**AIR FORCE SCHOOL ASTE**



**PROJECT FILE**

**2020-2021**

**TOPIC: QUIZ APPLICATION**

**NAME: ANANYA**

**CLASS: XII SCIENCE**

**REG NO:**

**SUBJECT:COMPUTER SCIENCE**

**CERTIFICATE**

This is to certify that **ANANYA**, of class **XII Science** Air Force School, ASTE has successfully completed the practical work in **COMPUTER SCIENCE** under the guidance of Mrs.Subina P during the academic year 2020-21 in partial fulfilment of COMPUTER SCIENCE PRACTICAL EXAMINATION conducted by CBSE, Delhi.

Signature of Principal:

Signature of Internal Examiner:

Signature of External Examiner:

School Seal:

**ACKNOWLEDGMENT**

First of all, I am immensely indebted to almighty god for his blessing and grace without which I could not have undertaken this task and my efforts would never have been a success.

I humbly consider a privilege and honor to express my heartiest and profound gratitude to Principal Madam Mrs.Chumki Nath, Air Force School ASTE, Bengaluru for her appropriate direction, valuable suggestion, under judging assistance so generously extended to me.

I wish to express my deepest feelings of gratitude to Madam Subina.P, PGT CS for her erudite involvement and sustained guidance which has been pivotal in my project work. Her minute observation, precious insights, critical comments have indeed greatly helped me to shape my ideas.

This guidance and support received from my entire classmates who contributed and who are contributing to this project, is vital for the success of the project. I am grateful for their constant support and help.

I also owe a sense of gratitude to my parents for encouragement and support throughout the project.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TOPIC** | **PAGE NUMBER** |
| 1 | Introduction | 5 |
| 2 | System Design | 6 |
| 3 | Source code | 11 |
| 4 | Output | 31 |
| 5 | Testing | 39 |
| 6 | Future scope | 40 |
| 7 | Bibliography | 41 |

**INTRODUCTION**

**Ankrity : Effective exams made effortless for you!**

Over the last year, our dependency and indulgence towards online education has become immense. While this has opened up the scope for a new revolution in the field of online classes and examination it has also created a huge demand for fool proof and fair exam taking

Our project Ankrity revolves around exploiting this opportunity, bridging this gap and making the task of online exam conduction flawless.

Individual teachers or educational institutions of all sizes can use Ankrity to [automate exam creation and evaluation](https://megaexams.com/adopting-the-right-online-tools-for-sustainable-education/) so that they can focus on teaching and improving their students’ performance.

They can Create customized exams in just a few clicks and allocate them to students who can attempt them remotely, irrespective of their location.

Ankrity’s automated evaluation gives immediate results and insights to save the precious time of both teachers and students while they tweak their preparation.

**SYSTEM DESIGN**

**ANALYSIS:**

Ankrity, a simple quiz app that contains a set of curated questions and its answers and checks for the correctness of the answer given by the user. It navigates through the questions using dynamic programming. Entertain and educate users with a quick quiz on any theme. Mark the correct answers and the app will automatically assign the participants with points. You can make an educational trivia app for students or kids, quiz about movies, books, programming or medicine. Think outside the box to find the niche for your quiz.

**DESIGN:**

This section of the project involves the division of project in some modules or parts, thus making the presentation of the project clear and easily understandable. We have discussed below the various modules and parts of the project.

This project consists of a main menu, which is divided into two parts, namely:

* Create a Quiz
* Take a Quiz

Create a Quiz is used to create your own personalized quiz.

Take a Quiz is used to attempt an existing quiz.

**CLASSES USED:**

* App

**The Class App Contains The Following:**

1. **Data members:**

* title
* left
* top
* height
* width
* a
* code
* quiz\_code
* l
* x1
* x2
* count
* no\_of\_crct
* no\_of wrong
* unattempted
* total
* flag
* quiz\_label
* create\_btn
* create\_label
* create\_expanded
* back\_button\_c
* deco
* welcome
* qname
* name\_input
* start\_create\_btn
* req
* create\_q\_frame
* next\_btn
* submit\_btn
* q\_frame
* question
* q\_input
* marks
* marks\_input
* ans\_a
* option\_a
* ans\_b
* option\_b
* ans\_c
* option\_c
* ans\_d
* option\_d
* group1
* ty
* exit
* ref
* join\_btn
* join\_label
* join\_expanded
* back\_button\_j
* code\_input
* qcode\_input
* quiz\_frame
* Okay\_btn
* title
* a\_frame
* quest
* qlabel
* crct
* ans\_1
* option\_1
* ans\_2
* option\_2
* ans\_3
* option\_3
* ans\_4
* option\_4
* group
* okay\_btn
* next\_q
* exit\_screen
* thnx
* instructions
* questn
* mark
* opt\_a
* opt\_b
* opt\_c
* opt\_d
* answr
* csv\_w
* ryt\_ans
* series
* chart
* chartview
* csv\_r
* record
* L
* msg

1. **Member functions:**

* initUI( )
* \_\_init\_\_( )
* create\_clicked( )
* join\_clicked( )
* back\_button\_clicked( )
* start\_c\_clicked( )
* next\_clicked()
* submit\_clicked()
* create\_quiz()
* OK\_clicked()
* submit\_btn\_clicked()
* next\_q\_clicked()
* piechart()
* access\_quiz()
* radio()

**Data design:**

**design.qss**

qt style sheet file used for app designing

**Functions used:**

**User defined functions:**

* initUI(self ) - Used to define and initialize all the widgets used in the main window.
* \_\_init\_\_( self ) –It is a special method that python calls when you create a new instance of this class. It is also used to define the dimensions of the window.
* create\_clicked(self,event ) –Used to make the create window visible.
* join\_clicked(self,event ) –Used to make the join window visible.
* back\_button\_clicked(self,event ) – Used to navigate between different windows.
* start\_c\_clicked( self ) – Used to initiate the create quiz process.
* next\_clicked(self ) – Used to enter/add the next question
* submit\_clicked( self) – Used to submit the created quiz.
* create\_quiz( self,a) –Used to create a csv file containing the quiz inputted by the user.
* OK\_clicked(self ) –Used to access an existing quiz from the file.
* submit\_btn\_clicked( self) –Used to calculate the marks scored after the quiz is attempted by the user.
* next\_q\_clicked(self ) –Used to display the next question in the quiz.
* piechart( self) –Used to display a pictorial representation of the user’s performance in the quiz as a piechart.
* access\_quiz( self) –Used to obtain all the quiz details from the csv file.
* radio( self) –An empty function used to toggle between various radio buttons.

