**Tutorial Performance Test System**

Submitted By: **Helen O’Brien**

Student Number: **T00183586**

Computing with Multimedia: **Stage 2**

Date Submitted: **Friday 22nd April 2016**

**Table of Contents**

[1. Overview 5](#_Toc449095936)

[2. Functional Components 6](#_Toc449095937)

[3. Functional User Requirements 8](#_Toc449095938)

[3.1. Tutorial Performance Test System will Login the User 8](#_Toc449095939)

[3.2. Tutorial Performance Test System will process student. 8](#_Toc449095940)

[3.3. Tutorial Performance Test System will process questions. 8](#_Toc449095941)

[3.4. Tutorial Performance Test System will manage tests. 8](#_Toc449095942)

[4. Functional System Requirements 9](#_Toc449095943)

[4.1. System Level Use Case Diagram 9](#_Toc449095944)

[Tutorial Performance Test System 9](file:///X:\students\t00183586\Software%20Engineering\SWEngProject\Documentation\Dogs.docx#_Toc449095945)

[4.2. Login 10](#_Toc449095946)

[4.2.1. Login 10](#_Toc449095947)

[4.3. Manage Students 13](#_Toc449095948)

[4.3.1. Register Student 14](#_Toc449095949)

[4.3.2. Amend Student 17](#_Toc449095950)

[4.3.2.1. Amend Student Activity Diagram 17](#_Toc449095951)

[4.3.2.2. Amend Student Use Case Narrative 18](#_Toc449095952)

[4.3.3. Deregister Student 21](#_Toc449095953)

[4.4. Process Questions 25](#_Toc449095954)

[4.4.1. Create Question 26](#_Toc449095955)

[4.4.1.1. Create Question Use Case Narrative 27](#_Toc449095956)

[4.4.2. Amend Question 29](#_Toc449095957)

[4.4.3. Delete Question 32](#_Toc449095958)

[4.5. Manage Tests 34](#_Toc449095959)

[4.5.1. Take Test 35](#_Toc449095960)

[4.5.1.1. Take Test Activity Diagram 36](#_Toc449095961)

[4.5.1.2. Take Test Use Case Narrative 37](#_Toc449095962)

[4.5.2. Display Student Profile 40](#_Toc449095963)

[4.5.2.1. Display Student Profile Use Case 40](#_Toc449095964)

[4.5.2.2. Display Student Test Profile Use Case Narrative 41](#_Toc449095965)

[5. System Model 43](#_Toc449095966)

[External Entities: 43](#_Toc449095967)

[Data Stores: 43](#_Toc449095968)

[Processes: 43](#_Toc449095969)

[Level-0 DFD 44](#_Toc449095970)

[Level-1 DFD 45](#_Toc449095971)

[Level-2 DFD (Process P1: Login) 45](#_Toc449095972)

[Level-2 DFD (Process P2: Process Students) 46](#_Toc449095973)

[Level-2 DFD (Process P3: Process Questions) 48](#_Toc449095974)

[Level-2 DFD (Process P4: Process Tests) 49](#_Toc449095975)

[6. Data Model (Class Diagram) 50](#_Toc449095976)

[7. Database Design 51](#_Toc449095977)

[8. Conclusion 54](#_Toc449095978)

[9. Appendices 55](#_Toc449095979)

[Appendix A - Actor Glossary 55](#_Toc449095980)

[Appendix B – Use Case Glossary 56](#_Toc449095981)

# Overview

The Institute of Technology Tralee wish to implement a Tutorial Performance Test System to assess the test results of students upon completion of a module tutorial. The college has stipulated that students should be able to review their collective test results to assess their knowledge of the subject material.

The faculty should be able to amend and deregister a student. They should also be able to access student test profiles.

# Functional Components

The Tutorial Performance Test System consists of functions to manage student registration, the formulation and maintenance of multiple choice questions, the formulation of tests and the processing of test results. The system will generate a student test profile detailing the performance of a student in every test taken and be accessible by the student and the lecturer.

