**Western Institute of Technology**

**The Setting**

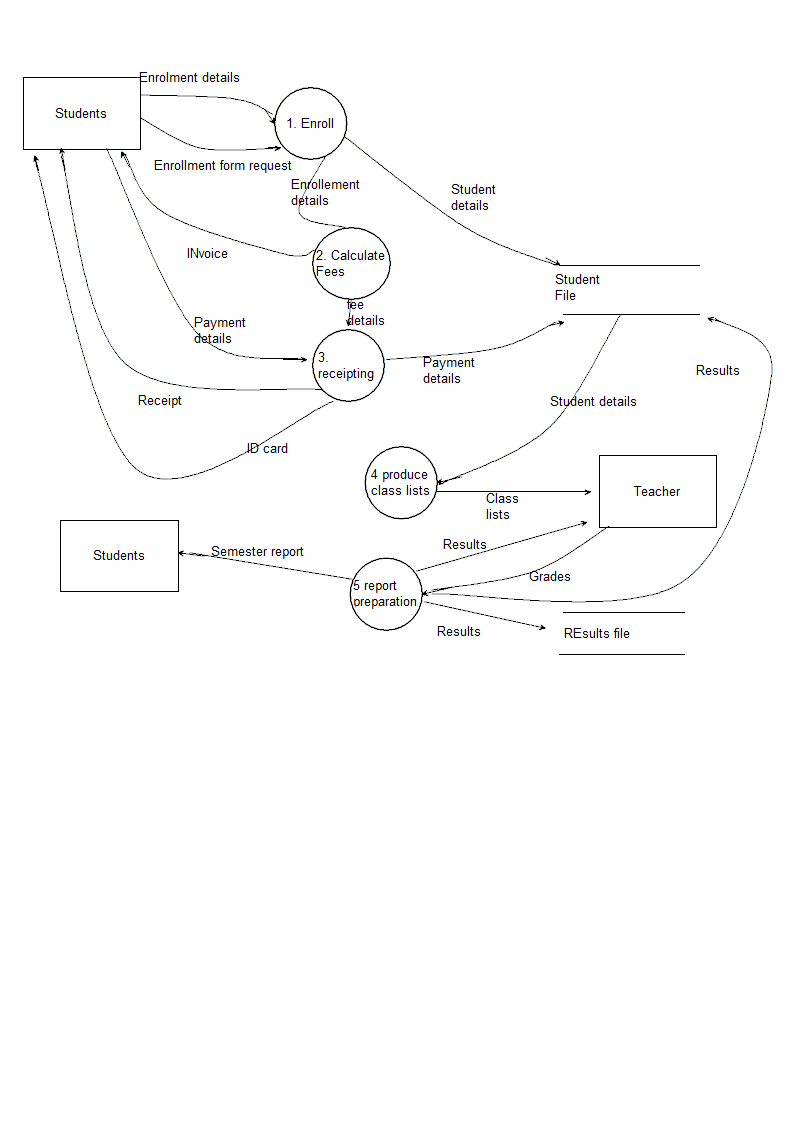
The Western Institute of Technology (WIT) is a small registered training organisation (RTO) in the western suburbs of Melbourne. They cater for local and international students wishing to study VET courses as well as receive recognition of prior learning (RPL) for previous work and study experience. Students can study full time, part time or even online.

There is one managing director, four office administrators and four mobile enrolment staff.

The WIT office looks after the student administration side of the business. The four mobile enrolment staff visit prospective students at their workplaces if they are unable to come into the office to enroll. Classes are conducted through various local secondary schools after hours and several other RTOs with classrooms.

**Current Practices**

After students at WIT enroll, administration staff in the WIT office calculate the enrolment fees owed by the students and then email their enrolment advice/invoices to the student’s as per the dfd below.



**The proposed Fee Payment System**

Students pay their fees at the WIT office and the administration staff gives them a receipt of payment. At this point students will receive an identification card, which contains their student number.

As part of the enrollment process, you are to write a module to calculate course fees for each student. The information to calculate course fees is contained in course files.

The course files used to calculate fees contain the student’s unique 5 digit ID number, their first and last name and a field indicating whether or not they have paid their fees for that course. The first line of the file contains the course code, course name and whether the course is Full Time (FT) , Part Time (PT) or Online.( OL)

A sample of the course files are below.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 | Commerce | FT |  |  | 200 | Cooking | PT |  |
| 23803 | Damen | Ray | true |  | 12535 | Daniels | Lou | false |
| 12535 | Daniels | Lou | false |  | 13610 | Grech | Kerrie | true |
| 20214 | Davidson | Kerrie | false |  | 17744 | Green | Tobin | false |
| 23805 | Davie | Laura | false |  | 17029 | Egglestone | Kim | true |
| 18356 | Beaton | Paul | true |  | 10105 | Guala | Jan | false |
| 18064 | Deller | Zoe | true |  | 19965 | Dickson | Lindsey | false |
| 19864 | Di Mieri | Andrew | false |  | 10124 | Hamilton | Stephanie | false |
| 19965 | Dickson | Lindsey | false |  | 18356 | Beaton | Paul | false |
| 20197 | Drape | Christine | true |  | 10146 | Hatfield | Erin | false |
| 17029 | Egglestone | Kim | false |  |  | | | |

Course fees per semester are

Full time $2500

Part time $1500

Online $1000

Students may enroll in up to 5 courses, but because this is a prototype solution you only need to consider two courses. The course names are stored in the file because next semester the courses might change and could be Art or Literacy for example.

