# COMP1811 – Python Project Report

| Name | MD MOSTAFIZUR RAHMAN BADHON | Student ID | 001134460 |
|---|---|---|---|
| Partner's name | | Partner SIDs | |

## 1. Brief statement of features you have completed

*(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)*
***Indicate the feature each partner implemented by replacing "developed by" in red below with partner name.***

| 1.1 Circle the parts of the coursework you have **fully completed and are fully working**. Please be accurate. | **Features**<br>***[Developed by]*** **F1:** i ☒ ii ☐ iii ☒<br>***[Developed by]*** **F2:** i ☐ ii ☐ iii ☐ |
|---|---|
| 1.2 Circle the parts of the coursework you have **partly completed or are partly working.** | **Features**<br>**F1:** i ☐ ii ☒ iii ☐<br>**F2:** i ☐ ii ☐ iii ☐ |
| Briefly explain your answer if you circled any parts in 1.2 | |

# 2. Concise List of Bugs and Weaknesses

*A concise list of bugs and/or weaknesses in your work (if you don't think there are any, then say so). Bugs that are declared in this list will lose you fewer marks than ones that you don't declare! (**100-200 word**, but word count depends heavily on the number of bugs and weaknesses identified.)*
*(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)*

## 2.1 Bugs

While managing or deleting the files, all the data present in the file is complete removed to again reuse the past data, it needs to manually copy externally in SQLITE3 format to be able to work properly in the solution.

## 2.2 Weaknesses

The data is stored in the text file formatting manner. However, the data format stored in the text file is that of the SQLITE3 file format. It was done for the ease of question handling in MCQ quiz sections.

## 3. DESCRIPTION OF THE FEATURES IMPLEMENTED

- The user will be able to use the quiz in beautifully structured GUI interface
- The user should be able to add or manage new modules

# 4. CLASSES AND OOP FEATURES

*List all the classes used in your program and include the attributes and behaviours for each. You may use a class diagram to illustrate these classes. Your narrative for section 3.2 should describe the design decisions you made and the OOP techniques used. Each partner must list the classes they developed separately and provide an exposition on the choice of classes, class design and OOP features implemented. (**200-400 words for each partner**). (THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)*

## 3.1 CLASSES USED

In the current section, two classes were primarily used They are class Module and Question.

## 3.2 BRIEF EXPLANATION OF CLASS DESIGN AND OOP FEATURES USED

Class question is used to work with the question genarte  section of the project where the program will note the question of each quiz in time stamped manner.

Class Module will handle all the functions of the project including GUI implementation, and handling the following features:

- add or manage new modules
- Update and view current module result data
- exit from the GUI interface by selecting quit button

# 5. CODE FOR THE CLASSES CREATED

*Add the **code for each of the classes you have implemented yourself** here. If you have contributed to parts of classes, please highlight those parts in a different colour. Copy and paste relevant code - actual code please, no screenshots! Make it easy for the tutor to read. Add explanation if necessary – though your in-code comments should be clear enough. You will lose marks if screenshots are provided instead of code.*
*(COMPLETE THIS SECTION INDIVIDUALLY – only list the code for the classes you developed individually. DO NOT provide a listing of the entire code. You will be marked down if a full code listing is provided.)*

