Assessment 4 (Team-based)

Assignment 4: Team-based

Assignment name: Assignment 4 (Team-based)

Course code: ISYS3413/ISYS3475/ISYS1118

Weighting: 25%

Due date: **Week 12 (22 February , 11:59 pm)**

1. Course Learning Outcomes Assessed

This assessment supports the following learning outcomes:

- CLO 1: explain how iterative software engineering processes can facilitate software development
- CLO 2: evaluate requirements for a software system
- CLO 5: communicate effectively with others, especially regarding the progress of the system development and the content of the design by means of reports and presentations. Use appropriate design, version control and collaboration tools to work effectively as a team.
- CLO 6: recognise and describe current trends in the area of software engineering.
- CLO 7: Design and carry out tests using various testing techniques and tools.

2. Overview of Assessment

This assignment will assess your ability in terms of writing quality unit tests, test cases, user stories, and acceptance tests, as well as working with GitHub and CI/CD pipelines using GitHub Actions. You will be asked to implement and test some small functions in the **DigtialID** platform. Then you need to move your source code to GitHub. Also, you need to show that the GitHub Actions workflow is triggered to automatically run the Unit tests once code is pushed to GitHub. Furthermore, following the templates and structures

taught in the lectures, you must write **eight user stories** for the **DigtialID** platform, along with their acceptance criteria. Finally, you need to make a video of your Java program.

3. Assessment Activities

General Description

Based on the requirements of the **DigtialID** platform, you need to do the following activities for the **DigtialID** platform.

Activities in Assessment 4

Activity 1: Git/GitHub/GitHub Actions and Unit Test + a Video of your program (17 points): In the **DigtialID** platform, there is a class called **Person**. The **Person** class has three functions called **"addPerson", "updatePersonalDetails"** and **"addID"**. You are asked to implement and test these three functions. The **Person** class has the following methods and attributes.

Important Note: The code written below is just a placeholder. You are allowed to edit it and add new functions (e.g., add getter and setter, new constructors, etc.), new attributes, new classes, etc. You are completely FREE on how you want to implement and test these functions. The key point is to write enough test cases to test the conditions discussed below. You are free to consider any other conditions.

1. addPerson function. This method stores information about a person in a TXT file. However, the following conditions should be considered when adding persons. If the Person's information meets the below conditions and any other conditions you may want to consider, the information should be inserted into a TXT file, and the addPerson function should return **true**. Otherwise, the information should not be inserted into the TXT file, and the addPerson function should return **false**.

Condition 1: **personID** should be exactly 10 characters long; the first two characters should be numbers between 2 and 9, there should be at least two special characters between characters 3 and 8, and the last two characters should be uppercase letters (A-Z). Example: "56s_d%&fAB"
Condition 2: The address of the Person should follow the following format: Street Number|Street|City|State|Country. The State should be only Victoria. Example: 32|Highland Street|Melbourne|Victoria|Australia.
Condition 3: The format of the birthdate of the person should follow the following format: DD-MM-YYYY. Example: 15-11-1990.

2. updatePersonalDetails function. This method allows updating a given person's ID, firstName, lastName, address and birthday in the TXT file. Changing personal details will not affect their demerit points or the suspension status. All relevant conditions discussed for the addPerson function also need to be considered and checked in the **updatePersonalDetails** function. If the Person's updated information meets the below conditions and any other conditions you may want to consider, the Person's information should be updated in the TXT file with the updated information, and the updatePersonalDetails function should return **true**. Otherwise, the Person's information should not be updated in the TXT file, and the updatePersonalDetails function should return **false**.

Condition 1: If a person is under 18, their address cannot be changed.
Condition 2: If a person's birthday is going to be changed, then no other personal detail (i.e, person's ID, firstName, lastName, address) can be changed.
Condition 3: If the first character/digit of a person's ID is an even number, then their ID cannot be changed.

3. addID function. This method stores information about a persons ID in a TXT file. However, the following conditions should be considered when adding ID. If the ID information meets the below conditions and any other conditions you may want to consider (see stretch goals), the information should be inserted into a TXT file, and the addID function should return **true**. Otherwise, the information should not be inserted into the TXT file, and the addID function should return **false**. To extend this method (stretch goals) you can create separate classes to represent the data.