**Inbuilt functions:**

* super( ) – Used to access methods of the base class.
* setFixedSize( ) - Used to prevent the resizing of the window.
* setStyleSheet( ) - Used to change the layout of the window.
* read( ) - Reads from a qss file.
* setObjectName( ) – Used to address the buttons by their object names.
* uuid4 - Used to generate a alphanumeric string.
* setWindowTitle( ) - Used to set the title of the window.
* setGeometry( )– Used to set the geometry of the window.
* setText( )– Used to change the text of a pre-existing label.
* move( ) - Used to move the label position .
* setVisible( )- Used to set the visibility of the label by accepting bool as an argument .
* setTextFormat( )- Used to set the format of the text.
* setRange( ) - Used to set the range of a spin box .
* addButton( ) - Used to add a button to a button group.
* setWordWrap() – Used to wrap lines along word boundaries in the text content of a q label by taking bool as an argument.
* show()- Used to display the widget on the monitor screen.
* text()- Used to retrieve the text of a QLabel.
* exec\_() – Used for dynamic execution of a python program .
* append( ) - Used to append an elements to a list .
* toPlainText( )– Used to retrieve the text of a QTextEdit.
* value( )- Used to retrieve the value from a spin box.
* isChecked( )- Used to know if the check box is checked or not .
* setChecked( ) - Used to change the state of the check box.
* setExclusive( ) - Used to check whether the button group is exclusive or not .
* clear( ) – Used to clear label.
* format()- Used for string formatting.
* open()- Used to open a file.
* writer() – Used to create a writer object.
* writerows( ) - Used to write all the given rows to the csv file.
* len( ) – Returns the number of characters in a string .
* addSeries( )- Used for adding data to the piechart .
* SetAnimationOptions( )- Used to set the piechart animation .
* SetBackgroundBrush( ) - Used to define the colourfot the background.
* legend( ) - Used to label data series plotted on the piechart.
* setRenderHint( ) - Used to paint by using with qpaint.
* reader( )- Used to create a reader object and read a csv file .
* close( )- Used to close an open file.
* argv( ) – Is an array for command line arguments .

**Imported Libraries used:**

* sys
* csv
* uuid
* QApplication
* QWidgets
* QMainWindow
* QLabel
* QFrame
* QLineEdit
* QRadioButton
* QTextEdit
* QPushButton
* QSpinBox
* QButtonGroup
* QMessageBox
* QChart
* QChartView
* QPieSeries
* QFont
* QPainter
* QColor
* Qt
* QRectF

**SOURCE CODE:**

import sys

from PyQt5.QtWidgets import QApplication, QWidget, QMainWindow, QLabel, QFrame, QLineEdit, QRadioButton, QPushButton, QTextEdit, QSpinBox, QButtonGroup,QMessageBox

from PyQt5.QtChart import QChart,  QChartView, QPieSeries

from PyQt5.QtGui import QFont, QPainter, QColor

from PyQt5.QtCore import Qt, QRectF

import csv

import uuid

class App(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title = 'Ankrity'

        self.left = 670

        self.top = 190

        self.width = 560

        self.height = 839

        self.setFixedSize(self.width, self.height)

        with open("design.qss",'r') as f:

            stylesheet = f.read()

        self.setStyleSheet(stylesheet)

        self.a = 0

        self.code = str(uuid.uuid4)

        self.quiz\_code = self.code[20:28]

        self.setObjectName("main-window")

        self.initUI()

        self.l = []

        #self.lst = []

        self.x1 = 1

        self.x2 = 0

        self.count = 0

        self.no\_of\_crct = 0

        self.no\_of\_wrong = 0

        self.unattempted = 0

        self.total = 0

        self.flag = 0

    def initUI(self):

        self.setWindowTitle(self.title)

        self.setGeometry(self.left, self.top, self.width, self.height)

    #create

        self.quiz\_label = QLabel(self)

        self.quiz\_label.setText("Ankrity")

        self.quiz\_label.setObjectName("quiz-label")

        self.quiz\_label.move(155,110)

        self.create\_btn = QFrame(self)

        self.create\_btn.setObjectName("buttons")

        self.create\_btn.move(100,350)

        self.create\_btn.mousePressEvent = self.create\_clicked

        self.create\_label = QLabel(self.create\_btn)

        self.create\_label.setObjectName("frame-heading")

        self.create\_label.setText("Create a Quiz")

        self.create\_label.move(85,25)

        self.create\_expanded = QLabel(self)

        self.create\_expanded.setObjectName("frame-exp")

        self.create\_expanded.move(75,100)

        self.create\_expanded.setVisible(False)

        #second page

        self.back\_button\_c = QLabel(self.create\_expanded)

        self.back\_button\_c.move(20,5)

        self.back\_button\_c.setTextFormat(Qt.RichText)

        self.back\_button\_c.setText("&#8592;")

        self.back\_button\_c.setObjectName("back\_button")

        self.back\_button\_c.mousePressEvent = self.back\_button\_clicked

        self.deco = QLabel(self.create\_expanded)

        self.deco.setObjectName("deco")

        self.deco.setText("ecmce")

        self.deco.move(175,12)

        self.welcome = QLabel(self.create\_expanded)

        self.welcome.move(45,50)

        self.welcome.setText("Welcome!")

        self.welcome.setObjectName("welcome")

        self.qname = QLabel(self.create\_expanded)

        self.qname.setText("Name your quiz\*:")

        self.qname.setObjectName("input\_prompts")

        self.qname.move(80,250)

        self.name\_input = QLineEdit(self.create\_expanded)

        self.name\_input.setObjectName("input")

        self.name\_input.move(70,300)

        self.start\_create\_btn = QPushButton(self.create\_expanded)

        self.start\_create\_btn.setText("Start Creating!")

        self.start\_create\_btn.setObjectName("push-buttons")

        self.start\_create\_btn.move(70,530)

        self.start\_create\_btn.clicked.connect(self.start\_c\_clicked)

        self.req = QLabel(self.create\_expanded)

        self.req.setText("\*:Required field")

        self.req.setObjectName("ps")

        self.req.move(120,590)

        self.create\_q\_frame = QLabel(self)

        self.create\_q\_frame.setObjectName("frame-exp")

        self.create\_q\_frame.move(75,100)

        self.create\_q\_frame.setVisible(False)

        self.next\_btn = QPushButton(self.create\_q\_frame)

        self.next\_btn.setObjectName("push-buttons")

        self.next\_btn.setText("Next")

        self.next\_btn.move(15,530)

        self.next\_btn.clicked.connect(self.next\_clicked)

        self.submit\_btn = QPushButton(self.create\_q\_frame)

        self.submit\_btn.setObjectName("push-buttons")

        self.submit\_btn.setText("Submit")

        self.submit\_btn.move(250, 530)

        self.submit\_btn.clicked.connect(self.submit\_clicked)

        self.q\_frame = QLabel(self.create\_q\_frame)

        self.q\_frame.setObjectName("q-frame")

        self.q\_frame.move(5,30)

        self.q\_frame.setVisible(False)

        self.question = QLabel(self.q\_frame)

        self.question.setObjectName("input\_prompts")

        self.question.setText("Question:")

        self.question.move(10,15)

        self.q\_input = QTextEdit(self.q\_frame)

        self.q\_input.setObjectName("question")

        self.q\_input.move(10,50)

        self.marks = QLabel(self.q\_frame)

        self.marks.setObjectName("input\_prompts")

        self.marks.setText("Marks:")

        self.marks.move(10,110)

        self.marks\_input = QSpinBox(self.q\_frame)

        self.marks\_input.setRange(1, 100)

        self.marks\_input.setObjectName("time\_input")

        self.marks\_input.move(105,115)

        self.ans\_a = QRadioButton(self.q\_frame)

        self.ans\_a.move(10, 200)

