# COMP1811 – Python Project Report

| **Name** | **Morales Huaygua, Luis David** | **Student ID** | **001137079** |
| --- | --- | --- | --- |
| **Partner’s name** | **Muthaliph, Gnei Tiara Rahiza** | **Partner SIDs** | **001166581** |

# 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 *[Luis David] [Gnei Tiara]*** | **F1:** i ii  iii  **F2:** i ii  iii |
| --- | --- | --- |
| 1.2 Circle the parts of the coursework you have **partly completed or are partly working.** | **Features** | **F1:** i ii  iii  **F2:** i ii  iii |
| Briefly explain your answer if you circled any parts in 1.2 | | |

# 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)*

## Bugs

*List each bug plus a brief description*

Incorrect feedback table foreign key: During the debugging stage of the project, I encountered an error in an SQLite query that would push the feedback from each given answer with a default foreign key that would not change as the user adds more questions. Although, this won’t raise an error, this issue would result in every feedback entity having the same ‘quest\_id’ (foreign key) which would later cause trouble when returning the right feedback to the user in the quiz. The bug initiated to happen due to the way I built the adding question function which would pass the module id as the ‘quest\_id’ parameter for **other function**. This function would then execute the query and setting the ‘quest\_id’ (column) of each feedback entity to the module id. I solved this issue by first pushing the question to the database, finding the id of that question and using this id as a parameter for the **other function**.

## Weaknesses

*List each weakness plus a brief description*

# Description of the features implemented

*Describe your implementation of the required features and how well do they work. Provide some exposition of the design decisions made and indicate how the features developed were integrated.   
(THIS SECTION SHOULD BE THE SAME FOR BOTH PARTNERS)*

# 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)*

## Diagram Description automatically generatedClasses Used

Figure Class Design

## Brief Explanation of Class Design and OOP Features Used

I used three classes for this project and multiple tkinter windows. I decided to use tkinter windows for almost every feature (add module, delete module…) because it made it much simpler to pass information from one window frame to another window frame than the conventional class attributes. This is the reason I do not use attributes in each class. However, I believe the behavior part of each class was good, as I did use a lot of them.

One OPP technique that particularly that stand out in the app is the inheritance technique. Both classes, (adminHomeFeatures and Log In Features) have availability to use part of the behavior of QuizzAPP class such as to change to another frame if necessary.

In terms of the tkinter windows, I did not find any solution to pass information from class to class without having to access the main (QuizzApp) class. This means from the adminHomeFeatures, we can only access attributes from the QuizAPP. Therefore, I used tkinter windows. Also, I could not figure out how to build a sort of tree structure with the classes. For instance, I wanted to build a child class for AdminHomeFeatures lass, but I couldn’t. I would need to build a child class for QuizzApp instead, so the only solution I came out with was using tkinter windows inside the AdminHomeFeatures. I later discovered how class would inherit their attributes while still being frames. Unfortunately, I could not re do all the work I’ve done so far and deliver on time.

# 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 …

This is the class I worked in. Unfortunately, I can’t provide the entire code of the class as I developed most necessary functions for the program to work inside this class and there are 1200+ lines of code only in this class. The entire class will be in the appendix at the end.

1. class AdminHomePage(tk.Frame):  
    def \_\_init\_\_(self, parent, controller):  
    tk.Frame.\_\_init\_\_(self, parent)  
    self.isActive = False

## Class …

1. class QuizzApp(tk.Tk):  
    def \_\_init\_\_(self, \*args, \*\*kwargs):  
    tk.Tk.\_\_init\_\_(self, \*args, \*\*kwargs)  
    self.currScore = 0  
    #Create the window  
    window = tk.Frame(self)  
    window.pack()  
     
    window.grid\_rowconfigure(0, minsize=700)  
    window.grid\_columnconfigure(0, minsize=700)  
     
    """  
    This idea of maintaining the frames in a constant for loop came from this video.  
    https://www.youtube.com/watch?v=tpGjHRDEjCE&t=1153s&ab\_channel=IGTechTeam  
      
    Basically, it creates a dictionary to store all the classes of the questionnaire.  
    Then with the "change\_frame" function it changes to which class you want to visit.   
    """  
    self.containerOfFrames= {}  
    for f in (LogInPage, UserHomePage, AdminHomePage):  
     
    frame = f(window, self)  
    self.containerOfFrames[f] = frame  
    frame.grid(row=0, column=0, sticky="nsew")  
    self.change\_frame(AdminHomePage)  
     
    def change\_frame(self, page):  
    frame = self.containerOfFrames[page]  
    frame.tkraise()  
    """  
    This function can be used to get the score of the user at any point in time whenever a  
    quiz has been initialized.  
    There is got to be another function to update the score to 0.  
      
