**IT7510 Capstone Semester 2 Year 2025**

**Learning Outcomes Tracking Application (LOTA)**



**Group Name: FAB**  
**Group Members:**

Frances Liu

Amarsh Tyagi

Bowang Ding



**Client:** Chalinor Baliuag

**Organization:** Whitireia and WelTec



Table of Content

[Sprint 1 Unit Testing 3](#_Toc210422411)

[Manage Tutor 3](#_Toc210422412)

[Manage Trimester 7](#_Toc210422413)

[Manage Course 13](#_Toc210422414)

[Manage Qualification 20](#_Toc210422415)

[Manage Student 25](#_Toc210422416)

[Sprint 2 Unit Testing 32](#_Toc210422417)

[LO Scores 32](#_Toc210422418)

[LO Results 34](#_Toc210422419)

[Sprint 3 Unit Testing 38](#_Toc210422420)

[Tutor 38](#_Toc210422421)

[Sprint 4 Unit Testing 40](#_Toc210422422)

[Student 40](#_Toc210422423)

The Learning Outcomes Tracking Application (LOTA) is a technological solution to help the Information Technology School of Whitireia and WelTec to monitor and manage students Learning Outcomes (LOs) achievements in IT programs. The aim of developing this application is to enable stakeholders to track whether students pass or fail each individual LO evaluated in each assignment and exam across different selected courses. Stakeholders include students, tutors, and administrators.

There are four sprints in the project. During the sprints, the team adopts Scrum and XP methodologies. Test-Driven Development (TDD) is one of the core practices of Extreme Programming (XP). So, under TDD practice, software development approach should be a cycle, including:

1. Write a failing unit test for the new functionality.
2. Write the minimum code needed to make the test pass.
3. Run all tests to ensure no failures.
4. Refactor the code while keeping tests passing.

(Wikipedia: [Test-driven development - Wikipedia](https://en.wikipedia.org/wiki/Test-driven_development?utm_source=chatgpt.com))

# Sprint 1 Unit Testing

This part is the collection of Sprint 1 unit testing results, whose focus is administrative functions. Both the failing and passing tests are included.

## Manage Tutor

|  |  |  |
| --- | --- | --- |
| **1.1** |  | The screenshots illustrate the TDD process for the AddTutorCourseAsync\_ShouldAddCoursesAndSave test. This is to verify the AddTutorCourseAsync method correctly adds multiple courses to a tutor and saves the changes to the database.  Figure 1.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 1.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that courses are correctly added to a tutor and changes are saved to the database. |
| **1.2** |  |
| **2.1** |  | The screenshots illustrate the TDD process for the GetTutorByIdAsync\_ShouldReturnTutor test. This is to verify that the GetTutorByIdAsync method correctly retrieves a tutor by their ID.  Figure 2.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 2.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that tutor is correctly retrieved when providing a valid ID. |
| **2.2** |  |
| **3.1** |  | The screenshots illustrate the TDD process for the GetTutorCoursesAsync \_ShouldReturnCoursesForTutor test. This is to verify that the GetTutorCoursesAsync method correctly retrieves all courses associated with a given tutor.  Figure 3.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 3.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that courses are correctly retrieved when providing a valid tutor. |
| **3.2** |  |
| **4.1** |  | The screenshots illustrate the TDD process for the RemoveAllTutorCoursesAsync\_ShouldRemoveAndSave\_WhenCoursesExist test. This is to verify that the RemoveAllTutorCoursesAsync method successfully removes all courses assigned to a tutor and saves the changes to the database when such courses exist.  Figure 4.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 4.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that all assigned courses are removed and all changes are saved when courses exist. |
| **4.2** |  |
| **5.1** |  | The screenshots illustrate the TDD process for the SearchTutorsAsync\_ShouldReturnFilteredTutors\_WhenSearchTermProvided test. This is to verify that the SearchTutorsAsync method return only tutors whose details match the given search term.  Figure 5.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 5.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that only tutors, whose details match the given search term, are returned. |
| **5.2** |  |
| **6.1** |  | The screenshots illustrate the TDD process for the GetAllTutorsAsync\_WithExistingTutors\_ReturnAllTutors test. This is to verify that the GetAllTutorsAsync method correctly fetches all existing tutors and returns them as a list.  Figure 6.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 6.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that all tutors currently in the system are retrieved successfully. |
| **6.2** |  |
| **7** |  | This figure shows that all test cases were run again. All passed. |