        self.option\_a = QTextEdit(self.q\_frame)

        self.option\_a.setObjectName("answers")

        self.option\_a.move(30, 200)

        self.ans\_b = QRadioButton(self.q\_frame)

        self.ans\_b.move(10, 250)

        self.option\_b = QTextEdit(self.q\_frame)

        self.option\_b.setObjectName("answers")

        self.option\_b.move(30, 250)

        self.ans\_c = QRadioButton(self.q\_frame)

        self.ans\_c.move(10, 300)

        self.option\_c = QTextEdit(self.q\_frame)

        self.option\_c.setObjectName("answers")

        self.option\_c.move(30, 300)

        self.ans\_d = QRadioButton(self.q\_frame)

        self.ans\_d.move(10, 350)

        self.option\_d = QTextEdit(self.q\_frame)

        self.option\_d.setObjectName("answers")

        self.option\_d.move(30, 350)

        self.group1 = QButtonGroup()

        self.group1.addButton(self.ans\_a)

        self.group1.addButton(self.ans\_b)

        self.group1.addButton(self.ans\_c)

        self.group1.addButton(self.ans\_d)

        self.ty = QLabel(self)

        self.ty.setObjectName("frame-exp")

        self.ty.move(100, 100)

        self.ty.setVisible(False)

        self.exit = QLabel(self.ty)

        self.exit.setObjectName("welcome")

        self.exit.setText("Thank you!!")

        self.exit.move(15, 150)

        self.ref = QLabel(self.ty)

        self.ref.setObjectName("frame-heading")

        self.ref.setText("Your quiz code is:  {}".format(self.quiz\_code))

        self.ref.setWordWrap(True)

        self.ref.move(75, 300)

    #join

        self.join\_btn = QFrame(self)

        self.join\_btn.setObjectName("buttons")

        self.join\_btn.move(100,470)

        self.join\_btn.mousePressEvent = self.join\_clicked

        #self.join\_btn.setVisible(False)

        self.join\_label = QLabel(self.join\_btn)

        self.join\_label.setObjectName("frame-heading")

        self.join\_label.setText("Take a Quiz")

        self.join\_label.move(85,25)

        self.join\_expanded = QLabel(self)

        self.join\_expanded.setObjectName("frame-exp")

        self.join\_expanded.move(75,100)

        self.join\_expanded.setVisible(False)

        #second page

        self.back\_button\_j = QLabel(self.join\_expanded)

        self.back\_button\_j.move(20,5)

        self.back\_button\_j.setTextFormat(Qt.RichText)

        self.back\_button\_j.setText("&#8592;")

        self.back\_button\_j.setObjectName("back\_button")

        self.back\_button\_j.mousePressEvent = self.back\_button\_clicked

        self.deco = QLabel(self.join\_expanded)

        self.deco.setObjectName("deco")

        self.deco.setText("ecmce")

        self.deco.move(175,12)

        self.welcome = QLabel(self.join\_expanded)

        self.welcome.move(55,50)

        self.welcome.setText("Welcome!")

        self.welcome.setObjectName("welcome")

        self.code\_input = QLabel(self.join\_expanded)

        self.code\_input.setText("Enter QUIZ CODE")

        self.code\_input.setObjectName("input\_prompts")

        self.code\_input.move(95, 250)

        self.qcode\_input = QLineEdit(self.join\_expanded)

        self.qcode\_input.setObjectName("input")

        self.qcode\_input.move(75, 300)

        self.quiz\_frame = QLabel(self)

        self.quiz\_frame.setObjectName("frame-exp")

        self.quiz\_frame.move(75, 100)

        self.quiz\_frame.setVisible(False)

        self.Okay\_btn = QPushButton(self.join\_expanded)

        self.Okay\_btn.setText("OK")

        self.Okay\_btn.setObjectName("push-buttons")

        self.Okay\_btn.move(170,530)

        self.Okay\_btn.clicked.connect(self.OK\_clicked)

        self.qtitle = QLabel(self.quiz\_frame)

        #self.qtitle.setObjectName("input\_prompts")

        self.qtitle.setStyleSheet("color: rgb(193, 191, 191); font-size: 50px;")

        self.qtitle.setFixedWidth(300)

        self.qtitle.move(100,5)

        self.a\_frame = QLabel(self.quiz\_frame)

        self.a\_frame.setObjectName("q-frame")

        self.a\_frame.move(5,65)

        self.a\_frame.setVisible(True)

        self.quest = QLabel(self.a\_frame)

        self.quest.setObjectName("input\_prompts")

        self.quest.setFixedWidth(200)

        self.quest.move(20,15)

        self.qlabel = QLabel(self.a\_frame)

        self.qlabel.setObjectName("input\_prompts")

        self.qlabel.setWordWrap(True)

        self.qlabel.setFixedSize(350, 100)

        self.qlabel.move(20, 30)

        self.crct = QLabel(self.a\_frame)

        self.crct.setTextFormat(Qt.RichText)

        self.crct.setText("&#10003;")

        self.crct.setObjectName("back\_button")

        self.crct.move(10,190)

        self.crct.setVisible(False)

        self.ans\_1 = QRadioButton(self.a\_frame)

        self.ans\_1.move(30,200)

        self.ans\_1.toggled.connect(self.radio)

        self.option\_1 = QLabel(self.a\_frame)

        self.option\_1.setObjectName("options")

        self.option\_1.setFixedWidth(400)

        self.option\_1.setWordWrap(True)

        self.option\_1.move(50, 190)

        self.ans\_2 = QRadioButton(self.a\_frame)

        self.ans\_2.move(30, 250)

        self.ans\_2.toggled.connect(self.radio)

        self.option\_2 = QLabel(self.a\_frame)

        self.option\_2.setObjectName("options")

        self.option\_2.setFixedWidth(400)

        self.option\_2.setWordWrap(True)

        self.option\_2.move(50, 240)

        self.ans\_3 = QRadioButton(self.a\_frame)

        self.ans\_3.move(30, 300)

        self.ans\_3.toggled.connect(self.radio)

        self.option\_3 = QLabel(self.a\_frame)

        self.option\_3.setObjectName("options")

        self.option\_3.setFixedWidth(400)

        self.option\_3.setWordWrap(True)

        self.option\_3.move(50, 290)

        self.ans\_4 = QRadioButton(self.a\_frame)

        self.ans\_4.move(30, 350)

        self.ans\_4.toggled.connect(self.radio)

        self.option\_4 = QLabel(self.a\_frame)

        self.option\_4.setObjectName("options")

        self.option\_4.setFixedWidth(400)

        self.option\_4.setWordWrap(True)

        self.option\_4.move(50, 340)

        self.group = QButtonGroup()

        self.group.addButton(self.ans\_1)

        self.group.addButton(self.ans\_2)

        self.group.addButton(self.ans\_3)

        self.group.addButton(self.ans\_4)

        self.okay\_btn = QPushButton(self.quiz\_frame)

        self.okay\_btn.setObjectName("push-buttons")

        self.okay\_btn.setText("Submit")

        self.okay\_btn.move(250, 550)

        self.okay\_btn.clicked.connect(self.submit\_btn\_clicked)

        self.next\_q = QPushButton(self.quiz\_frame)