    """  
    def getScore(self):  
    print(self.currScore)

## Class …

1. class LogInPage(tk.Frame):  
    def \_\_init\_\_(self, parent, controller):  
    tk.Frame.\_\_init\_\_(self, parent)  
    # --------------------------LOGIN---------------------------------  
    fontBG = tkFont.Font(  
    family="Arial",  
    size=16,  
    weight='bold',  
    )  
     
    font\_small = tkFont.Font(  
    family="Arial",  
    size=12,  
    weight='bold',  
    )  
    """  
    The idea came from: https://www.youtube.com/watch?v=tpGjHRDEjCE&t=1153s&ab\_channel=IGTechTeam  
    I used part of the code from the video to develop the essential log in page that would  
    be the bridge between the user interface and admin interface.  
    I fully understand the small parts I replicated from the video.  
      
    """  
    border = tk.LabelFrame(self, text="Log In", bg='#02203c', fg="white", bd=1, font=fontBG)  
    border.pack(fill='both', expand='yes', padx=20, pady=150)  
     
    username = tk.Label(border, text="username", font=fontBG, fg="white", bg='#02203c')  
    username.place(x=50, y=20)  
     
    userInput = tk.Entry(border, width=30, bd=5)  
    userInput.place(x=180, y=20)  
     
    password = tk.Label(border, text="password", font=fontBG, fg="white", bg='#02203c')  
    password.place(x=50, y=80)  
     
    passInput = tk.Entry(border, show="\*", width=30, bd=5)  
    passInput.place(x=180, y=80)  
    testing = tk.Label(border, text="To enter the admin page use: admin as username and admin as password",  
    font=font\_small, fg="white", bg='#02203c')  
    testing.place(x=20, y=200)  
     
    # -------------------------------SUBMIT LOGIN--------------------------  
    def verify():  
    *"""  
    OPTION 1 ->  
    username and password match  
    in the database File for normal users  
    Should take them to the UI of normal users  
     
    """* with open("credential.txt", "r") as f:  
    # ["username, password", "username,password"]  
    info = f.readlines()  
    for user in info:  
     
    # u -> username, p -> password  
    # split them such that u -> "username" and p -> "password"  
    u, p = user.split(",")  
    # strip -> removes spaces at the end and beginning  
    # if u match our username input and p match our password input take user to next page  
    if u.strip() == userInput.get() and p.strip() == passInput.get():  
    controller.change\_frame(UserHomePage)  
    return  
    """  
    # OPTION 2 ->  
    # username and password match  
    # in the database file for ADMINS users  
    # Should take them to the UI for Admin Users  
    """  
    with open("AdminCredential.txt", "r") as f:  
    # ["username, password", "username,password"]  
    info = f.readlines()  
    for user in info:  
    # u -> username, p -> password  
    # split them such that u -> "username" and p -> "password"  
    u, p = user.split(",")  
    # strip -> removes spaces at the end and begining  
    # if u match our username input and p match our password input take user to next page  
    if u.strip() == userInput.get() and p.strip() == passInput.get():  
    controller.change\_frame(AdminHomePage)  
    return  
    messagebox.showinfo("Error", "Please provide a correct username and password")  
     
    # ---------------------REGISTRATION-----------------------------  
    submitBtn = tk.Button(border, text="Submit", command=verify, font=fontBG)  
    submitBtn.place(x=275, y=120)  
     
    def registerUser():  
    window = tk.Tk()  
     
    # make the window not resizable  
    window.resizable(0, 0)  
     
    window.title("Register")  
    l1 = tk.Label(window, text="Username: ", font=fontBG)  
    l1.place(x=10, y=10)  
     
    e1 = tk.Entry(window, width=30, bd=5)  
    e1.insert(tk.END, "username")  
    e1.place(x=200, y=10)  
     
    l2 = tk.Label(window, text="Password: ", font=fontBG)  
    l2.place(x=10, y=80)  
     
    e2 = tk.Entry(window, show="\*", width=30, bd=5)  
    e2.insert(tk.END, "password")  
    e2.place(x=200, y=80)  
     
    l3 = tk.Label(window, text="Confirm Password: ", font=fontBG)  
    l3.place(x=10, y=150)  
    e3 = tk.Entry(window, show="\*", width=30, bd=5)  
    e3.place(x=200, y=150)  
     
