Coursework Documentation

Haaris Iqbal

16952858

Table of contents

* Note: Images may appear blurry due to conversion

Project Specifications

The primary goal of this coursework was to design, build, test and evaluate a GUI Application. The project specifications were broken down into four main goals;

1. Design, build and test a standalone GUI. The GUI must demonstrate the use of labels, entry boxes, buttons, treeviews, etc.

2. Design, build and test a standalone Database. The Database must demonstrate the use of CRUD operations (Create, Read, Update, Delete).

3. The GUI must be seamlessly attached to the database application, such that interaction with the “front end” GUI will result in the operation and manipulation of the “back end” database.

4. Testing must be performed on all elements of the project (front end, back end and linking) to ensure project is functioning as intended and without fault.

Alongside these four conditions, the creator is required to submit documentation demonstrating the final project and its full life cycle, along with a transcript of the testing. The final deadline for the submission of a zipped file containing all of the requirements was 10th April, 2019.

As long as these axiomatic requirements were met, the project may be developed in any additional way, shape or form (at the discretion of the creator).

Project Development

The specifications for the project were received on 21st January, 2019. While basic steps were immediately taken (creating a plan document to begin development schedule, creating Log document to note changes in projects along its development), no major action could be taken due to limited knowledge in the Python language, hence many weeks were spent in focusing on studying and improving knowledge in python, as well as installing necessary modules on system (PIP, PILLOW, etc.)

Console:

On 19th February 2019, knowledge in python was sufficient enough to begin work on “back end” database. By implementing basic “CRUD” functionality from lectures and tutorials, testing of each CRUD function began, along with construction of basic menu for manipulation of CRUD in console.

On March 10th 2019, a fully functioning “back end” database had been completed. This was completed in the form of two .py modules.

The first module “main.py” would initially try to connect to the second module “functions.py”, along with the module “sqlite3”. In event of failure, custom error message would be displayed to user. If successful, it would then try to connect to “database.db”. If “database.db” did not exist, the .db file would be created from scratch. The central commands for menu were then defined, and in an infinite loop the user would be prompted to enter a choice from the available menu. Based on entrance of choice, a function would be called from “functions.py”

The second module “functions.py” was designed to contain all the functions needed for “main.py”, and is not intended to run on its own. After attempting and successfully connecting to the database, a total of nine functions were defined. A brief summary of each function is as follows;

1. menu()
   This is a function that simply contained the menu screen for “main.py” to call upon when needed

2. add()
   This function, when called, would prompt the user to input four more variables (Name, Grade, Image, Status). It would then add these variables as a new entry into the database

3. select()
   This function, when called, would prompt the user to input a student ID. It would then attempt to search and pull up (display) the ID, along with all associated details from the database. If it is unable to do so, the user will be informed that an invalid ID was entered.

4. update()
   This function, when called, would prompt the user to input a student ID to update. Upon successful, conversion to integer, and confirmation of existence in database, the entry is pulled up, and the user may then input 4 new variables to replace the variables in the selected entry.

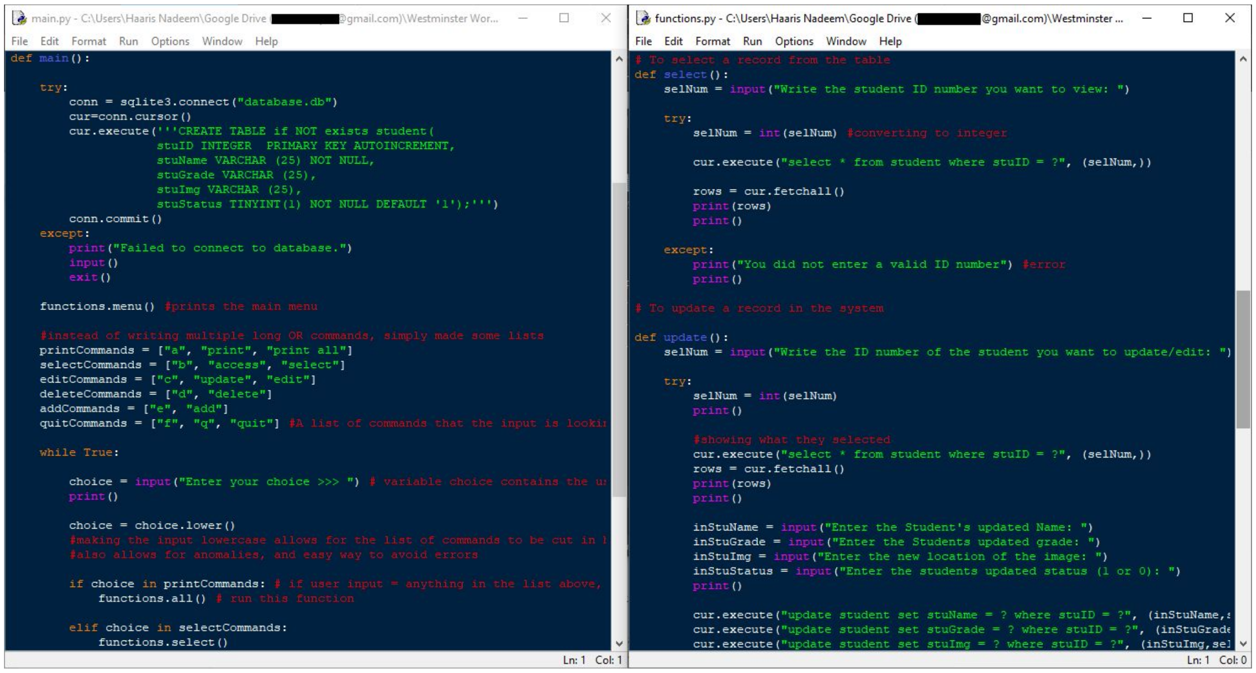
5. delete()
   This function, when called, would prompt the user to input an ID to be deleted from the database. Once a valid ID has been inputted, the entry is deleted. If input is invalid, user is informed and no action is taken.

6. all()
   This function, when called, simply displayed all the entries in the database. This was often called after every false entry from the user to remind them of which entries are present. It could also specifically be called with menu command

7. quit()
   This function, when called, would simply shut down the program.

8. error()
   Upon false entries in the menu screen, this function would be called to display the error message of having typed in an invalid command. It was often accompanied by calling the menu function, to display the list of valid commands

9. test()
   This was simply a test function used during construction to ensure that the two modules “main.py” and “functions.py” were linked