# Functional User Requirements

## Tutorial Performance Test System will Login the User

## Tutorial Performance Test System will process student.

* + 1. Tutorial Performance Test System will register students.
    2. Tutorial Performance Test System will amend student details.
    3. Tutorial Performance Test System will deregister student.

## Tutorial Performance Test System will process questions.

* + 1. Tutorial Performance Test System will create questions.
    2. Tutorial Performance Test System will amend questions.
    3. Tutorial Performance Test System will delete questions.

## Tutorial Performance Test System will manage tests.

* + 1. Tutorial Performance Test System will compile a test.
    2. Tutorial Performance Test System will display a student’s test results.
    3. Tutorial Performance Test System will display a student’s profile.

# Functional System Requirements

The Tutorial Performance Test System consists of functions for student registration, the formulation and maintenance of questions, producing a test and reviewing test performance. Students will be maintained on a list with functions to register, amend and delete. Questions will be maintained on a list with functions to create, amend and delete. The questions will be set at one of three levels, basic, intermediate and advanced. Tests will be generated by randomly selected questions. Test results can be reviewed collectively.

## System Level Use Case Diagram

### Tutorial Performance Test System

Lecturer

Student

## Login

The Tutorial Performance Test System login function will enable students and lecturers registered with the system admittance, verifying their id and password.

## Login

Student

<<include>>

<<extends>>

Lecturer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Login** | |
| **Use Case Id** | 01 | |
| **Priority** | 1 | |
| **Source** | User | |
| **Primary Business Actor** | Student | |
| **Other Participating Actors** | Lecturer | |
| **Description** | User gain access to the system by entering Id and password. | |
| **Preconditions** | User must be registered. | |
| **Trigger** | Clicking icon entry link | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The user invokes the Login function.  **Step 3:** The user enters their Id.  **Step 4:** The user enters their password.  **Step 5:** The user submits details. | **Step 2:** The system displays the login UI.  **Step 6:** The system validates the details entered:   * All fields must be entered * Id and password are correct   **Step 7:** The system validates user category i.e. student or lecturer.  **Step 8:** The system loads the relevant Main Menu. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid entry** |  | **Step 6:** The system identifies an invalid entry.  **Step 7:** The system displays an error message.  **Step 8:** The system prompts the student to re-enter the details. |

|  |  |  |
| --- | --- | --- |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The user is now in the system. | |
| **Post conditions** | The user may now use the test facilities. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Students

The Tutorial Performance Test System registration function will enable the student to register and amend their details should any of those details change. Registration expires one week after the module finishes.

The lecturer is able to amend a students’ details if necessary and following disciplinary action the lecturer is also able to deregister a student.

### Register Student

<<include>>

<<extends>>

Student

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Register Student** | |
| **Use Case Id** | 02 | |
| **Priority** | 1 | |
| **Source** | Student | |
| **Primary Business Actor** | Student | |
| **Other Participating Actors** |  | |
| **Description** | This function registers a students’ details in the system and saves the details in the **Student File**. | |
| **Preconditions** | A student must complete and submit the online registration form. | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The student invokes the Register Student function.  **Step 4:** The student enters their details:   * Surname * Forename * Email * DOB * Password * Password Confirmation | **Step 2:** The system retrieves the next available **Student ID.**  **Step 3:** The system displays the **Student ID** and Student Registration UI. |
|  | **Step 5:** The student submits the details to be registered. | **Step 6:** The system validates the details entered:   * All fields must be entered * Email must be valid format * DOB must be date format * Password and Password confirmation must match   **Step 7:** The system sets a default student status of ‘r’ for registered.  **Step 8:** The system assigns the current system date as **Registration Date.**  **Step 9:** The system sets **Expiry Date** to **Registration Date** plus 13 weeks.  **Step 10:** The system saves the students details in the **Student File.**   * **Student ID** * Email * Password * Surname * Forename * DOB * Student status * Registration Date * ExpiryDate   **Step 11:** The system displays a confirmation message.  **Step 12:** The system clears the UI. |

|  |  |  |
| --- | --- | --- |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid entry of Student Details** |  | **Step 6:** The system identifies an invalid entry.  **Step 7:** The system displays an error message.  **Step 8:** The system prompts the student to re-enter the details. |
| **Conclusions** | The student is now registered in the system. | |
| **Post conditions** | The student may now use the test facilities. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Amend Student