    """  
    This function open the database and  
    check whether the username of the user  
    is already in use returns True if it is,  
    otherwise False  
    """  
    def isNameUsed(name):  
    with open("credential.txt", 'r') as f:  
    info = f.readlines()  
    for user in info:  
    # u -> username, p -> password  
    # split them such that u -> "username" and p -> "password"  
    u, p = user.split(",")  
    # strip -> removes spaces at the end and beginning  
    # if u match our username input  
    # and p match our password input take user to next page  
    if u.strip() == name:  
    return True  
    return False  
     
    # ----------------------------submit registration --------------------------  
    def check():  
    isUsed = isNameUsed(e1.get())  
    if e1.get() != 'username' and e2.get() != "password" and isUsed is False:  
    if e2.get() == e3.get():  
    with open("credential.txt", "a") as f:  
    f.write(e1.get() + ',' + e2.get() + "\n")  
     
    messagebox.showinfo("Welcome", "You are now a fully registered")  
    else:  
    messagebox.showinfo("Error", "Your password didn't get match!")  
    else:  
    if isUsed:  
    messagebox.showinfo("Error", "This username is already in use")  
    return  
    messagebox.showinfo("Error", "Some field is missing, Please fill of all of them")  
     
    e4 = tk.Button(window, text="Sign In", command=check, font=fontBG)  
    e4.place(x=330, y=180)  
    window.geometry("480x250")  
     
    registerBtn = tk.Button(self, text="Register", bg='#02203c', font=fontBG, fg="white",  
    command=registerUser)  
    registerBtn.place(x=550, y=170)

## Class …

…

# 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)*

Unfortunately, my partner did not deliver any functional feature so I could not integrate my code to her’s.

As the admin features developer, I had to interact with the database in multiple occasions to fetch, delete, update and add data. Therefore, I decided to use a white box technique to verify the flow of inputs and outputs through the questionnaire app before developing any feature. I used this technique every time I needed to add a new feature to the project which made the implementation simpler as I could see the steps from top to bottom.