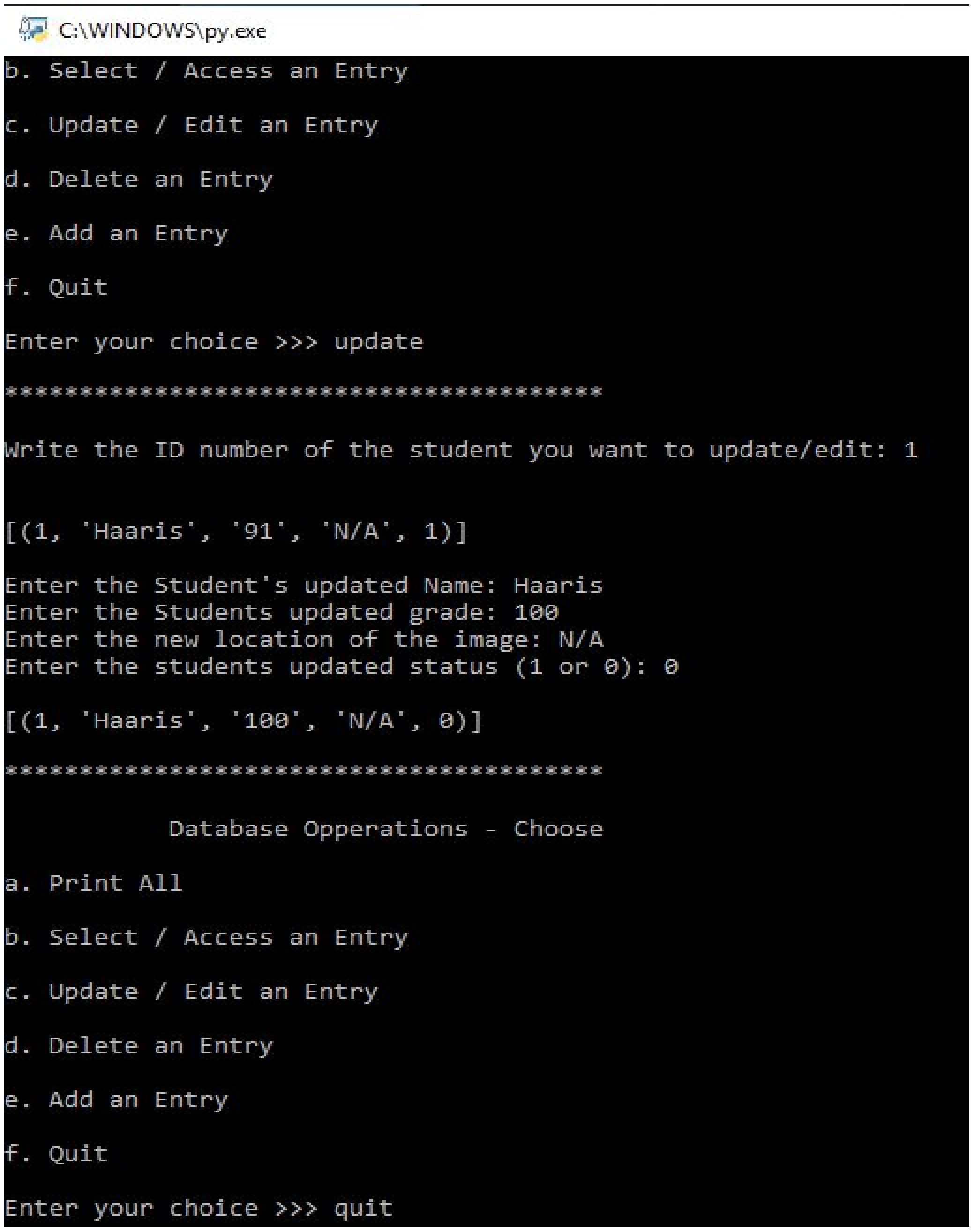

main.py - C:\Users\Haaris Nadeem\Google Drive ([redacted]@gmail.com)\Westminster Wor...
File Edit Format Run Options Window Help
Ln: 1 Col: 1
functions.py - C:\Users\Haaris Nadeem\Google Drive ([redacted]@gmail.com)\Westminster ...
Ln: 1 Col: 0

```python
def main():

    try:
        conn = sqlite3.connect("database.db")
        cur=conn.cursor()
        cur.execute('''CREATE TABLE if NOT exists student(
                    stuID INTEGER  PRIMARY KEY AUTOINCREMENT,
                    stuName VARCHAR (25) NOT NULL,
                    stuGrade VARCHAR (25),
                    stuImg VARCHAR (25),
                    stuStatus TINYINT(1) NOT NULL DEFAULT '1');''')
        conn.commit()
    except:
        print("Failed to connect to database.")
        input()
        exit()

    functions.menu() #prints the main menu

    #instead of writing multiple long OR commands, simply made some lists
    printCommands = ["a", "print", "print all"]
    selectCommands = ["b", "access", "select"]
    editCommands = ["c", "update", "edit"]
    deleteCommands = ["d", "delete"]
    addCommands = ["e", "add"]
    quitCommands = ["f", "q", "quit"] #A list of commands that the input is looki

    while True:

        choice = input("Enter your choice >>> ") # variable choice contains the u
        print()

        choice = choice.lower()
        #making the input lowercase allows for the list of commands to be cut in 
        #also allows for anomalies, and easy way to avoid errors

        if choice in printCommands: # if user input = anything in the list above,
            functions.all() # run this function

        elif choice in selectCommands:
            functions.select()
```

```python
# To select a record from the table
def select():
    selNum = input("Write the student ID number you want to view: ")

    try:
        selNum = int(selNum) #converting to integer

        cur.execute("select * from student where stuID = ?", (selNum,))

        rows = cur.fetchall()
        print(rows)
        print()

    except:
        print("You did not enter a valid ID number") #error
        print()

# To update a record in the system

def update():
    selNum = input("Write the ID number of the student you want to update/edit: ")

    try:
        selNum = int(selNum)
        print()

        #showing what they selected
        cur.execute("select * from student where stuID = ?", (selNum,))
        rows = cur.fetchall()
        print(rows)
        print()

        inStuName = input("Enter the Student's updated Name: ")
        inStuGrade = input("Enter the Students updated grade: ")
        inStuImg = input("Enter the new location of the image: ")
        inStuStatus = input("Enter the students updated status (1 or 0): ")
        print()

        cur.execute("update student set stuName = ? where stuID = ?", (inStuName,s
        cur.execute("update student set stuGrade = ? where stuID = ?", (inStuGrade
        cur.execute("update student set stuImg = ? where stuID = ?", (inStuImg,sel
```

Of course, the primary goal of this task was to ensure that all SQL functions were fully operational as they would later be linked to the GUI, and used to interact with the database. Nonetheless, some of the additional work that was put into ensuring the “main.py” module functioned beyond its intended specifications include;

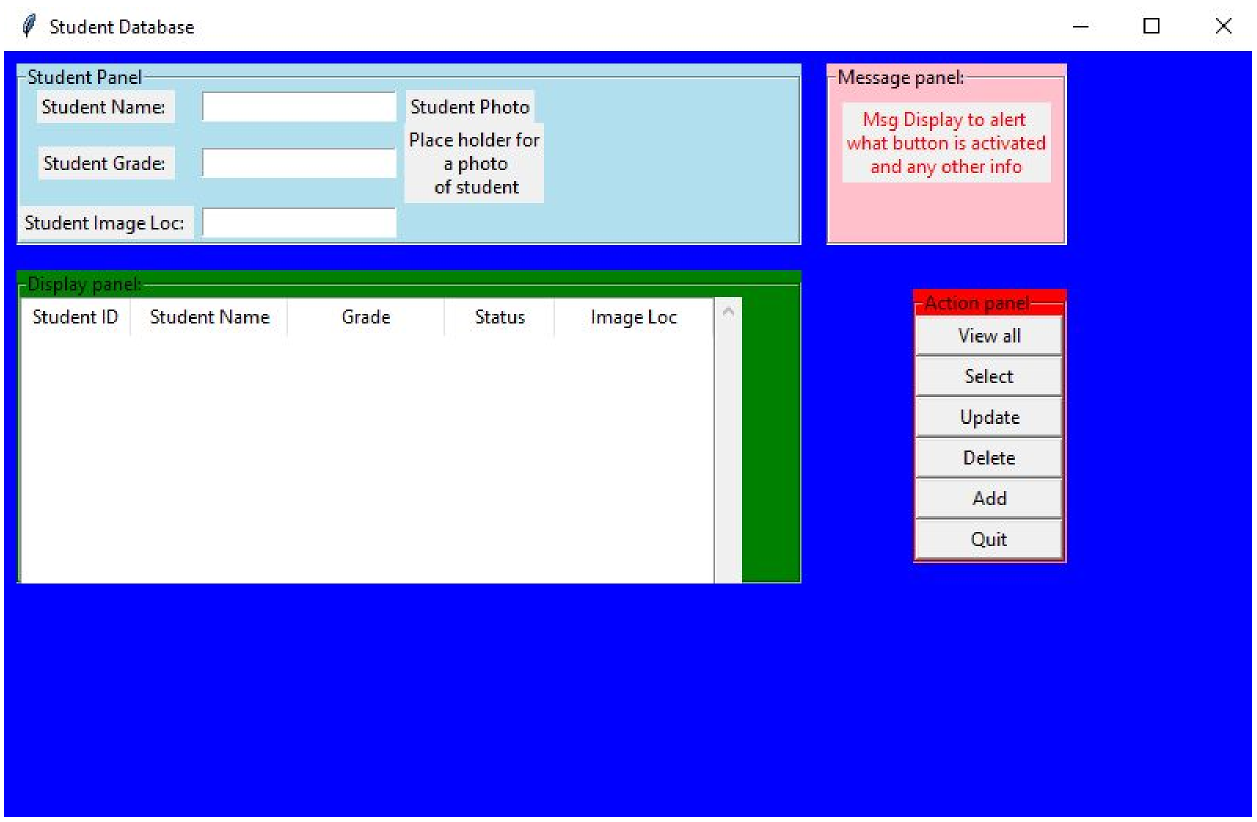
1. Having a list for each menu command, along with auto-shifting to lowercase, to allow for a large variety of ways to enter the same command.
2. Ensuring easy readability and neatness by making sure spacing between inputs and output were appropriate, along with using a separating line (in the form of stars) to visually define the end of an old output/input, and the beginning of a new output/input.



```
C:\WINDOWS\py.exe
b. Select / Access an Entry

c. Update / Edit an Entry

d. Delete an Entry

e. Add an Entry

f. Quit

Enter your choice >>> update

****************************************

Write the ID number of the student you want to update/edit: 1


[(1, 'Haaris', '91', 'N/A', 1)]

Enter the Student's updated Name: Haaris
Enter the Students updated grade: 100
Enter the new location of the image: N/A
Enter the students updated status (1 or 0): 0

[(1, 'Haaris', '100', 'N/A', 0)]

****************************************

           Database Opperations - Choose

a. Print All

b. Select / Access an Entry

c. Update / Edit an Entry

d. Delete an Entry

e. Add an Entry

f. Quit

Enter your choice >>> quit
```

