Software Major Work Documentation **Matthew Hill**

Problem Definition 1.1

My app, 'Custom Quizzer' solves the issue of paywalls and in app purchases in self-customisable quizzes for Saint Augustine's students. Apps with such paywalls and in app purchases include Kahoot, Quizlet and Blooket. My app aims to help students from Saint Augustine's test their knowledge and education. The app will function as a self-customisable quiz where users can either test themselves or others by making their own quiz or using the example quizzes provided. The quiz will check all answers given from the user and compile their score and provide feedback on their understanding and progress of the topic. My app utilises Excel spreadsheets to compile questions in the quiz, so the user of the app must have at minimum, a limited knowledge within Excel spreadsheets in order to create their own quizzes. The primary issue that is to be tackled with my app is that it allows Saint Augustine's students to compile their own quizzes with their own topics that they want to be tested on without facing paywalls, in app purchases and advertisements.

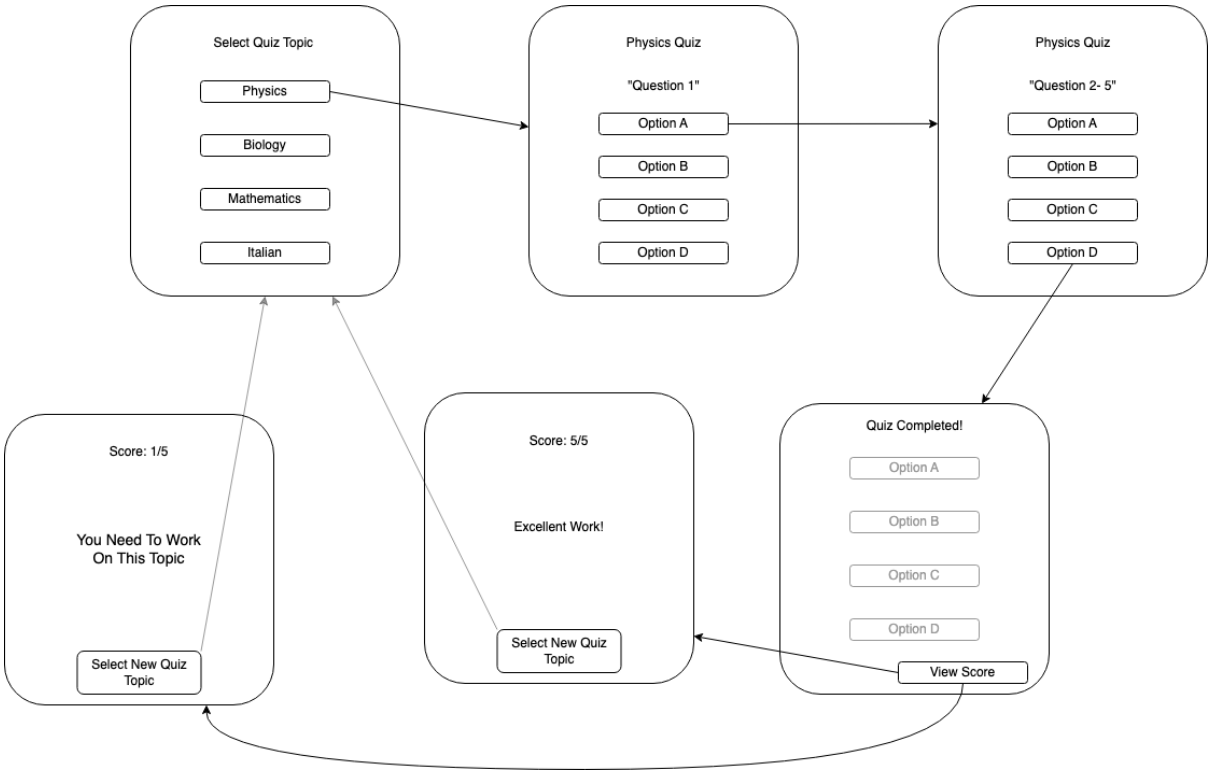
Legal and ethical considerations 1.2

My app, 'Custom Quizzer' does not require any personal information as there is no log in or registering features, therefore it doesn't breach any of the user's data and privacy. The software is somewhat accessible as it functions as a knowledge quiz on certain areas the user is studying or trying to extend their knowledge on. However, the user must be able to operate Excel spreadsheets in order to compile their own sets of quiz questions, options and answers, this limitation of my software should be minimal as the extent of usage on Excel spreadsheets is limited and only requires a basic knowledge of Excel for the user to type their quizzes into the file. 'Custom Quizzer' is similar to other online quiz applications and webpages that allow you create and test yourself on your own topics. This could result in some issues regarding intellectual property. However, as the concept of an online customisable quiz is very common it cannot be claimed as intellectual property of one person, therefore 'Custom Quizzer' is not an infringement of any intellectual property.

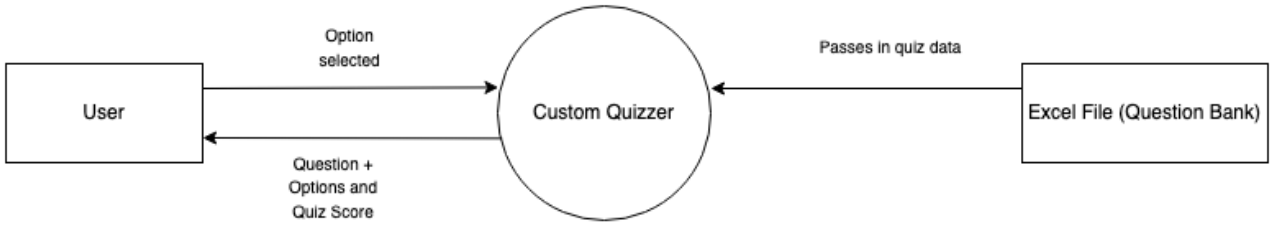
Functional and non-functional requirements 1.3

Functional:	Non-functional:
The quiz will not have any paywalls or in app purchases	App will have simple multiple-choice quiz not requiring a tutorial
Users must make custom questions through an Excel file	The app will compile questions inputted by user from an Excel csv file
App will allow user to select topic of quiz	A choice of 4 subjects the user can choose to be quizzed on
The app will give user feedback on how they performed on the quiz	Once completed, the quiz will provide the user with a score out of 5
User should be notified once quiz is completed	The quiz will print 'Quiz completed' and prompt the user to view their score on the quiz