***CLASS …***

```
import tkinter as tk
from tkinter import ttk


class StartPage(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent, bg='green')

        self.controller = controller
        # To paint the modules.
        self.button_F1 = tk.Button(self, text='F1', command=lambda:
self.controller.display_frame(ModulePage))
        self.button_F1.grid(column=0, row=0, sticky=tk.NSEW, padx=10, pady=10)
        self.rowconfigure(0, weight=1)
        self.columnconfigure(0, weight=1)

        self.button_F2 = tk.Button(self, text='F2', command=lambda: print("pressed
F2"))
        self.button_F2.grid(column=1, row=0, sticky=tk.NSEW, padx=10, pady=10)
        self.rowconfigure(0, weight=1)
        self.columnconfigure(1, weight=1)


class ModulePage(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent, bg='red')
        self.controller = controller

        # inform label
        self.title_label = tk.Label(self, text='AVAILABLE MODULES FOR QUIZ')
        self.title_label.grid(row=0, column=1, columnspan=3, sticky=tk.NSEW,
padx=10, pady=10)
        self.rowconfigure(0, weight=1)
        self.columnconfigure(1, weight=1)

        self.up_buttom = tk.Button(self, text='MODULES', command=lambda:
controller.show_frame(StartPage))
        self.up_buttom.grid(row=0, column=0, sticky='nswe', padx=10, pady=10)
        self.rowconfigure(0, weight=1)
        self.columnconfigure(0, weight=1)

        # using a treeview
        # 4 line protocol
        self.treeview = ttk.Treeview(self, selectmode='browse', columns=('NAME',
'DESCRIPTION'), show='headings',
                                     style="mystyle.Treeview")
        self.treeview.grid(row=1, column=0, columnspan=4, sticky=tk.NSEW, padx=10,
```

```
pady=10)
        self.rowconfigure(1, weight=5)
        self.columnconfigure(0, weight=1)

        self.treeview.heading('NAME', text='NAME')
        self.treeview.heading('DESCRIPTION', text='DESCRIPTION')

        #  binding module_selected
        self.treeview.bind('<<TreeviewSelect>>', self.treeview_select)
        self.treeview.bind('<Double-1>', self.treeview_double_1)

        # using an scroll bar
        # 4 line protocol
        scrollbar = tk.Scrollbar(self, orient=tk.VERTICAL,
command=self.treeview.yview)
        scrollbar.grid(row=1, column=4, sticky=tk.NSEW)
        self.rowconfigure(1, weight=25)
        self.columnconfigure(1, weight=1)

        # linking
        self.treeview.configure(yscroll=scrollbar.set)

        name_label = tk.Label(self, text=" Name :")
        name_label.grid(row=2, column=0, sticky=tk.NSEW, padx=10, pady=10)
        self.rowconfigure(2, weight=1)
        self.columnconfigure(0, weight=1)

        self.name_entry = tk.Entry(self, textvariable=tk.StringVar())
        self.name_entry.grid(row=2, column=1, sticky=tk.NSEW, padx=10, pady=10)
        self.rowconfigure(2, weight=1)
        self.columnconfigure(1, weight=1)

        description_label = tk.Label(self, text=" Description :")
        description_label.grid(row=2, column=2, sticky=tk.NSEW, padx=10, pady=10)
        self.rowconfigure(2, weight=1)
        self.columnconfigure(2, weight=1)

        self.description_entry = tk.Entry(self, textvariable=tk.StringVar())
        self.description_entry.grid(row=2, column=3, sticky=tk.NSEW, padx=10,
pady=10)
        self.rowconfigure(2, weight=1)
        self.columnconfigure(3, weight=1)

        # Buttonery for command actions contained in an auxiliar pannel.
        self.button_Frame = tk.Frame(self, bg='lightblue')
        self.button_Frame.grid(row=1, column=5, columnspan=1, rowspan=1,
sticky=tk.NSEW, padx=10, pady=10)
        self.rowconfigure(0, weight=1)
        self.columnconfigure(5, weight=1)

        # self.rowconfigure(0, weight=1)
        # self.columnconfigure(6, weight=1)

        add_button = tk.Button(self.button_Frame, bg='cyan', text="Add",
command=lambda: self.add_row())
        add_button.grid(row=0, column=0, sticky=tk.NSEW, padx=10, pady=10)
        self.button_Frame.rowconfigure(0, weight=1)
        self.button_Frame.columnconfigure(0, weight=1)

        remove_button = tk.Button(self.button_Frame, bg='cyan', text="Remove",
command=lambda: self.remove_row())
        remove_button.grid(row=1, column=0, sticky=tk.NSEW, padx=10, pady=10)
```

```
        self.button_Frame.rowconfigure(1, weight=1)
        self.button_Frame.columnconfigure(0, weight=1)

        edit_button = tk.Button(self.button_Frame, bg='cyan', text="Edit",
command=lambda: self.edit_row())
        edit_button.grid(row=2, column=0, sticky=tk.NSEW, padx=10, pady=10)
        self.button_Frame.rowconfigure(2, weight=1)
        self.button_Frame.columnconfigure(0, weight=1)

        exit_button = tk.Button(self.button_Frame, bg='cyan', text="Exit",
command=lambda: print("to do"))
        exit_button.grid(row=3, column=0, sticky=tk.NSEW, padx=10, pady=10)
        self.button_Frame.rowconfigure(3, weight=1)
        self.button_Frame.columnconfigure(0, weight=1)

        # geting data from DDBB
        # controller
        for module in self.controller.get_Modules_from_db():
            self.treeview.insert('', tk.END, values=(module.name,
module.description))
            print(module.description, module.name)

    def treeview_select(self, event):
        # Fill in the contents.
        selected_modules = self.treeview.selection()
        print(selected_modules)
        # assert (len(selected_modules) == 1)
        for selected_row in selected_modules:
            selected_row = self.treeview.item(selected_modules[0])
            name, description = selected_row['values'][0],
selected_row['values'][1]
            print(name, description)
            self.name_entry.delete(0, tk.END)
            self.name_entry.insert(0, name)
            self.description_entry.delete(0, tk.END)
            self.description_entry.insert(0, description)

    def treeview_double_1(self, event):
        # Fill in the contents.
        selected_modules = self.treeview.selection()
        assert len(selected_modules) == 1
        selected_row = self.treeview.item(selected_modules[0])
        code = selected_row['values'][0]
        print('RAFA', code)
        self.controller.show_frame(StartPage, code)

    def add_row(self):
        name = self.name_entry.get()
        description = self.description_entry.get()
        self.treeview.insert('', tk.END, values=(name, description))
        self.controller.add_Module_to_db((name, description))

    def remove_row(self):
        name = self.name_entry.get()
        description = self.description_entry.get()
        self.treeview.insert('', tk.END, values=(name, description))
        self.controller.remove_Module_from_db((name, description))

    def edit_row(self):
        name = self.name_entry.get()
        description = self.description_entry.get()
        self.treeview.insert('', tk.END, values=(name, description))
```

```
        self.controller.edit_Module_from_db((name, description))
```