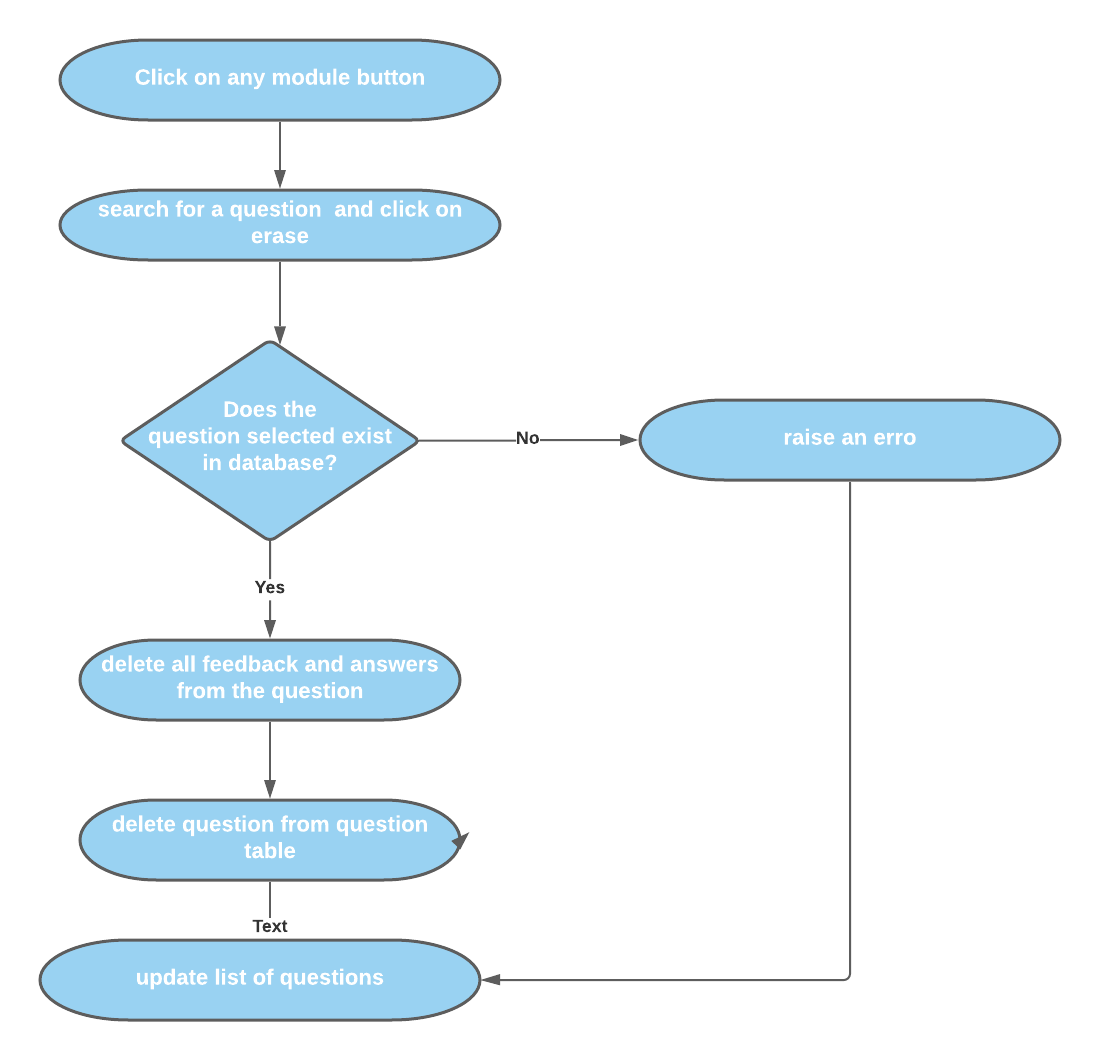
On some occasions multiple SQLite commands would be combined. For instance, when deleting a module, the program, first needs to delete all feedback and questions linked to the module and finally the module itself from the database. Having, the white box diagrams made this extremely easier to visualize and understand before starting to code.

Figure white box, shows the process of deleting a question

See below some of the white box diagrams.