GUI:

With the database fully completed, extensive work began on the GUI. Research on tkinter was done, and necessary modules (such as PILLOW) were installed on system. Due to material in lectures and tutorials, a prototype for GUI interface was created very quickly, however initial design was very generic. Sketches were created to envision optimal design for database.

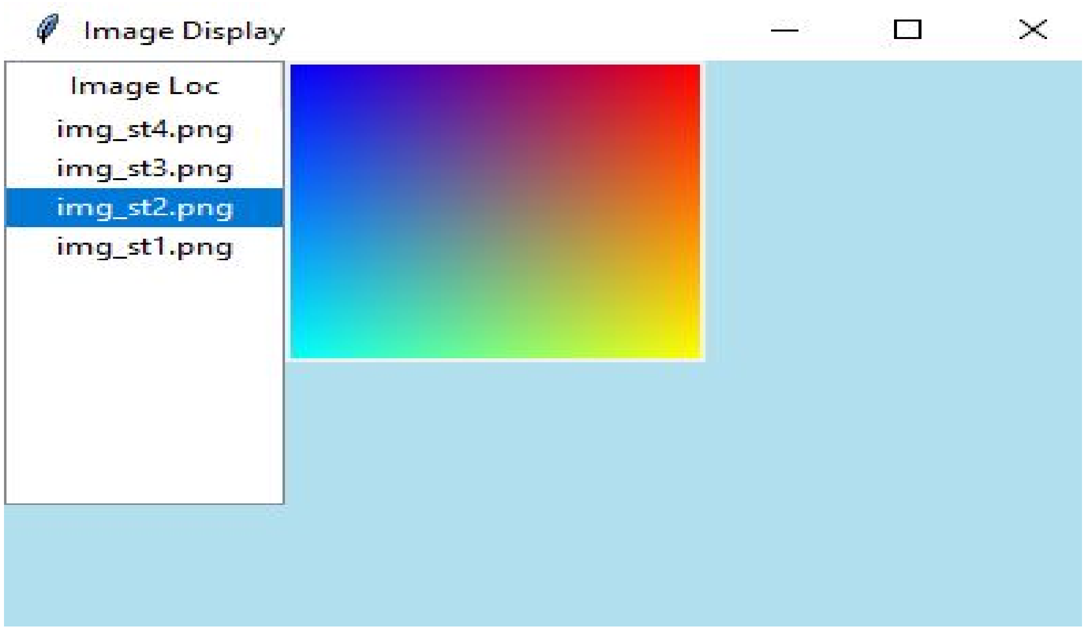


Before pursuing further work on GUI design, the decision was made to focus on a seamless link between the GUI and Database. Hence, focus was put on modification of "functions.py" code to work with "GUI.py", and modification of "GUI.py" code to work with "functions.py". The unification of these two modules was a simple task. Below is a list of the more unique challenges in finishing the final product;

1. Implementation of image

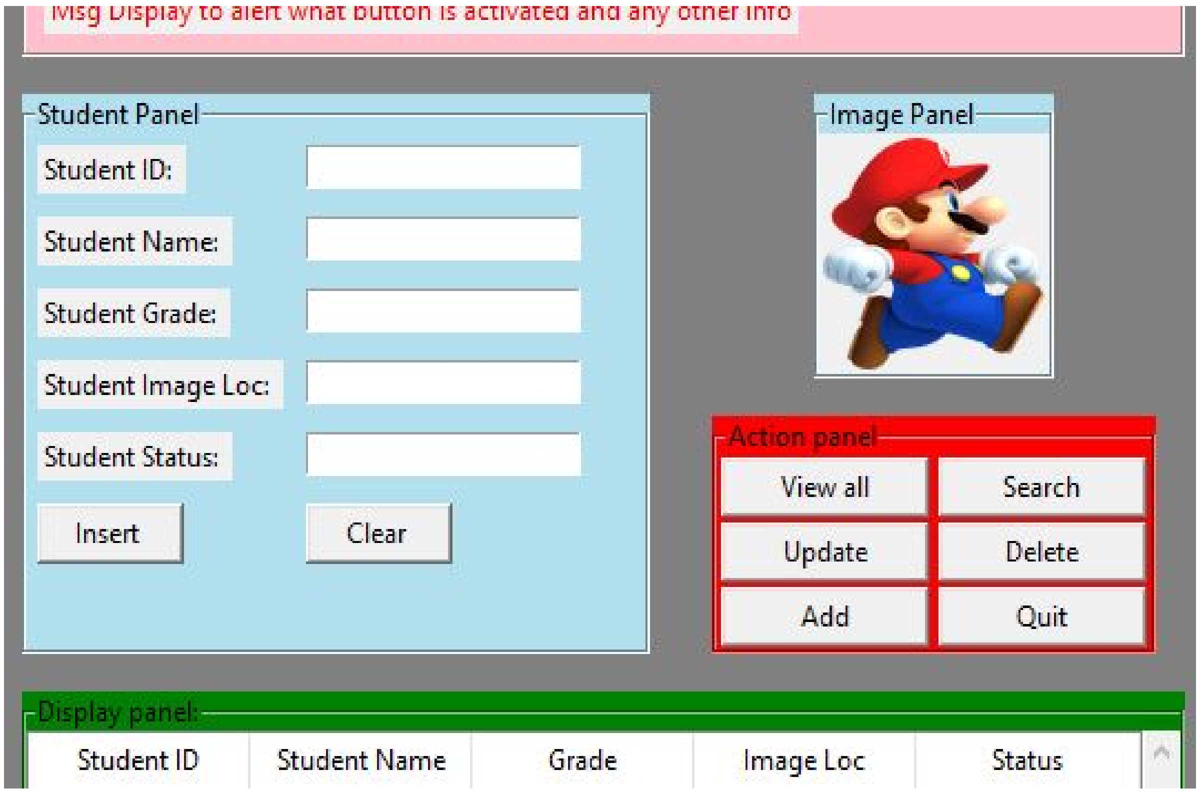
Implementation of images themselves were not a difficult task, as the code on doing so were taught during lectures and tutorials. However there were two main goals relating to the image that had to be achieved;

- A rescaling of any imputed images to a constant, predefined size to ensure that it could be seamlessly designed into the GUI, along with ensuring that any image inputted by a user would function without the user having to crop it to make it fit for purpose
- Making the GUI work with a default image, and how it should act in situation where user provides no image or invalid image location



Through testing on an alternate .py file, a solution was found to achieve both these goals. For the issue of rescaling, research on the topic yielded the ".resize" command. By implementing this in the code, a definitive size for the image could be set, such that every image handed by the code would automatically be scaled to the defined parameters of "WIDTH, HEIGHT".