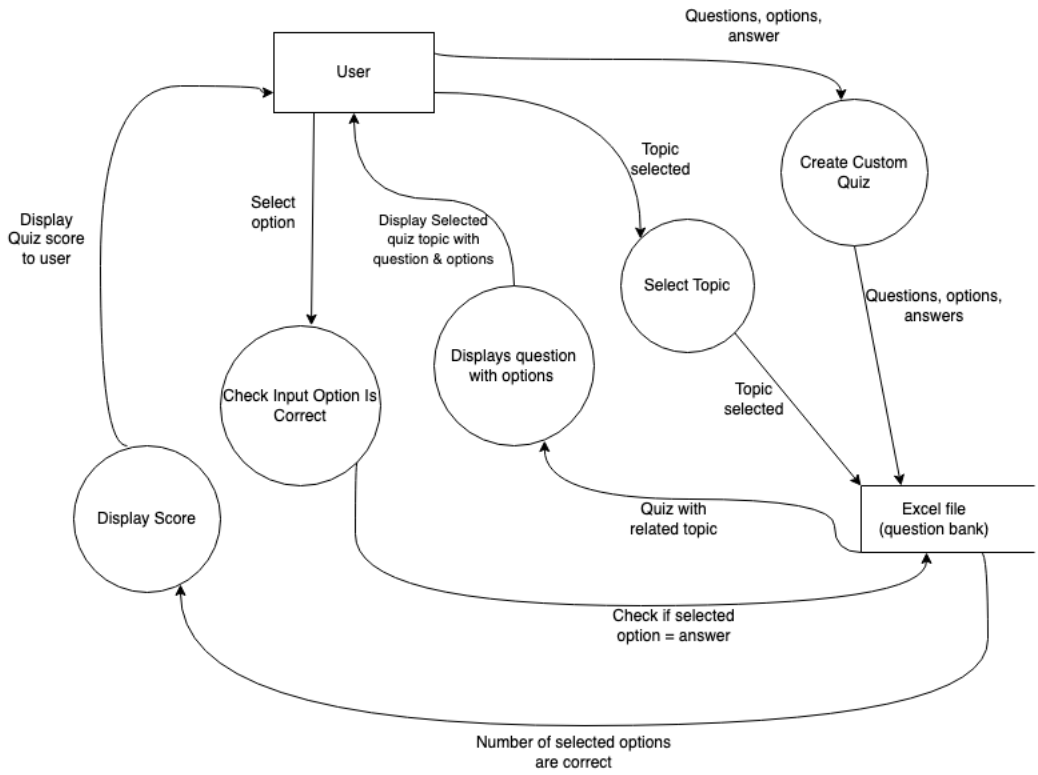
Story Board 2.1



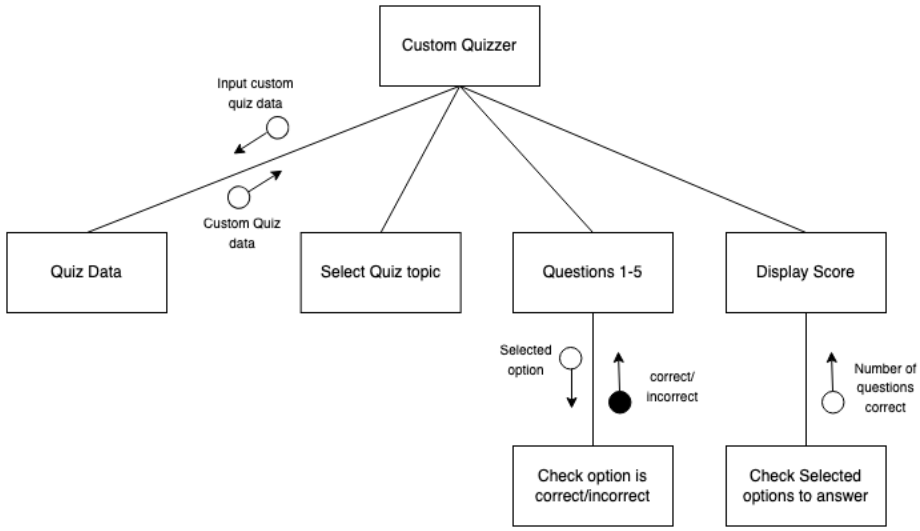
Context Diagram 2.2



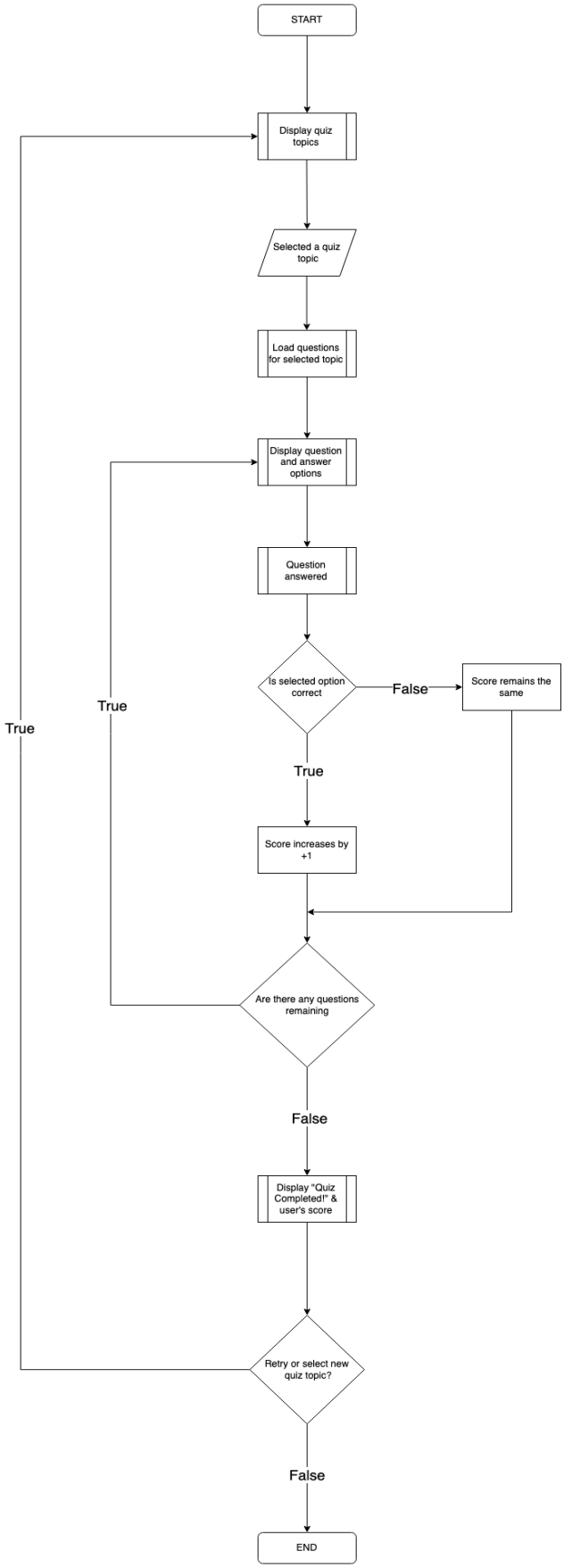
Data Flow Diagram 2.3



Structure Chart 2.4



Algorithms 2.5



Algorithmic description examples:

check_answer function:

- Receive Input: User selects an option to answer quiz question
- Check Input: User input is checked against correct answer
- Validate Input: If selected option = correct increase score by +1, if not score remain the same

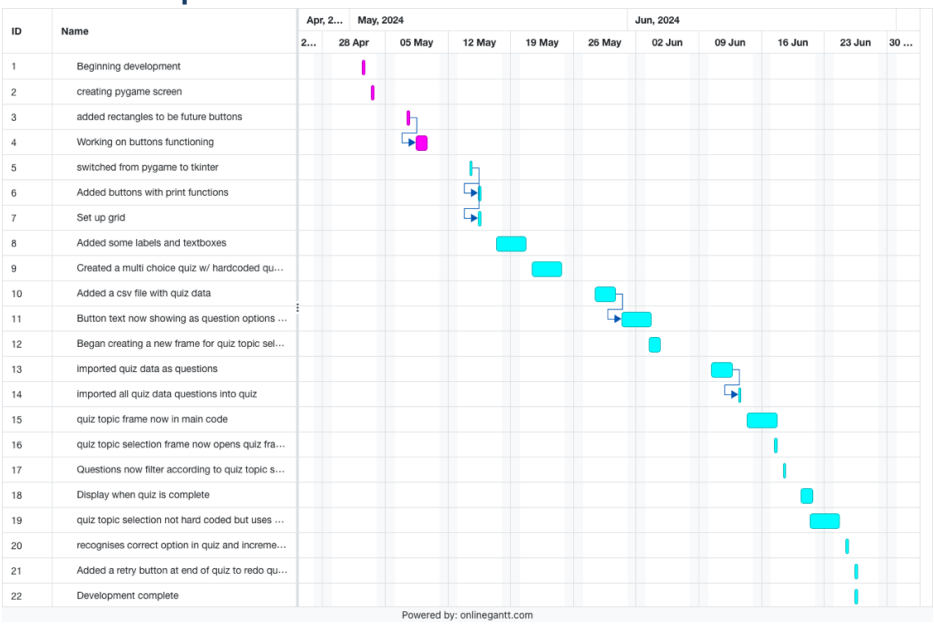
option_buttons function:

- Display buttons: subject selection buttons are displayed within the frame
- Receive Input: User selects a subject selection button
- The selected button then opens quiz frame with related questions

Gantt Charts 2.6

Pre-Development

Post-Development



(Please refer to documentation folder in GitHub repo to access Gantt chart files and open in onlinegantt.com to view further)

Reflection:
In the development of my project, I faced many issues throughout, right from the start I encountered import errors with pygame, after this I was struggling with development in pygame and made the switch to tkinter. Many other issues occurred resulting in stunts in development progression, this ultimately led to a large part of the development being done within the last 2 weeks of the due date. However, I was able to finish development of my project although the cramming of development towards the end may have affected the overall limitations of the code.

Development Log 3.1

Log entry 1 2/5/24 Week 1

Update:
I began to develop my major work with the idea of creating a quiz app that users can input their own questions to test themselves on. I began developing my code through Pygame and I had created a frame which was viewable on screen.

Multi choice pygame interface.py

Multi choice pygame interface.py > ...

```python
import pygame

pygame.init()

screen_width = 800
screen_height = 800

screen = pygame.display.set_mode((screen_width, screen_height))

run = True
while run == True:

    for event in pygame.event.get():
