BIT502 *Fundamentals of Programming*

BIT502

Assessment

2

Assessment 2

Weighting

30%

Learning outcomes

1 Create simple applications using fundamental programming logic, constructs and mathematical operations to solve business problems.

3 Utilise a range of software tools commonly used during the software development life cycle to create, test and document simple working code.

Instructions

Complete and submit your assessment according to the Open Polytechnic's [Assessments web page](https://www.openpolytechnic.ac.nz/current-students/assessments/). This includes information on academic integrity, word limits and referencing.

* Include your name, student number and the assessment number.
* Number your pages.

Submission

* Submit your assessment in two files. The code file should be submitted as a zip file.
* Submit your work through your iQualify course.
* Emailed assessments will not be accepted.
* You will receive an automated notice following submission.

**By submitting your assessment, you confirm that it is your own, original work.**

Introduction

Read the following case study and complete the tasks. Refer to the marking schedule for additional information about the tasks.

You need to submit your assessment in two files: the report as an MS Word file and the Python code file as a zip file.

Case study

City Gym has been in business for two years and is seeing a steady increase in the number of members. The gym offers its members access to a standard range of exercise equipment and personalised training.

New members have an initial fitness assessment where their personal fitness goals are set (for example, weight loss, toning, bulking up, improving cardio), and soon after, a personal trainer takes them through their personalised fitness programme.

City Gym also provides online fitness videos that its clients can access. As the number of members has increased, it has become more difficult to keep track of client information. Currently, the owners have a rather old-fashioned approach to signing up new members. A paper membership form would be provided to the customer (see Appendix A), and the employee would calculate the fees manually using a calculator. The customer would then be given one copy, and another copy would be stored in a filing cabinet.

City Gym is looking to purchase new computers for its employees and is interested in pursuing a digital approach that will allow new members to sign up through the new computer application. The new application would allow the user to enter all their personal details as required from their current membership form (refer to City Gym membership form) and calculate their membership costs.

Your application should have the following:

* This should only be a single-page Python application.
  + The user should be able to enter all details that are contained in the paper membership form (Appendix A). No details should be omitted.
* There should be a calculate button that provides the user with:
* the base cost of their membership
* the total amount of extra charges
* the total discount
* the net membership cost
* the regular payment amount (based on payment frequency).
* There should be a submit button to save the data into a text file (.txt).

**Sample calculation**

Refer to the end of this document for sample calculations.

**Rules brief**

For all the membership rules and values, refer to the sample paper form in Appendix A.

Task 1: Create the gym membership user interface (UI)

Build a GUI application for City Gym using a GUI library for Python. This application will have a single screen: the member registration form. This is the screen for which you created a prototype in Assessment 1, Task 2.

The membership form will calculate the membership costs and write the new membership details in a text file, as outlined in the client brief.

It should include the following information.

Part 1: User interface

Create the interface by using the GUI library. It should include all items in the City Gym’s paper form. Apply principles of good UI practice.

**(Part 1: 8 marks)**

Part 2: Coding

1. The main coding will be included inside the calculate button and must perform the following:

* calculating the base membership cost
* calculating the total extras as selected by the user (for example, diet)
* calculating the duration discount
* calculating the payment type discount
* calculating the total membership cost
* calculating the regular payment amount based on the weekly or monthly selection.

(36 marks)

1. The main coding must display all the above calculations.

(10 marks)

1. The main coding should include data validation/error checking. The following criteria must be considered:

* No text field should be empty.
* No radio button component should go unselected.
* The first and last name fields should not contain any number.

(15 marks)

1. There must be a code associated with the submit button. The main purpose of this code is to push all the data (that is, all the calculation values and the text fields) and save it to a text file. A sample of the data inside a text file is given at the end of this assessment.   
   The following points need to be considered when coding this part:

* opening a connection to the text file
* saving the data into the text file
* closing a connection to a text file
* not overriding the previously saved data inside the text file
* error-checking/data validation by making sure all the data is saved and to avoid saving missing data.

(10 marks)

1. Provide appropriate comments within your code. Apply principles of good practice for writing comments.

(6 marks)

**(Part 2: 77 marks)**

[Task 1 total: 85 marks]

**Checklist**

* Make sure you have included all the components and features required in the client brief.
* You can enhance the user experience for the GUI by selecting the appropriate colours and positioning elements.
* Name the file for saving data **membersdatasaved.txt**. Note that the path for saving this text file should be C:\Temp.