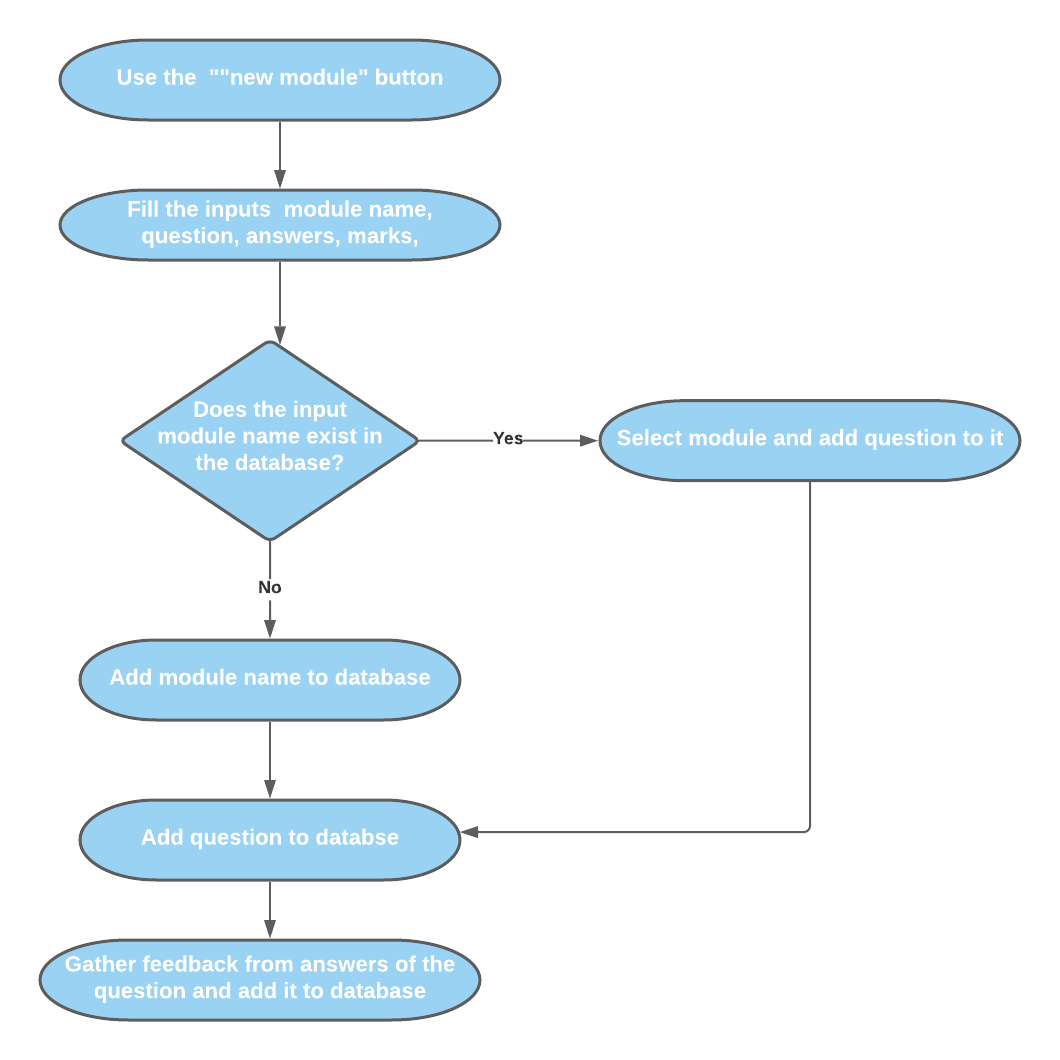


Figure white box, shows the process of adding a new module

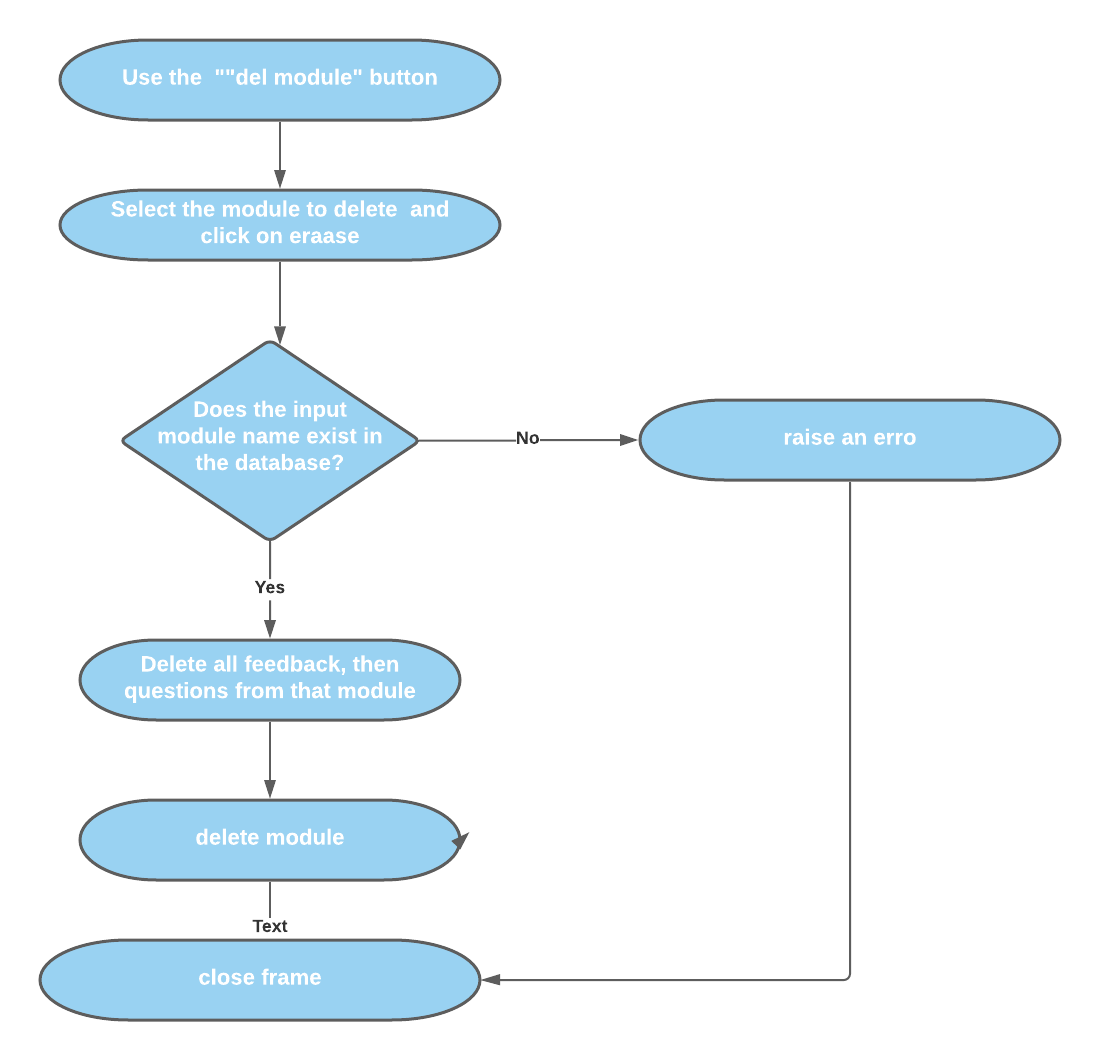


Figure white box, shows the process of deleting a module

# 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)*

## Feature F1 (Luis David Morales Huaygua)

## Sub-feature i- screenshots …

## Sub-feature ii- screenshots …

## Sub-feature iii- screenshots …

## Feature F2 (Muthaliph, Gnei Tiara Rahiza)

## Sub-feature i- screenshots …

## Sub-feature ii- screenshots …

## Sub-feature iii- screenshots …

# 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)*

## Evaluate how well your design and implementation meet the requirements

This quiz has been tested for all sub-features of admin features and it passed all the test cases. However, due to my teammate’s performance, the questionnaire features could not be completed.

## Evaluate you own and your group’s performance

Although, my teammate and I met on some occasions to discuss the vision of the project I cannot say her performance reached my expectations. She was unable to implement a simple feature of the quiz by 3 days prior submission. Subsequently, she proceeds to block me on our main source of communication, presumably because of me asking her for updates on the project. As far as I am concerned, she did not ask for help, even though I was offering her my support in multiple occasions.

On the other hand, I was spending long hours building the admin feature and sending it to her whenever I update something so she can work on top of that. I managed to develop a fully functional Admin Frame and I believe it was because of my dedication and commitment on the project.

## What went well?

One of the things that I struggled the most was the interaction between the SQL file and python itself. The admin features required a lot of SQL manipulation because simply there were a lot of entities to be changed each time a button is clicked, literally. However, once I learned how to implement few of these queries, I found a way of doing things with less code. For instance, during the development of the “edit question” sub-feature I was doing a lot of calls to the database to update data, and therefore creating lot of functions for each of these calls. I made this simpler with a single function that would take the column we want to update and the new value as parameter.