Condition 1: **passport** should be exactly 8 characters long; the first two characters should be should be uppercase letters (A-Z), and the other characters should be numbers 0-9. You could create a Passport class with Name, Date of Birth, Country, Data of Issue, Date of Expiry, and Authority fields.
Condition 2: **drivers licence** should be exactly 10 characters long; the first two characters should be should be uppercase letters (A-Z), and the other characters should be numbers 0-9. You could create a Drivers Licence class with Date of Birth, Version, Name, and VehicleType fields.
Condition 3: **medicare card** should be exactly 9 characters long. All characters should be numbers 0-9. You could create a Medicare Card class including valid to field.
Condition 4: **student card** if a person is under 18 a student card can instead be added (assuming they have no passport, drivers licence, medicare card) which should be exactly 12 characters long. All characters should be numbers 0-9.

In this activity (#1), each team is requested to do the following activities:

Activity 1.1: Create a Maven-based Java project. You are free to use this **pom.xml** file as a guide when creating the project.

Activity 1.1.1 (Unit Tests and Test Cases for three functions): Develop and write **15 test cases**. Each function should have 5 test cases (**use this template**). Each test case should have at least one test data. You need to implement the unit tests for these THREE functions using JUnit that test the conditions (**7.5 points** in total: unit tests and test cases for each function have **2.5 points**).

Activity 1.1.2 (The implementation of three functions in Java): Implement these THREE functions in Java that meet the conditions. You can use any Java IDEs (**6 points** in total: each function has **2 points**).

Activity 1.2 (Presentation, GitHub, GitHub Actions, Gource): Create a **GitHub** repository and move the implemented functions, their unit tests, and POM.XML to the GitHub repository. Then, add a **GitHub Actions** workflow to the GitHub repository and automatically run tests using the GitHub Actions workflow. You are free to use this **github-actions-demo.yml** as a guide. Also, record your screen while **(1)** running and testing your program on your local machine with the changes made to a TXT file/TXT files (1 minute), **(2)** running and testing your program using GitHub Actions (1 minute), and **(3)** showing the **commits** of each team member on GitHub (2 minutes)

Create a *seperate* **Gource** (**https://gource.io/**) video showing who made what contributions to your repository. We gave demonstrations of this tool in class. You need to download and install this software then point it to your GitHub repo.

(**3.5 points**).

Recorded video characteristics: The recorded video should also show that your project is linked with the remote GitHub repository correctly. No need to explain the content inside the pom.xml file and the github-actions-demo.yml file. You need to show how the GitHub Actions workflow is triggered when you push your code to the repository and the unit tests are automatically executed. Show that all tests have passed. The total recording of the *first* video should not be more than **4 minutes** and 100 MB. The recorded video should have a 720p resolution. The total recording of the *second* video should not be more than **2 minutes** and 50 MB. You need to upload BOTH videos to Canvas as part of your submission. Alternatively, once you have created the videos (e.g first video via SharePoint or MS Teams), you can insert the link to the recorded video in a WORD/PDF file and give your tutor access to the recorded video. The WORD/PDF file should be uploaded as part of your submission to Canvas.

Activity 1 (NOTES):

Note 1. There is no need to have your face visible in the recorded video.

Note 2. There is no need for all team members to talk in the recorded video. This can be done by one member. It is fine if all team members want to talk in the recorded video.

Note 3. You can use a TXT file or multiple TXT files to implement and test the Person class. The data in the TXT file(s) should be readable. It is up to you how you want to structure the data in the TXT files. There is no need to store the post's information in a Database Management System like SQL Server, MySQL, Oracle, etc.

Note 4. You MUST write enough meaningful comments to explain your code in JUnit 5 and Java.

Note 5. You CAN develop your code (implementing functions and JUnit test code) in **any Java IDEs** such as Eclipse, VS Code, IntelliJ and use JUnit for tests.

Note 6. You need to develop a Maven Java project. There is no need to add a Graphical User Interface, a menu, or even the main function to your program (to interact with end-users).