# 6. TESTING

*Describe the process you took to test your code and to make sure the program functions as required. Provide the detailed test plan used. Also, indicate the testing you did after integrating your code with your partner's. (COMPLETE THIS SECTION INDIVIDUALLY)*

# 7. Annotated Screenshots Demonstrating Implementation

*Provide screenshots that demonstrate the features implemented. Annotate each screenshot and if necessary, provide a brief description for **each** (**up to 100 words**) to explain the code in action. Make sure the screenshots make clear what you have implemented and achieved.*
*(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)*

## 7.1 Feature F1 (*indicate name of partner that developed feature here*)

### a. Sub-feature I- screenshots …

### b. Sub-feature II- screenshots …

### c. Sub-feature III- screenshots …

## 7.2 Feature F2 (*indicate name of partner that developed feature here*)

### a. Sub-feature I- screenshots …

### b. Sub-feature II- screenshots …

### c. Sub-feature III- screenshots …

# 8. EVALUATION

*Give a reflective, critical self-evaluation of your experience developing the project and discuss what you would do if you had more time to work on the project. Answer the following questions for the reflection and write **350-400 words overall**. Please include an actual word count for this section.*
*(COMPLETE THIS SECTION INDIVIDUALLY)*

## 8.1 EVALUATE HOW WELL YOUR DESIGN AND IMPLEMENTATION MEET THE REQUIREMENTS

## 8.2 EVALUATE YOU OWN AND YOUR GROUP'S PERFORMANCE

### 8.2.1 WHAT WENT WELL?

GUI design

### 8.2.2 WHAT WENT LESS WELL?

Data handling

### 8.2.3 WHAT WAS LEARNT?

OOP features

### 8.2.4 HOW WOULD A SIMILAR TASK BE COMPLETED DIFFERENTLY?

### 8.2.5 HOW COULD THE MODULE BE IMPROVED?

Incorporating for both text base and GUI based features as per the user criterion

## 8.3 Self-assessment

*Please assess yourself objectively for each section shown below and then enter the total mark you expect to get. Marks for each assessment criteria are indicated between parentheses.*

### Code development (70)

#### Features Implemented [30]

##### Sub-feature i (up to 8)

Sub-features have not been implemented – 0
Attempted, not complete or very buggy – 1 or 2
Implemented and functioning without errors but not integrated – 3 or 4
Implemented and fully integrated but buggy – 5 or 6
Implemented, fully integrated and functioning without errors – 7 or 8

##### Sub-feature ii (up to 10)

Sub-features have not been implemented – 0
Attempted, not complete or very buggy – 1 or 2
Implemented and functioning without errors but not integrated – 3 to 5
Implemented and fully integrated but buggy – 6 to 8
Implemented, fully integrated and functioning without errors – 9 or 10