        if event.type == pygame.quit:
            run = False

pygame.quit()
```

This is a screenshot of how my code is currently going. The dimensions of the frame have been established along with the app being able to close.

Problems & solutions
When I began developing through Pygame I ran into some issues with importing Pygame and had to uninstall VisualStudioCode and reinstall to fix the issue.

Log entry 2 13/5/24 Week 3

Update:
I have begun to add buttons into the frame which will become the buttons the user will press to select an option within their quiz. Although there have been some difficulties with Pygame the buttons are now successfully appearing on the screen and I have been changing colours, size and placement of rectangle buttons within the frame.

multi choice btns.py

Users > matthewhill > Desktop > multi choice btns.py > ...

```python
import pygame
import sys

# Initialize Pygame
pygame.init()

# Set up the display
width, height = 400, 300
screen = pygame.display.set_mode((width, height))
pygame.display.set_caption("Multiple choice quiz")

# Colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
GRAY = (200, 200, 200)

# Fonts
font = pygame.font.SysFont(None, 40)

# Button class
class Button:
    def __init__(self, text, pos):
        self.text = text
        self.pos = pos
        self.width, self.height = 200, 50
        self.rect = pygame.Rect(self.pos, (self.width, self.height))

    def draw(self):
        pygame.draw.rect(screen, GRAY, self.rect)
        text_surface = font.render(self.text, True, BLACK)
        text_rect = text_surface.get_rect(center=self.rect.center)
        screen.blit(text_surface, text_rect)

# Create buttons
A_button = Button("Option A", (100, 100))
B_button = Button("Option B", (100, 200))

# Main loop
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    screen.fill(WHITE)
    A_button.draw()
    B_button.draw()
    pygame.display.flip()

pygame.quit()
sys.exit()
```

this is the current state of my code, showing the two different rectangles being displayed on the frame, along with the colours and the title of the frame.