        self.next\_q.setObjectName("push-buttons")

        self.next\_q.setText("Next")

        self.next\_q.move(250,550)

        self.next\_q.clicked.connect(self.next\_q\_clicked)

        self.next\_q.setVisible(False)

        self.exit\_screen = QLabel(self)

        self.exit\_screen.setObjectName("frame-exp")

        self.exit\_screen.move(75, 100)

        self.exit\_screen.setVisible(False)

        self.marks\_obtained = QLabel(self.exit\_screen)

        self.marks\_obtained.setObjectName("options")

        self.marks\_obtained.setFixedWidth(200)

        self.marks\_obtained.move(100, 60)

        self.thnx = QLabel(self.exit\_screen)

        self.thnx.setText("Thank you for taking the quiz :)")

        self.thnx.setWordWrap(True)

        self.thnx.setStyleSheet("color: rgb(193,191,191); font-size:25px; font=weight: bold;")

        self.thnx.move(70,555)

        self.show()

    def create\_clicked(self, event):

        self.create\_expanded.setVisible(True)

        self.join\_btn.setVisible(False)

        self.create\_btn.setVisible(False)

    def join\_clicked(self, event):

        self.join\_expanded.setVisible(True)

        self.join\_btn.setVisible(False)

        self.create\_btn.setVisible(False)

    def back\_button\_clicked(self, event):

        self.create\_expanded.setVisible(False)

        self.join\_expanded.setVisible(False)

        self.join\_btn.setVisible(True)

        self.create\_btn.setVisible(True)

    def start\_c\_clicked(self):

        self.instructions = QMessageBox()

        self.instructions.setText("--Fill in  the answers and choose the right one--")

        self.instructions.setObjectName("input\_prompts")

        self.instructions.setWindowTitle("Instructions")

        #self.instructions.move(20,95)

        self.instructions.exec\_()

        self.create\_expanded.setVisible(False)

        self.create\_q\_frame.setVisible(True)

        self.q\_frame.setVisible(True)

        title = self.name\_input.text()

        self.l.append([title])

    def next\_clicked(self):

        self.a += 1

        questn = self.q\_input.toPlainText()

        mark = self.marks\_input.value()

        opt\_a = self.option\_a.toPlainText()

        opt\_b = self.option\_b.toPlainText()

        opt\_c = self.option\_c.toPlainText()

        opt\_d = self.option\_d.toPlainText()

        if self.ans\_a.isChecked():

            answr = self.option\_a.toPlainText()

        elifself.ans\_b.isChecked():

            answr = self.option\_b.toPlainText()

        elifself.ans\_c.isChecked():

            answr = self.option\_c.toPlainText()

        else:

            answr = self.option\_d.toPlainText()

        self.l.append([self.a, questn, opt\_a, opt\_b, opt\_c, opt\_d, answr, mark])

        self.group1.setExclusive(False)

        self.ans\_a.setChecked(False)

        self.ans\_b.setChecked(False)

        self.ans\_c.setChecked(False)

        self.ans\_d.setChecked(False)

        self.group1.setExclusive(True)

        self.q\_input.clear()

        self.marks\_input.clear()

        self.option\_a.clear()

        self.option\_b.clear()

        self.option\_c.clear()

        self.option\_d.clear()

    def submit\_clicked(self):

        self.a += 1

        questn = self.q\_input.toPlainText()

        mark = self.marks\_input.value()

        opt\_a = self.option\_a.toPlainText()

        opt\_b = self.option\_b.toPlainText()

        opt\_c = self.option\_c.toPlainText()

        opt\_d = self.option\_d.toPlainText()

        if self.ans\_a.isChecked():

            answr = self.option\_a.toPlainText()

        elifself.ans\_b.isChecked():

            answr = self.option\_b.toPlainText()

        elifself.ans\_c.isChecked():

            answr = self.option\_c.toPlainText()

        else:

            answr = self.option\_d

        self.l.append([self.a, questn, opt\_a, opt\_b, opt\_c, opt\_d, answr, mark])

        #print(self.l)

        self.create\_quiz(self.l)

    def create\_quiz(self,a):

        with open("{}.csv".format(self.quiz\_code),"w",newline='') as f :

            csv\_w=csv.writer(f,delimiter=',')

            csv\_w.writerows(a)

        #print("successfully created")

        self.ty.setVisible(True)

    def OK\_clicked(self):

        self.q\_code = self.qcode\_input.text()

        self.access\_quiz()

        if self.flag == 0:

            self.join\_expanded.setVisible(False)

            self.quiz\_frame.setVisible(True)

            self.qtitle.setText(self.l[0][0])

            self.len = len(self.l)

            self.L = self.l[self.x1]

            self.quest.setText("Question "+self.L[0])

            self.qlabel.setText(self.L[1])

            self.option\_1.setText(self.L[2])

            self.option\_2.setText(self.L[3])

            self.option\_3.setText(self.L[4])

            self.option\_4.setText(self.L[5])

    def submit\_btn\_clicked(self):

        self.x2 += 1

        if self.x2 <self.len:

            self.marks = int(self.l[self.x2][7])

            self.total += self.marks

            self.ryt\_ans = self.l[self.x2][6]

            if self.ans\_1.isChecked():

                if self.option\_1.text() == self.ryt\_ans:

                    self.no\_of\_crct += 1

                    self.count += self.marks

                    self.crct.setVisible(True)

                    self.crct.move(10, 190)

                else:

                    self.no\_of\_wrong += 1

                    if self.option\_1.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 190)

                    elifself.option\_2.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 240)

                    elifself.option\_3.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 290)

                    else:

                        self.crct.setVisible(True)

                        self.crct.move(10, 340)

            elif self.ans\_2.isChecked():

                if self.option\_2.text() == self.ryt\_ans:

                    self.no\_of\_crct += 1

                    self.count += self.marks

                    self.crct.setVisible(True)

                    self.crct.move(10, 240)

                else:

                    self.no\_of\_wrong += 1

                    if self.option\_1.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 190)

                    elifself.option\_2.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 240)

                    elifself.option\_3.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 290)

                    else:

                        self.crct.setVisible(True)

                        self.crct.move(10, 340)

            elif self.ans\_3.isChecked():

                if self.option\_3.text() == self.ryt\_ans:

                    self.no\_of\_crct += 1

                    self.count += self.marks

                    self.crct.setVisible(True)

                    self.crct.move(10, 290)

                else:

                    self.no\_of\_wrong += 1

                    if self.option\_1.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 190)

                    elifself.option\_2.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 240)

                    elifself.option\_3.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 290)

                    else:

                        self.crct.setVisible(True)

                        self.crct.move(10, 340)

            elif self.ans\_4.isChecked():

                if self.option\_4.text() == self.ryt\_ans:

                    self.no\_of\_crct += 1

                    self.count += self.marks

                    self.crct.setVisible(True)

                    self.crct.move(10, 340)

                else:

                    self.no\_of\_wrong += 1

                    if self.option\_1.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 190)

                    elifself.option\_2.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 240)

                    elifself.option\_3.text() == self.ryt\_ans:

                        self.crct.setVisible(True)

                        self.crct.move(10, 290)

                    else:

                        self.crct.setVisible(True)

