**||Jai Sri Gurudev||**

**Sri Adichunchanagiri Shikshana Trust ®**

**BGS NATIONAL PUBLIC SCHOOL**

**(Affiliated to Central Board of Secondary Education, New Delhi)**

**Hulimavu, Bannerghatta Road, Bengaluru - 560 076**

****

**Computer Science Investigatory Project**

Year: 2023-24

Topic: Tax Calculator

**Name:** SHASHANK S PATIL

**Class:** XII A

**Exam Roll No:**

# **Certificate**

This is to certify that  **SHASHANK S PATIL** of class XII of BGS National Public School has successfully completed the Investigatory Project in Computer Science for **ALL INDIA SENIOR SECONDARY CERTIFICATE EXAMINATION (AISSCE)** prescribed by CBSE in the year 2023-2024.

Date:

Principal Sign

External Examiner Internal Examiner

# **Acknowledgements**

I would like to express my gratitude to our Respected **Principal Dr Malini M Dutta**, **and Vice Principal Ms Savitha Suverna** for being a constant pillar of support.

I would then like to thank our **Computer Science teacher,**

**Ms. Babitha E Z** for helping me with this project and guiding me throughout and helping me in every step of the project.

I would also like to thank God Almighty and my parents for always being by my side and my fellow mates who were always ready to help.

# **Table of Contents**

|  |  |  |
| --- | --- | --- |
| S.No | Title | Page No |
| **1** | Introduction | 4 |
| **2** | Project Selection | 5 |
| **3** | Working Environment | 6 |
| **4** | Libraries & Modules | 7 |
| **5** | Data Dictionary | 8 |
| **6** | Source Code | 11 |
| **7** | Log of project | **47** |
| **8** | Sample Output | 50 |
| **9** | Bibliography | 53 |

# **Introduction**

This project mainly focuses on the development of an application program to calculate Income Tax. This program makes it easy and convenient for individuals and companies to calculate their income tax.

The features of the project include the ability to choose the most favorable tax scheme by calculating both the old and new scheme. It can also account for the exceptions provided by the government.

It can calculate individual tax returns for the common folk as well as for large companies with many employees.

# **Project Selection**

The inspiration behind choosing this particular topic stems from a dual purpose – to facilitate corporations in streamlining their tax processes for increased efficiency and speed, while also delving deep into the intricate workings of taxation to foster a comprehensive understanding. Recognizing the complexities and challenges that businesses face in the realm of taxation, we were driven by the belief that developing tools or strategies to simplify tax-related tasks could significantly benefit corporations.

We envision our project as a practice tool that can help us understand taxes easier and understand the deep underlying processes that goes in the finance. this can act as a motivator towards and push us forward.

# **Working Environment**

**Optimum Requirements**

* **Operating System** – Windows 10
* **Processor** – Must be clocked over 1.5 GHz
* **Graphics Driver** – Intel Integrated Graphics or equivalent or greater
* **RAM** – 4 GB or more.
* **Hard Disk** – 1 TB
* **Python interpreter** – Python IDLE 3.6
* **MySQL**

# **Libraries & Modules**

|  |  |
| --- | --- |
| **Libraries** | **Purpose** |
| Kivy Module | It is used for GUI |
| openpyxl | To display data in excel |
| Mysql.connector | It is used to access mySQL |