## Manage Trimester

|  |  |  |
| --- | --- | --- |
| **1.1** |  | The screenshots illustrate the TDD process for the CreateAsync\_ShouldReturnCreatedTrimesterDTO test. This is to verify that the CreateAsync method properly creates a new trimester and returns the corresponding DTO of the created trimester.  Figure 1.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 1.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that trimesters can be created and returned successfully. |
| **1.2** |  |
| **2.1** |  | The screenshots illustrate the TDD process for the DeleteAsync\_ShouldRemoveTrimester test. This is to verify that the DeleteAsync method correctly removes the specified trimester from the system.  Figure 2.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 2.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that trimester is removed from the database. |
| **2.2** |  |
| **3.1** |  | The screenshots illustrate the TDD process for the GetActiveTrimestersAsync\_ShouldReturnDTOList test. This is to verify that the GetActiveTrimestersAsync method correctly filters and maps active trimesters into a trimester list.  Figure 3.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 3.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that when the method is called, it retrieves only the active trimesters and returns the results as a list of objects. |
| **3.2** |  |
| **4.1** |  | The screenshots illustrate the TDD process for the GetByAcademicYearAndTrimesterAsync\_ShouldReturnDTO test. This is to verify that the GetByAcademicYearAndTrimesterAsync method can accurately find the trimester for the given year and number, and return it as a TrimesterDTO.  Figure 4.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 4.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method correctly retrieves the matching trimester from the data source. |
| **4.2** |  |
| **5.1** |  | The screenshots illustrate the TDD process for the GetByAcademicYearAsync\_ShouldReturnDTOList mester test. This is to verify that the GetByAcademicYearAsync method correctly returns a list of trimester DTOs for the specified academic year.  Figure 5.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 5.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that when the method is called with a given academic year, all trimesters belonging to that year should be retrieved, and the result is a list of objects. |
| **5.2** |  |
| **6.1** |  | The screenshots illustrate the TDD process for the GetByIdAsync\_ShouldReturnDTO test. This is to verify that the GetByIdAsync method correctly fetches and maps a trimester by its ID into a TrimesterDTO.  Figure 6.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 6.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that when the method is called with a valid ID, matching trimesters should be retrieved, and the result is returned as a TrimesterDTO object. |
| **6.2** |  |
| **7.1** |  | The screenshots illustrate the TDD process for the GetCurrentTrimesterAsync\_ShouldReturnCurrentTrimesterDTO test. This is to verify that the GetCurrentTrimesterAsync method identifies the ongoing trimester and returns it as a properly mapped TrimesterDTO.  Figure 7.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 7.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method correctly determines which trimester is the current one. |
| **7.2** |  |
| **8.1** |  | The screenshots illustrate the TDD process for the UpdateAsync\_ShouldReturnUpdatedTrimesterDTO test. This is to verify that the UpdateAsync method correctly updates a trimester and returns the updated trimester as a TrimesterDTO.  Figure 8.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 8.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the trimester is correctly updated in the database, |
| **8.2** |  |
| **9** |  | This figure shows that all test cases were run again. All passed. |