The default image was an even simpler task. An image function was created that attempted to load the new image, and if failed, would simply fall back to the default image. This function was then called in other menu functions, therefore ensuring that with a selection or button press, this action would run, and either an associated image would be displayed by the provided valid location, or the default image would imply appear in its place.

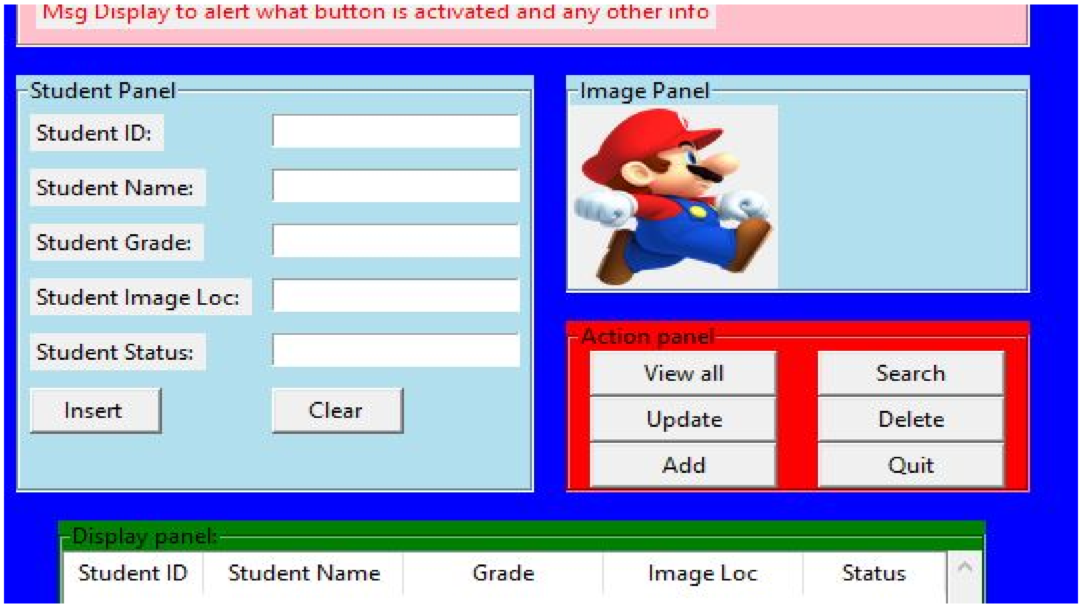


A new widget was also made to accommodate the image. Merging both the student panel and image display caused issues with columns not spacing as intended. However, this would have not been too difficult to fix, and the primary reason for separating these two columns was to allow for a more neater / streamlined design

2. Widget Issues

There were two particularly curious issues that arose in development that were necessary to address, both a consequence of using "grid" in tkinter;

- Issue with "sticky" in image panel causing the image panel to demorth upon receiving and displaying an image.
- Resizing the the created tkinter window did nothing to the widgets as they were fixed in place; hence user could hide content or create a lot of unused extra space by dragging window, causing the GUI to look very unsightly

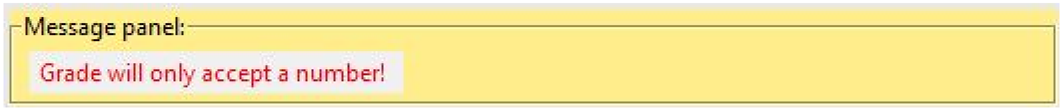


In both cases, the best solution seemed to be to remove and reduce. The issue with "sticky" in the image panel was resolved by simply removing the "sticky" parameters and command altogether. For this reason, the image panel is the only panel without "sticky" parameters, and this allows it to sit centered above the action panel.

In regards to the issue of resizing, the ability for the user to manually resize was removed completely. This was achieved by attaching .resizable to the main window name, and setting width and height parameters to "false". With the ability to resize removed, greater care was then made to ensure that the whole panel appeared in an exact, pixel perfect size that would accommodate all the widgets and look visually appealing.

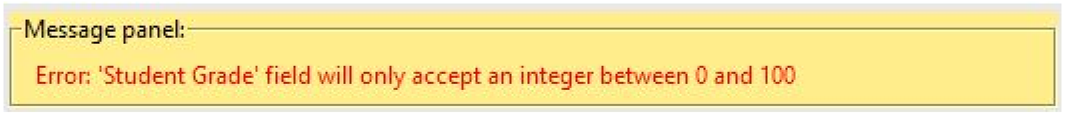
3. Message Panel

As message panel was worked on early in development, initially there was insufficient understanding on manipulation of text in the label. As a result, a very rudimentary approach was taken, where the new label would simply be pasted over the last label, giving the illusion of change. This had the limitation of forcing all unique text alerts to be of the same size.


Message panel:

Grade will only accept a number!

Later in development, this was revisited. Through the separation of the creation of the label, and the text variable, manipulation of the text was achieved. The message panel was then totally revamped to be colour coded for various actions, to display far more descriptive messages, etc. (full capability of message panel can be found Final Product description)


Message panel:

Error: 'Student Grade' field will only accept an integer between 0 and 100

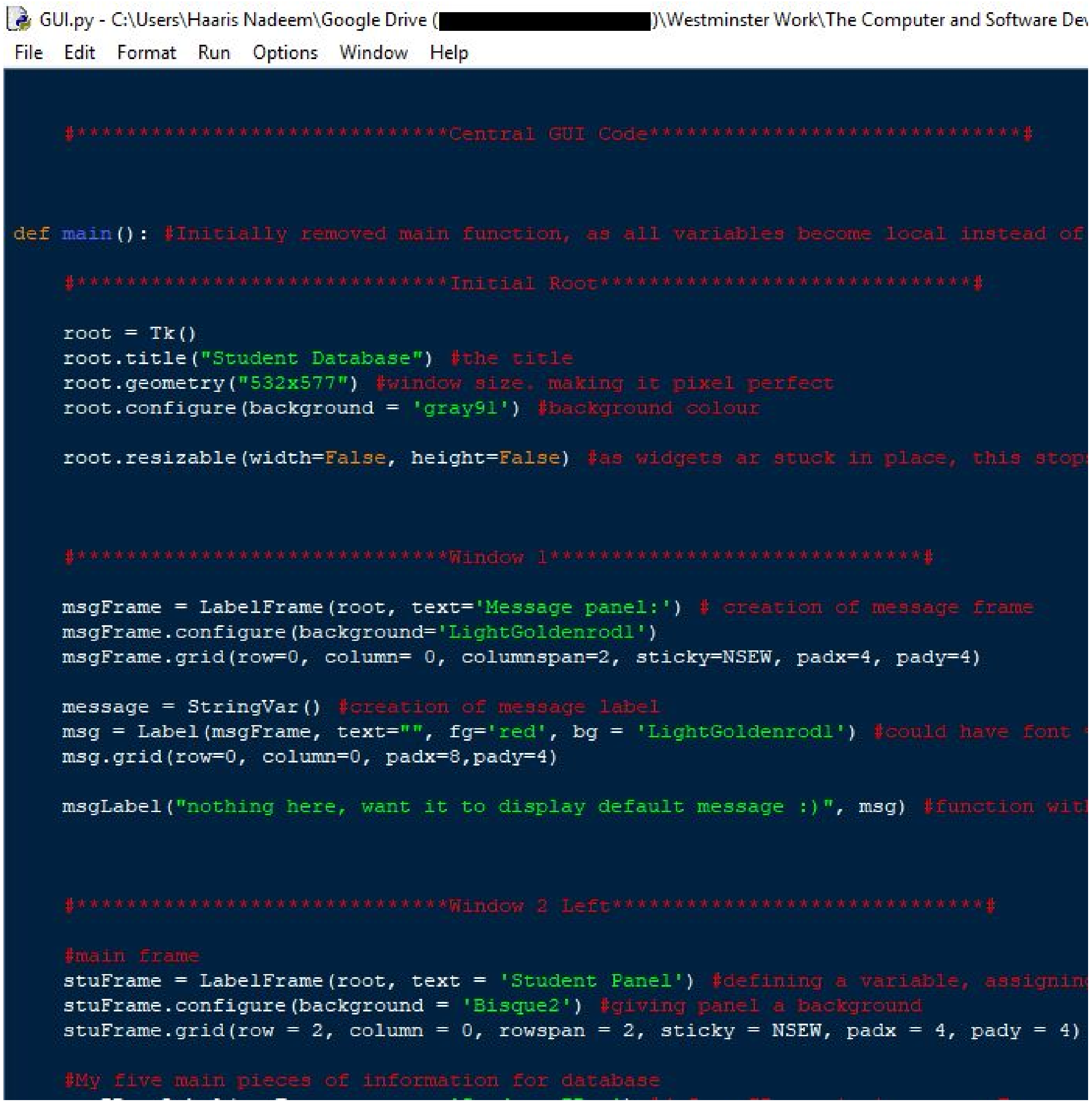
4. Full Code Overhaul (Global to Local)

While working on the coursework, the central code was initially defined as a function, “main()”. It was very quickly revealed that there was a difference between local and global variables, which is why the central code was shifted to be global early in development. It is in later lectures where the dangers of global code and (especially) global variables along with the safety and good practice of writing code locally prompted an endeavour to shift the entire completed central code to a main function, thereby having the whole project built in functions, with not a single variable globally available.