Only students who have not paid their fees should be charged.

The program should produce a list on screen of all students in a particular course ( user selected). You should give the user a choice of listing **all** students in the course, students who have **not** paid their course fees or students who **have** paid their fees. It should contain the student’s ID, first and last name and amount of fees owing for that course. If they have paid their fees, their fees owing will be $0.

.

A student should be able to pay their fees owing for a particular course. Your program should be able enter a student ID number and by selecting the course they are attending, display the fees owing for that course and give them the option to pay their fees. The payment field should then be marked as true and the course list on the screen updated. The updated course files will then need to be saved back to disk. You should also create a new file called payments.txt which records the students ID, the date of payment, the course ID and the amount paid. Payments should be added to this file, not overwrite old payments. This file will then be used by another program to produce the receipts for the students. You do not have to produce the receipting program.

Functional requirements

* Read the course files and store the data in appropriate data structures (3)
* Calculate the course fees for the students in each course based on whether they have already paid or not and the type of course it is. (3)
* Display either all students in a course, those who have paid or those who haven’t paid.( 3)
* Allow students to pay their fees for a course. (5)
* Update the course file showing students who have now paid. (3)
* Save the fee payments to a text file. (3)

**Non-functional requirements**

* All required information is displayed in a clear and logical fashion
* The program should be easy to use with a logical flow of input and output.
* The layout of the screen should be clear with appropriate labelling.

**Constraints**

It is anticipated that at some stage lecturers will be able to access fee information on their mobile devices, so that students who have not paid their fees by a certain date will not be allowed to attend classes.

* The software solution will be running on a mobile computing device yet to be determined. There are a number of candidates – all are tablets capable of running at least a screen size of 800 x 600 pixels. For this reason, the interface should be confined to a screen size of 800 x 600 (landscape) and should be designed with touch interaction in mind.
* All input will be via touch (as this is a prototype, single mouse clicks will simulate touches of the display). An onscreen keyboard will allow data to be entered on the tablet.
* Storage space on the device will be limited, so the software solution should take this into account.

**Scope**

* The WIT could have up to 50 courses, but in this prototype you will only be required to handle two courses.
* Courses can contain up to 25 students, but once again you only need to have enough students in each course to be able to fully test your solution.
* Your solution does not need to take security, such as authentication into account.
* You can assume all data provided has been validated, so your solution does not have to validate any input.
* You do not have to produce the programs which create the course files or the receipts for student payments.

(Logo: http://www.westerniot.com/index.html)

**The Task**

Represent a software design and apply a range of functions and techniques using a programming language to develop a prototype solution to meet a specific need.

**The design section( tasks 1,2 and 3) must be completed before programming commences. It must be submitted for marking no later than the end of the third period.**

**Complete the following:**

1. Create a data dictionary to describe the format of data in the text file/s that will be read in and produced by the software solution. If files have duplicate designs, only one file needs to be designed.
2. Identify evaluation criteria appropriate to the solution requirements
3. Construct an algorithm to represent the operation of the software solution.
4. Construct a prototype software solution that meets the specifications in the SRS, ensuring that the prototype solution:

* contains efficient and effective programming techniques.
* contains efficient and effective use of data types and data structures.
* has intuitive and consistent navigation.
* contains clear and appropriate internal documentation.
* has a logical and attractive user interface.
* produces the correct output.

1. Create a testing table that contains a number of tests, the test data used, the expected outcome of each test and the actual results of the tests. The tests should cover all functions of the solution. If your final solution is incomplete, or not working, identify the testing tasks that you would use, test data and the expected outcomes.

**Scope of Task**

**Dates:**  21/5-30/5 **Time allowed: 8 periods.**

**Resources:** You may refer to class notes, programs, your text book and practice SACs. You may **NOT** use the internet or email during the SAC, nor seek or give assistance to any other student.

**Authentication:** No SAC work is to be undertaken out of class time. The teacher must be able to authenticate that all work submitted belongs to that student.

**Saving work:** All SAC work is to be saved on the S: drive coursework SD folder. No work is to be saved on your USB or H: drive. Make sure you save regularly throughout the SAC. You should set up the saving options in VB to point to S: drive.

**Marks:** This SAC is worth 60% of Unit 3 marks.