Student

<<include>>

<<extends>>

Lecturer

## Amend Student Activity Diagram

|  |  |
| --- | --- |
| Student | System |
| Click Amend Link  Enters Password  Confirms Password  Submit Details | Assigns Amend Date  Update Details  Confirmation Message  Error Message  Validates Details  Y  N  Valid?  Retrieves Details  Display Details  Amends Details |

## Amend Student Use Case Narrative

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Amend Student Details** | |
| **Use Case Id** | 03 | |
| **Priority** | 1 | |
| **Source** | User | |
| **Primary Business Actor** | Student | |
| **Other Participating Actors** | Lecturer | |
| **Description** | Student details can be updated should there be any changes. | |
| **Preconditions** | Student details change. | |
| **Trigger** | i.e. change of address. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The student invokes the Amend Student function.  **Step 4:** The student amends their details.  **Step 5:** The student enters their password.  **Step 6:** The student confirms their password.  **Step 7:** The student submits the details to be amended. | **Step 2:** The system retrieves the Students Details.   * Surname * Forename * Email * DOB   **Step 3:** The system displays the Students Details.  **Step 8:** The system validates the details entered:   * Email must be the correct format * Date of Birth must be in date format * Passwords must match   **Step 9:** The system assigns the current system date as **Amend Date.**  **Step 10:** The system updates the details in the **Student File.**  **Step 11:** The system displays a confirmation message.  **Step 12:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Lecturer Amends Student Details** | **Step 1:** The lecturer invokes the Amend Student function.  **Step 3:** Thelecturer enters the Student Id or the Surname of the student to be amended.  **Step 5:** The lecturer selects the student.  **Step 8:** The lecturer amends the students’ details.  **Step 9:** The lecturer enters their password.  **Step 10:** The lecturer confirms their password.  **Step 11:** The lecturer submits the details to be amended. | **Step 2:** The system displays the Amend Student UI.  **Step 4:** The system displays the student relevant to the Id or a list of students by surname.  **Step 6:** The system retrieves the students’ details from the **Student File.**   * Surname * Forename * Email * DOB   **Step 7:** The system displays the students’ details to be amended.  **Step 12:** The system validates all the entries:   * Email must be the correct format * Date of Birth must be in date format * Passwords must match   **Step 13:** The system updates the student’s details in the **Student File.**  **Step 14:** The system displays a confirmation message.  **Step 15:** The system clears the UI |

|  |  |  |
| --- | --- | --- |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Student Details are not found** |  | **Step 2:** The systemsfails to retrieve the student details.  **Step 3:** The systemsdisplays and error message.  **Step 4:** The systemsresets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Student Details entered** |  | **Step 8:** The system identifies an invalid entry.  **Step 9:** The system displays an error message.  **Step 10:** The system prompts the student to re-enter the details. |
| **Conclusions** | Students details are updated in the **Student File** | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Deregister Student

Lecturer

Student

<<extends>>

<<include>>

<<include>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Deregister Student** | |
| **Use Case Id** | 04 | |
| **Priority** | 1 | |
| **Source** | Lecturer | |
| **Primary Business Actor** | Lecturer | |
| **Other Participating Actors** |  | |
| **Description** | Student status will be set to ‘Deregistered’ in the **Student File.** | |
| **Preconditions** | Misconduct leading to deregistration. | |
| **Trigger** | Misconduct | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The student invokes the Deregister Student function.  **Step 5:** The student enters their password.  **Step 6:** The student confirms their password.  **Step 7:** The student clicks the delete button. | **Step 2:** The system displays the Deregister Student UI.  **Step 3:** The system retrieves the student’s details.  **Step 4:** The system displays the student’s details.  **Step 8:** The system validates the passwords.   * The fields are filled * The passwords match * The password is correct   **Step 9:** The system displays a warning message.  **Step 10:** The system sets student status to ‘d’ for ‘Deregistered’ in the **Student** **File**.  **Step 11:** The system displays a confirmation message.  **Step 12:** The system clears the UI. |