This was a difficult task to achieve, as every function would now have to be defined with parameters, and some functions would even run another function within them, which would only increase the number of parameters necessary to run. However, the endeavour was successfully completed with the implementation of two primary solutions;

1. In the case of buttons, any and all functions could be run with parameters the incorporation of “lambda:” before calling the function. While before any attempt to add parameters would instantly run the function and cause errors, lambda allowed for each function to receive its necessary variables to function.

2. In the case of .bind commands, there was a higher level of difficulty, as simply incorporating lambda was not a fix to the solution, due to the command handling an event. However, the solution that was discovered was “lambda event: function(event, x, y, z)”, where the event was now handled properly, and additional parameters could be added to the function.

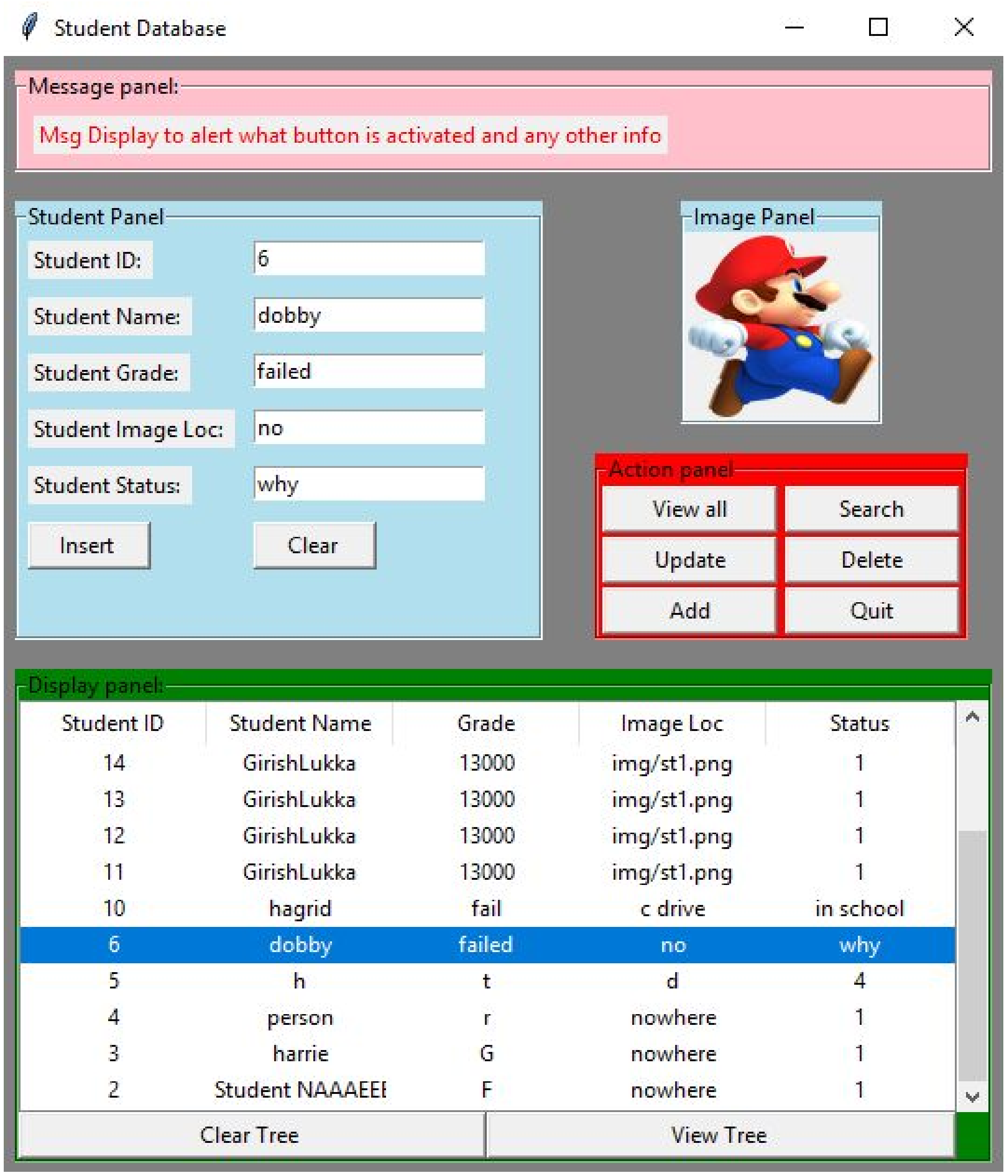


GUI.py - C:\Users\Haaris Nadeem\Google Drive ([redacted])\Westminster Work\The Computer and Software De

File Edit Format Run Options Window Help

```
    #********************************Central GUI Code********************************#


def main(): #Initially removed main function, as all variables become local instead of

    #********************************Initial Root********************************#

    root = Tk()
    root.title("Student Database") #the title
    root.geometry("532x577") #window size. making it pixel perfect
    root.configure(background = 'gray91') #background colour

    root.resizable(width=False, height=False) #as widgets ar stuck in place, this stop


    #********************************Window 1********************************#

    msgFrame = LabelFrame(root, text='Message panel:') # creation of message frame
    msgFrame.configure(background='LightGoldenrod1')
    msgFrame.grid(row=0, column= 0, columnspan=2, sticky=NSEW, padx=4, pady=4)

    message = StringVar() #creation of message label
    msg = Label(msgFrame, text="", fg='red', bg = 'LightGoldenrod1') #could have font
    msg.grid(row=0, column=0, padx=8,pady=4)

    msgLabel("nothing here, want it to display default message :)", msg) #function wit


    #********************************Window 2 Left********************************#

    #main frame
    stuFrame = LabelFrame(root, text = 'Student Panel') #defining a variable, assignin
    stuFrame.configure(background = 'Bisque2') #giving panel a background
    stuFrame.grid(row = 2, column = 0, rowspan = 2, sticky = NSEW, padx = 4, pady = 4)

    #My five main pieces of information for database
```

It was now later in the development of the project that primary focus was shifted to design of GUI. The generic design was manipulated to slowly morph closer to final design.


