Assessment 2 – Part 2: Software Development

Due date: Monday May 13, 2024, by 09:00 AM
Weight: 50 out of total assessment 1 weight 70
Project: University Application (CLI system and GUI system)
Collaboration: Group of three. Group and individually assessed.

1- Project Abstract

A local university wants to develop a new interactive system. The system should offer access to two interactive sub-system menus for students and admins. The university interactive system is called "CLIUniApp". CLIUniApp stores students' data into a local file "students.data". All CLIUniApp CRUD operations should be operated with the storage file "students.data".

The university main system should offer access to students and admins. Students accessing the sub-system have the choice to login to the system (if they are previously registered) or register to the system. A registered student can enrol in maximum 4 subjects. A registered student can remove subject. A registered student can change their access password. A registered student can view their enrolment.

Admins already exist in the system and do not need to register. Admins using the sub-system can remove a student from the "students.data" file. Admins can view students partitioned as Pass/Fail. Admins can group and show students organized by grade Admins can view all students from file. Admins can clear all file data.

Develop the case study project using your preferred IDE with Java or Python and submit the software application as a .zip file by the due date (refer to submission section). The application should be development in a group of 3 students. Develop the application based on the requirement analysis, modelling and design completed by your team in Part 1 of the case study. Part 2 requirements are further explained in the "System Menu Requirements" section and in the "Model Classes" section. Feel free to add more classes following your class diagram of Part 1. It is important to refer to the "Sample I/O" section to understand the system menu options and possible output.

The case study has a GUI challenge task. The GUI challenge task is a standalone GUI application called GUIUniApp. Refer to section 5 for more information.

2- System Menu Requirements

The program system is composed of four system menus: The University System, The Student System, The Student Course System, and the Admin System

The University System

The university menu system should enable users to choose to go the Admin menu or Student Menu

(A) Admin
(S) Student
(X) exit

The Student System

The student menu system should enable students to Login and Register as follows:

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

- **Register**
  Students should be able to register. Email and password should be verified against pattern rules. Then the student should be checked if they exist. Only register students that do not exist in the Database file. On registration, student data should be stored in "students.data".

- **Login:**
  Students should be able to login. Then the student should be checked if they exist. Only a registered student should login. Upon login, students' data should be read from "students.data". After login, a student goes to Student Course Menu that offers the choices:

The Student Course System

Logged in students can access this menu to perform the following actions:
(c) change: enables a student to change their password
(e) enrol: Student enrols in a subject. A student can enrol in maximum 4 subjects.
(r) remove: Student can remove a subject from the subjects' enrolment list
(s) show: Shows the enrolled subjects and the marks and grades for each subject
(x) exit

The Admin System

Admin menu offers the following actions:
(c) clear database file: enables admin to clear the data file "students.data"
(g) group students: shows the students organized with respect to the grade
(p) partition students: shows the pass/fail distribution
(r) remove student: enables admin to remove a student by ID
(s) show: show the students from the data file
(x) exit

NOTE: Admins already exist in the system. They do not need to login into the system. Admins can simply use the admin sub-system.

3- Sample Model Classes

The program model has at least the following classes: Student, Subject, and Database. These classes are responsible for storing the program data and for supplying the program controllers with functionalities and data. You may add more classes based on Part 1 design.

The Student Class

- The Student class has the fields:
    - ID randomly generated 1 <= ID <= 999999, unique and formatted as 6-digits width. IDs less than 6-digits width should be completed with zeroes from the left.
    - name, email, password, and a list of subjects
- A student can only enrol in 4 subject maximum (A course of 4 subjects).
- A student can enrol/drop a subject at any time.
- Upon enrolment in a subject a random mark is generated for this subject 25<= mark <= 100
- Upon enrolment in a subject, the grade of that subject is calculated based on the mark
- A student pass/fail a course if the average mark of the subjects is greater or equal to 50
- A student can change their password at any time.

The Subject Class

- The Subject class has the fields:
    - ID randomly generated 1 <= ID <= 999, unique and formatted as 3-digits width IDs less than 3-digits width should be completed with zeroes from the left.
    - mark is randomly generated where 25<= mark <= 100
    - grade is determined based on the mark

The Database Class

The Database class should contain the functionalities to:

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

NOTE: All program menu(s) actions (Student and Admin) should use the "students.data" data and perform CRUD operations with this data.

BEST PRACTICES AND RECOMMENDATIONS

The program is best developed using classes (controllers) to manage the data exchange between the model classes (Student, Subject, Database) and the menu(s) actions. The controllers (for example: StudentController, SubjectController, etc…) are normal classes that use the "students.data" data and perform CRUD operations with this data. The controllers contain the system menus that use the model objects and work with the data file "students.data".

NOTE: You may add any controller classes based on Part 1 case study design. Groups can implement the controllers the way that suits their program.

