PROFESSIONAL INDUSTRIAL TRAINING/ INTERNSHIP REPORT **ON**

**“Python”**

AT

REMARK SKILL

DELHI

AN INDUSTRY INTERNSHIP REPORT SUBMITTED IN

PARTIAL FULFILLMENT OF THE REQUIREMENTS

FORTHE AWARDOF DEGREE OF

BACHELOR OF ENGINEERING In

Department of Computer Science and

Engineering

SUBMITTED BY – DIVYAJOT KOUR

ROLL:NO – 2020A1R108



CANDIDATES’ DECLARATION

I, DIVYAJOT KOUR, Roll Number (2020A1R108) hereby declare that the work

which is being presented in the Industry Internship Report entitled, ―PYTHON

INTERNSHIP” in partial fulfillment of requirement for the award of degree of

B.E. (CSE) and submitted in the Department of Computer Science And Engineering,

Model Institute of Engineering and Technology (Autonomous), Jammu is an

authentic record of my own work carried by me at ―REMARKSKILL PVT. LTD,

DELHI‖ under the supervision and mentorship of Saksham Madam.

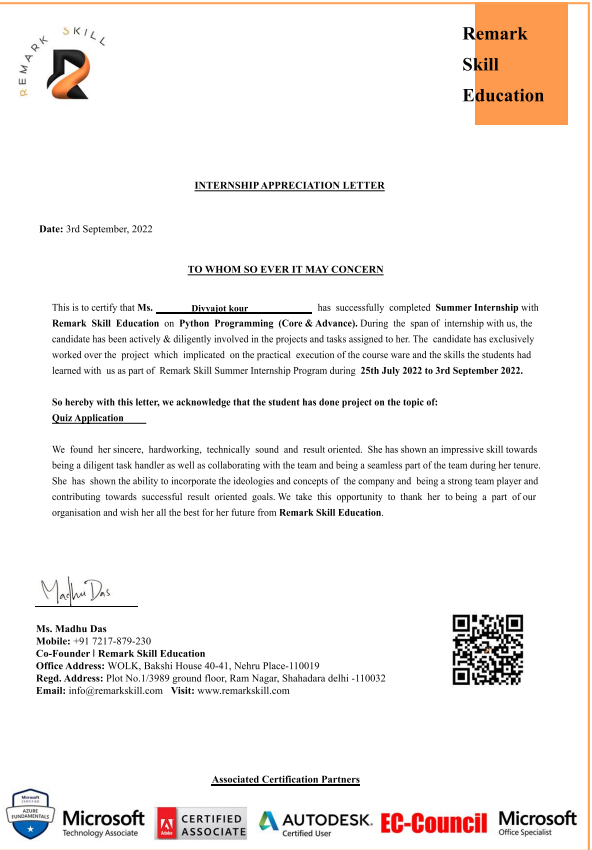
The matter presented in this report has not been

submitted in thisor any other University / Institute for the award of B.E. Degree.

Signature of student Dated

Divyajot Kour 15 OCT, 2022

2020A1R108



****

**Department Of Computer Science and Engineering**

**Model Institute of Engineering and Technology (Autonomous) Kot Bhalwal, Jammu, India**

***(NAAC“A” GradeAccredited)***

**Ref. No                                                                                         Date: 15 OCT 2022**

**CERTIFICATE**

Certified that this Industry Internship Report entitled **“python programming** “is the bonafide work of “**DIVYAJOT KOUR, Roll No. 2020A1R108, of 5thSemester,Department of Computer Science And Engineering, Model Institute of Engineering and Technology (Autonomous), Jammu”,** who carried out the Industry Internship at **“REMARK SKILL PVT LTD DELHI”** work under my mentorship during 25thJuly,2022 to 3 Septemeber , 2022.

**Saurabh sharma**

**Assistant professor**

Department of Computer Science and Engineering, MIET

*This is to certify that the above statement is correct to the best of my knowledge.*

**ACKNOWLEDGEMENTS**

This summer internship opportunity was a great chance for learning and professional development. I am grateful for having a chance to meet so many wonderful people and professionals who led me though this internship period.

It is my pleasant duty to pay my heartfelt gratitude to Mr. saksham madam, mentor   remark skill PVT.LTD who has guided me through the course of this Internship.