Task 2: Create gym membership

Part 1: Test plan

Once you have completed the application, you will need to demonstrate how you conducted the user acceptance testing.

To do this:

* add eight test items to the test plan and submit it as a Word document or pdf file. The test plan must be a table and build on the user acceptance testing by using the following steps:
* In the first column of the table, list the test item.
* Place the expected results in the second column.
* Place the actual output in the third column and comments and actions required in the last column.
* Make sure the test plan covers all the requirements and acceptance criteria given in the case study.

**(Part 1: 8 marks)**

**Part 2: Applying the steps in the software development lifecycle**

Describe the steps you have taken to ensure you have provided the client with a viable product. The response should include:

* a brief description of each phase of the software development process, outlining the key decisions that were made at the design, implementation and testing stages of this application
* an explanation of how you have made the application as easy to use as possible (you may refer to the user experience testing results here).

*[Word count 100–400]*

**(Part 2: 7 marks)**

**[Task 2 total: 15 marks]**

**[Assessment 2 total: 100 marks]**

SAMPLE CALCULATIONS

**Scenario 1**

Membership type: basic

Duration: three months

Extras: 24/7, diet consultations

Payment method: bank

Frequency: weekly﻿﻿

﻿

**Calculation 1**

﻿Base membership cost: **$10** (for basic type)

﻿

Extra charges: $1 + $20 = **$21** (for the extras above)

﻿

Total discount: **$0.1** (see the explanation below)

$0 (because it is a three-month membership) +

(1 × 10/100) (because it is a bank payment, hence 1% on the base membership). So, the result is $0.1.

Net membership cost: **$30.9** (see the explanation below)

This equals base membership cost + extra – total discount

$10 + $21 – $0.1 = $30.9

Regular payment amount: **$30.9** (see the justification below)

Because this is a weekly payment, this cost will always be equal to the net membership cost.

**Scenario 2**

Membership type: premium

Duration: three months

Extras: 24/7, personal trainer, diet consultations, fitness video

Payment method: credit

Frequency: monthly

﻿

**Calculation 2**

﻿Base membership cost: **$20** (for premium type)

﻿

Extra charges: $1 + $20 + $20 + $2 = **$43** (for the extras above)

﻿

Total discount: **$0** (see the explanation below)

It is premium, but because it is three months, there is no discount. Also, there is no 1% discount as the payment method is through credit card.

Net membership cost: **$63** (see the explanation below)

This equals base membership cost + extra – total discount

$20 + $43 – $0 = $63

Regular payment amount: **$252** (see the justification below)

Because this is a monthly payment, this cost will be equal to (net membership cost × 4). So, $63 × 4 = $252.

**Scenario 3**

Membership type: regular

Duration: 12 months

Extras: 24/7, fitness video

Payment method: bank

Frequency: monthly

﻿

**Calculation 3**

﻿Membership cost: **$15** (for regular type)

﻿

Extra charges: $1 + $2 = **$3** (for the extras above)

﻿

Total discount: **$2.15** (see the explanation below)

Because the membership is for 12 months, there is a discount of $2. Also, there is a 1% discount as the payment method is through the bank. So, adding both discounts will give you the total discount.

$2 (discount 1) + 1 × 15/100 (discount 2) = $2 + $0.15 = $2.15

Net membership cost: **$15.85** (see the explanation below)

This equals base membership cost + extra – total discount

$15 + $3 – $2.15 = $15.85

Regular payment amount: **$63.4** (see the justification below)

Because this is a monthly payment, this cost will be equal to (net membership cost × 4). So, $15.85 × 4 = $63.4.