                        self.crct.move(10, 340)

            else:

                self.unattempted += 1

                if self.option\_1.text() == self.ryt\_ans:

                    self.crct.setVisible(True)

                    self.crct.move(10, 190)

                elifself.option\_2.text() == self.ryt\_ans:

                    self.crct.setVisible(True)

                    self.crct.move(10, 240)

                elifself.option\_3.text() == self.ryt\_ans:

                    self.crct.setVisible(True)

                    self.crct.move(10, 290)

                else:

                    self.crct.setVisible(True)

                    self.crct.move(10, 340)

        self.okay\_btn.setVisible(False)

        self.next\_q.setVisible(True)

    def next\_q\_clicked(self):

        self.x1 += 1

        self.group.setExclusive(False)

        self.ans\_1.setChecked(False)

        self.ans\_2.setChecked(False)

        self.ans\_3.setChecked(False)

        self.ans\_4.setChecked(False)

        self.group.setExclusive(True)

        if self.x1 <self.len:

            self.crct.setVisible(False)

            self.L = self.l[self.x1]

            self.quest.setText("Question "+self.L[0])

            self.qlabel.setText(self.L[1])

            self.option\_1.setText(self.L[2])

            self.option\_2.setText(self.L[3])

            self.option\_3.setText(self.L[4])

            self.option\_4.setText(self.L[5])

            self.next\_q.setVisible(False)

            self.okay\_btn.setVisible(True)

        else:

            self.exit\_screen.setVisible(True)

            self.marks\_obtained.setText("Marks:"+str(self.count)+"/"+str(self.total))

            self.piechart()

    def piechart(self):

        self.crct.setVisible(False)

        self.series = QPieSeries()

        self.series.append("Correct", self.no\_of\_crct)

        self.series.append("Wrong", self.no\_of\_wrong)

        self.series.append("Unattempted", self.unattempted)

        self.chart = QChart()

        self.chart.addSeries(self.series)

        self.chart.setAnimationOptions(QChart.SeriesAnimations)

        self.chart.setBackgroundBrush(QColor(10,10,10))

        self.chart.legend().setLabelColor(Qt.gray)

        self.chartview = QChartView(self.chart,self.exit\_screen)

        self.chartview.setGeometry(5,100,400,459)

        self.chartview.setStyleSheet("background-color: rgb(10,10,10);")

        #self.chartview.setBackgroundBrush(Qt.darkGray)

        self.chartview.setRenderHint(QPainter.Antialiasing)

        self.chartview.show()

    def access\_quiz(self):

        try:

            with open("{}.csv".format(self.q\_code),"r",newline='') as f :

                csv\_r = csv.reader(f)

                for record in csv\_r:

                    self.l.append(record)

                print(self.l)

                self.flag = 0

                return self.l

        except FileNotFoundError:

            self.flag = 1

            self.msg = QMessageBox()

            self.msg.setText("Invalid code")

            self.msg.setObjectName("input\_prompts")

            self.msg.setWindowTitle("Message")

            self.msg.exec\_()

    def radio(self):

        pass

if \_\_name\_\_ == '\_\_main\_\_':

    app = QApplication(sys.argv)

    ex = App()

    sys.exit(app.exec\_())

**SOURCE CODE OF QSS FILE:**

#main-window

{

    background-image: url(bg.jpg);

    background-color: black;

}

#quiz-label

{

    color: rgb(193, 191, 191);

    font-size: 70px;

    font-weight: bold;

    min-height: 100px;

    min-width: 300px;

    max-height: 110px;

    max-width: 130px;

}

#buttons

{

    background-color:#202020;

    min-width: 360px;

    max-width: 360px;

    min-height: 100px;

    max-height: 100px;

}

#frame-heading

{

    color: rgb(193, 191, 191);

    font-size: 30px;

}

#frame-heading:hover

{

    color: rgb(73, 69, 29);

}

#frame-exp

{

    background-color:#202020;

    min-width: 360px;

    max-width: 360px;

    min-height: 500px;

    max-height: 500px;

}

#back\_button

{

    color: white;

    font-weight: bold;

    font-size: 25px;

    border-radius: 20px;

}

#back\_button:hover

{

    color: #202020;

    background-color: rgb(193, 191, 191);

    border-radius: 20px;

}

#deco

{

    color: rgb(193, 191, 191);

    font-family: fantasy;

    font-size: 15px;

}

#welcome

{

    color: rgb(193, 191, 191);

    font-size: 40px;

    font-weight: bold;

}

#input\_prompts

{

    color: rgb(193, 191, 191);

    font-size: 17px;

}

#input

{

    background-color: rgb(193, 191, 191);

    min-width: 150px;

    max-width: 150px;

    min-height: 20px;

    max-height: 20px;

}

#time\_input

{

    background-color: rgb(193, 191, 191);

    min-width: 30px;

    max-width: 30px;

    min-height: 20px;

    max-height: 20px;

}

#push-buttons

{

    background-color: rgb(193, 191, 191);

    color: #202020;

    font-size: 25px;

    font-weight: bold;

}

#ps

{

    color: rgb(193, 191, 191);

}

#q-frame

{

    background-color: rgb(73, 69, 29);

    color: rgb(193, 191, 191);

    min-width: 350px;

    max-width: 350px;

    min-height: 400px;

    max-height: 400px;

}

#question

{

    background-color: rgb(193, 191, 191);

    font-size: 15px;

    min-width: 330px;

    max-width: 330px;

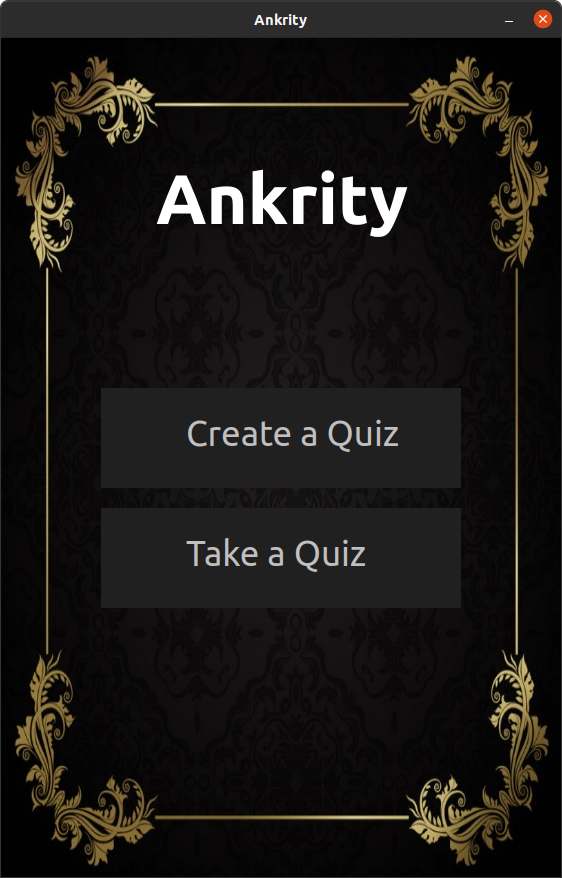
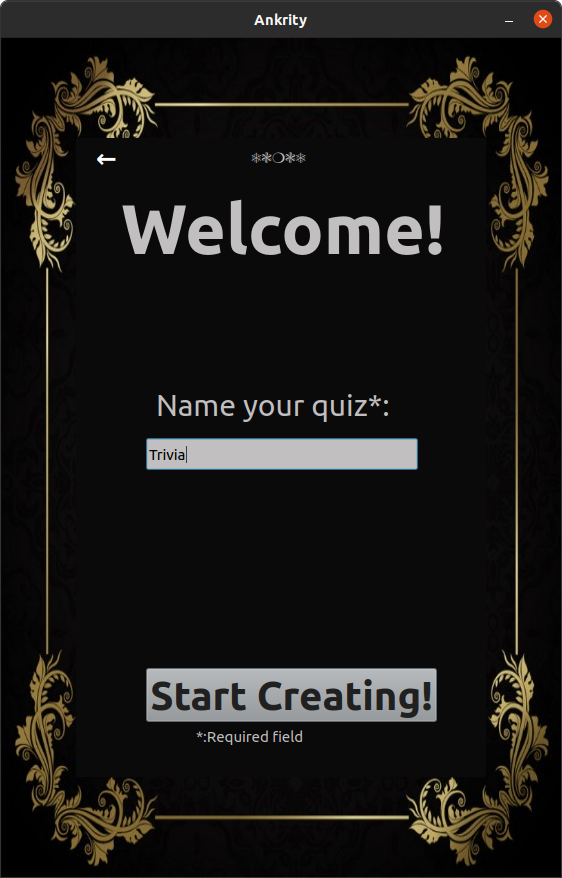
    min-height: 40px;