I must record my deep sense of gratitude to Prof. (Dr.)  Ankur  Gupta (Director, MIET)  and Prof.(Dr.)AshokKumar(DeanAcademics&HODCSE,MIET) forth guidance, constant inspiration and encouragement, and for their keen involvement throughout the course of present work.

Gratitude and thanks although mean a who have always given me a parental source of love, motivation and strength right from the journey of my life.

Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to the teachers who in spite of being extra ordinarily busy with their duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my project at their esteemed organization and extending during the training.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future.

I express my sincere gratitude to Remark skill Pvt. Ltd, Delhi and Model Institute of Engineering and Technology (Autonomous), Jammu for giving me the opportunity.

**DIVYAJOT KOUR**

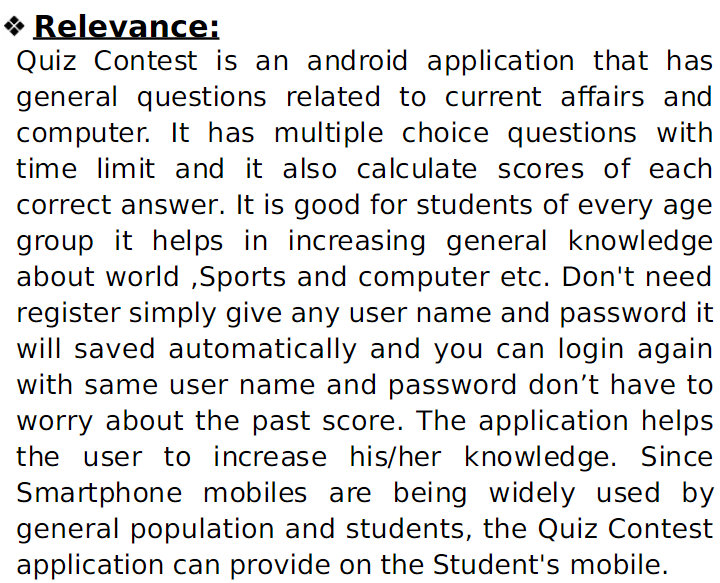
**2020A1R108**

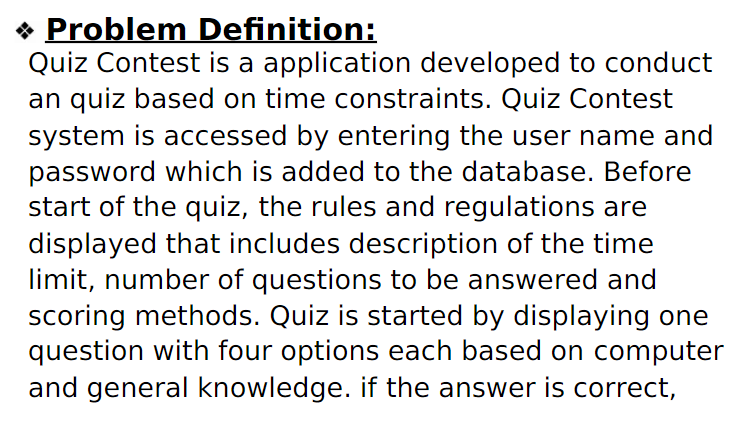
**CONTENTS p.no**

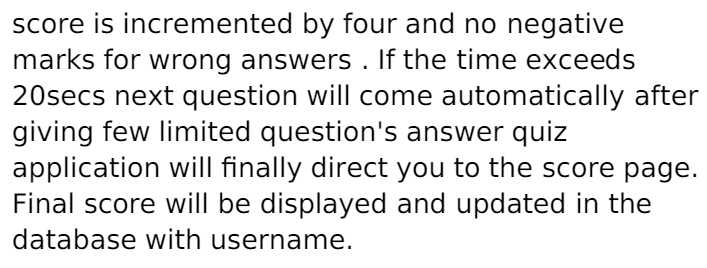
1. **INTRODUCTION 8 - 11**
   1. **RELEVANCE**
   2. **PROBLEM DEFINITION**
   3. **OBJECTIVE**
2. **Basic concepts & tools 11-12**
   1. **Introduction to Python**
   2. **Android**
3. **System analysis 13-14**
   1. **Functional Modelling**
   2. **Quantitative system**
4. **Software Requirements 14-15**
5. **Hardware Requirements 16**
6. **Features 16**
7. **Implementation 16 -24**
8. **Activity Diagram 25**