# **Data Dictionary**

# **User defined Functions**

|  |  |
| --- | --- |
| **Functions** | **PURPOSE** |
| **gui\_main.py** | |
| \_\_init\_\_(self, \*\*kwargs) | Initializes the screen |
| show\_tost\_msg(self,text) | Creates Toast messages |
| place\_button(self, x, y, button\_label, font\_size) | Places Buttons |
| add\_text\_input(self, x, y, text\_input\_name) | Places Text box |
| on\_text\_input\_validate(self, instance, text\_input\_name) | Gets Text box data |
| delete\_widget\_by\_label(self, label) | Deletes unnecessary labels |
| on\_button\_press(self, instance, button\_label) | Checks which button is pressed |
| def build(self) | Runs the GUI |
| **detail**.**py** | |
| \_\_init\_\_(self, \*\*kwargs) | Initializes the screen |
| Tost(self,text) | Creates Toast messages |
| create\_label(self, message, x\_rel, y\_rel, font\_size=50, halign='center', valign='middle', widget=None) | Creates labels |
| add\_styled\_text\_input(self, x, y, text\_input\_name) | Places Text box |
| on\_text\_input\_validate(self, instance, text\_input\_name) | Gets Text box data |
| place\_button(self, x, y, button\_label, font\_size) | Places Buttons |
| delete\_widget\_by\_label(self) | Deletes unnecessary labels |
| on\_button\_press(self, instance, button\_label) | Checks which button is pressed |
| switch\_to\_tables(self, instance) | Changes the Scree to Table screen |
| **test.py** | |
| \_init\_\_(self, \*\*kwargs) | Initializes the screen |
| create\_label(self, message, x\_rel, y\_rel, font\_size=50, halign='center', valign='middle', widget=None) | Creates labels |
| place\_button(self, x, y, button\_label, font\_size, widget\_name) | Places Buttons |
| on\_widget\_press(self, instance, widget) | Checks which button is pressed |
| clear\_labels(self) | Deletes unnecessary labels |
| switch\_to\_home\_screen(self, instance) | Changes to home screen |
| **User\_check.py** | |
| execute(q) | Executes the Querry |
| test(q) | Prints data of querry |
| INSERT(username, password) | Inserts data into Login database |
| create\_new\_user(username, password) | Creates new user |
| delete\_account(username) | Deletes account |
| Update\_user(username,new\_password) | Updates user |
| check(USERNAME, passw) | Checks if it is correct password |
| add\_employee(name,id,doj,ctc,C\_80Exemption,D\_80Exemption,rent) | Adds employee to employee tabke |
| info(id) | Prints the data of Employee table |
| tax\_add(name,id,b,hra,sa,te,np,pf,it,td) | Adds data to tax table |
| **tax\_calc.py** | |
| tax\_calc(data,id) | Calculates Tax and returns Excel |

# **Source Code**

**gui\_main.py**

from kivy.app import App

from kivy.uix.label import Label

from kivy.uix.relativelayout import RelativeLayout

from kivy.uix.image import Image

from kivy.uix.button import Button

from kivy.uix.textinput import TextInput

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.screenmanager import ScreenManager, Screen

from kivy.clock import Clock

from test import NextScreen

import User\_check

username=""

password=""

Username=""

New\_password=""

class HomeScreen(Screen):

def \_\_init\_\_(self, \*\*kwargs):

super(HomeScreen, self).\_\_init\_\_(\*\*kwargs)

background\_image = Image(source='2.png', allow\_stretch=True)

self.add\_widget(background\_image)

# Place buttons

self.place\_button(0.13, 0.127, 'Create Account',60)

self.place\_button(0.68, 0.33, 'login', 40)

self.place\_button(0.7, 0.127, 'Forgot Password',60)

# Add text input boxes

self.add\_text\_input(0.42, 0.56, 'Username')

self.add\_text\_input(0.42, 0.38, 'Password')

def show\_tost\_msg(self,text): # shows tost messages

background\_layout = BoxLayout(

orientation='horizontal',

pos\_hint={'center\_x': 0.5, 'top': 0.93}, # Centered at the top of the screen

size\_hint=(None, None),

size=(500, 60),

)

background\_layout.background = 'button\_bg.png'

invalid\_label = Label(

text=text,

font\_size=100,

color=(0, 0, 0, 1), # Black text color

)

background\_layout.add\_widget(invalid\_label)

self.add\_widget(background\_layout)

# removal of the label and background after 3 seconds

Clock.schedule\_once(lambda dt: self.remove\_widget(background\_layout), 3)

def place\_button(self, x, y, button\_label, font\_size):

if font\_size == 40:

button = Button(

text=button\_label,

size\_hint=(None, None),

size=(667, 150),

pos\_hint={'x': x, 'y': y},

font\_size=font\_size,

background\_color=(1, 1, 1, 0), # Fully transparent background color

background\_normal='', # No background image

color=(0, 0, 0, 1) # Text color

)

button.bind(on\_press=lambda instance: self.on\_button\_press(instance, button\_label))

self.add\_widget(button)

else:

button = Button(text=button\_label, size\_hint=(None, None), size=(667, 150),

pos\_hint={'x': x, 'y': y}, font\_size=font\_size,background\_normal='button\_bg.png',color=(0, 0, 0, 1))

button.bind(on\_press=lambda instance: self.on\_button\_press(instance, button\_label))

self.add\_widget(button)

def add\_text\_input(self, x, y, text\_input\_name):

box\_layout = BoxLayout(orientation='horizontal', spacing=20, size\_hint=(None, None), size=(600, 150),

pos\_hint={'x': x, 'y': y})

text\_input = TextInput(

multiline=False,

background\_normal="normal\_bg\_text\_box.png",

background\_active='active\_bg\_text\_box.png',

foreground\_color=(0, 0, 0, 1), # Text color

cursor\_color=(0, 0, 0, 1), # Cursor color

hint\_text=f'{text\_input\_name}...',

hint\_text\_color=(0.5, 0.5, 0.5, 1), # Hint text color

padding=(20, 10),

font\_size=50,

size=(600, 150),

on\_text\_validate=lambda instance: self.on\_text\_input\_validate(instance, text\_input\_name)