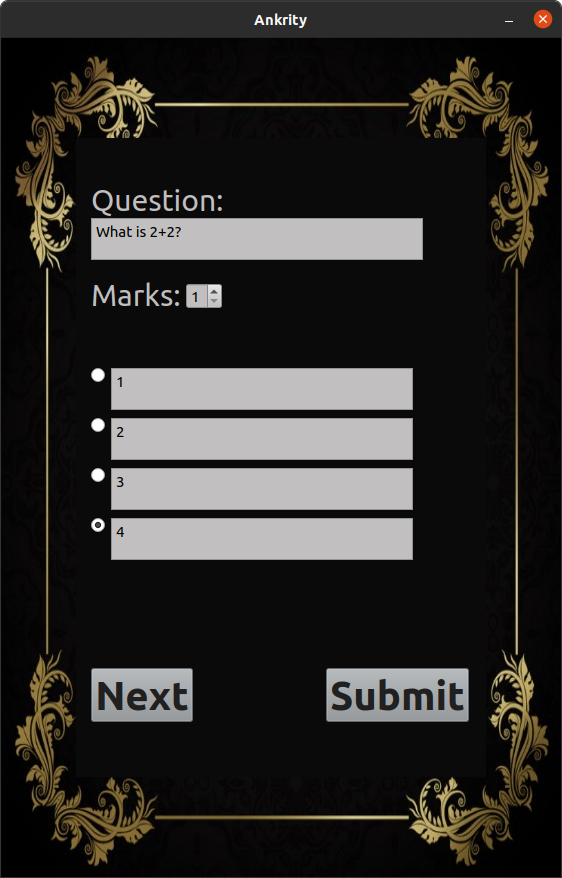
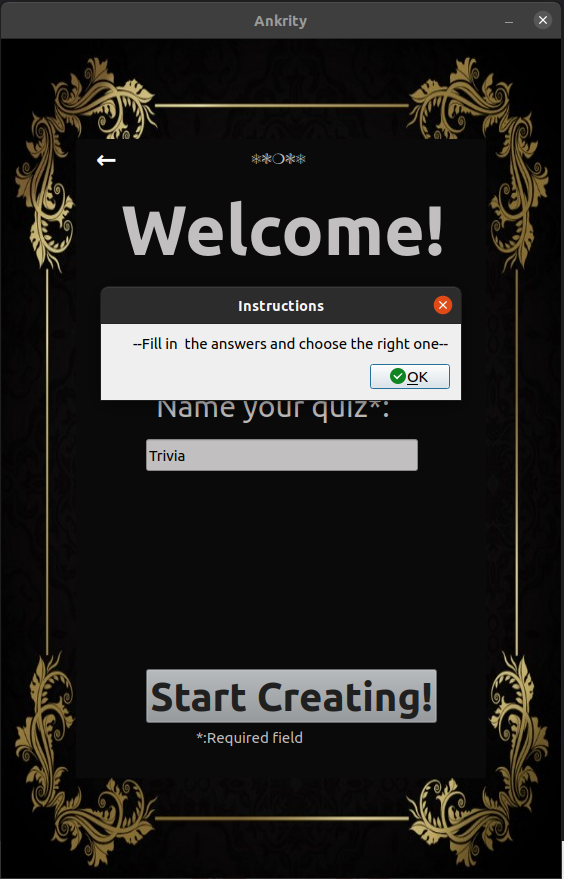
    max-height: 40px;

