu	1
Admin	Can go to the admin menu	1
Browsing	Student/Admin can browse between menus	1
Matching I/O	I/O wording and indentation match the sample output	1

b. The Student System (9 Marks)

Entity	Criteria	Mark
Register	Student can register – data saved to file	2
Login	Student can login – data read from file	2
RegEx	For login and register	2
Error Handling	Exceptions, errors, logical scenarios are handled	2
Matching I/O	I/O wording and indentation match the sample output	1

c. The Subject Enrolment System (15 Marks)

Entity	Criteria	Mark
Enroll	Student can enroll in a maximum of 4 subjects	2
Tracking	Subject enrolment is tracked	2

Remove a subject	Student can remove an enrolled subject by its ID	2
Show subjects	Student can list all enrolled subjects	1
Change password	Student can change the login password	2
Read/Write to file	Student and subject data are read/written from/to a file	3
Error Handling	Exceptions, errors, logical scenarios are handled	2
Matching I/O	I/O wording and indentation match the sample output	1

d. The Admin System (15 Marks)

Entity	Criteria	Mark
Show students	Admin list all students	1
Group students	Admin can group students according to the grade	2
Partition students	Admin can partition students into Pass/Fail categories	2
Remove a student	Admin can remove a student by the student ID	2
Clear file	Admin can remove all students and clear the file	2
Read/Write to file	Student and subject data are read/written from/to file	3
Error Handling	Exceptions, errors, logical scenarios are handled	2
Matching I/O	I/O wording and indentation match the sample output	1

e. GUIUniApp (7 Marks)

Entity	Criteria	Mark
Login window	Main window. Upon logging in, students are taken to the enrolment window. The registered students are read from `students.data`.	2
Enrolment window	Enrollment window allows students to enroll in a maximum of four (4) subjects.	2
Subject window	Enrolled subjects with their marks and grades are listed in subject window.	2
Exception window	Notifies users of format errors and when a student tries to enroll in more than four (4) subjects.	1

7- Deliverables and Contribution

You will be **marked individually on Assessment 1 - Part 2 and Part 3**. This assessment requires collaboration and **equal contribution from all group members**. One possible way to divide the work in a 4-member group is by assigning tasks based on the Marking Scheme above: Tasks a, b -> Student 1, Task c -> Student 2, Task d -> Student 3, Task e -> Student 4 – despite handling only the View which is worth 7 Marks, Student 4 needs to understand how the Model and Controller work and integrate them with the View, which justifies the workload. Alternatively, the work can be divided into Model, View, and Controller components, or in another preferred manner. Please specify how you divide the work among group members and the individual contributions in your Part 2 submission. Students who fail to demonstrate adequate contribution will receive penalties on a case-by-case basis. All individual contributions should be integrated into a **single, cohesive piece of software**. Group members must collectively agree on one programming language, and the final

source code should be submitted as a **single group submission**. If any part cannot be integrated into the final software, all group members will incur a **10% penalty**.

During the Part 3 Showcase, each group member must **attend in person and present their own part** of the project. Any student who fails to appear and present in this compulsory Showcase will receive a final individual mark of zero (0) for Assessment 1 – Part 2 & Part 3.

Do NOT use AI-generated code. If we believe your code is AI-generated, you may receive zero (0) for your assignment.

The submitted CLI/GUI software must be fully functional to receive marks. If your team chooses to do the GUI task, both CLI and GUI applications can be included in the same project folder and submitted as a single .zip file. You can include a README file to specify any additional libraries, specific programming language versions, etc, required for running your code.

8- Assessment Submission

Submit Assessment 1 – Part 2 code as a **single .zip file per group**.

File naming convention: **group<group-number>-Cmp1<lab-number>.zip**

Upload your assessment file to Canvas under Task 1: Case Study/Part 2: Software Development by the due date and time specified on Canvas.

9- Assessment Showcase

Total Assessment 1 - Part 3 mark is 5/70.

The Showcase will be conducted during your usual lab hours in Week 13. All team members must attend the showcase. Any student who fails to appear and present in this compulsory Showcase will receive a final individual mark of zero (0) for Assessment 1 – Part 2 & Part 3.

Your showcase will be marked according to the following marking schema:

f. Showcase (5 Marks)

Criteria	Mark
Demonstrate the working of the software with all group components	2
Speech is clear and easy to follow	1
Includes a walkthrough of the source code to demonstrate individual contributions	1
Overall correctness of answers to the questions	1

10- Special Consideration

Special consideration for deadline extensions must be arranged beforehand with the subject co-ordinator. Please refer to the UTS Special Consideration Procedure (https://www.uts.edu.au/current-students/managing-your-course/classes-and-assessment/special-circumstances/special-consideration).

11- Late Penalty

Please refer to the UTS Late Submission Penalties page (https://www.uts.edu.au/current-students/current-students-information-faculty-law/assessment/late-submission-penalties).

12- Assessment Misconduct

Please see the Subject Information for plagiarism and academic integrity in conjunction with UTS policy and procedures for the assessment for coursework subjects.