)

box\_layout.add\_widget(text\_input)

self.add\_widget(box\_layout)

def on\_text\_input\_validate(self, instance, text\_input\_name):

global username, password , Username , New\_password

print(f"Enter key pressed for {text\_input\_name}. Text Input string: {instance.text}")

if text\_input\_name == "Username":

username=instance.text

if text\_input\_name == "Password":

password=instance.text

if text\_input\_name == "Username":

Username=instance.text

if text\_input\_name == "New Password":

New\_password = instance.text

def delete\_widget\_by\_label(self, label):

for widget in self.children:

if isinstance(widget, Button) and widget.text == label:

self.remove\_widget(widget)

print(f"Widget with label '{label}' deleted.")

break

def on\_button\_press(self, instance, button\_label):

print(f"{button\_label} pressed!")

if button\_label == "Forgot Password":

self.delete\_widget\_by\_label("login")

self.place\_button(0.68, 0.33, 'Next', 40) # red color, font size 18

self.place\_button(0.7, 0.127, 'Delete Account', 60) # blue color, font size 20

self.add\_text\_input(0.42, 0.56, 'Username')

self.add\_text\_input(0.42, 0.38, 'New Password')

if button\_label == "Delete Account":

User\_check.delete\_account(Username)

self.show\_tost\_msg("Account deleted")

if button\_label == "Create Account":

self.delete\_widget\_by\_label("Next")

self.delete\_widget\_by\_label("login")

self.delete\_widget\_by\_label("Create Account")

self.place\_button(0.13, 0.127, 'Back to home', 60)

self.add\_text\_input(0.42, 0.56, 'Username')

self.add\_text\_input(0.42, 0.38, 'Password')

self.place\_button(0.68, 0.33, 'Create', 40)

if button\_label == "Back to home":

self.delete\_widget\_by\_label("Create")

self.delete\_widget\_by\_label("Back to home")

self.place\_button(0.68, 0.33, 'login', 40)

if button\_label == "Create":

User\_check.INSERT(username,password)

self.show\_tost\_msg("Account Created")

self.delete\_widget\_by\_label("Create")

self.place\_button(0.13, 0.127, 'Create Account', 60)

self.place\_button(0.68, 0.33, 'login', 40)

self.place\_button(0.7, 0.127, 'Forgot Password', 60)

self.add\_text\_input(0.42, 0.56, 'Username')

self.add\_text\_input(0.42, 0.38, 'Password')

if button\_label == "Next":

self.delete\_widget\_by\_label("Next")

User\_check.Update\_user(Username,New\_password)

self.place\_button(0.13, 0.127, 'Create Account', 60) # green color, font size 24

self.place\_button(0.68, 0.33, 'login', 40) # red color, font size 18

self.place\_button(0.7, 0.127, 'Forgot Password', 60)

self.add\_text\_input(0.42, 0.56, 'Username')

self.add\_text\_input(0.42, 0.38, 'Password')

if button\_label == 'login' and User\_check.check(username,password) == "access granted":

# Create an instance of NextScreen and switch to it

next\_screen = NextScreen(name='next\_screen')

self.manager.add\_widget(next\_screen)

self.manager.current = 'next\_screen'

global page

page=1

if button\_label == 'login' and User\_check.check(username, password) == "access denied":

self.show\_tost\_msg("Invalid login credentials")

#def on\_text\_input\_validate(self, instance):

# print(f"Enter key pressed. Text Input string: {instance.text}")

class MyApp(App):

def build(self):

global username, password

# Create the screen manager

sm = ScreenManager()

# Add the home screen to the screen manager

home\_screen = HomeScreen(name='home\_screen')

sm.add\_widget(home\_screen)

# Add the screen manager to the layout

layout = RelativeLayout()

layout.add\_widget(sm)

return layout

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

MyApp().run()

**details.py**

from kivy.uix.screenmanager import Screen

from kivy.uix.image import Image

from kivy.uix.button import Button

from kivy.uix.label import Label

from kivy.uix.boxlayout import BoxLayout

import User\_check

from kivy.uix.textinput import TextInput

from kivy.clock import Clock

import tax

import sys

data3 =User\_check.test("select \* from employee;")

length = len(data3)

p=0

print(User\_check.info(1))

ename=""

CTC=0

DOJ=""