## Manage Course

|  |  |  |
| --- | --- | --- |
| **1.1** |  | The screenshots illustrate the TDD process for the AddStudentsToCourseAsync\_ShouldAddStudents\_WhenNotEnrolled test. This is to verify that the AddStudentsToCourseAsync method correctly enrolls new students into a course only when they are not already enrolled.  Figure 1.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 1.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that a list of not enrolled students are successfully added in the course, when the method is called. |
| **1.2** |  |
| **2.1** |  | The screenshots illustrate the TDD process for the CreateCourseAsync\_WithExistingCourses\_ReturnsCreatedCourse test. This is to verify that the CreateCourseAsync method can add a new course to the system that already has courses, and returns the created course with correct details.  Figure 2.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 2.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that a new course is successfully created even when other courses already exist. |
| **2.2** |  |
| **3.1** |  | The screenshots illustrate the TDD process for the GetAllCoursesAsync\_WithExistingCourses\_ReturnsAllCourses test. This is to verify that the GetAllCoursesAsync method correctly fetches all existing courses and returns them as a list.  Figure 3.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 3.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that all courses currently in the system are retrieved successfully. |
| **3.2** |  |
| **4.1** |  | The screenshots illustrate the TDD process for the GetCourseByCodeAsync\_WithExistingCourses\_ReturnsCourse test. This is to verify that the GetCourseByCodeAsync method correctly finds and returns the course with the specified code as a DTO.  Figure 4.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 4.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method retrieves the course that matches the given code. |
| **4.2** |  |
| **5.1** |  | The screenshots illustrate the TDD process for the GetCourseByIdAsync\_WithExistingCourses\_ReturnsCourse test. This is to verify that the GetCourseByIdAsync method correctly finds and returns the course with the specified ID as a DTO.  Figure 5.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 5.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method retrieves the course that matches the given ID. |
| **5.2** |  |
| **6.1** |  | The screenshots illustrate the TDD process for the GetCoursesByNameOrCodeAsy\_WithExistingCourses\_ReturnsCourse test. This is to verify that the GetCoursesByNameOrCodeAsync method correctly finds courses by name or code and returns them as DTOs.  Figure 6.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 6.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method retrieves all courses that match the given name or code and returns a list. |
| **6.2** |  |
| **7.1** |  | The screenshots illustrate the TDD process for the RemoveCourse\_ValidId\_CallsRemoveAndSaveAsync test. This is to verify that the RemoveCourse method correctly triggers both the remove and save operations for a valid course ID.  Figure 7.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 7.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called with a valid course ID and the method correctly calls the Remove and Save. |
| **7.2** |  |
| **8.1** |  | The screenshots illustrate the TDD process for the RemoveStudentFromCourseAsync\_ValidId\_CallsRemoveAndSaveAsync test. This is to verify that the RemoveStudentFromCourseAsync method properly removes a student from a course and saves the change.  Figure 8.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 8.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called with a valid student ID and course ID, the student is successfully removed from the specified course and save the removal. |
| **8.2** |  |
| **9.1** |  | The screenshots illustrate the TDD process for the UpdateCourse\_ValidId\_CallsRemoveAndSaveAsync test. This is to verify that the UpdateCourse method correctly updates a course’s details and saves the changes.  Figure 9.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 9.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called with a valid course ID and updated data. |
| **9.2** |  |
| **10** |  | This figure shows that all test cases were run again. All passed. |

## Manage Qualification

|  |  |  |
| --- | --- | --- |
| **1.1** |  | The screenshots illustrate the TDD process for the CreateQualificationAsync\_ShouldAddAndReturnDTO test. This is to verify that the CreateQualificationAsync method correctly adds a new qualification and returns it as a DTO.  Figure 1.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 1.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that new qualifications are successfully added to the system |
| **1.2** |  |
| **2.1** |  | The screenshots illustrate the TDD process for the DeleteQualificationAsync\_ShouldRemoveAndReturnTrue test. This is to verify that the DeleteQualificationAsync method correctly deletes the specified qualification and confirms success by returning true.  Figure 2.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 2.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called with a valid qualification ID, and the qualification is successfully removed from the system |
| **2.2** |  |
| **3.1** |  | The screenshots illustrate the TDD process for the GetAllQualificationsAsync\_ShouldReturnMappedDTOs test. This is to verify that the GetAllQualificationsAsync method correctly fetches all qualifications and maps them to DTOs.  Figure 3.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 3.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that all qualifications currently in the system are retrieved successfully. |
| **3.2** |  |
| **4.1** |  | The screenshots illustrate the TDD process for the GetAllQualificationTypesAsync\_ShouldReturnListOfStrings test. This is to verify that the GetAllQualificationTypesAsync method correctly fetches all qualification types and returns them as a list of strings.  Figure 4.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 4.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that all qualification types currently in the system are retrieved successfully. The results are returned as a list of strings |
| **4.2** |  |
| **5.1** |  | The screenshots illustrate the TDD process for the GetQualificationByIdAsync\_ShouldReturnQualificationDTO test. This is to verify that the GetQualificationByIdAsync method correctly finds and returns the specified qualification as a DTO.  Figure 5.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 5.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that when the method is called with a valid qualification ID, it retrieves the qualification matching the given ID. |
| **5.2** |  |
| **6.1** |  | The screenshots illustrate the TDD process for the IsQualificationNameExistsAsync\_ShouldReturnTrue test. This is to verify that the IsQualificationNameExistsAsync method can detect existing qualification names and returns true when the name is already in use.  Figure 6.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 6.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method correctly checks the database to see if a qualification with that name exists. |
| **6.2** |  |
| **7.1** |  | The screenshots illustrate the TDD process for the UpdateQualificationAsync\_ShouldUpdateAndReturnDTO test. This is to verify that the UpdateQualificationAsync method correctly updates a qualification and returns the updated DTO.  Figure 7.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 7.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that when the method is called with a valid qualification ID and updated data, the qualification is successfully updated in the system. |
| **7.2** |  |
| **8** |  | This figure shows that all test cases were run again. All passed. |