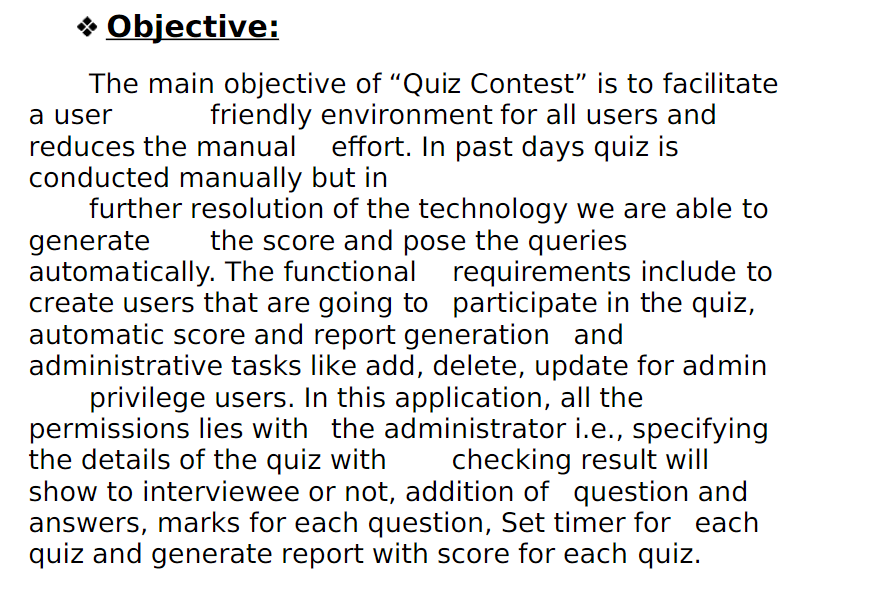
1. **Technology Used 26**
2. **Conclusions 26**
3. **References 27**

**Introduction**

****

****

****

****

**BASIC CONCEPTS &TOOLS:**

**Introduction To Python :**

**Python** is a [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Its design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with the use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule).[[33]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-7-33)

Python is [dynamically-typed](https://en.wikipedia.org/wiki/Type_system#DYNAMIC) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [structured](https://en.wikipedia.org/wiki/Structured_programming) (particularly [procedural](https://en.wikipedia.org/wiki/Procedural_programming)), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). It is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).[[34]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-About-34)[[35]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-35)

[Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) began working on Python in the late 1980s as a successor to the [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)) and first released it in 1991 as Python 0.9.0.[[36]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-36) Python 2.0 was released in 2000 and introduced new features such as [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension), [cycle-detecting](https://en.wikipedia.org/wiki/Cycle_detection) garbage collection, [reference counting](https://en.wikipedia.org/wiki/Reference_counting), and [Unicode](https://en.wikipedia.org/wiki/Unicode) support. Python 3.0, released in 2008, was a major revision that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility) with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

ANDROID

python-for-android is a packaging tool for Python apps on Android. You can create your own Python distribution including the modules and dependencies you want, and bundle it in an APK or AAB along with your own code.

Features include:

* Different app backends including Kivy, PySDL2, and a WebView with Python webserver.
* Automatic support for most pure Python modules, and built in support for many others, including popular dependencies such as numpy and sqlalchemy.
* Multiple architecture targets, for APKs optimised on any given device.
* AAB: Android App Bundle support.

SYSTEM ANALYSIS

* 1. Functional Modelling Analysis:

Functional modeling is an established technique to model engineered systems. The modeled system is viewed as a set of interconnected functions.  
Each function is transforming inputs into outputs, with the behavior possibly dependent upon parameters. Graphically this is typically depicted by a functional block diagram.

To analyze quantitatively such functional block diagram, an executable mathematical representation is needed. Python is well suited for this task. Not just for coding and executing the mathematical model, but actually performing the analysis, plotting the results, and sharing the work via a reproducible computing environment. This article introduces a coding pattern in Python to concisely model a functional block diagram as part of a quantitative system analysis. The coding pattern is applied in the context of a simple engineering analysis: the analysis of the error sources in a signal transformation chain.