##### Sub-feature iii (up to 12)

Sub-features has not been implemented – 0
Attempted, not complete or very buggy – 1 to 3
Implemented and functioning without errors but not integrated – 4 to 6
Implemented and fully integrated but buggy – 7 to 9
Implemented, fully integrated and functioning without errors – 10 to 12

| **For this criterion I think I got: 20 out of 30** |
|---|

#### Use of OOP techniques [25]

##### Abstraction (up to 10)

No classes have been created – 0
Classes have been created superficially and not instantiated or used – 1 or 2
Classes have been created but only some have been instantiated and used – 3 or 4
Useful classes and objects have been created and used correctly – 5 to 7
The use of classes and objects exceeds the specification – 8 to 10

##### Encapsulation (up to 10)

No encapsulation has been used – 0
Class variables and methods have been encapsulated superficially – 1 to 3
Class variables and methods have been encapsulated correctly – 4 to 6
The use of encapsulation exceeds the specification – 7 to 10

##### Inheritance (up to 5)

No inheritance has been used – 0
Classes have been inherited superficially – 1
Classes have been inherited correctly – 2 to 4
The use of inheritance exceeds the specification – 5

Bonus marks will be awarded for the appropriate use of polymorphism (bonus marks up to 10)

**For this criterion I think I got: 20 out of 25**

### Quality of Code [15]

#### Code Duplication (up to 8)

Code contains too many unnecessary code repetition – 0
Regular occurrences of duplicate code – 1 to 3
Occasional duplicate code – 4 to 5
Very little duplicate code – 6 to 7
No duplicate code – 8

#### PEP8 Conventions and naming of variables, methods and classes (up to 4)

PEP8 and naming convention has not been used – 0
PEP8 and naming convention has been used occasionally – 1
PEP8 and naming convention has been used, but not regularly – 2
PEP8 and naming convention has been used regularly – 3
PEP8 convention used professionally and all items have been named correctly – 4

#### In-code Comments (up to 3)

No in-code comments – 0
Code contains occasional in-code comments – 1
Code contains useful and regular in-code comments – 2
Thoroughly commented, good use of docstrings, and header comments describing.py files – 3

**For this criterion I think I got: 10 out of 15**

## DOCUMENTATION (20)

### Design (up to 10) clear exposition about the design and decisions for OOP use

The documentation cannot be understood on first reading or mostly incomplete – 0
The documentation is readable, but a section(s) are missing – 1 to 3
The documentation is complete – 4 to 6
The documentation is complete and of a high standard – 7 to 10

### Testing (5)

Testing has not been demonstrated in the documentation – 0
Little white box testing has been documented – 1 or 2
White box testing has been documented for all the coursework – 3 or 4
White box testing has been documented for the whole system – 5

### Evaluation (5)

No evaluation was shown in the documentation – 0
The evaluation shows a lack of thought – 1 or 2
The evaluation shows thought – 3 or 4
The evaluation shows clear introspection, demonstrates increased awareness – 5

**For this criterion I think I got: 15 out of 20**

## ACCEPTANCE TESTS - DEMONSTRATIONS (10)

### Final Demo (up to 10)

Not attended or no work demonstrated – 0
Work demonstrated was not up to the standard expected – 1 to 3
Work demonstrated was up to the standard expected – 4 to 7

Work demonstrated exceeded the standard expected – 8 to 10

| For this criterion I think I got: | 10 out of 10 |
|---|---|

| I think my overall mark would be: | out of 100 |
|---|---|

## 9. GROUP PRO FORMA

*Describe the division of work and agree percentage contributions. The pro forma must be signed by all group members and an identical copy provided in each report. If you cannot agree percentage contributions, please indicate so in the notes column and provide your reasoning.*
*(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)*

| Partner ID | Tasks/Features Completed | %Contribution | Signature | Notes |
|---|---|---|---|---|
| 001134460 | F1 | 100% | Mostafizur Rahman | |
| 2 | | | | |
| | **Total** | 0 | | |

# APPENDIX A: CODE LISTING

*Provide a complete listing of all the *.py files in your PyCharm project. Make sure your code is well commented and applies professional Python convention (refer to PEP 8 for details). The code listed here must match that uploaded to Moodle. Please copy and paste the actual code – no screenshots please! You will lose marks if screenshots are provided instead of code.*
*(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)*