## Manage Student

|  |  |  |
| --- | --- | --- |
| **1.1** |  | The screenshots illustrate the TDD process for the CreateStudentAsync\_SuccessfullyCreatesStudent test. This is to verify that the CreateStudentAsync method correctly adds a new student to the system and returns the created student.  Figure 1.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 1.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that new student is successfully added to the database. |
| **1.2** |  |
| **2.1** |  | The screenshots illustrate the TDD process for the DeleteStudentAsync\_SucceedsWhenNoEnrolledCourses test. This is to verify that the DeleteStudentAsync method correctly deletes a student when they are not enrolled in any courses.  Figure 2.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 2.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called on a student who is not enrolled in any courses, and the student is successfully removed from the system. |
| **2.2** |  |
| **3.1** |  | The screenshots illustrate the TDD process for the GetAllStudentsAsync\_ReturnOnlyStudentsWithStudentNo test. This is to verify that the GetAllStudentsAsync method filters out students without a student number and returns only valid students.  Figure 3.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 3.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that only students who have a valid student number are returned. |
| **3.2** |  |
| **4.1** |  | The screenshots illustrate the TDD process for the GetEnrolledStudentsAsync\_ReturnsMappedStudents test. This is to verify that the GetEnrolledStudentsAsync method correctly fetches enrolled students for a course and returns them as mapped list.  Figure 4.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 4.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called with a with a specific course ID and retrieves all students enrolled in the course. |
| **4.2** |  |
| **5.1** |  | The screenshots illustrate the TDD process for the GetStudentByIdAsync\_ReturnStudentDTO\_WhenStudentHasStudentNo test. This is to verify that the GetStudentByIdAsyn method correctly returns a student as a DTO when the student has a valid StudentNo.  Figure 5.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 5.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called with a valid student ID for a student who has a StudentNo and retrieves the correct student from the system. |
| **5.2** |  |
| **6.1** |  | The screenshots illustrate the TDD process for the GetStudentsByNameOrEmailAsync\_ReturnFilteredStudents test. This is to verify that the GetStudentsByNameOrEmailAsync method correctly filters students by name or email.  Figure 6.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 6.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called with a name or email filter and retrieves only students whose name or email matches the filter. |
| **6.2** |  |
| **7.1** |  | The screenshots illustrate the TDD process for the IsStudentEmailExistsAsync\_ReturnsTrueIfExists test. This is to verify that the IsStudentEmailExistsAsync method can detect existing student emails and returns true when the email is already registered.  Figure 7.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 7.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called with a student email that already exists. |
| **7.2** |  |
| **8.1** |  | The screenshots illustrate the TDD process for the IsStudentNoExistsAsync\_ReturnsTrueIfExists test. This is to verify that the IsStudentNoExistsAsync method can detect existing student numbers and returns true when the StudentNo is already registered.  Figure 8.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 8.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called with a student number that already exists in the system. |
| **8.2** |  |
| **9.1** |  | The screenshots illustrate the TDD process for the UpdateStudentAsync\_SuccessfullyUpdatesStudent test. This is to verify that the UpdateStudentAsync method orrectly updates a student’s details and returns the updated DTO.  Figure 9.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 9.2 (green) screenshot demonstrates the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method is called with a valid student ID and updated student data |
| **9.2** |  |
| **10** |  | This figure shows that all test cases were run again. All passed. |

# Sprint 2 Unit Testing

This part is the collection of Sprint 2 unit testing results, whose focus is still administrative functions. Both the failing and passing tests are included.