* 1. QUANTITATIVE ANALYSIS:

Python is no exception: it has taken the world of numerical computing by storm, becoming a serious contender to established programming languages such as MATLAB or R. And indeed, the exact mode of interaction is supported in Python too with the use of IPython Notebooks.

SOFTWARE REQUIREMENTS

* **GUI TOOLKITS:**
  + First thing you’ll need is GUI Toolkits to design the User Interface(UI) for your Application. For a traditional UI **Qt**is the clear winner; it’s powerful, looks native, have the largest community and runs on all the major Platforms(i.e. Windows, Mac, Linux etc.). There are two different bindings: [PyQt](https://riverbankcomputing.com/software/pyqt/intro" \o "riverbankcomputing.com" \t "_blank) is older and more mature but is only free if your software is open source. While, [PySide](http://qt-project.org/wiki/PySide" \o "qt-project.org" \t "_blank)is newer and is better licensed with effective features. [Qt Designer Manual](http://qt-project.org/doc/qt-4.8/designer-manual.html) is a drag and drop manual to design your UI in an effective manner.
  + If you need to design a simple UI, **Tkinter**is a good option. It is installed as a part of Python. Details to install this Interface is provided in this document provided by Python; [tkinter - Python interface documentation](https://docs.python.org/3/library/tkinter.html" \o "docs.python.org" \t "_blank).
  + pygame is a popular for designing simple 2D Games and [Free 3D Game Engine](https://www.panda3d.org/) is best framework for 3D Designing.
* **Packaging and Distribution:**
  + The applications developed using python is in as Python package. It relies upon Python to get installed. But with the help of Packaging and Distribution tools you can make your application independent, it can be installed in any system.
  + [cx\_Freeze](http://cx-freeze.sourceforge.net/) is a freeze tool; it makes an Executable program(.Exe) out of your Python Package Programs. It works on Windows, Mac and Linux but you need separate version of Applications to create Executable Files for each Platform.
  + [PyInstaller](http://www.pyinstaller.org/) is similar as cx\_Freeze but it doesn’t support Python 3 as of now. It’s specialty is, it can create a Multi platform executable file(.Exe) from a single application.
  + [py2exe](http://www.py2exe.org/) is a Windows only Freezing Module.
  + [py2app](http://pythonhosted.org/py2app/) is a Mac only Freezing Module.
  + Packaging in Linux is the most complicated task, i won’t recommend Linux OS to develop Python Based App/Software. The main reason is Linux distribution don’t have a single Packaging format, the most popular format is deb packages; used by Debian, Ubuntu and Mint. Whereas, rpm packages are used by Fedora and Red Hat.
* **Update:**
  + [Python Package Index](https://pypi.python.org/pypi/esky)(Esky) is a framework which is used to update the Frozen(Packaged) Python Applications. That’s how you get the updates of common Software Applications.

HARDWARE REQUIREMENTS

* Intel Core i5 processor or equivalent
* 4 GB RAM (8 GB preferred)
* 15 GB available hard disk space
* Internet connection

FEATURES

* Features Save results as a CSV file. Read questions and answers (entire quiz) from CSV files. Multiple Choice Questions ...
* Next Steps Why don't you make your own quiz with the syntax of the pre-loaded CSV Quiz files. There will soon be a tutorial in this README on doing so. ...
* Authors I created this repository with the help of Py08 ...
* Check out Our other reporitories too!

IMPLEMENTATION

Let’s see what all functionalities we will be developing in this project (for two roles – user and admin):

**User:**

* Sign up (register)
* Login
* Quiz page (with timer)
* Result page (with score, time taken, percentage score, total questions, correct answers, wrong answers)

**Admin:**

* Login
* Sign up
* Add Question
* Quiz Page
* Result Page

Project Prerequisites

You should have a basic knowledge of the following technologies:

* HTML: to display content on web page
* CSS: to style HTML document
* JavaScript: to make a timer
* Bootstrap: A front-end framework
* Python: Programming language.
* Django: Python framework

**To install Django:**

pip install Django

Download Quiz App Python Code

Please download the source code of quiz web application: [**Quiz Web Application Python Code**](https://data-flair.s3.ap-south-1.amazonaws.com/python-projects/quiz-web-application-django.zip)

Steps to Build Quiz App Project

1. Starting project:

Commands to start the project and app:

django-admin startproject DjangoQuiz

cd DjangoQuiz

django-admin startapp Quiz

2. Writing Models

**Code:**

from django.db import models

# Create your models here.

**class** QuesModel(models.Model):

question = models.CharField(max\_length=200,**null**=**True**)

op1 = models.CharField(max\_length=200,**null**=**True**)

op2 = models.CharField(max\_length=200,**null**=**True**)

op3 = models.CharField(max\_length=200,**null**=**True**)

op4 = models.CharField(max\_length=200,**null**=**True**)

ans = models.CharField(max\_length=200,**null**=**True**)

**def** \_\_str\_\_(self):

**return** self.question

We just need one model for this project, QuesModel.