## What went less well?

The implementation of more than 3 classes was not possible. This is because I believe that opening a new window and passing the necessary parameters to this new function window was easier than implementing a new class for each window I needed. This resulted in the F1 (my part) being less OOP oriented but overall, I am happy with the results.

## What was learnt?

During the development of the project, I learned a ton about best python practices, OOD and OOP which helped me improve my design skills when building projects overall. I also learned that it is very important to look at specifications of the problem firstly and tackle the main features before everything else. If you can build a solid base in your project, you can always come back to it if you need to re-do the project again.

## How would a similar task be completed differently?

A similar task could be done using different libraries/technologies for the GUI but same OOD would be very helpful.

## How could the module be improved?

During the development of my project, I faced many issues with my partner. She wasn’t doing particularly well delivering any feature, and I think that was because she wasn’t taking the project seriously. Therefore, I believe that by doing more progress checks during the term would be very helpful to each team and unstuck those who are facing challenges.

## 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: 30 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: 15 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: 12 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: 5 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: 8 out of 10**

**I think my overall mark would be: 70 out of 100**

# 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** |
| --- | --- | --- | --- | --- |
| **1** | Administration Features | 50% | Luis David Morales Huaygua | Sub-feature i.  Sub-feature ii.  Sub-feature iii. |
| **2** | Quiz Features | 0% |  | Sub-feature i.  Sub-feature ii.  Sub-feature iii. |
|  | **Total** | 50% |  |  |

# 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*](https://www.python.org/dev/peps/pep-0008/) *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)*