## LO Scores

|  |  |  |
| --- | --- | --- |
| **1.1** |  | The screenshots illustrate the TDD process for the BatchSaveStudentLOScoresAsync\_ShouldCallSaveAsync test. This is to verify the BatchSaveStudentLOScoresAsync method correctly save student LO scores in batch.  Figure 1.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 1.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that the system can behave correctly, the scores can saved successfully. |
| **1.2** |  |
| **2.1** |  | The screenshots illustrate the TDD process for the GetCourseOfferingDetailsByCourseOfferingId\_ReturnsCourseOfferingDetailsDTO test. This is to verify the GetCourseOfferingDetailsByCourseOfferingId method can return course offering details.  Figure 2.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 2.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method can work correctly and giving back the expected course details. |
| **2.2** |  |
| **3.1** |  | The screenshots illustrate the TDD process for the GetCourseOfferingWithAssessmentsAsync\_ShouldReturnResult test. This is to verify the GetCourseOfferingWithAssessmentsAsync method correctly return the course offering along with its assessments.  Figure 3.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 3.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method can work and give back the expected course and assessment details. |
| **3.2** |  |
| **4** |  | This figure shows that all test cases were run again. All passed. |

## LO Results

|  |  |  |
| --- | --- | --- |
| **1.1** |  | The screenshots illustrate the TDD process for the GetFailedAssessmentsForRetakeAsync\_ReturnsList test. This is to verify the GetFailedAssessmentsForRetakeAsync method correctly return a list of assessments that a student needs to retake.  Figure 1.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 1.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method can retrieve the retake assessments successfully. |
| **1.2** |  |
| **2.1** |  | The screenshots illustrate the TDD process for the GetLatestTrimesterCourseOfferingsAsync\_ReturnsCourseOfferings test. This is to verify that the GetLatestTrimesterCourseOfferingsAsync method can return the course offerings for the latest trimester.  Figure 2.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 2.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method retrieved the latest courses correctly and successfully. |
| **2.2** |  |
| **3.1** |  | The screenshots illustrate the TDD process for the GetLOResultsByCourseOfferingAsync\_ReturnsLOResultDTO test. This is to verify that the GetLOResultsByCourseOfferingAsync method should return the learning outcome results for a course offering.  Figure 3.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 3.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method can give back the expected learning outcome results successfully. |
| **3.2** |  |
| **4.1** |  | The screenshots illustrate the TDD process for the UpdateRetakeScoresAsync\_Success test. This is to verify that the UpdateRetakeScoresAsync method successfully update the retake scores for the student.  Figure 4.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 4.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method works correctly and the retake scores are updated as expected. |
| **4.2** |  |
| **5** |  | This figure shows that all test cases were run again. All passed. |

# Sprint 3 Unit Testing

This part is the collection of Sprint 3 unit testing results, whose focus is tutors related functions. Both the failing and passing tests are included.

## Tutor

|  |  |  |
| --- | --- | --- |
| **1.1** |  | The screenshots illustrate the TDD process for the GetAllAssessmentsByTutorIdAsync\_ReturnsAssessmentsForTutor test. This is to verify the GetAllAssessmentsByTutorIdAsync method correctly returns the list of assessments that belong to that tutor when given correct tutor ID.  Figure 1.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 1.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that the assessments can be returned correctly with specific tutor ID. |
| **1.2** |  |
| **2.1** |  | The screenshots illustrate the TDD process for the GetTrimesterCoursesByTutorAndTrimesterAsync\_ReturnsTrimesterCourses test. This is to verify GetTrimesterCoursesByTutorAndTrimesterAsync method can correctly return the list of courses that the tutor is assigned in the trimester when given a tutor ID and a trimester.  Figure 2.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 2.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that the method can return the expected list of trimester courses for the tutor. |
| **2.2** |  |

# Sprint 4 Unit Testing

This part is the collection of Sprint 4 unit testing results, whose focus is students related functions. Both the failing and passing tests are included.

## Student

|  |  |  |
| --- | --- | --- |
| **1.1** |  | The screenshots illustrate the TDD process for the GetStudentLOResultsAsync\_ReturnsStudentLOResult test. This is to verify the GetStudentLOResultsAsync method correctly returns the LO results for a student with specific student ID.  Figure 1.1 (red) shows the initial failing test, where a NotImplementedException occurs, representing the TDD Red phase. Figure 1.2 (green) screenshot shows the TDD Green phase, where the test passes successfully after implementing the method, confirming that the function can return specific LO results successfully by given correct student ID. |
| **1.2** |  |