**Attributes of QuesModels:**

* question: This stores the question, we have also defined maximum length of the question i.e. 200
* op1: As the quiz contains Multiple choice questions. This stores option 1 value.
* op2: option2
* op3: option3
* op4: option4
* ans: This indicates which among the four options is the correct ans.

The \_\_str\_\_() method returns a string representation of any object of QuesModel.

In Django when we use Sqlite3 database, we don’t have to write table definitions we just have to write models and after that, we have to run the following commands

Py manage.py makemigrations

Py manage.py migrate

3. forms.py

**Code:**

from django.forms import ModelForm

from .models import \*

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

**class** createuserform(UserCreationForm):

**class** Meta:

model=User

fields=['username','password']

**class** addQuestionform(ModelForm):

**class** Meta:

model=QuesModel

fields="\_\_all\_\_"

To access, update, create or delete entries in the database (to implement CRUD functionality), we need forms.

Form makes it easy to implement CRUD functionality as we neither have to create forms to accept information from users nor we have to validate information manually, we can just use is\_valid() method to validate the information before updating it in the database.

4. admin.py

**Code:**

from django.contrib import admin

from .models import \*

# Register your models here.

admin.site.register(QuesModel)

Here, we are registering our model to the admin site, so that we can update or access the database from the admin panel also. But we need a superuser to access the admin site.

To create a staff (admin) user, run the below command

py manage.py createsuperuser

5. settings.py

To use javascript we have to specify the path of static files. Therefore, add the following code in settings.py

STATIC\_URL = '/static/'

STATICFILES\_DIRS=[BASE\_DIR/'static']

6. urls.py

In this file we are just defining urls. To define urls:

* import view functions: from Quiz.views import \*
* add url to urlpatterns: path(‘login/’, loginPage,name=’login’),

**Code:**

"""DjangoQuiz URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/3.1/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import path

from Quiz.views import \*

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path('admin/', admin.site.urls),

path('', home,name='home'),

path('addQuestion/', addQuestion,name='addQuestion'),

path('login/', loginPage,name='login'),

path('logout/', logoutPage,name='logout'),

path('register/', registerPage,name='register'),

]

urlpatterns += static(settings.MEDIA\_URL,document\_root=settings.MEDIA\_ROOT)

7. views.py :

**Code:**

from django.shortcuts import redirect,render

from django.contrib.auth import login,logout,authenticate

from .forms import \*

from .models import \*

from django.http import HttpResponse

# Create your views here.

**def** home(request):

**if** request.method == 'POST':

print(request.POST)

questions=QuesModel.objects.all()

score=0

wrong=0

correct=0

total=0

**for** q **in** questions:

total+=1

print(request.POST.get(q.question))

print(q.ans)

print()

**if** q.ans == request.POST.get(q.question):

score+=10

correct+=1

**else**:

wrong+=1

percent = score/(total\*10) \*100

context = {

'score':score,

'time': request.POST.get('timer'),

'correct':correct,

'wrong':wrong,

'percent':percent,

'total':total

}

**return** render(request,'Quiz/result.html',context)

**else**:

questions=QuesModel.objects.all()

context = {

'questions':questions

}

**return** render(request,'Quiz/home.html',context)

**def** addQuestion(request):

**if** request.user.is\_staff:

form=addQuestionform()

**if**(request.method=='POST'):

form=addQuestionform(request.POST)

**if**(form.is\_valid()):

form.save()

**return** redirect('/')

context={'form':form}

**return** render(request,'Quiz/addQuestion.html',context)

**else**:

**return** redirect('home')

**def** registerPage(request):

**if** request.user.is\_authenticated:

**return** redirect('home')

**else**:

form = createuserform()