Problems & solutions

Right now, my issues with adding rectangles to later become buttons is that they are only rectangles and do not function as buttons yet, I am working towards a solution to this and making them function as buttons when pressed.

Log entry 3 14/5/24 Week 3

Update:
I have decided to switch from Pygame to tkinter as I am struggling with pygame to creating the quiz frame. Iggy has helped me switching over and progress seems to be moving a lot smoother as I have now created a frame, set up a grid and changed the appearance and colour of the frame.

multi choice tk.py

Users > matthewhill > Desktop > multi choice tk.py > page > __init__

```python
import customtkinter


class page():
    def __init__(self):
        self.app = customtkinter.CTk("Mutiple Choice Quiz") #creating the app
        self.frame = customtkinter.CTkFrame(self.app)
        self.app.geometry("400x400")
        customtkinter.set_appearance_mode('light')
        customtkinter.set_default_color_theme('dark green')

        self.frame.pack(pady=20, padx=60, fill='both', expand=True)


        for i in range (12):
            self.frame.grid_columnconfigure
```

Screenshot of code showing the progress I have started in tkinter.

Problems & solutions:
Now that I have switched over to tkinter it will take some time to learn and figure it out, but I can learn it through help from others and Customtkinter widgets.

Log entry 4 30/5/24 Week 5

Update:
I have added a question bank in an Excel file which the user will access to and will be able to enter their own questions into. There are currently some example questions in there. I have also used pandas to bring the questions into the main python file, the main python file now functions as a simple multiple-choice quiz with hard coded questions.

	A	B	C	D	E	F	G
1	subject	question	option_a	option_b	option_c	option_d	answer
2	Biology	What is the powerhouse of the cell?	mitochondria	nucleus	cell wall	cytoplasm	a
3	Biology	Which organelle is responsible for energy production in the cell?	nucleus	ribosome	mitochondrion	golgi apparitus	c
4	Biology	What is the process by which a DNA sequence is copied into mRNA?	translation	replication	transcription	transformation	c
5	Biology	Who is known as the father of evolution?	Gregor Mendel	Charles Darwin	Louis Paster	Carl Lnneaus	b
6	Biology	What are the building blocks of proteins?	nucleotides	monosaccharides	fatty acids	amino acids	d
7	Physics	What is the unit of force in the International System of Units (SI)?	Pascal	Newton	Joule	Watt	b
8	Physics	What is the SI unit of electric current?	Volt	Coulomb	Ampere	Ohm	c
9	Physics	Who developed the theory of relativity?	Issac Newton	Albert Einstien	Niels Bohr	James Clerk Maxwell	b
10	Italian	What is the Italian word for 'book'?	Penna	Tavolo	Libro	Sedia	c
11	Italian	How do you say "How are you?" in Italian?	Come stai?	Che ore sono?	Dove vai?	Cosa fai?	a
12	Italian	What is the Italian word for "red"?	Blu	Verde	Giallo	Rosso	d
13	Physics	What is the acceleration due to gravity on Earth's surface?	8.9m/s/s	9.8m/s/s	10.2m/s/s	11.1m/s/s	b
14	Physics	What particle is emitted during beta decay?	Proton	Neutron	Electron	Photon	c
15	Italian	How do you say the number "fifteen" in Italian?	quattordici	tre	quindici	nova	c
16	Italian	Which of the following is the Italian word for "Wednesday"?	Mercoledi	Lunedi	Giovedi	Martedi	a
17	Mathematics	What is the sum of the interior angles of a triangle?	90 Degrees	180 Degrees	270 Degrees	360 Degrees	b
18	Mathematics	What is the derivative of f(x) =x^2?	1/2x	2x	x	1x	b
19	Mathematics	What is 25% of 200?	100	40	50	25	c
20	Mathematics	What is the value of sin(90)?	-1	0	-2	1	d
21	Mathematics	What is the next number in the sequence: 2, 4, 8, 16, ___?	20	24	64	32	d

This screenshot shows the question bank with example questions in it with the necessary subject filters, options and answers to the questions.

```python
import pandas as pd


# Load the data from the CSV file
df = pd.read_csv('questions.csv')

quiz_data = []
for index, row in df.iterrows():
    question_dict = {
        "subject": row["subject"],
        "question": row["question"] if pd.notna(row["question"]) else "No question provided",
        "options": [row["option_a"], row["option_b"], row["option_c"], row["option_d"]],
        "correct": row["answer"]
    }
    quiz_data.append(question_dict)

# Print the quiz data to verify
for question in quiz_data:
    if question["subject"] == "Physics":
        print(question)
```

This screenshot shows the pandas import which pulls the questions from the Excel file and brings them into the main python quiz file.

```python
import customtkinter

class Page(customtkinter.CTk):  # Inherit from customtkinter.CTk
    def __init__(self):
        super().__init__()  # Initialize the superclass

        self.geometry("750x750") # Window size
        customtkinter.set_appearance_mode("dark")  # Setting appearance mode
        customtkinter.set_default_color_theme("blue")  # Setting theme

        self.frame = customtkinter.CTkFrame(self)
        self.frame.pack(pady=20, padx=20, fill='both', expand=True)

        self.feedback_label = customtkinter.CTkLabel(master=self.frame, text='Quiz Title', fg_color='transparent') #setting label dimensions + label name/title
        self.feedback_label.grid(row=0, column=4, sticky='nsew') #label in grid

        self.option_buttons = []
        for i in range(4):
            buttonA = customtkinter.CTkButton(master=self.frame, text='A', command=lambda i=i: self.check_answer(i)) #button commands + labels
            self.option_buttons.append(buttonA)
            buttonA.grid(row=i + 1, column=4, padx=10, pady=10)    #setting buttons in grid

        for i in range(9):
            self.frame.grid_columnconfigure(i, weight=3)
            self.frame.grid_rowconfigure(i, weight=1)


        self.questions = [
            {
                "question": "What is the capital of France?",
                "options": ["Paris", "London", "Rome", "Berlin"],
                "correct": 0
            },
            {
                "question": "What is 2 + 2?",
                "options": ["3", "4", "5", "6"],
                "correct": 1
            }
            # Add more questions as needed
        ]

        self.current_question_index = 0
        self.load_question()

    def load_question(self):
        question_data = self.questions[self.current_question_index]
        self.feedback_label.configure(text=question_data["question"])
```

This screenshot is of the main python quiz file and how it is functioning as a simple multiple-choice quiz

Problems & solutions:

The questions in the quiz are hard coded and need to be taken from the excel bank of questions for it to function as a custom quiz. Also, a subject choice frame needs to be added so the user can choose the topic of questions they will be asked.