Student Database

Message panel:

Msg Display to alert what button is activated and any other info

Student Panel

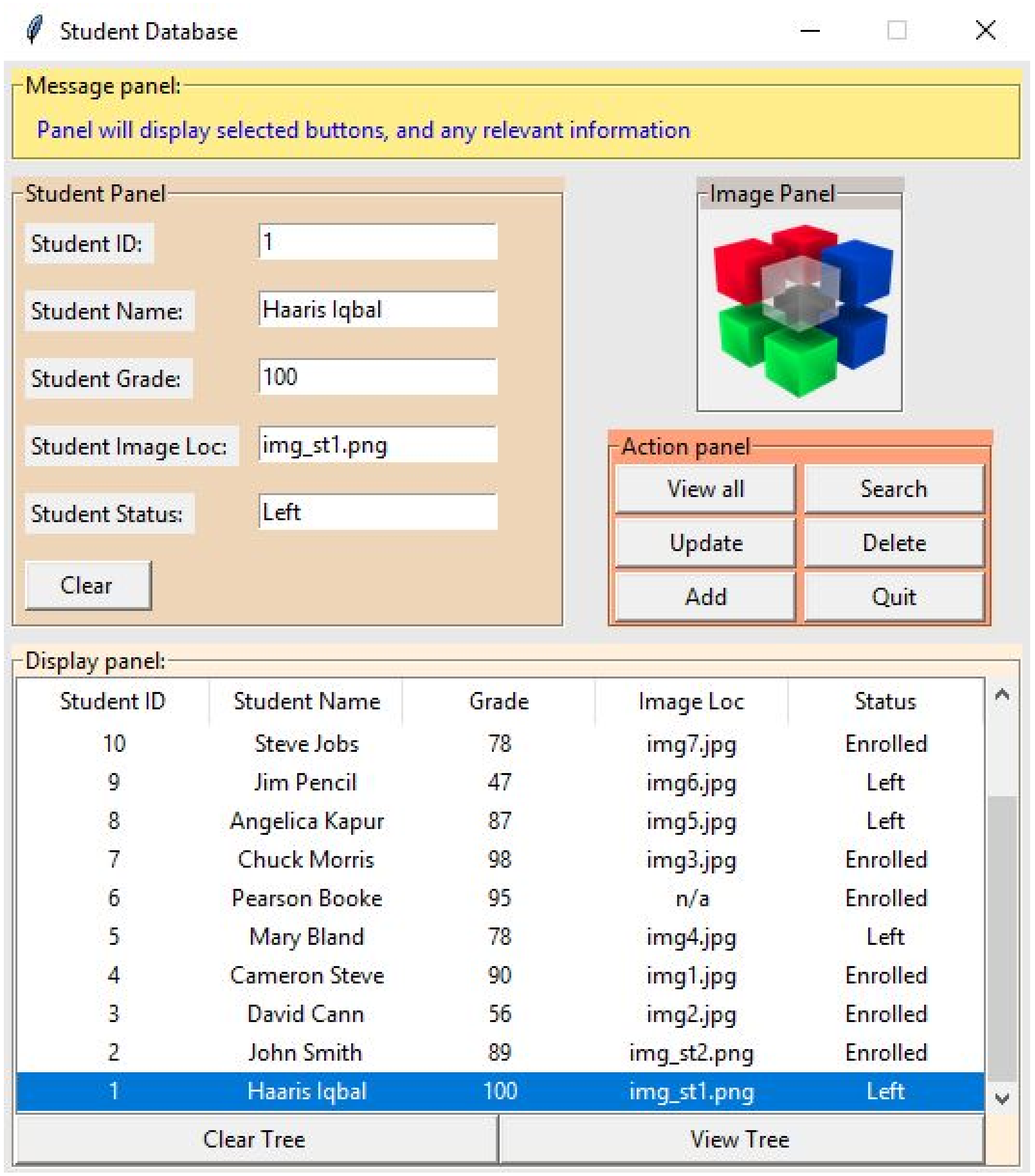
Student ID:	6
Student Name:	dobby
Student Grade:	failed
Student Image Loc:	no
Student Status:	why

Display panel:

Student ID	Student Name	Grade	Image Loc	Status
14	GirishLukka	13000	img/st1.png	1
13	GirishLukka	13000	img/st1.png	1
12	GirishLukka	13000	img/st1.png	1
11	GirishLukka	13000	img/st1.png	1
10	hagrid	fail	c drive	in school
6	dobby	failed	no	why
5	h	t	d	4
4	person	r	nowhere	1
3	harrie	G	nowhere	1
2	Student NAAAEEI	F	nowhere	1

Initially, the plan was to create a GUI with a dark theme, as illustrated above. However, there are a number of reasons this idea was abandoned in favour of a light theme;

- Light theme allowed for a more professional and clean look
- Light theme was less visually heavy and more easy to interact with
- Darker themes in Tkinter would look too cartoonish



Panel will display selected buttons, and any relevant information

Student ID: 1
Student Name: Haaris Iqbal
Student Grade: 100
Student Image Loc: img_st1.png
Student Status: Left

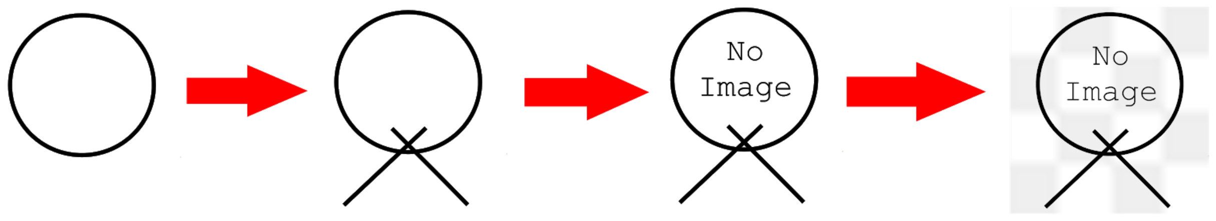
Student ID	Student Name	Grade	Image Loc	Status
10	Steve Jobs	78	img7.jpg	Enrolled
9	Jim Pencil	47	img6.jpg	Left
8	Angelica Kapur	87	img5.jpg	Left
7	Chuck Morris	98	img3.jpg	Enrolled
6	Pearson Booke	95	n/a	Enrolled
5	Mary Bland	78	img4.jpg	Left
4	Cameron Steve	90	img1.jpg	Enrolled
3	David Cann	56	img2.jpg	Enrolled
2	John Smith	89	img_st2.png	Enrolled
1	Haaris Iqbal	100	img_st1.png	Left

Above is a screenshot of the final GUI product. There are a number of reasons this layout was chosen. The message panel will always display important and relevant information, however there is no way to directly interact with the widget, which is why it has been placed on top. The middle section contains three widgets. The right space has been dedicated entirely to the student panel, allowing for easy interaction and viewing of information. The left space has the image panel on top, and the action panel on the bottom. This allows for intuitive interaction with action panel, while at the same time, a prominent view of the image. The display panel has the entire lower section dedicated to it, allowing for easy interaction of the full database, along with prominent view of entries. For each and every element extensive work was performed to ensure optimal functionality and quality.

The message panel, as well as having variable text, is colour coded and will shift colour depending upon the function called, and action performed. There are also unique colours and messages for unique actions, such as filling in all information but status, or filling in an invalid grade, etc. A detailed log of its capabilities were tested, and can be found in the test section below.

The Message panel works by calling to a function and handing it two inputs; one being a string key, used to call a custom message. And one being the message panel itself, to allow for the text to be variable (as label was defined in main function). If the string key does not correspond with any of the defined variables, the function will display the default message. The message panen has been given a column span of “2”, and is at the very top of the application (row = 0, column = 0). Upon the start of the GUI, the default message is displayed. The background of the text is set to match the colour of the message panel.

The Student panel is on the left, and features five main entries to be added as a record in the database. There are also two inputs programmed into the student panel; a clear button, and the ability to double click to clear the panel. Both actions have a custom message associated with them. Using the “child” command, all buttons and columns have been aligned to fit neatly in panel. The panel has been configured to have a row span of “2”. This panel has the dual purpose of also displaying information, as well as allowing one to modify and delete information.



The Image panel has been configured to be on the upper right. In the central code, a default width and height for every image to crop to is defined (in pixels), and the default image is opened upon bootup. It calls an image function, that will attempt to display an associate image with each selection. If no associated image is present, the default image is used. This is also the case for when the user has no longer selected an item (such as after clicking clear, delete, view, etc.) The default image is custom made.

The Action panel is on the lower right, beneath the Image Panel. It is built as a two by three grid, and houses a total of 6 buttons. All of these interactive elements have been tested extensively, and further information of each button can be found in test section below. Each button calls upon a function, and multiple variables are handed to the function parameters to allow for actions to take place, such as modifications to the database, messages from message panel, etc.

The Display panel has been given five main headings to correspond with the data of the database. In tree view, the database entries are displayed. select functionality has been programmed in, along with a scrollbar and two buttons; clear tree and view tree. A detailed testing and description of each interactive element can be found below, in test section.