Note 7. The code written above is just a placeholder. You are allowed to edit it and add new functions (e.g., add getter and setter, new constructors, etc.) and new attributes. You are completely FREE on how you want to implement and test these functions. The key point is to write enough test cases to test the conditions. You are free to consider any other conditions.

Note 8. You only need to write test cases explicitly to test the above mentioned conditions. There is no need to test other conditions. However, if you want to write test cases for any other conditions, feel free to do so.

Activity 2: User Story and Acceptance Criteria (8 points): Based on the requirements collected for the **DigitalID** platform, each team is asked to write **EIGHT** user stories and **THREE** acceptance criteria for each user story. Each user story and its acceptance criteria have **1 point**. In total, you need to write **8 user stories** and **24 acceptance criteria**. The user stories and acceptance criteria should be written based on the template and principles taught in Week 8. See the **template**.

4. Submission Instructions & Feedback

As this assignment is team-based, each team must submit a report that includes all activities described above. The team needs to upload the contribution form, along with other documents. The grade of each team member is determined based on their level of contribution. Each team member is expected to contribute equally to this team-based assignment. **Click here to download the Statement of Contributions form**.

Submission Type: A Zip file includes the following items:

1. A written report that contains (a) Test Cases developed for testing the functions and (b) User Stories and Acceptance Criteria of the User Stories.

2. The whole Java project that includes the implementation of the specified functions and Unit Tests

3. A recorded video (4 minutes, 720p resolution, 100 MB) or a WORD/PDF file that includes a link to the recorded video (give access to ONLY your tutor to watch this video)

4.A Gource video showing the contributions made in your GitHub repository by each member of your team.

5. Completed the contribution form

- **Note**: It is acceptable to put your names in the signature part of the contribution statement form. There is **NO** need to do a physical or digital signature.

Report Length: The criteria are in the correctness and completeness of the artefacts in the report.

Late submission: Assignments received late and without prior extension approval or special consideration will be penalised by a deduction of 10% of the total score possible per calendar day late for that assessment

5. Required Software Tools

- For Activity 1, You can develop your code (the implementation of functions and JUnit test code) in any Java IDEs such as **Eclipse** and **IntelliJ** and use **JUnit for tests.**
- For Activity 2, you can use MS Word.

6. Important Notes

- You can get ideas and learn from all online resources for Assignments. It is your responsibility to ensure that your submission is original and solely your own work. Otherwise, you will lose some marks.
- You need to work on Assignment 4 as part of a group. You are not allowed to submit Assignment 4 individually. The individual submission in Assignment 4 will **NOT be marked (Zero Mark).**
- Extension requests should be made **one working day** before the deadline.

- **Extensions are for individuals, not per group**. In other words, if a member of a group gets an extension, the group should submit the assignment by the deadline, and that member should submit the updated version of the assignment to me via email. I will check the differences between the original version and the updated version.

7. Assessment Criteria

Your report will be assessed on the following criteria:

Criteria	Ratings					Pts
Activity 1.1.1 (Unit Tests and Test Cases for three functions) **Note: 7.5 points in total: unit tests and test cases for each function have 2.5 points**	**7.5 Pts** **Full Marks** The Unit tests developed for the functions are correct. The test cases (15 test cases) and at least one test data for each test case are correctly written, without any mistakes. The written codes have enough meaningful comments.	**7.5 to >5.64 Pts** **Partial Mark** The Unit tests developed for the functions are, to a large extent, correct. The majority of test cases (15 test cases) and at least one test data for each test case are correctly written, with 1-3 minor mistakes. The written codes have enough meaningful comments.	**5.64 to >3.75 Pts** **Partial Mark** The Unit tests developed for the functions are, to some extent, correct. At least half of the test cases (15 test cases) are correctly written, with 1-3 mistakes. The written codes have to some extent meaningful comments.	**3.75 to >0.0 Pts** **Partial Mark** The Unit tests developed for the functions are, to some extent, correct. Less than 50% of the test cases (15 test cases) are correctly written, with more than three mistakes. The written codes have to some extent meaningful comments.	**0 Pts** **No Marks** The student team does not submit anything, or a student submits the assignment individually (not as part of a group).	7.5 pts
Activity 1.1.2 (The implementation of three functions in Java) **Note: 6 points in total: each function implementation has 2 points**	**6 Pts** **Full Marks** The codes developed for the functions are correct without any mistakes. The written codes have enough meaningful comments.	**6 to >4.5 Pts** **Partial Mark** The codes developed for the functions are, to a large extent, correct with 1-3 minor mistakes. The written codes have enough, to a large extent, meaningful comments.	**4.5 to >3.0 Pts** **Partial Mark** The codes developed for the functions are, to some extent, correct with 1-3 mistakes. The written codes have enough, to some extent, meaningful comments.	**3 to >0.0 Pts** **Partial Mark** The codes developed for the given function are, to some extent, correct, with more than 3 mistakes. The written codes have enough, to some extent, meaningful comments.	**0 Pts** **No Marks** The student team does not submit anything, or a student submits the assignment individually (not as part of a group).	6 pts
Activity 1.2 (Presentation, GitHub, GitHub Actions,	**3.5 Pts** **Full Marks** The presentation	**3.5 to >2.63 Pts** **Partial Mark** The presentation is,	**2.63 to >1.75 Pts** **Partial Mark** The presentation is, to some	**1.75 to >0.0 Pts** **Partial Mark** The presentation is,	**0 Pts** **No Marks** The student	3.5 pts