Log entry 5 13/6/24 Week 7

Update:
Now working on adding a subject choice frame which allows the user to select what topic of questions they want to be tested on

```python
from typing import Optional, Tuple, Union
import customtkinter

class Page(customtkinter.CTk):
    def __init__(self):
        super().__init__()

        self.geometry('500x500')
        customtkinter.set_appearance_mode("dark")
        customtkinter.set_default_color_theme('blue')

        self.frame2 = customtkinter.CTkFrame(self)
        self.frame2.pack(pady =10, padx =10, fill='both', expand=True)

        self.Label2 = customtkinter.CTkLabel(master=self.frame2, text='Select Quiz Topic', fg_color='transparent')
        self.Label2.grid(row=0, column=3, sticky='nsew')

        self.options_buttons = []
        for i in range(4):
            Butto (variable) options_buttons: list =self.frame2, text='', command=lambda i=i: self.check_answer(i))
            self.options_buttons.append(Button1)
            Button1.grid(row=i + 1, column=1, pady=5, padx=5)

        for i in range(10):
            self.frame.grid_columnconfigure(i, weight=3)
            self.frame.grid_rowconfigure(i, weight=1)


        self.questions = [
            {
                'question': 'Select Quiz Topic',
                'options': ['physics', 'biology', 'mathematics', 'italian'],
                'correct': 0
            }
        ]


        self.current_question_index = 0
        self.load_question()

    def load_question(self):
        question_data = self.questions[self.current_question_index]
        options = question_data['options']
        for i, option in enumerate(options):
            self.options_buttons[i].configure(text=option)




def run():
    app = Page()
    app.mainloop()

if __name__ == "__main__":
    run()
```

Problems & solutions:
The code does acknowledge when an option has been selected but there is no function that runs when this happens. When an option is selected it should open a new frame with the appropriate questions.

Log entry 6 15/6/24 Week 7

Update:

Code now will open a new frame when an option of quiz topic has been selected, the new frame will have all the quiz questions from the question bank.

```python
import customtkinter
from get_questions import import_quiz_data

quiz_data = import_quiz_data()


class Page(customtkinter.CTk):  # Inherit from customtkinter.CTk
    def __init__(self):
        super().__init__()  # Initialize the superclass

        self.geometry('500x500')
        customtkinter.set_appearance_mode("dark")
        customtkinter.set_default_color_theme('blue')

        self.subject_frame = customtkinter.CTkFrame(self)
        self.subject_frame.pack(pady =10, padx =10, fill='both', expand=True)

        self.Label2 = customtkinter.CTkLabel(master=self.subject_frame, text='Select Quiz Topic', fg_color='transparent')
        self.Label2.grid(row=0, column=3, sticky='nsew')

        self.options_buttons2 = []
        self.options = ['Physics', 'Biology', 'Mathematics', 'Italian']
        for i, option in enumerate(self.options):
            Button = customtkinter.CTkButton(master=self.subject_frame, text=option, command=lambda opt=option: self.open_frame(opt))
            self.options_buttons2.append(Button)
            Button.grid(row=i + 1, column=1, pady=5, padx=5)

        for i in range(10):
            self.subject_frame.grid_columnconfigure(i, weight=3)
            self.subject_frame.grid_rowconfigure(i, weight=1)


        self.frame = customtkinter.CTkFrame(self)
        self.frame.pack_forget()

        self.questions = quiz_data
        self.current_question_index = 0

    def open_frame(self, option):
        print(f"Selected option: {option}")
        self.subject_frame.pack_forget()
        self.frame.pack(pady=20, padx=20, fill="both", expand=True)

        self.label = customtkinter.CTkLabel(master=self.frame, text="Quiz Title", fg_color="transparent")
        self.label.grid(row=0, column=0, columnspan=4, sticky="news")

        self.option_buttons = []
        for i in range(4):
            buttonA = customtkinter.CTkButton(master=self.frame, text='A', command=lambda i=i: self.check_answer(i)) #button commands + labels
            self.option_buttons.append(buttonA)
            buttonA.grid(row=i + 1, column=4, padx=10, pady=10)    #setting buttons in grid

        for i in range(9):
            self.frame.grid_columnconfigure(i, weight=3)
            self.frame.grid_rowconfigure(i, weight=1)


        # self.questions = quiz_data
        self.questions = [

            quiz_data[0],
            quiz_data[1],
            quiz_data[2],
            quiz_data[3],
            quiz_data[4],
            quiz_data[5],
            quiz_data[6],
            quiz_data[7],
            quiz_data[8],
            quiz_data[9],
            quiz_data[10],
            quiz_data[11],
            quiz_data[12],
            quiz_data[13],
            quiz_data[14],
            quiz_data[15],
            quiz_data[16],
            quiz_data[17],
            quiz_data[18],
            quiz_data[19],
        ]

        self.current_question_index = 0
        self.load_question()

    def load_question(self):
        question_data = self.questions[self.current_question_index]
        self.feedback_label.configure(text=question_data["question"])

        for i, option in enumerate(question_data["options"]):
            self.option_buttons[i].configure(text=option)

    def check_answer(self, selected_option_index):
        correct_option_index = self.questions[self.current_question_index]["correct"]

        if selected_option_index == correct_option_index:
            self.feedback_label.configure(text="Correct!", text_color="green")
        else:
            self.feedback_label.configure(text="Incorrect!", text_color="red")

        self.current_question_index += 1
        if self.current_question_index < len(self.questions):
            self.load_question()
        else:
            self.feedback_label.configure(text="Quiz Completed!")
            for button in self.option_buttons:
                button.configure(state='disabled')
            self.feedback_label.configure(text='')

def run():
    app = Page()
    app.mainloop()

if __name__ == "__main__":
    run()
```

Problems & solutions:

The questions that have been brought into the quiz as quiz data are unfiltered and when the quiz frame is opened by topic selection all questions are displayed one after the other, irrespective of the topic selection. The questions need to be filtered by each subject.