**if** request.method=='POST':

form = createuserform(request.POST)

**if** form.is\_valid() :

user=form.save()

**return** redirect('login')

context={

'form':form,

}

**return** render(request,'Quiz/register.html',context)

**def** loginPage(request):

**if** request.user.is\_authenticated:

**return** redirect('home')

**else**:

**if** request.method=="POST":

username=request.POST.get('username')

password=request.POST.get('password')

user=authenticate(request,username=username,password=password)

**if** user is not None:

login(request,user)

**return** redirect('/')

context={}

**return** render(request,'Quiz/login.html',context)

**def** logoutPage(request):

logout(request)

**return** redirect('/')

View.py is the file which contains the main or we can say the business logic of the project as this file can only access the database and also it can pass the information to templates so that it can be displayed on the web browser.

Home

This is the most important view of our project as this method is responsible for displaying the quiz page as well as the result page.

**For Quiz Page:** It renders all the questions from QuesModel and then it just passes all the questions to home.html

**For result page:** It calculates the score by giving ten marks for every correct question and zero (i.e. no negative marking) for every incorrect question. And then it calculates the percentage score.

Finally, it sends all these calculated values along with time elapsed, number of correct answers, number of incorrect answers to result.html. Result.html displays all this information.

Add Question:

This method is only for admin, if any user tries to access this method, the user is redirected to the home page. We are basically using the add question form in this to create a form object.

To validate the information entered by the user we are using is\_valid method and after validating the information we are adding another question in the database using save() method.

Result.html

{% **extends** 'Quiz/dependencies.html' %}

{% block content %}

<div **class**="container ">

<div **class**="card-columns" style="padding: 10px; margin: 20px;">

<div **class**="card" align="centre " style="width: 32rem; border:5px black solid">

<img **class**="card-img-top" src="http://kmit.in/emagazine/wp-content/uploads/2018/02/karnataka-results.jpg" alt="Card image cap">