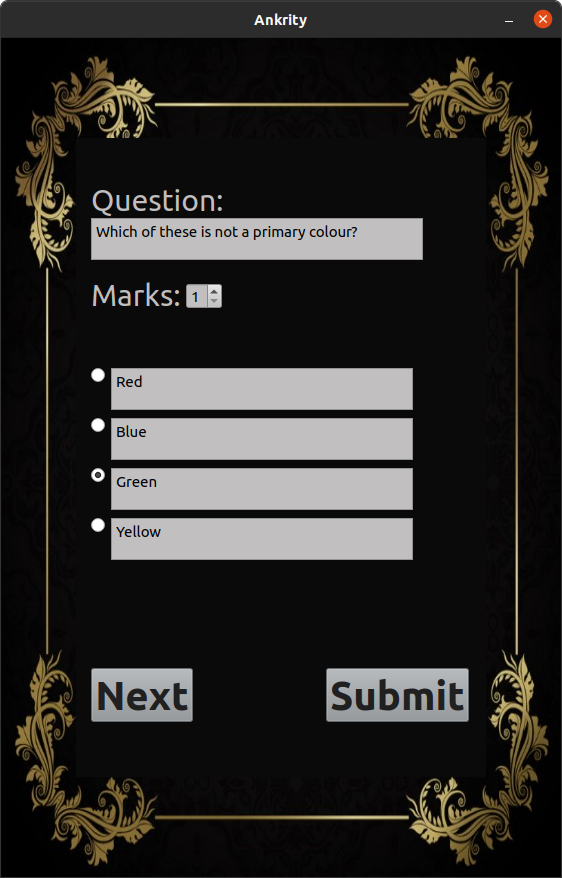
}

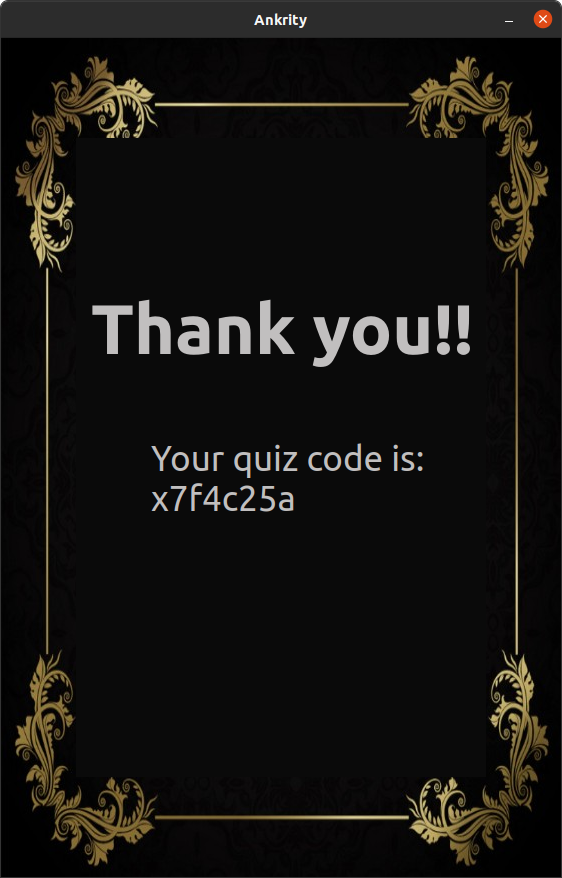
**OUTPUT**

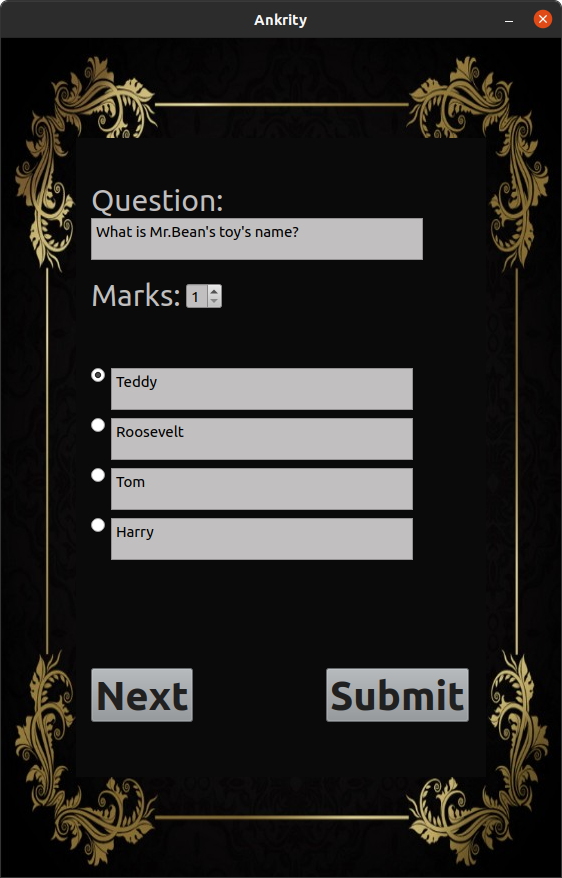
**CREATE A QUIZ OUTPUT:**



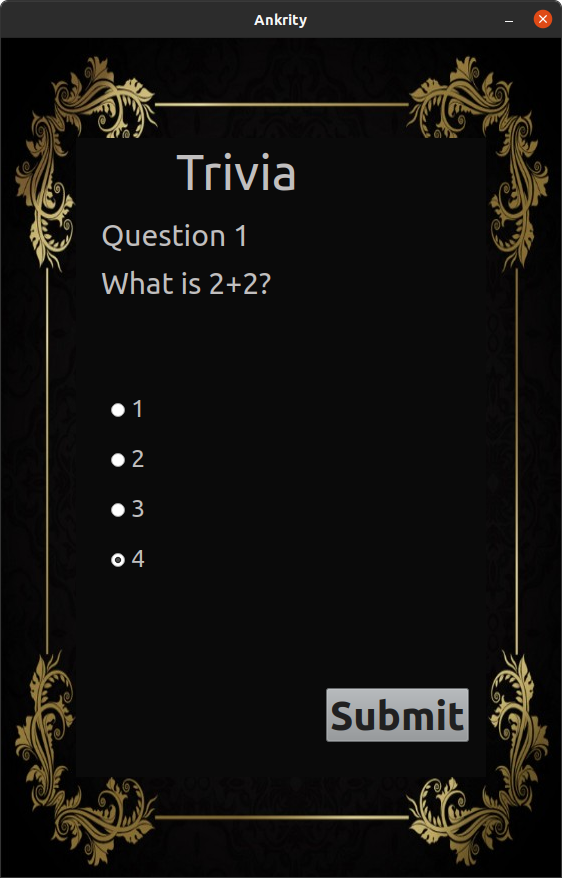
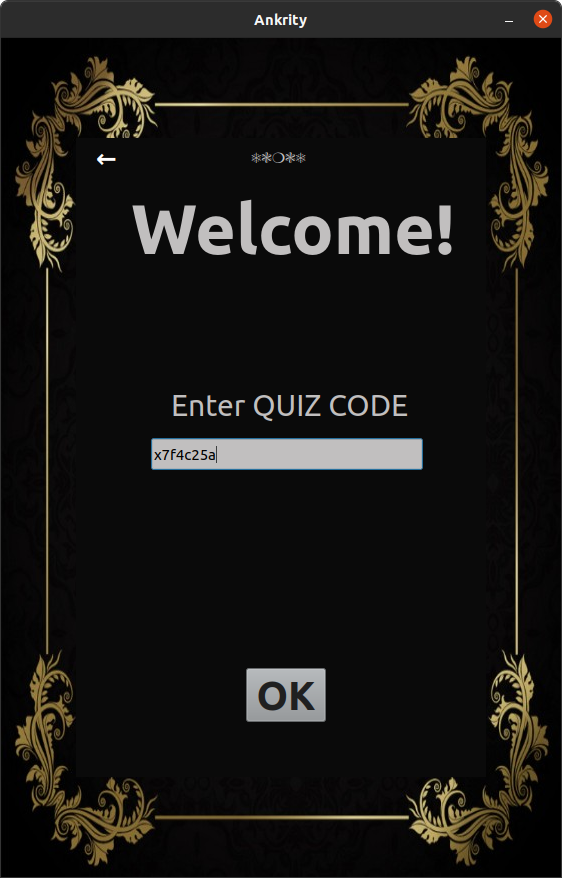