|  |  |  |
| --- | --- | --- |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  | **Step 1:** The lecturer invokes the Deregister Student function.  **Step 3:** The lecturer enters the student’s id or surname.  **Step 5:** The lecturer selects the student to be deleted.  **Step 8:** The lecturer enters their password.  **Step 9** The lecturer confirms their password.  **Step 10:** The lecturer deletes the student.  . | **Step 2:** The system displays the Delete Student UI.  **Step 4:** The system displays the student relevant to the Id or a list of students by surname.  **Step 6:** The system retrieves the students’ details from the **Student File.**   * Surname * Forename * Email * DOB   **Step 7:** The system displays the students’ details.  **Step 11:** The system validates all the entries:   * Password and confirmation have been entered * Password is correct   **Step 12:** The system displays a warning message.  **Step 13:** The system sets student status to ‘d’ for ‘Deregistered’ in the **Student** **File**.  **Step 14:** The system displays a confirmation message.  **Step 15:** The system clears the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Student Details are not found** |  | **Step 7:** The systemsfails to retrieve the student details.  **Step 8:** The systemsdisplays and error message.  **Step 9:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Password or Password confirmation entered** |  | **Step 13:** The system identifies an invalid entry.  **Step 14:** The system displays an error message.  **Step 15:** The system prompts the lecturer to re-enter the password. |
| **Conclusions** | Student will be unable to use the system. | |
| **Post conditions** |  | |
| **Business Rules** | Deregistered Students can be amended by the lecturer and ‘reactivated’. | |
| **Implementation Constraints** |  | |

## Process Questions

The Tutorial Performance Test System questions function will maintain a list all the questions available for a test. Questions are set at one of three levels, basic, intermediate and advanced. Questions can be created, amended and deleted.

### Create Question

Tests are multiple choice and each question is created with one correct answer and three incorrect alternative answers. The questions are divided into categories and set at one of three levels, basic, intermediate and advanced (‘B’,’I’ and ‘A’). When a question is created a Question Id will be generated and saved together with the question details in the **Question File**.

<<include>>

<<extends>>

Lecturer

## Create Question Use Case Narrative

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Create Question** | |
| **Use Case Id** | 05 | |
| **Priority** | 1 | |
| **Source** | Lecturer | |
| **Primary Business Actor** | Lecturer | |
| **Other Participating Actors** |  | |
| **Description** | Question will be created, level set and saved in the **Question File.** | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The lecturerinvokes the Create Question function.  **Step 6:** The lecturerenters the question data:-   * **Level** * **Question Text** * **Answer 1** * **Answer 2** * **Answer 3** * **Answer 4** * **Correct Answer**   **Step 7:** The lecturersubmits the question data. | **Step 2:** The system allocates the next available **Question Id**.  **Step 3:** The system retrieves level details from the **Level File.**  **Step 4:** The system loads the levels.  **Step 5:** The system displays the Create Question UI.  **Step 8:** The system validates all fields are filled in.  **Step 9:** The system assigns the current system date as **QAdd Date**.  **Step 8:** The system sets question status to ‘a’ for ‘Active’.  **Step 9:** The system saves the question details in the **Question File.**  **Step 10:** The system displays a confirmation message.  **Step 11:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid entry of Question Details** |  | **Step 5:** The system identifies an invalid entry.  **Step 6:** The system displays an error message.  **Step 7:** The system prompts the lecturer to re-enter the data. |
| **Conclusions** | The question is created and saved in the **Question** **File**. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Amend Question

