Assessment 1 – Part 1: Requirements Analysis

Due date: Friday September 22, 2023, by 11:59 PM
Weight: 15 out of total assessment 1 weight 70
Project: Case-study requirements analysis and software design
Collaboration: Group of three. Group and individually assessed.

1- Project Brief

A local university wants to develop a new interactive system. The university wants to allow students to self-enrol into a semester subjects. The students would need to register in the new application before they can access the system and enrol in subjects. A student can enrol in subjects between 1 and 4. The enrolment is only for one semester at the time. Hence, choice of multiple semester enrolment is outside the application scope.

Students must register first (using valid email and password). A student must enter their name, email, and password into the registration form. On sign up, a unique student ID will be auto generated for each student. The student unique ID is between 1 and 999999. If the size of the generated ID is not 6-digits, then the ID should be pre-fixed with zeroes to complete 6-digits size (e.g., 002340 is a valid ID. 2345 is not a valid ID). Registered students' data should be saved into a file data store "students.data".

Students' emails should have the extension "**@university.com**". The students' emails and the password should be validated against existing patterns, for example:
Student email: **firstname.lastname@university.com** is a valid email.
Student email: **firstname.lastname@university** is not a valid email.
Student password is considered valid if it matches the following pattern:

- Starts with upper-case character.
- Minimum 5 letters
- Followed by 3 of more digits.

Registered students can then login into the application and perform the following operations: (i) enrol into a subject, (ii) remove a subject from enrolment list, (iii) show current enrolment list, (iv) change their password. Students enrolling into subject do not need to select a subject to enrol into (to simplify the application). Once a student selects the enrol command a new subject will be added to their enrolment list. The enrolment system will keep track of the subjects in the student's list and will notify the student if the subject count exceeds 4.

Subjects should be available for students on enrolment. When a student selects the enrolment command/action, a new subject will be added to their enrolment list (given the list has less than 4 subjects). A subject is identified by a unique 3-digits auto-generated ID (1 <= ID <= 999).

On enrolment, a random subject mark (between 25 and 100) will be autogenerated and allocated for the subject. Then the subject grade will be auto calculated based on the mark. Refer to UTS grading system (mark < 50 → Z; 50 <= mark < 65 → P; 65 <= mark < 75 → C; 75 <= mark < 85 → D; mark >= 85 → HD).

Registered students and their enrolment data should be saved into a file data store "students.data". The data store file "students.data" should also be available to Admins to perform students' management operations with the students' data.

Admins are existing university staff (do not require registration). Admins have their own sub-system within the new application to perform student management operations. Admins can view all registered students. Admins can organize and view students by grade. Admins can partition and view students based on PASS/FAIL categories (using the students grades and marks). Admins can remove a student or clear the entire students.data file store.

The university is requesting a CLI application students and admins actions. The university is also requesting a GUI component (at smaller scale).

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

The case study GUI software implementation is a challenge task of Part 2. The GUI application is called "GUIUniApp". The GUI application is a prototype designed only for students to simplify the implementation. 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.

In the GUI application, assume that the students are already registered (create and add few student accounts to the application for testing). In GUIUniApp, the rules for student enrolment into a subject are the same rules as CLIUniApp. There is no need to store students' data into a file when using GUIUniApp.

You team is expected to develop the application in two parts, Part 1 and Part2, then demonstrate the result to the stakeholders in Part 3.

In Part 1, your team is expected to complete and deliver a comprehensive software requirements analysis report which include: (i) Transform the requirements into user-stories and map the user-stories to a requirements table (or backlog); (ii) documentation testing (black box testing) of the university requirements; (iii) Create a UML use-case diagram and explain in details the goals, actors, cases their relationships in the diagram; (iv) Create a UML class-diagram and explain in details the classes, their properties and their relationships.

In Part 2, your team is expected to develop and implement the university application following Part 1 design. The university application is composed of CLIUniApp and GUIUniApp (challenge task). The application should be submitted by the due date and demonstrated in Part 3.