SAMPLE DATA FOR TEXT FILE

**Text

Description automatically generated**

Marking schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TASK 1: Create the gym membership UI** | | | | | |
| **Part 1** | **7–8** | **6–6** | **4–5** | **3–3** | **1–2** |
| **User interface** | User interface produced using GUI library. All items included, with no errors present. Good user-interface practice is applied. | User interface produced using GUI library. All items included, with no errors present. Good user-interface practice is applied with minor inaccuracies. | User interface produced using GUI library. All items included, with minor errors present. Good user-interface practice is applied with inaccuracies. | User interface produced using GUI library. All items included, with significant errors present. Good user-interface practice is applied with inaccuracies. | User interface produced using GUI library. Not all items included with significant errors present. |
| **Part 2a** | **33–36** | **25–32** | **18–24** | **10–17** | **1–9** |
| **Main coding: Calculations** | Coding has been included inside the calculate button.  All required calculations are present, functional and accurate. | Coding has been included inside the calculate button.  All required calculations are present and functional.  Minor errors in coding are present. | Coding has been included inside the calculate button.  All required calculations are present and functional.  Minor errors in coding and functionality are present. | Coding has been included inside the calculate button.  All required calculations are present.  Significant errors in coding and functionality are present. | Coding has been included inside the calculate button.  Not all required calculations are present.  Significant errors in coding and functionality are present. |
| **Part 2b** | **9–10** | **7–8** | **5–6** | **3–4** | **1–2** |
| **Display the calculations** | All calculations are accurately displayed in the code. | All calculations are displayed in the code. Few errors in the display exist. | All calculations are displayed in the code. Minor errors in the display exist. | All calculations are displayed in the code. Significant errors in the display exist. | Not all calculations are displayed in the code. |
| **Part 2c** | **14–15** | **11–13** | **8–10** | **5–7** | **1–4** |
| **Error-checking /data validation** | The process and the code for error-checking/data validation are thorough and well-executed without any errors. | The process and the code for error-checking/data validation are complete and cover most of the elements. Executed with a few errors. | The process and the code for error-checking/data validation are mostly complete and cover most of the elements. Executed with some errors. | The process and the code for error-checking/data validation are partially complete and cover only some of the elements.  Executed with many errors. | The process and the code for error-checking/data validation are token and cover very few or none of the elements. |
| **Part 2d** | **9–10** | **7–8** | **5–6** | **3–4** | **1–2** |
| **Text file** | Submit button has been coded. No errors in coding and functionality exist.  Data saves correctly. | Submit button has been coded. No errors in coding and functionality exist.  Minor data-saving issues are present. | Submit button has been coded. Minor errors in coding and functionality exist.  Missing data is saved. | Submit button has been coded. Significant errors in coding and functionality exist.  Some data is saved. | Submit button has been coded. Significant errors in coding and functionality exist.  No data is saved. |
| **Part 2e** | **5–6** | **4–4.5** | **3–3.5** | **2.5–2.5** | **1–2** |
| **Comments** | All lines of code are readable and well-maintained by comments so a non-programmer can understand what each line of code in the program does and the intentions behind it. | 75% of lines of code are readable and well-maintained by comments so a non-programmer can understand what each line of code does. | 50% of lines of code are readable and well-maintained by comments so a non-programmer can understand what each line of code does. | 25% of lines of code are readable and well-maintained by comments so a non-programmer can understand what each line of code does. | None of the lines of code are readable and well-maintained by comments so a non-programmer can understand what each line of code does. |
| **TASK 2: Create the gym membership** | | | | | |
| **Part 1** | **7–8** | **6–6** | **4–5** | **3–3** | **1–2** |
| **Test plan** | Evidence of user acceptance testing provided.  No errors in structure present.  Test plan accurately covers all requirements. | Evidence of user acceptance testing provided.  No errors in structure present.  Test plan covers all requirements. Some ambiguity in how some requirements are covered exists. | Evidence of user acceptance testing provided.  Minor errors in structure present.  Test plan covers all requirements. Some ambiguity in how some requirements are covered exists. | Evidence of user acceptance testing provided.  Significant errors in structure present.  Test plan covers all requirements. Some ambiguity in how all requirements are covered exists. | Evidence of user acceptance testing provided.  Significant errors in structure present.  Test plan does not cover all requirements. |
| **Part 2** | **7–7** | **5–6** | **4–4** | **3–3** | **1–2** |
| **Applying the steps in the software development life cycle** | Description of steps to ensure the production of a viable product.  Phases of software development accurately provided.  Explanation as to how the application was designed to be easy to use is provided. Explanation is justified with reference and comparison to accessibility principles. | Description of steps to ensure the production of a viable product.  Phases of software development accurately provided.  Explanation as to how the application was designed to be easy to use is provided. | Description of steps to ensure the production of a viable product.  Phases of software development provided with minor inaccuracies.  Explanation as to how the application was designed to be easy to use is provided, but ambiguity around main points exists. | Description of steps to ensure the production of a viable product.  Phases of software development provided with significant inaccuracies.  Explanation as to how the application was designed to be easy to use is provided, but ambiguity around main points exists. | Description of steps to ensure the production of a viable product.  Phases of software development provided with significant inaccuracies.  Explanation as to how the application was designed to be easy to use is not provided. |