<div **class**="card-body">

<h5 **class**="card-title">Score: {{score}}</h5>

<p **class**="card-text">Percentage: {{percent}}%</p>

<p **class**="card-text">Time Taken: {{time}} seconds</p>

<p **class**="card-text">Correct answers: {{correct}}</p>

<p **class**="card-text">Incorrect answers: {{wrong}}</p>

<p **class**="card-text">Total questions: {{total}}</p>

<h5>All the best **for** next quiz!</h5>

</div>

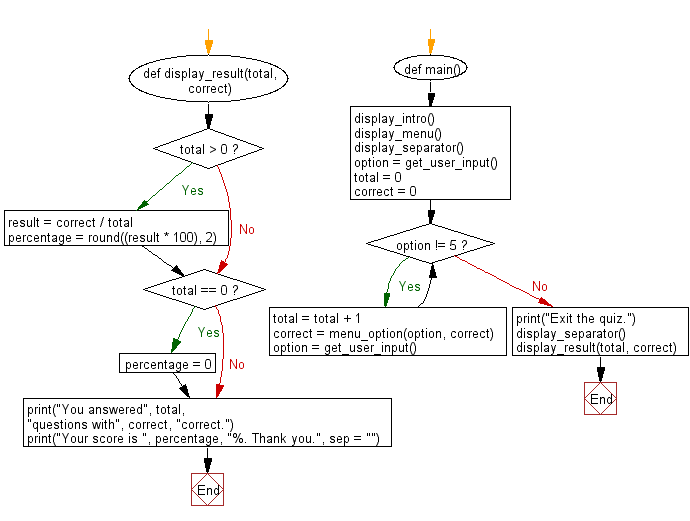
</div>

</div>

</div>

{% endblock %}

FLOWCHART



Technology used

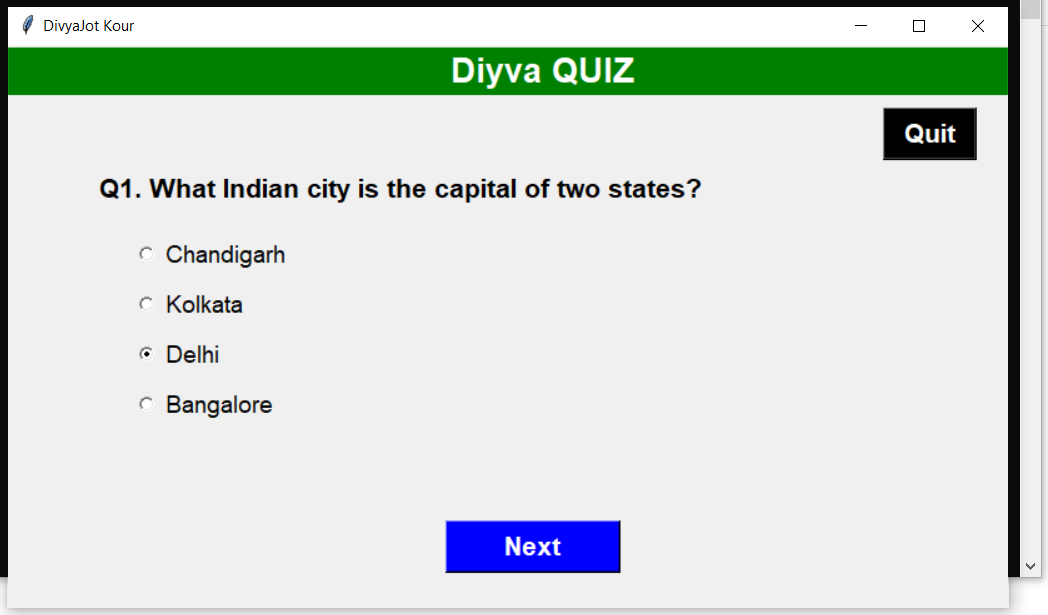
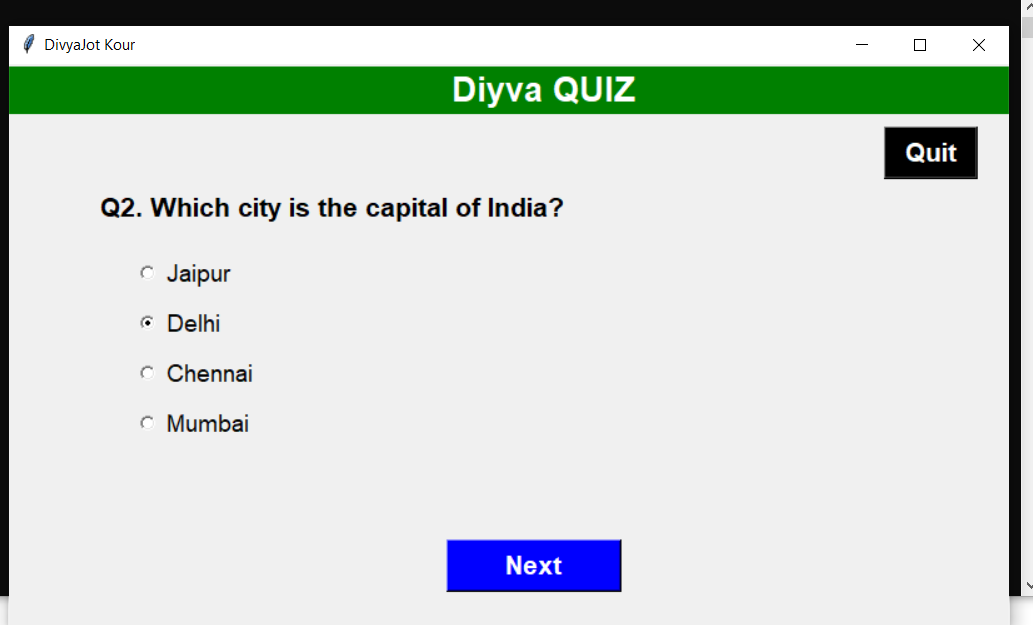
Object-oriented programming