Lecturer

<<include>>

<<extends>>

<<include>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Amend Question** | |
| **Use Case Id** | 06 | |
| **Priority** | 1 | |
| **Source** | Lecturer | |
| **Primary Business Actor** | Lecturer | |
| **Other Participating Actors** |  | |
| **Description** | Question will be amended and updated in the **Question File.** | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The lecturerinvokes the Amend Question function.  **Step 3:** The lecturer enters the Question Id for the question to be amended.  **Step 7:** The lecturer enters the amendments to the Question:-   * **LevelCode** * **Question Text** * **Answer 1** * **Answer 2** * **Answer 3** * **Answer 4** * **Correct Answer**   **Step 8:** The lecturer submits the amendments to the Question. | **Step 2:** The system displays the Amend Question UI.  **Step 4:** The system retrieves the question details from the **Question File**.  **Step 5:** The system loads the question levels.  **Step 6:** The system displays the question details.  **Step 9:** The systemvalidates the amendments:-   * **LevelCode** * **Question Text** * **Answer 1** * **Answer 2** * **Answer 3** * **Answer 4** * **Correct Answer**   **Step 10:** The system assigns the current system date as Q**Amd Date**.  **Step 11:** The system updates the question details in the **Question File.**  **Step 12:** The system displays a confirmation message.  **Step 13:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Question is not found.** |  | **Step 4:** The systemfails to retrieves the question details from the **Question File**.  **Step 5:** The systemsdisplays an error message.  **Step 6:** The systemresets the UI. |

|  |  |  |
| --- | --- | --- |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Question Details entered** |  | **Step 9:** The system identifies an invalid entry.  **Step 10:** The system displays an error message.  **Step 11:** The system prompts the student to re-enter the details. |
| **Conclusions** | A question is updated in the **Questions File**. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Delete Question

Lecturer

<<include>>

<<extends>>

<<include>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Delete Question** | |
| **Use Case Id** | 07 | |
| **Priority** | 1 | |
| **Source** | Lecturer | |
| **Primary Business Actor** | Lecturer | |
| **Other Participating Actors** |  | |
| **Description** | Question status will be updated to ‘Deregistered’ in the **Question File.** | |
| **Preconditions** | Question no longer included in tests. | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The lecturerinvokes the Delete Question function.  **Step 3:** The lecturerenters the Question Id of question to be deleted.  **Step 6:** The lecturer enters their password.  **Step 7:** The lecturer deletes the question.  **Step 10:** The lecturer verifies the deletion. | **Step 2:** The system displays the Delete Question UI.  **Step 4:** The system retrieves the question from the **Question File**.  **Step 5:** The system displays the question text to be deleted.  **Step 8:** The system validates the password.  **Step 9:** The system displaysa verification message.  **Step 11:** The system updates the question status to ‘d’ for deleted in the **Question File**.  **Step 12:** The system displays a confirmation message.    **Step 13:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Question is not found.** |  | **Step 4:** The systemfails to retrieves the question from the **Question File**.  **Step 5:** The systemsdisplays an error message.  **Step 6:** The systemresets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Password** |  | **Step 8:** The system identifies the entry as invalid.  **Step 9:** The system displays an error message.  **Step 10:** The system prompts the Password to be re-entered. |
| **Conclusions** | The Question Status is updated in the **Question File**. | |
| **Post conditions** | The Question is disabled. | |
| **Business Rules** | Deleted questions cannot be included in a test.  A deregistered question can be amended and updated to ‘Active’. | |
| **Implementation Constraints** |  | |

## Manage Tests

The Tutorial Performance Test System test function will compile tests of randomly generated questions. The tests are set at one of three levels, basic, intermediate and advanced. Once the student has completed a test the system will provide an opportunity to display the test score. The student can review the student test profile which shows the results of all the tests the student has taken.

The faculty also have access to the student test profiles to review a student’s test results.

## Take Test

Tests are compiled of randomly generated questions and presented in multiple choice format, they are set at one of three levels, basic, intermediate and advanced. A choice of four answers will be displayed for each question and there will be four questions per test.

<<include>>

<<extends>>

Student

<<include>>

## Take Test Activity Diagram

|  |  |
| --- | --- |
| **Student** | **System** |
| Click Take Test Link  Choose level  Start Test  Answer Question  Next Question | Y  Score Incremented  Correct?  Save Answer  Progress Bar Incremented  Y  N  Answers  <4?  ?  Displays Score  Saves Question Data  Saves Test Data  Progress Bar Incremented  N  Score Incremented  Correct?  Save Answer  Load Question  N  Questions  <4?  ?  Y  Save Question Id  Check Number Unique  Generate Random Number  Retrieve Level Question Ids  Level Questions Counted  Display Test UI  Load Levels  Retrieve Levels  Gets Next Test Id |