Part 3 is the assessment formal showcase. Each team will present their Part 2 working application based on their collaborative Part 1 design. Team members must equally participate and collaborate in all assessment parts.

2- User-Story Table (Backlog)

Your team is expected to read thoroughly the customer (university) requirements and transform the requirements into user-story. The user-story should be simple so that each story is later translated into a function (or action). Each story will have a unique 3-digits ID. If a group of stories related to the same features, then the hundreds (number) will match for all those stories. For example:

Consider the Login feature. All the following stories are related to the same Login feature. Hence their ID should start with the same hundreds number.
Story: match username and password with the ones on file → *101*
Story: verify username and password against patterns → *105*
Story: show error message if credentials do not match → *106*
Story: take student to student sub-menu if credentials are correct → *100*

The refined user-stories should be mapped into a requirements table (or backlog). The table is formatted as follows:

ID	User	Action	Result	Function
A unique 3 digits user story ID.	The person or entity taking the action	The action taken by the user	The result or outcome of the action	The action name

3- Documentation Testing

Your team is expected to read thoroughly the customer (university) requirements and conduct documentation testing (Black-Box Testing). The documentation testing aims to determining whether the stated requirements are unclear, incomplete, inconsistent, ambiguous, or contradictory.

Refer to lecture 1 documentation testing criteria and develop a test table "Black-Box Test table". The test results should be recorded in the table and discussed with the stakeholders. Your test table should be thorough and composed of test-cases. For example:

Black-Box Test

Test Case ID	Criteria	Score (0/5)	Recommendation
i.e: T-001, T101, …	The criteria that apply to the case	Satisfactory score between 0 and 5	Improvement actions

4- UML Use-Case Diagram

Your team is expected to develop a comprehensive UML use-case diagram. To successfully develop the diagram, identify the actors, the goals, the case, and their relationships. Provide explanation for each actor, goal, case, relationship. Ensure that your diagram is consistent and align with the provided explanations about all involved entities.

5- UML Class Diagram

Your team is expected to develop a comprehensive UML class diagram. To successfully develop the diagram, identify the classes, fields, methods, visibility, multiplicity, and their relationships. Provide explanation for each actor, goal, case, relationship. Ensure that your diagram is consistent and align with the provided explanations about all involved entities.

6- Marking Scheme

Total assessment Part 1 mark is 10/70. All team members are expected to equally contribute to the development of the project (all parts).

a. User-Story Table (3 Marks)

Entity	Criteria	Mark
User stories are specific	User stories are decomposed into simple story = action	1
User stories consistency	User stories align with the project requirements	1
Backlog correctness	User stories are correctly mapped into the backlog	1

b. Documentation Testing (2 Marks)

Entity	Criteria	Mark
Test cases correctness	Test cases are correctly mapped into the table	1
Test cases validity	Test cases score is accurate. Test cases are valid defects	1

c. Use-case Diagram (5 Marks)

Entity	Criteria	Mark
Entities identification	Goals, cases, actors, relationships correctly identified	1
Entities description	Entities are correctly explained and reported	1
Actors action	Actors initiate accurate cases	1
Case relationships	Accurate and consistent cases relationship	1
Multiplicity	Cases multiplicity is accurate.	1

d. Class Diagram (5 Marks)

Entity	Criteria	Mark
Class	Class properly identified and explained	1
Fields	Properly identified. Accurate visibility choice	1
Methods	Correct method naming, type, visibility	1
Relationships	Consistent class relationships	1
Multiplicity	Accurate relationship multiplicities	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 or 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 assessment 1 part 1 (**single submission only**) as a PDF file (**group<number>.pdf**) to Canvas/Assignments/Task1/Part1.

Submit your assessment PDF file by the due date: Friday 22/09/2023 by 11:59 PM

9- Special Consideration

Special consideration, for late submission, must be arranged beforehand with the subject co-ordinator (email: georges.boughantous-1@uts.edu.au). Please also see the UTS Special Consideration Process: www.sau.uts.edu.au/assessment/consideration

10- Late Penalty

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

11- 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.