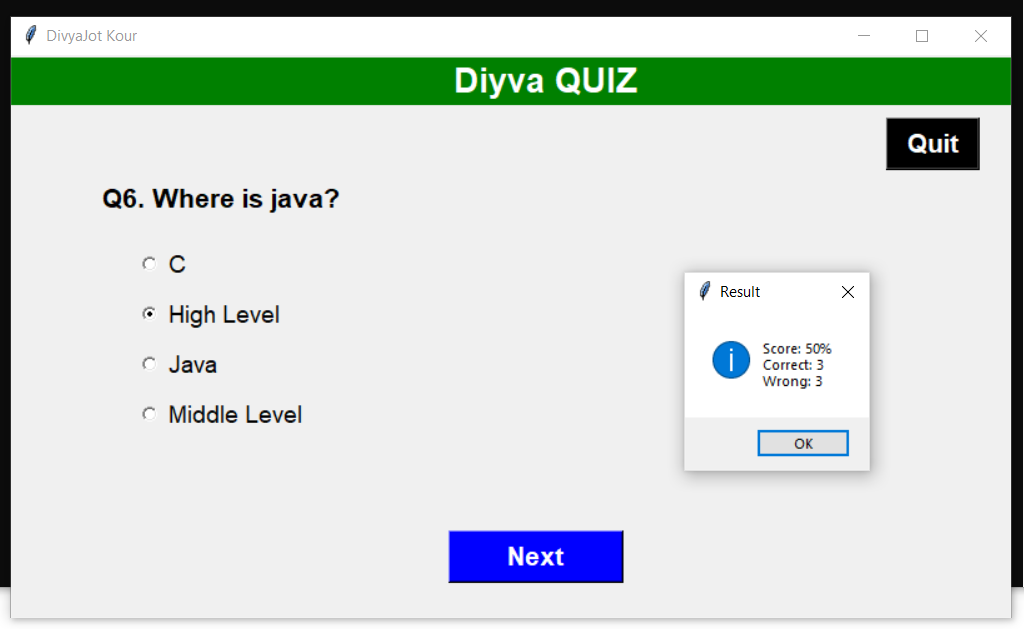
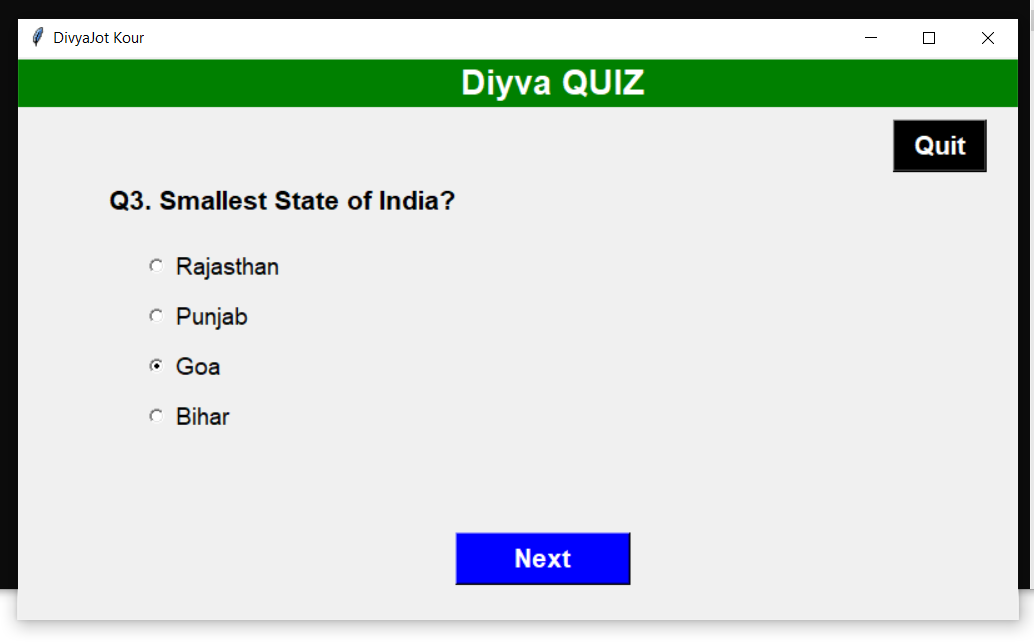
Python Tkinter Quiz Application Code Description The code is written using**Object-oriented programming** and the class name is Quiz. In the below code we have created a constructor that holds the values like the number of questions, the title of the window, questions with multiple options using the radio button, functions, etc.

CONCLUSIONS

**In this tutorial, you’ve learned how to:**

* **Interact** with the user in the terminal
* **Improve** the usability of your application
* **Refactor** your application to continuously improve it
* **Store** data in dedicated data files





**REFERENCES**

1. [Python Crash Course](https://realpython.com/best-python-books/#python-crash-course)
2. [Python Cookbook](https://realpython.com/best-python-books/#python-cookbook)