## Take Test Use Case Narrative

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Take Test** | |
| **Use Case Id** | 08 | |
| **Priority** | 1 | |
| **Source** | Student | |
| **Primary Business Actor** | Student | |
| **Other Participating Actors** |  | |
| **Description** | A Test is compiled with randomly generated questions of one of three levels**.** | |
| **Preconditions** | Student must be registered with the system. | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The student invokes the Take Test function.  **Step 5:** The student selects the test level.  **Step 6:** The student ‘Starts Test’.  **Step 14:** The student answers the question.  **Step 15:** The student clicks ‘Next Question’.  **Step 21:** The student submits the test. | **Step 2:** The system retrieves the next available Test Id.  **Step 3:** The system retrieves test level data from the **Level File.**  **Step 3:** The system loads the levels.  **Step 4:** The system displays the Take Test UI.  **Step 7:** The system counts the questions of the selected level.  **Step 8:** The system retrieves all the question ids of the selected level.  **Step 9:** The system generates a random number within the range of question id’s picks **Question Id’s** of the correct level.  **Step 10:** The system checks the number is not repeated.  **Step 11:** The system saves the question id.  **Step 12:** Repeat from **Step 9** until 4 question ids have been saved.  **Step 13:** The system loads a question.  **Step 16:** The system saves the question answer.  **Step 17:** The system compares the answer given with the correct answer.  **Step 18:** If the answer given is correct the system increments the score.  **Step 19:** The system increments the progress bar.  **Step 20:** Repeat from **Step 13** until 4 questions have been answered.  **Step 22:** Repeat from **Step 16** to **Step 19.**  **Step 23:** The system displays the test score.  **Step 24:** The system sets current system date as TestDate.  **Step 25:** The system saves the test data in the **Test File**   * **Test Id** * **Date** * **Score** * **Student Id** * **Level**   **Step 26:** The system saves the test question data in the **Test Question File**   * **Test Id** * **Question Id** * **Answer Given**   **Step 27:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Question is not found** |  | **Step 6:** The systemfails to retrieves the question from the **Question File**.  **Step 7:** The systemsdisplays an error message.  **Step 8:** The systemresets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid entry** |  | **Step 10:** The system identifies an invalid entry.  **Step 11:** The system displays an error message.  **Step 12:** The system prompts the student to re-enter the details. |
| **Conclusions** | A test is completed and the details saved in the **Test File** and the **Test Question File**. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Display Student Profile

The review student profile function displays a table of all the students test result for review.

## Display Student Profile Use Case

User

Student

<<include>>

## Display Student Test Profile Use Case Narrative

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Display Student Test Profile** | |
| **Use Case Id** | 10 | |
| **Priority** | 1 | |
| **Source** | User | |
| **Primary Business Actor** | Student | |
| **Other Participating Actors** | Lecturer | |
| **Description** | The Student Test Profile results are displayed**.** | |
| **Preconditions** | Student must be registered with the system | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The student invokes the Student Test Profile function.  **Step 4:** The student reviews a table of all their test results.  **Step 5:** The student exits the function. | **Step 2:** The system retrieves the details for all tests for that student from the **Test File**.   * Test Id * Date Taken * Score * Test Level   **Step 3:** The system displays all the students test results in a table**.**  **Step 6:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Profile reviewed by Lecturer** | **Step 1:** The lecturer invokes the Student Profile function.  **Step 3:** The lecturerenters either the **Students Id** or **Students Surname**.  **Step 5:** The lecturer selects the specific student.  **Step 8:** The lecturer reviews a table of all the students test results. | **Step 2:** The system displays the Student Profile UI.  **Step 4:** The system displays either the student by student id or a list of students by surname.  **Step 6:** The system displays all the test results for that student.  **Step 7:** The system retrieves the test details from the **Test File**.   * Test Id * Date Taken * Score * Test Level |

|  |  |  |
| --- | --- | --- |
|  | **Step 9:** The lecturer exits the function. | **Step 10:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Student Profile not found during student profile review** |  | **Step 3:** The systemsfails to retrieve the tests from the **Test File**.  **Step 4:** The systemsdisplays and error message.  **Step 5:** The systemsoffers a choice of Student Profile, Main Menu or Exit System. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Student Profile not found during lecturer profile review** |  | **Step 6:** The systemsfails to retrieve the tests from the **Test File**.  **Step 7:** The systemsdisplays and error message.  **Step 8:** The systemsoffers a choice of Student Profile, Main Menu or Exit System. |
| **Conclusions** | A Student Test Profile is provided for review. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# System Model