Criteria	Ratings					Pts
Gource)	is delivered well. The team describe well (1) the execution and testing of the program on their local machine with the changes made to a TXT file/TXT files(1 minute), (2) running and testing the program using GitHub Actions with the changes made to a TXT file/TXT files on GitHub (1 minute), and (3) showing the commits of each team member on GitHub (2 minutes). This part should have no mistakes. Includes Gource video.	to a large extent, delivered well. The team to a large extent well describe (1) the execution and testing of the program on their local machine with the changes made to a TXT file/TXT files (1 minute), (2) running and testing the program using GitHub Actions with the changes made to a TXT file/TXT files on GitHub (1 minute), and (3) showing the commits of each team member on GitHub (2 minutes). This part may include 1-3 minor mistakes. Includes Gource video.	extent, delivered well. The team to some extent well describe (1) the execution and testing of the program on their local machine with the changes made to a TXT file/TXT files (1 minute), (2) running and testing the program using GitHub Actions with the changes made to a TXT file/TXT files on GitHub (1 minute), and (3) showing the commits of each team member on GitHub (2 minutes). This part may include 1-3 mistakes. No Gource video.	to some extent, delivered well. The team to some extent well describe (1) the execution and testing of the program on their local machine with the changes made to a TXT file/TXT files (1 minute), (2) running and testing the program using GitHub Actions with the changes made to a TXT file/TXT files on GitHub (1 minute), and (3) showing the commits of each team member on GitHub (2 minutes). This part may include more than 3 mistakes. No Gource video.	team does not submit anything, or a student submits the assignment individually (not as part of a group).	
Activity 2 (User Stories and their acceptance	**8 Pts** **Full Marks** The 8 developed user stories and their acceptance	**8 to >6.0 Pts** **Partial Mark** The 8 developed user stories and their acceptance	**6 to >4.0 Pts** **Partial Mark** The 8 developed user stories and their acceptance	**4 to >0.0 Pts** **Partial Mark** The 8 developed use stories and their acceptance	**0 Pts** **No Marks** The student team does not submit anything, or a student	8 pts

Criteria	Ratings					Pts
criteria) Note: 8 User Stories and their 3 respective acceptance criteria (8 points – each user story and its three acceptance criteria 1 point)**	criteria are meaningful, well-described, and in the scope of the project and fully follow the template and principles, without any mistakes.	criteria are, to a large extent, meaningful, well-described, and in the scope of the project and to a large extent follow the template and principles, with 1-3 minor mistakes.	criteria are, to some extent, meaningful, well-described, and in the scope of the project and to some extent follow the template and principles, with 1-3 mistakes.	criteria are, to some extent, meaningful, well-described, and in the scope of the project and to some extent follow the template and principles, with more than 3 mistakes.	submits the assignment individually (not as part of a group).	
Deduction	**0 Pts** **Deduction**		**0 Pts** **No Marks**			0 pts
						Total points: 25