Log entry 7 18/6/24 Week 8

Update:

My app now acknowledges when the user has finished their questions and displays a message and their score on the quiz. When the user selects a certain topic, the questions will now filter and only show the questions under that topic.

```python
import customtkinter
from get_questions import import_quiz_data

global quiz_data

quiz_data = import_quiz_data()

global score

score = 0


class Page(customtkinter.CTk):  # Inherit from customtkinter.CTk
    def __init__(self):
        super().__init__()  # Initialize the superclass

        self.geometry('500x500')
        customtkinter.set_appearance_mode("dark")
        customtkinter.set_default_color_theme('blue')

        self.subject_frame = customtkinter.CTkFrame(self)
        self.subject_frame.pack(pady =10, padx =10, fill='both', expand=True)

        self.Label2 = customtkinter.CTkLabel(master=self.subject_frame, text='Select Quiz Topic', fg_color='transparent')
        self.Label2.grid(row=0, column=5, sticky='nsew')

        self.options_buttons2 = []
        self.options = ['Physics', 'Biology', 'Mathematics', 'Italian']
        for i, option in enumerate(self.options):
            Button = customtkinter.CTkButton(master=self.subject_frame, text=option, command=lambda opt=option: self.open_frame(opt))
            self.options_buttons2.append(Button)
            Button.grid(row=i + 1, column=5, pady=5, padx=5)

        for i in range(10):
            self.subject_frame.grid_columnconfigure(i, weight=3)
            self.subject_frame.grid_rowconfigure(i, weight=1)


        self.frame = customtkinter.CTkFrame(self)
        self.frame.pack_forget()

        self.questions = quiz_data
        self.current_question_index = 0

    def open_frame(self, option):

        # get questions based on chosen option


        filtered_questions = []
        for i in quiz_data:
            if i['subject'] == option:
                filtered_questions.append(i)

        print(filtered_questions)


        print(f"Selected option: {option}")
        self.subject_frame.pack_forget()
        self.frame.pack(pady=20, padx=20, fill="both", expand=True)

        self.label = customtkinter.CTkLabel(master=self.frame, text=f"{option} Quiz", fg_color="transparent")
        self.label.grid(row=0, column=3, columnspan=4, sticky="nsws")


        self.feedback_label = customtkinter.CTkLabel(master=self.frame, text='Quiz Title', fg_color='transparent') #setting label dimensions + label name/title
        self.feedback_label.grid(row=1, column=4, sticky='nsew') #label in grid

        self.option_buttons = []

        for i in range(4):
            buttonA = customtkinter.CTkButton(master=self.frame, text='A', command=lambda i=i: self.check_answer(i)) #button commands + labels
            self.option_buttons.append(buttonA)
            buttonA.grid(row=i + 2, column=4, padx=20, pady=10)    #setting buttons in grid

        for i in range(9):
            self.frame.grid_columnconfigure(i, weight=3)
            self.frame.grid_rowconfigure(i, weight=1)


        self.questions = filtered_questions


        self.current_question_index = 0
        self.load_question()

    def load_question(self):
        question_data = self.questions[self.current_question_index]
        self.feedback_label.configure(text=question_data["question"])

        for i, option in enumerate(question_data["options"]):
            self.option_buttons[i].configure(text=option)

    def check_answer(self, selected_option_index):


        correct_option_index = self.questions[self.current_question_index]["correct"] # accesses the correct answer for the question
        print("Test",correct_option_index)

        print(self.questions[self.current_question_index]["correct"])

        if selected_option_index == correct_option_index:
            score=score+1
            # self.feedback_label.configure(text="Correct!", text_color="green")
        # else:
        #     self.feedback_label.configure(text="Incorrect!", text_color="red")

        self.current_question_index += 1

        if self.current_question_index < len(self.questions):
            self.load_question()
        else:

            for button in self.option_buttons:
                button.configure(state='disabled')
            self.feedback_label.configure(text="Quiz Completed!\n Your score:  ")
            print(score)
```

Problems & solutions:

The select topic options are still hard coded and need to be able to change depending on what the user inputs into the subject column. A function will need to collate the subjects and compile them into the buttons to achieve this.

Log entry 8 24/6/24 Week 9

Update:

The select topic buttons now use the quiz data and are no longer hard coded to those 4 options, so the select topic buttons will now change when different subjects are added into the quiz data Excel file.

```python
import customtkinter
from get_questions import import_quiz_data

global quiz_data

quiz_data = import_quiz_data()

global score
score = 0

unique_subjects = list(set(question['subject'] for question in quiz_data))

class Page(customtkinter.CTk):  # Inherit from customtkinter.CTk
    def __init__(self, subject):
        super().__init__()  # Initialize the superclass

        self.geometry('500x500')
        customtkinter.set_appearance_mode("dark")
        customtkinter.set_default_color_theme('blue')

        self.subject_frame = customtkinter.CTkFrame(self)
        self.subject_frame.pack(pady =10, padx =10, fill='both', expand=True)

        self.Label2 = customtkinter.CTkLabel(master=self.subject_frame, text='Select Quiz Topic', fg_color='transparent')
        self.Label2.grid(row=0, column=5, sticky='nsew')

        self.options_buttons2 = []
        for i, option in enumerate(subject):
        # self.options = ['Physics', 'Biology', 'Mathematics', 'Italian']
            Button = customtkinter.CTkButton(master=self.subject_frame, text=option, command=lambda opt=option: self.open_frame(opt))
            self.options_buttons2.append(Button)
            Button.grid(row=i + 1, column=5, pady=5, padx=5)

        for i in range(10):
            self.subject_frame.grid_columnconfigure(i, weight=3)
            self.subject_frame.grid_rowconfigure(i, weight=1)


        self.frame = customtkinter.CTkFrame(self)
        self.frame.pack_forget()

        self.questions = quiz_data
        self.current_question_index = 0

    def open_frame(self, option):

        # get questions based on chosen option

        filtered_questions = [q for q in quiz_data if q['subject'] == option]




        print(f"Selected option: {option}")
        self.subject_frame.pack_forget()
        self.frame.pack(pady=20, padx=20, fill="both", expand=True)

        self.label = customtkinter.CTkLabel(master=self.frame, text=f"{option} Quiz", fg_color="transparent")
        self.label.grid(row=0, column=3, columnspan=4, sticky="news")


        self.feedback_label = customtkinter.CTkLabel(master=self.frame, text='Quiz Title', fg_color='transparent') #setting label dimensions + label name/title
        self.feedback_label.grid(row=1, column=4, sticky='nsew') #label in grid

        self.option_buttons = []

        for i in range(4):
            buttonA = customtkinter.CTkButton(master=self.frame, text='A', command=lambda i=i: self.check_answer(i)) #button commands + labels
            self.option_buttons.append(buttonA)
            buttonA.grid(row=i + 2, column=4, padx=10, pady=10)    #setting buttons in grid

        for i in range(9):
            self.frame.grid_columnconfigure(i, weight=3)
            self.frame.grid_rowconfigure(i, weight=1)


        self.questions = filtered_questions


        self.current_question_index = 0
        self.load_question()

    def load_question(self):
        question_data = self.questions[self.current_question_index]
        self.feedback_label.configure(text=question_data["question"])

        for i, option in enumerate(question_data["options"]):
            self.option_buttons[i].configure(text=option)

    def check_answer(self, selected_option_index):


        correct_option_index = self.questions[self.current_question_index]["correct"] # accesses the correct answer for the question
        print("Test",correct_option_index)

        print(self.questions[self.current_question_index]["correct"])

        if selected_option_index == correct_option_index:
            global score
            score += 1
            # self.feedback_label.configure(text="Correct!", text_color="green")
        # else:
        #     self.feedback_label.configure(text="Incorrect!", text_color="red")

        self.current_question_index += 1

        if self.current_question_index < len(self.questions): #
            self.load_question()
        else:

            for button in self.option_buttons:
                button.configure(state='disabled')
            self.feedback_label.configure(text="Quiz Completed!\n Your score:  ")  # Displays Text in a label once all questions have been answered
            print(score)
```

Problems & solutions:

The app does recognise when a selected option within the quiz is correct, but it doesn't add to the user's score at the end. Still working out solutions for this.

Log entry 9 27/6/24 Week 9

Update:

All code has now been completed, everything is completed, and I have added a retry button at the end of the quiz which allows the user to retake the quiz or take a different quiz. All code has been commented, explaining each line.

```python
import customtkinter
import csv

def import_quiz_data():# import data from csv
    quiz_data = []
    with open('questions.csv', 'r') as file:
        reader = csv.DictReader(file) # reads csv file
        for row in reader:
            options = [row['option_a'], row['option_b'], row['option_c'], row['option_d']]
            correct_answer = row['answer'] # the correct answer is recognised by the string in answer column & the letter option selected match (option_a, b, c, d)
            quiz_data.append({
                'subject': row['subject'],
                'question': row['question'],
                'options': options,
                'correct_answer': correct_answer
                })
    return quiz_data

global quiz_data
quiz_data = import_quiz_data() # importing the data from csv file through get_questions

global score # global variable, score starts at 0
score = 0

unique_subjects = list(set(question['subject'] for question in quiz_data)) #filters questions through the subjects column in csv file

class PageI(customtkinter.CTk):  # Inherit from customtkinter.CTk
    def __init__(self, subject):
        super().__init__()  # Initialize the superclass

        self.geometry('500x500') # dimensions of frame
        customtkinter.set_appearance_mode("dark") # background/primary colour
        customtkinter.set_default_color_theme("blue") # colour theme/secondary colour

        self.subject_frame = customtkinter.CTkFrame(self) # initialising the subject selection frame
        self.subject_frame.pack(pady =10, padx =10, fill='both', expand=True)

        self.Label2 = customtkinter.CTkLabel(master=self.subject_frame, text='Select Quiz Topic', fg_color='transparent') # label within the subject frame
        self.Label2.grid(row=0, column=5, sticky='nsew') # label in grid, place in the frame

        self.options_buttons2 = [] # subject selection buttons
        for i, option in enumerate(subject):
            Button = customtkinter.CTkButton(master=self.subject_frame, text=option, command=lambda opt=option: self.open_frame(opt)) # initialising button. command lambda = used for defining an anon
            self.options_buttons2.append(Button) # can add more buttons if necessary
            Button.grid(row=i + 1, column=5, pady=5, padx=5) # place of buttons in the frame, buttons in the grid

        for i in range(10):
            self.subject_frame.grid_columnconfigure(i, weight=3)
            self.subject_frame.grid_rowconfigure(i, weight=1)


        self.frame = customtkinter.CTkFrame(self) #new frame for quiz
        self.frame.pack_forget()  # hide the frame until called

        self.questions = quiz_data # questions are taken from quiz_data/ csv file
        self.current_question_index = 0 # the quiz start at the first question. The index of questions starts at 0

        self.retry_button = customtkinter.CTkButton(master=self.frame, text='Retry Or Select New Topic', command=self.reset_quiz) # initialising return button

    def open_frame(self, option):# open quiz frame when subject is selected

        # get questions based on chosen option
        filtered_questions = [q for q in quiz_data if q['subject'] == option] # use the questions associated with the same subject

        print(f"Selected option: {option}") # print the selected subject
        self.subject_frame.pack_forget() # hide the subject frame
        self.frame.pack(pady=20, padx=20, fill='both', expand=True)

        self.label = customtkinter.CTkLabel(master=self.frame, text=f"{option} Quiz", fg_color="transparent") # displaying what quiz the user is doing, eg. 'Physics quiz'
        self.label.grid(row=0, column=3, columnspan=3, sticky="nsew") # setting label in the grid

        self.feedback_label = customtkinter.CTkLabel(master=self.frame, text='Quiz Title', fg_color='transparent') #setting label dimensions + label name/title
        self.feedback_label.grid(row=1, column=4, sticky='nsew') #label in grid

        self.option_buttons = []

        for i in range(4):
            buttonA = customtkinter.CTkButton(master=self.frame, text='A', command=lambda i=i: self.check_answer(i)) #button commands + text in button
            self.option_buttons.append(buttonA)
            buttonA.grid(row=i + 2, column=4, padx=10, pady=10)    #setting buttons in grid

        for i in range(9):
            self.frame.grid_columnconfigure(i, weight=3) # setting grid of frame
            self.frame.grid_rowconfigure(i, weight=1)


        self.questions = filtered_questions
        self.current_question_index = 0
        self.load_question()

    def load_question(self): # loads in questions when quiz frame is called
        question_data = self.questions[self.current_question_index]
        self.feedback_label.configure(text=question_data["question"]) # quiz questions use quiz data from csv file from questions column

        for i, option in enumerate(question_data["options"]):
            self.option_buttons[i].configure(text=option)  # buttons use quiz data from csv file in the option_a, b, c, d columns

    def check_answer(self, selected_option_index):
        question_data = self.questions[self.current_question_index]
        correct_option_index = ord(question_data['correct_answer']) - ord('a')  # Convert 'a', 'b', 'c', 'd' to 0, 1, 2, 3. string to integer

        global score
        if selected_option_index == correct_option_index: # if the selected option = correct option add +1 to score
            score += 1

        self.current_question_index += 1 # score is increased by +1 on each correct answer

        if self.current_question_index < len(self.questions): # if there are still unanswered questions remaining, load them
            self.load_question()
        else:                                                 # otherwise display that the quiz is completed and the user's score
            for button in self.option_buttons:
                button.configure(state='disabled')
            self.feedback_label.configure(text=f"Quiz Completed!\n Your score: {score} ")  # Displays Text in a label once all questions have been answered
            print(score)

            self.retry_button.grid(row=7, column=4, pady=10, padx=10) # place of retry button in the frame

    def reset_quiz(self):# if retry button is clicked hides the retry button and quiz frame
        global score # resets user's score when retry button is clicked
        score = 0
        self.retry_button.pack_forget()
```