Testing

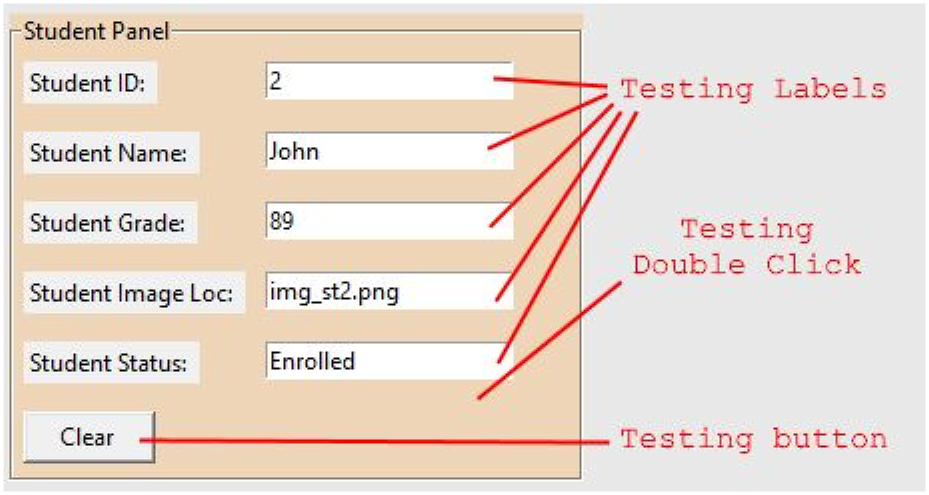
This section includes some of the testing that was completed for this coursework project.

Database Testing Date(s) Conducted: 10th - 13th March, 2019		
Test Description	**Expected Functionality**	**Observed Outcome**
Database created	With cur.execute() command implemented, database file should appear in same directory as .py file	Database file appeared in same directory as .py file
Table created	cur.execute() should also attempt to connect to existing table, or create new table if not present.	console displayed table, and blank table created
CRUD function - Insert	Menu recognizes multiple commands for "insert", based on list. "add()" function is called from functions.py file via "functions.add()". "add()" function prompts user for 4 more inputs, then plugs inputs into database.	A new record was successfully added to database Auto increment of ID is working as intended
Crud Function - Amend	Menu recognizes multiple commands	In case where Valid ID given, the record

	for “amend”, based on list. “update()” function is called from functions.py file via ”functions.update()”. “update()” function first prompts user to input valid ID. If no valid ID given, user is informed. If valid ID inputted, the old record is displayed, and user if given ability to add 4 more inputs to replace old inputs.	was amended with new details. In case where no valid ID given, custom error message was displayed to user, and menu booted up again
Crud Function - Delete	Menu recognizes multiple commands for “delete”, based on list. “delete()” function is called from functions.py file via ”functions.delete()”. “delete()” function first prompts user to input valid ID. If no valid ID given, user is informed. If valid ID inputted, the record is removed from database, and user is informed. All records are shown	In case where valid ID was given, record was removed from database, and all records were displayed. In case where no valid ID given, custom error message was displayed to user, and menu was booted up again
Crud Function - View	Menu recognizes multiple commands for “view”, based on list. “all()” function is called	All current available records were displayed Note;

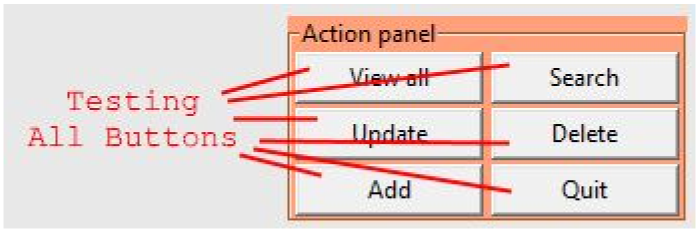
	from functions.py file via ”functions.all()”. “all()” function simply selects all records from database, and displays them to user	Alongside basic CRUD functions, more functions have also been implemented (such as Search, Quit, etc). These were tested in console, and ensured to be working correctly
GUI attached to database as evidenced in some of the GUI tests.	Interaction with GUI would manipulate the table (extensively tested in GUI tests below)	Through checking outcome first in console, then in GUI itself, the GUI and all of its purpose built buttons and design interact with database in background as intended; hence GUI has been successfully attached to the database, and all functions have been tweaked to continue to work as designed

Important Note: All tests automatically incorporate Message Panel and Image Panel, however user does not directly interact with them, which is why their tables are not distinct.



GUI Testing Section 1: Student Panel Date(s) Conducted: 05th - 06st April, 2019		
Test Description	**Expected Functionality**	**Observed Outcome**
Test of 5 labels	All 5 created with string variables, simply need to test if they are capable of holding variables	Yes, able to hold variables in labels
Clear Button Test	clearData function invoked with 8 variables. 5 used to pickup and erase data from all 5 labels. 1 used to call default image. Final two used to run message function with custom message	Upon clicking clear button, all variables present in label fields are erased. Default image is loaded, and user is informed of clear button having cleared the panel, with custom colour
Double Click on Frame Test	clearStu function invoked with 8 variables. 5 used to pickup and erase data from all 5 labels. 1 used to	Upon double clicking, all variables present in label fields are erased. Default image is loaded, and

	call default image. Final two used to run message function with custom message	user is informed of having cleared the panel via action of double click, with custom colour



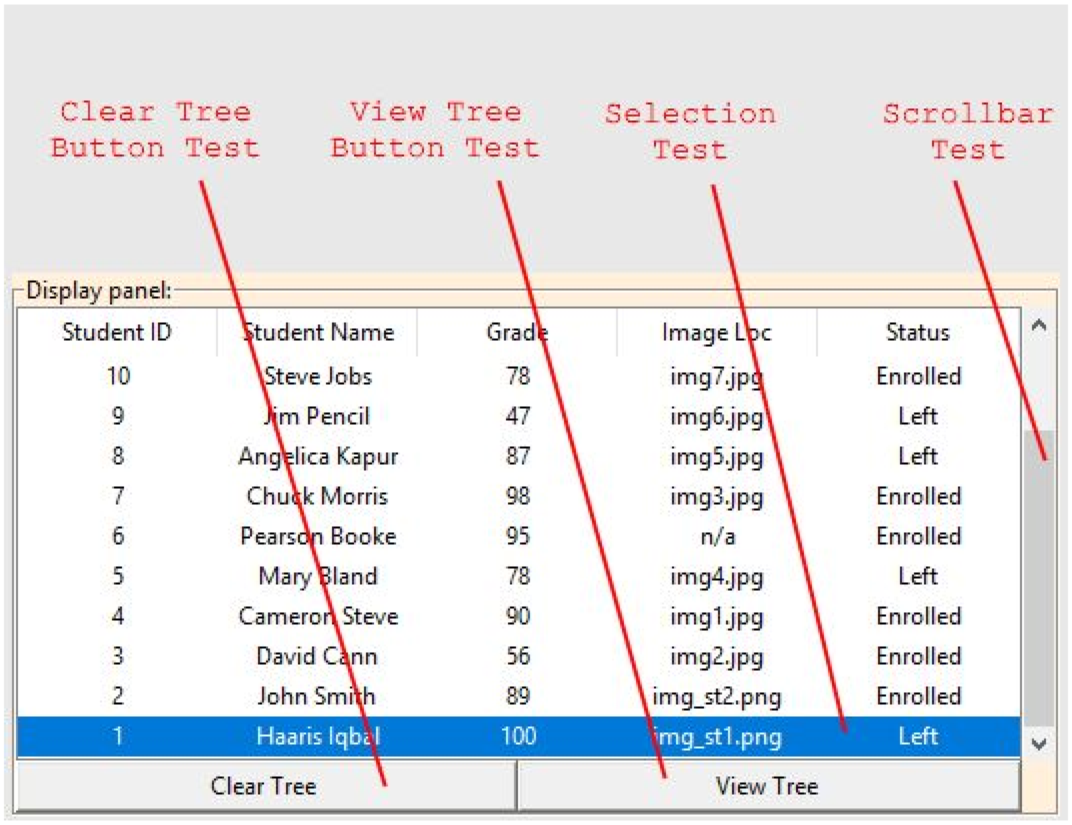
GUI Testing Section 2: Action Panel Date(s) Conducted: 05th - 06st April, 2019		
Test Description	**Expected Functionality**	**Observed Outcome**
View All button test	Upon click, viewAll function should be invoked with 4 variables in parameters. In viewAll function, before insertion of records, all 4 variables used on clearTree function to clear tree, then at end, 2 variables used to call custom message to message panel	In an instant, any information on the display panel is erased. It is then replaced with all the available record in database. The message panel informs user of action, with custom message and colour
Search button test	Upon click, search function should be invoked with 8 variables in parameters. 4 variables are passed	The information that is typed into the Student Panel is globally searched and displayed in display panel. A