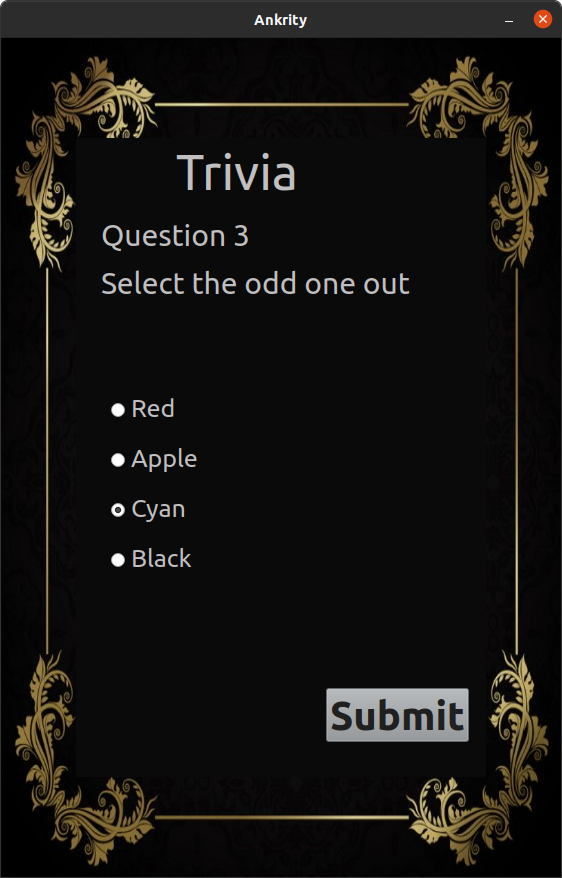


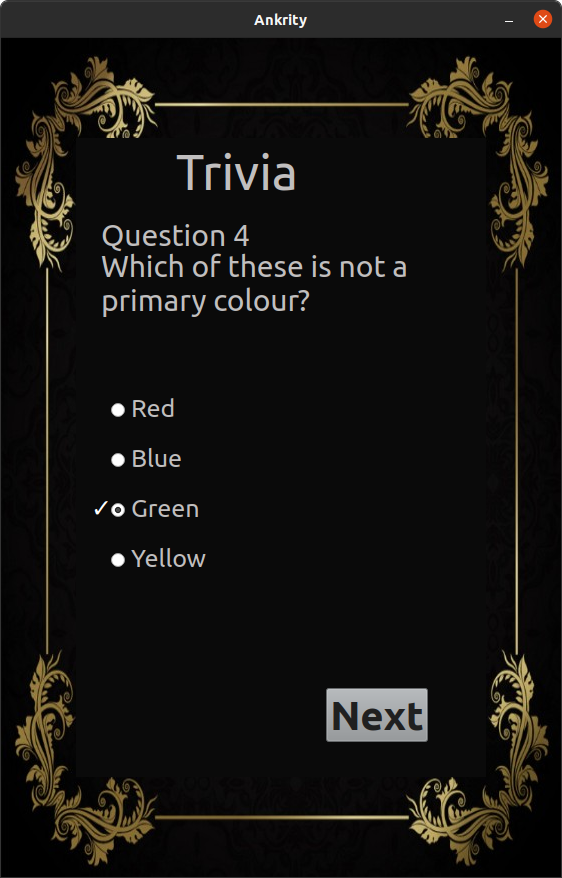
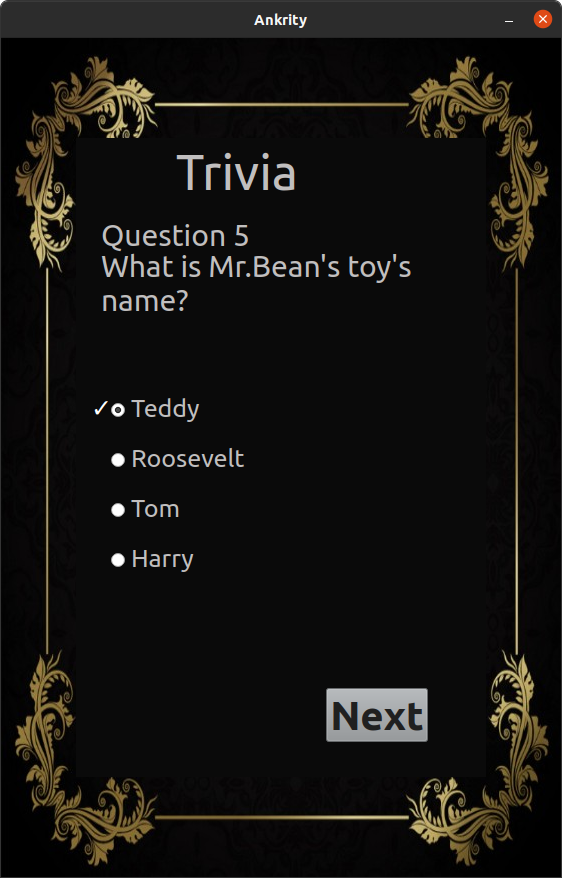




**TAKE A QUIZ OUTPUT:**



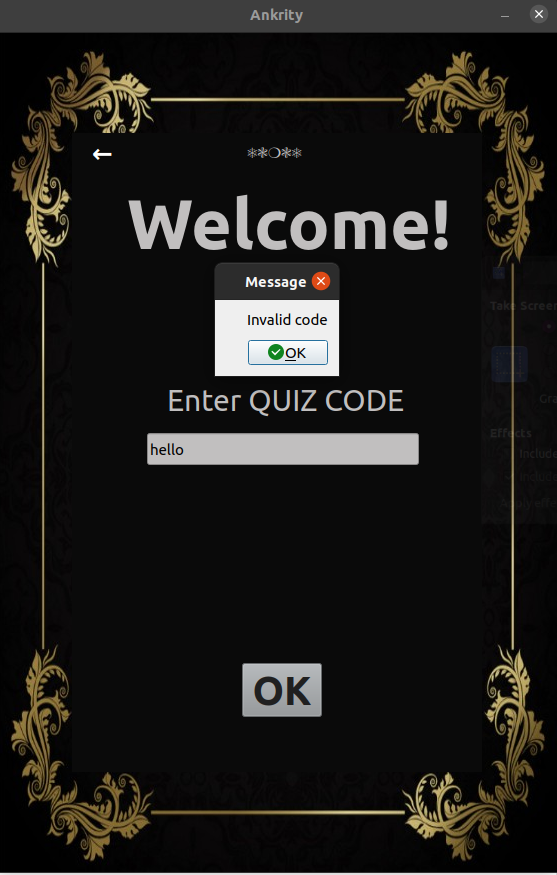






**TESTING:**

Upon clicking the ‘Take A Quiz’ option, when the user enters a wrong code, he/she will be presented with a dialogue box on their screen that reads ‘Invalid Code’.



**FUTURE SCOPE**

There is unlimited scope and opportunities to explore in the field of online examination. In near future Ankrity can be laced up and improvised with the following :

* The designed tests can be Assigned to multiple batches at once
* The teachers can have the facility to Track ongoing exams in one place
* A module that can [Track the performance](https://megaexams.com/personalised-learning-assistance-using-online-exam-software/) of each one of the students across exams, with intelligent insights and performance reports they can use to improve
* Real-time scoring and analytics for classes across time and exams to help teachers build effective and efficient teaching plans
* Facility to Instantly send report to student’s parents
* Identify scholars and create motivation with a well designed leaderboard
* A feedback section where the teacher’s can provide feedback that will help the students to improve.

**BIBLIOGRAPHY**

This project would be nearly incomplete if I had not used the information given in the following websites:

[https://pythonspot.com](https://pythonspot.com/)

<https://wiki.qt.io/Word_Wrap_of_Text_in_QLabel>

[https://www.geeksforgeeks.org](https://www.geeksforgeeks.org/how-to-create-a-quiz-app-in-android/)

<https://www.youtube.com/channel/UC4JX40jDee_tINbkjycV4Sg> (Tech With Tim)

<https://www.tutorialspoint.com/python/index.htm>

My special thanks to the up loader of information on these websites.