### External Entities:

Student

### Data Stores:

D1: Student File

D2: Lecturer File

D3: Level File

D4: Question File

D5: Test Questions File

D6: Test File

### **Processes:**

P1: Login

P2: Registrations

P1.1: Register Student

P1.2: Amend Student

P1.3: Delete Student

P3: Questions

P2.1: Create Question

P2.2: Amend Question

P2.3: Delete Question

P4:Tests

P3.1: Compile Test

P3.2: Review Student Profile

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

Student Details

Student

Tutorial

Performance

Test System

Test Results

## Level-1 DFD

Test Taken

Student

Student Details

Student

Details

Student Details

D1

Students File

Student Details

Test Details

**Process**

**Tests**

**P4**

**Process**

**Students**

**P2**

D6

Test File

Test Result

Student Details

D5

Test Question File

Question Level

D3

Level File

Questions

**Login**

**P1**

Question Level

**Process**

**Questions**

**P3**

D4

Question File

## Level-2 DFD (Process P1: Login)

Question Details

## Level-2 DFD (Process P2: Process Students)

Student Details

Student Details

**Login**

**P1**

Student

Student

Amended Details

Student Details

**Amend Student**

**Details**

**P2.2**

**Register**

**Student**

**P2.1**

Student Details

Student Details

D1

Students File

Amended Details

Student Status

Student Details

­

Password

**Delete**

**Student**

**P2.3**

## Level-2 DFD (Process P3: Process Questions)

**Create**

**Question**

**P3.1**

**Amend Question**

**P3.2**

Question Details

Question Details

D2

Question File

Amended Details

Question Status

Question Details

­

**Delete Question**

**P3.3**

## Level-2 DFD (Process P4: Process Tests)

Questions

D4

Question File

Question Details

D1

Students File

Student Details

Student Details

**Take**

**Test**

**P4.1**

**Student**

**Test Profile**

**P4.2**

Request Profile

Student

Request Test

Levels

D3

Level File

Test Details

Test Details

D6

Test File

Answers

D5

Test Question File

# Data Model (Class Diagram)

1

**Level**

Level

Description

1

1

**Lecturer**

LecId

Passwd

Sname

Fname

reviews a

has a

has a

0..\*

0..\*

0..\*

**Question**

QuestId

Text

Ans1

Ans2

Ans3

Ans4

CorrectAns

QAdd

QAmd

Status (‘A’,’I’)

**Test**

TestId

TestDate

Score

1

1

0..\*

**Student**

StudId

Email

Passwd

Sname

Fname

DOB

Status (‘R’ / ‘D’)

RegDate

AmdDate

ExpDate

takes a

belongs to

has

4..4

1

**Test Question**

AnsGiven

0..\*

# Database Design

**7.1. Relational Schema**

**Students** (StudId,Email, Passwd, Sname, Fname, DOB, Status, RegDate, AmdDate, ExpDate)

**Lecturers** (LecId,Passwd, Sname, Fname)

**Levels** (LevelCode, Description)

**Questions** (QuestId, LevelCode, Text, Ans1, Ans2, Ans3, Ans4, CorrectAns, QAdd, QAmd, Status)

**Tests** (TestId, TestDate, Score, StudId, LeveCode)

**TestQuestions** (TestId, QuestId, AnsGiven)

**7.2.** Database Schema

**Relation** **Students**

StudId numeric(4)

Email char(30),

Password char(8)

Sname char(20)

Fname char(20)

DOB date

Status char(1) DEFAULT ‘r’ (CHECK Status = ‘r’ OR Status = ‘d’)

RegDate date (RegDate = System Date)