4- Sample I/O

The sample I/O should be used as a reference and guidance to help you understand the scenarios of the program. The sample I/O outlines several scenarios to help you understand how the program should work and how the output should be designed and formatted. The Sample I/O is composed of the following scenarios:

The University System Scenario:

```
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:

In the login/register scenarios, students' credentials are verified against regular expressions stored as constants in the Utils class.

Email regex should ensure that the dot (.), the (@) and the (**university.com**) are present

Password regex should ensure that entered passwords are:

- Start with upper case
- Minimum 6 letters
- Following by minimum 3-digits

Registered users are saved in the "students.data". When logging in, your program should read the data from the file and verify the student 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):
```

The Student System – Login:

```
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 Student Course System – Enrolment:

```
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
```

In the enrolment scenario, no controls are needed to sort the subjects.

Every time a student enrols in a subject, the overall mark should be re-calculated.

In the enrolment scenario, you must keep track of the enrolled subjects

Subject enrolment data is saved in the file “students.data” within the Student objects

The Student Course System – Remove 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 Student Course System – Change 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
```

In the change password, you may reuse the same password.

The Admin System – View Students – Groups - Partitions:

```
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:

```
        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
```

The admin should read the students data from the file "students.data" and produce the outputs shown in the scenario (listing, grouping, partitioning).

When the admin removes a student, or remove all students, that data should be removed from the file "students.data".

5- Case Study GUI Implementation

The case study GUI software implementation is a challenge task of Part 2. Develop a GUI application called "GUIUniApp" only for students. GUIUniApp should allow students to login into the system. The login window is the GUI main window. Once a student logs in correctly they can enrol into subjects (4 subjects maximum). Every time a student is enrolled in a subject, the subject is added to the subject menu enrolment list. Handle possible exceptions for empty login fields and for enrolment in more than 4 subjects.

Your GUIUniApp should have at least 4 windows (as indicated in the marking scheme). The GUIUniApp requirements are the following:

- Access management: Login for students (No admin options)
- Enrolment management: Logged in student can enrol in 4 subjects (maximum). New subjects should be added to the subject GUI menu list.
- Exception handling. If there are any exceptions, for example: empty login fields, incorrect student credentials, incorrect email format.
- Data source management: You should work with registered students details already saved in the file 'students.data' and use the registered students (in the CLIUniApp) to login into the GUIUniApp.

6- Marking Scheme

Total assessment Part 2 mark is 50/70. The program must work so marks can be awarded for functionality, correctness, design, output matching with the sample I/O.

Your program solution (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, coloring, indentation matches the sample	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, coloring, indentation matches the sample	1

c. The Student Course system (15 Marks)

Entity	Criteria	Mark
Enroll	Student can enroll in a subject – 4 maximum	2
Tracking	Subject enrolment is tracked	2
Remove a subject	Remove by ID	2
Show subjects	Subject listing	1
Change password	Student can change their password	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, coloring, indentation matches the sample	1

d. The Admin System (15 Marks)

Entity	Criteria	Mark
Show students	Admin list all students	1
Group students	Admin groups students according to the grade	2
Partition students	Admin partitions students as Pass/Fail	2
Remove a student	Admin can remove a student by 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, coloring, indentation matches the sample	1

e. GUIUniApp (7 Marks)

Entity	Criteria	Mark
Login window	Login window works and on login, students are taken to the enrollment window. Use the registered students from students.data	2
Enrollment window	Enrollment window allows students to add subjects (maximum 4)	2
Subjects window	Enrolled subjects are added to the list in the subjects' window	2
Exception window	Handle incorrect format exception and max 4 subjects' exception	1

7- Deliverables and Contribution

The assessment requires collaboration and equal amount of contribution between all group members. The individual student contributions or parts will be collated in a group deliverable for submission and assessment (group submission but individual assessment). The deliverables of this assessment task also include a compulsory oral/visual presentation (no PowerPoint slides) of the individually implemented working software application during the scheduled assignment assessment or review session (showcase).

The submitted "case study software" CLI / GUI must fully work before marks can be awarded. CLI (and GUI) applications can be in the same project folder and submitted in the same ZIP file (if your team chose to complete the GUI challenge task.

8- Assessment Submission

Submit your assessment project code (**single submission only/group**) as a ZIP file.

Assessment file name: Assessment file name: (**group<group-number>-Cmp<lab-number>.zip**) to Canvas/Assignments/Task1/Part2. Submit your assessment ZIP project by the due date: Monday 13/05/2024 by 9:00 AM

9- Assessment Showcase

Assessment 1 working software project demonstration and showcase will be conducted during usual class hours in week 12. All team members must attend the showcase. Students who failed to appear and present in this compulsory assignment assessment and review sessions (Showcase) will receive zero (0) as a final individual mark for assessment 1 – Part 2.

10- Special Consideration

Special consideration, for late submission, must be arranged beforehand with the subject co-ordinator (email: yining.hu@uts.edu.au). Please also see the UTS Special Consideration Process: www.sau.uts.edu.au/assessment/consideration

11- Late Penalty

See subject outline for late submission penalty unless an extension has been approved by the subject coordinator.

12- Assessment Misconduct

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