	on to functions.search() to search database. Once found, results are returned to function, clearTree is invoked with 4 variables, returned records are inserted into tree, and message function is invoked with two variables	custom message pops up informing user of action.
Update button test	Upon click, update function should be invoked with 9 variables in parameters. There is a check that 5 variables (labels) are not empty. If they are empty, custom error message displayed with message function and 2 variables. If not, there is an attempt to change variable “grade” to integer. If unable to do so, custom error message displayed with message function and 2 variables If able, there is now a check to ensure grade is in range 0 to 100	There are many scenarios that were checked with update button. When no information entered, error message in red is displayed, along with instruction on what must be inputted. When grade is not integer, error message in red is displayed, along with instruction on what must be inputted. When grade is out of range, error message in red is displayed, along with instruction on what must be inputted When status is invalid, error

	inclusive. If outside range, custom error message displayed with message function and 2 variables If within range, variable “status” is made lowercase with first letter capitalized. There is a check to ensure status is either “Enrolled” or “Left”. If neither, custom error message displayed with message function and 2 variables If true, then finally 5 variables passed on to functions.update to update record with new information. custom message is displayed with message function and 2 variables.	message in red is displayed, along with instruction on what must be inputted. When all criterions filled, record is amended with new information
Delete button test	Upon click, delete function should be invoked with 5 variables in parameters. Using variable ID, record is passed to functions.py and record is removed. Next 4 variables used to call viewAll function. 2	Upon selecting record and clicking delete, a record is deleted from the database, and all the records are displayed in display panel .A custom message in custom colour is displayed

	variables used for message function	
Add button test	Upon click, add function should be invoked with 8 variables in parameters. There is a check that 5 variables (labels) are not empty. If they are empty, custom error message displayed with message function and 2 variables. If not, there is an attempt to change variable “grade” to integer. If unable to do so, custom error message displayed with message function and 2 variables If able, there is now a check to ensure grade is in range 0 to 100 inclusive. If outside range, custom error message displayed with message function and 2 variables If within range, variable “status” is made lowercase with first letter	There are many scenarios that were checked with add button. When no information entered, error message in red is displayed, along with instruction on what must be inputted. When grade is not integer, error message in red is displayed, along with instruction on what must be inputted. When grade is out of range, error message in red is displayed, along with instruction on what must be inputted When status is invalid, error message in red is displayed, along with instruction on what must be inputted. When all criterions filled, new record is added

	capitalized. There is a check to ensure status is either “Enrolled” or “Left”. If neither, custom error message displayed with message function and 2 variables If true, then finally 5 variables passed on to functions.add to add record to database. Custom message is displayed with message function and 2 variables.	
Quit button test	Upon click, viewAll function should be invoked with 1 variable in parameters. The variable is the central window “root”, which is closed with .destroy()	The quit button shuts the GUI



Student ID	Student Name	Grade	Image Loc	Status
10	Steve Jobs	78	img7.jpg	Enrolled
9	Jim Pencil	47	img6.jpg	Left
8	Angelica Kapur	87	img5.jpg	Left
7	Chuck Morris	98	img3.jpg	Enrolled
6	Pearson Booke	95	n/a	Enrolled
5	Mary Bland	78	img4.jpg	Left
4	Cameron Steve	90	img1.jpg	Enrolled
3	David Cann	56	img2.jpg	Enrolled
2	John Smith	89	img_st2.png	Enrolled
1	Haaris Iqbal	100	img_st1.png	Left

GUI Testing Section 3: Display Panel Date(s) Conducted: 05th - 06st April, 2019		
Test Description	**Expected Functionality**	**Observed Outcome**
Selection of data test	Upon selection event, select_item function is called with 13 parameters. Function selects correspond data from database. Student panel and all labels are cleared with clear function. Data from entry is inserted into student panel. Image	Upon selection of any data, the data appears in the student panel where it can be viewed and manipulated. A corresponding image also pops up if there is a valid location, else the default image pops up.

	function is run, message function is run. In image function, there is an attempt to load image from data provided in record. If location is invalid, default image is displayed in its place	Message panel displays its default message, as no major action has taken place yet.
Scrollbar test	Scrollbar is created horizontally on the left, within Display Panel	When there are enough entries, scrollbar allows user to scroll up and down the display panel
Clear Tree button test	Upon clicking button, clearTree function is invoked with 4 parameters. In function, each column of Display panel is erased. Image function is run, to display default image. Message function is run with 2 parameters.	Upon click of button, the display panel is erased. The default image pops up, and the message panel informs the user of the action that has taken place
View Tree button test	Upon clicking button, viewTree function is invoked with 4 parameters. In function, the viewAll function is simply run, and handed the 4 parameters. Message function is also run with 2 parameters.	Upon clicking the button, all information in the Display panel is erased in an instant. It is then replaced with all records of database. A custom message is displayed to the user, informing them

		of action that has taken place

Additional GUI Checklist Date(s) Conducted: 29th March, 2019	
GUI has appropriate title, size, color.	Yes - GUI is a Student Database, which is why title is fitting - Size has been made pixel perfect. As the widgets are stuck in place, I have also taken away the ability for the user to resize the window, and the Database will spawn with my custom tailored size, to ensure best experience - Color is suitable. Originally, I was going to implement dark design elements, but I have chosen light design elements as they are easier on the eyes, more pleasant to work on, and look more professional
GUI is visible and remains visible until quit button is activated.	Yes - This is done by simply implementing .mainloop() code at end - Quit button functions as intended, and immediately shuts program upon click
GUI is well designed and organized in a logical manner –use of 'panels' for table data, buttons, etc. & Widget alignment is fit for purpose	Yes - Over many iteration, a lot of planning went into the position of my widgets. (See photos in document) For final iteration; - Message board has been placed on top, as user will never have to interact with message board, they simply view it - On the middle-left, entire space has been dedicated to Student Panel, allowing for comfortable input, and it

	is best spot for user to primarily view as it will also display the output from the Display Panel - On the middle right, upper portion has been dedicated to showing the image, Lower portion has been dedicated to the buttons - The Action Panel buttons have been designed to be rectangular, and are spaced in a 2x3 grid, allowing for easy access, and visually looks neat - Finally, lower half has been dedicated entirely to the Display Panel. This is the most primary area of interaction for the user, so location is fitting for mouse presses - In miscellaneous; clear buttons are strategically placed in lower left locations to allow user to quickly clear if they wish to do so
Button widgets function as required - View, Add, Delete, Amend, Search, Clear	Yes - Every single one of these buttons has been implemented in the GUI, in appropriate locations
Scrollbars used	Yes - Scroll Bar has been implemented successfully in display panel
Widget binding to clear display panel, clear employee panel, select individual elements of a record for display in student panel	Yes - Rather than have a clear button on the Action Panel, I have incorporated two of them, each in the bottom left of their respective Panel - This has two purposes. It ensures that the Action Panel looks neat, as there are an even 6 buttons on it, rather than an odd 7 buttons - It also allows for faster access, and is more intuitive - Selecting individual records in the Display Panel has also been

	implemented, output appears on Student Panel (giving it dual purpose)
Default image display when first launched & Image changes depending on record highlighted	Yes - By incorporating code that will call for the default image in the buttons "View All", "Clear", etc, there is always a default image displayed, until a valid image location is provided - And for any entry without a valid image location, the default image is called in its place, once again ensuring it is always displayed - Made a custom image that will act as a default, and is fit for purpose in describing to user that no image is associated with selected record, while also looking visually neat