AmdDate date (AmdDate = System Date)

ExpDate date (ExpDate = System Date + 13 weeks)

**Primary Key:** StudId

**Relation** **Lecturers**

­LecId numeric(4)

Sname char(20) NOT NULL

Fname char(20)

Passwd char(8)

**Primary Key:** LecId

**Relation Levels**

LevelCode char(1)

Description char(15)

**Primary Key:** LevelCode

**Relation** **Questions**

QuestId numeric(4)

LevelCode char(1)

Text char(100)

Ans1 char(50)

Ans2 char(50)

Ans3 char(50)

Ans4 char(50)

CorrectAns numeric(1)

QAdd date (QAdd = System Date)

QAmd date (QAmd = System Date)

Status char(1) CHECK (Status = ‘a’ OR Status = ‘d’)

**Primary Key:** QuestId

**Foreign Key:** LevelCode References Levels

**Relation Tests**

TestId numeric(4)

TestDate date (TestDate = System Date)

Score numeric(3)

StudId numeric(4)

**Primary Key:** TestId

**Foreign Key:** StudId References Students

**Foreign Key:** LevelCode References Levels

**Relation TestQuestions**

TestId numeric(4)

QuestionId numeric(4)

AnsGiven numeric(1)

**Primary Key:** TestId, QuestionId

**Foreign Key:** TestId References Tests

**Foreign Key:** QuestId References Questions

# Conclusion

This document covers the requirements specified by Institute of Technology Tralee. An agile approach has been implemented.

The Tutorial Test Performance System enables students to register their details in the system in order to take a test. Should their information change they can update their details. A college lecturer is also able to update student details and if necessary, following disciplinary action, a student can be deregistered from the system.

Students can take a test to evaluate their knowledge of the module syllabus and also review their test results with the Student Test Profile.

Lecturers for the module can create, amend and delete questions from the database. They are also able to review a student’s progress with the Student Test Profile function.

The Institute of Technology Tralee will be given the recommendation to implement the Tutorial Performance Test System for all modules in the college curriculum.

Due to time constraints the lecturers have been hard coded into the system but I would have preferred to include an add lecturer function to increase functionality.

To improve the system, recommendations would include:-

* Timed tests to better simulate exam conditions.
* Lecturers Review module test results.
* The faculty should be able to compile statistics of the test results.
* Compile statistics of questions incorrectly answered to discern gaps in student knowledge.

# Appendices

### [Appendix A - Actor Glossary](#_For_Actor_Glossary)

The actor glossary is as follows:-

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Term** | **Synonym** | **Description** |
| **1** | Student | Test Taker | Student enrolled in the college, taking the module and registered with the system. |
| **2** | Lecturer |  | Employee of the college overseeing Lecturer of the system. |
| **3** | Lecturer | Faculty | Lecturer employed by the college and responsible for the module course material and teaching the module. |

### [Appendix B – Use Case Glossary](#_For_Use_Case)

The use case glossary is as follows:-

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Use Case Description** | **Participating Actors & Roles** |
| Register Student | This function provides the student with a form to enter their details, once correctly filled in and submitted. The student’s details are saved in the **Student** **File**. The student is then registered and able to use the system. | * Student |
| Amend Student | This function provides the user with a form to enter their amendments, once correctly filled in and submitted the students details are updated in the **Student** **File**. | * Student * Lecturer |
| Delete Student | This function provides Lecturer with an option to deregister a student. The student’s status is updated to ‘d’ in the **Student** **File**. | * Lecturer |
| Create Question | This function provides a lecturer with a form to create a question and saves it to the **Question File**. | * Lecturer |
| Amend Question | This function provides a lecturer with form to amend a question and update it in the **Question File**. | * Lecturer |
| Delete Question | This function provides a lecturer with an option to delete a question and update it to the question status to ‘d’ in the **Question File**. | * Lecturer |
| Compile Test | This function compiles a test of randomly generated questions, from a selected category and a selected level. The system calculates the score and saves the result in the **Test Question File**. | * Student |
| Student Test Profile | This function provides the user with an option to review a profile of test results. | * Student * Lecturer |