Problems & solutions:

All problems have been resolved as the code has been completed.

Test Table 4.1

Test ID	Category	Test Case Description	Input to Provide	Expected Output	Actual Output	Pass/Fail	Action Taken
Test 1	Path Coverage	Verify initialization of the main window	Run the application	Main window opens with default settings	Main window opens with default settings	Pass	N/A
Test 2	Path Coverage	Verify loading of quiz topics	Run the application and check for topic buttons	Buttons for Physics, Biology, Mathematics, and Italian appear	Buttons for Physics, Biology, Mathematics, and Italian appear	Pass	N/A
Test 3	Path Coverage	Verify selection of quiz topic	Click on a quiz topic (e.g., Physics)	Opens the quiz frame with the title "Physics Quiz"	Opens the quiz frame with the title "Physics Quiz"	Pass	N/A
Test 4	Boundary Value	Check behaviour with no questions in a topic	Select a subject without any questions assigned	Should display a message “no questions available”	No message is displayed	Fail	Check for empty question lists and display messages
Test 5	Boundary Value	Verify behaviour at the end of the quiz	Finish the quiz	Displays ‘Quiz completed!’ Your score:	Displays ‘Quiz completed!’ Your score:	Pass	N/A
Test 6	Faulty data	Input a non-integer as option index	Modify quiz data to include	Should handle or reject invalid data	Handles or rejects invalid data	Pass	N/A

			non-integer correct option index				
Test 7	Path coverage	Verify score increment on correct answer	Answer a question correctly	Increments score by 1	Increments score by 1	Pass	N/A
Test 8	Path coverage	Verify no score increment on wrong answer	Answer a question incorrectly	Score stays the same	Score stays the same	Pass	N/A
Test 9	Exception handling	Verify handling of index out of range	Change current_ question_ index to an out-of-range value	Should be able to handle out of range value	Crashes or is unable to handle out of range value	Fail	Check for current question index
Test 10	Faulty data	Verify handling of invalid quiz data	Provide quiz data with missing 'subject' or 'question' areas	Should be able to handle invalid or missing quiz data	Unable to load questions or missing quiz data will impact app	Fail	Check invalid or missing quiz data will not effect app

GitHub Repository 5.1

My GitHub repository link: https://github.com/Matt-H139/Software-major-work.git
README file is on my repository

Project Reflection 6.1

My app 'Custom Quizzer' had the aim of providing a self-customisable multiple-choice quiz to Saint Augustine's students, the reason for development of my app was to provide something that rivalled online self-customisable quizzes such as Kahoot and Quizlet but only for free, meaning no paywalls and in app purchases. Completely free for Saint Augustine's students. The app functions as the user will open the provided Excel file and add their own questions and filling in the criteria in the spreadsheet appropriately to test themselves on whatever they need to study, learn or revise. Once the quiz data has been inputted the quiz app will use such data and quiz the user in a multiple-choice style test and then displaying the user with their score once completed, this score can tell the user if their knowledge on the area needs work or is sound. By overcoming these issues while developing my project I have gained a further understanding of coding in python.

The development of my app took time to see progress as I faced some issues as initially, I had switched from pygame to tkinter due to struggles in developing through pygame. Once I had switched development began to move smoother as I created a simple multiple-choice quiz and

expanded upon that eventually utilising the csv file I had created with all the quiz data in there. This allowed for the quiz to be totally customisable, although there were still some issues regarding the program to recognise when a correct answer had been selected as answers in the csv file were stored as strings and the program was trying to use integers. Eventually with some help from peers I was able to overcome this issue.

I think there were areas for improvement within the development in my project as to begin with my abilities of coding were quite limited and they developed alongside my project. I think this could be improved in the future by just becoming more familiar with coding in python meaning by coding in python more frequently I could develop this knowledge and skill. In my code I think that the UI is slightly lacking as it is quite simple and doesn't feature any images and widgets other than buttons and labels. I think the way to improve this is again just to familiarise myself with coding in python and doing it more frequently.

In conclusion, the development of my app has shown me many things from how I overcame issues in developing to the areas that I need to work on and improve in the future. The main areas from my project are that I have gained a greater understanding of python just through developing my app. Additionally I think there are many things I can learn and acknowledge from the development of my project, including: knowledge gaps in tkinter and coding in python and designing a better UI by utilising more widgets.