C80=0

D80=0

rent=0

class Details(Screen):

def \_\_init\_\_(self, \*\*kwargs):

super(Details, self).\_\_init\_\_(\*\*kwargs)

global sid1

from test import NextScreen

sid1 = NextScreen.shared\_id

sys.stdout.write("\ntesting plz plz :"+str(sid1))

if sid1 == 1:

self.add\_widget(Image(source='details.png', allow\_stretch=True))

self.add\_widget(Button(text='Go Back', size=(400, 100), size\_hint=(None, None), pos=(300, 200),

on\_press=self.switch\_to\_tables))

self.add\_styled\_text\_input(0.34, 0.6, 'Employee Name')

self.add\_styled\_text\_input(0.34, 0.48, 'DOJ')

self.add\_styled\_text\_input(0.74, 0.638, 'CTC')

self.add\_styled\_text\_input(0.74, 0.5, '80C')

self.add\_styled\_text\_input(0.74, 0.42, '80D')

self.add\_styled\_text\_input(0.74, 0.37, 'Rent')

self.place\_button(0.74, 0.10, 'ADD PERSON', 60)

else:

print("just why:")

data = User\_check.info(sid1)

sys.stdout.write(str(sid1))

self.add\_widget(Image(source='details.png', allow\_stretch=True))

self.add\_widget(Button(text='Go Back', size=(400, 100), size\_hint=(None, None), pos=(300, 200),

on\_press=self.switch\_to\_tables))

self.place\_button(0.16, 0.21, 'Del', 1) # green color, font size 24

self.place\_button(0.32, 0.21, 'Up', 1) # red color, font size 18

self.place\_button(0.68, 0.21, 'Calc', 1) # blue color, font size 20

self.add\_widget(self.create\_label("%s" % (data[0][0]), 0.34, 0.64, widget="label1"))

self.add\_widget(self.create\_label("%s" % (data[0][1]), 0.34, 0.58, widget="label2"))

self.add\_widget(self.create\_label("%s" % (data[0][2]), 0.34, 0.514, widget="label3"))

self.add\_widget(self.create\_label("%s" % (data[0][3]), 0.74, 0.638, widget="label4"))

self.add\_widget(self.create\_label("%s" % (data[0][4]), 0.74, 0.582, widget="label5"))

self.add\_widget(self.create\_label("%s" % (data[0][5]), 0.74, 0.53, widget="label6"))

self.add\_widget(self.create\_label("%s" % (data[0][6]), 0.74, 0.47, widget="label7"))

def Tost(self,text):

background\_layout = BoxLayout(

orientation='horizontal',

pos\_hint={'center\_x': 0.5, 'top': 0.93}, # Centered at the top of the screen

size\_hint=(None, None),

size=(500, 60), # Set the size based on your background image

)

background\_layout.background = 'button\_bg.png'

invalid\_label = Label(

text=text,

font\_size=100,

color=(0, 0, 0, 1), # Black text color

)

background\_layout.add\_widget(invalid\_label)

self.add\_widget(background\_layout)

# Schedule the removal of the label and background after 3 seconds

Clock.schedule\_once(lambda dt: self.remove\_widget(background\_layout), 3)

def create\_label(self, message, x\_rel, y\_rel, font\_size=50, halign='center', valign='middle', widget=None):

label\_pos\_rel = {'center\_x': x\_rel, 'center\_y': y\_rel}

label = Label(text=message, font\_size=font\_size, halign=halign, valign=valign, pos\_hint=label\_pos\_rel,

color=(0, 0, 0, 1))

label.bind(on\_press=lambda instance: self.on\_widget\_press(instance, widget))

return label

def add\_styled\_text\_input(self, x, y, text\_input\_name):

box\_layout = BoxLayout(orientation='horizontal', spacing=20, size\_hint=(None, None), size=(600, 150),

pos\_hint={'x': x, 'y': y})

text\_input = TextInput(

multiline=False,

background\_normal="normal\_bg\_text\_box.png",

background\_active='active\_bg\_text\_box.png',

foreground\_color=(0, 0, 0, 1), # Text color

cursor\_color=(0, 0, 0, 1), # Cursor color

hint\_text=f'{text\_input\_name}...',

hint\_text\_color=(0.5, 0.5, 0.5, 1), # Hint text color

padding=(20, 10),

font\_size=50,

size=(600, 150),

on\_text\_validate=lambda instance: self.on\_text\_input\_validate(instance, text\_input\_name)

)

box\_layout.add\_widget(text\_input)

self.add\_widget(box\_layout)

def on\_text\_input\_validate(self, instance, text\_input\_name):

global ename, CTC , DOJ , C80, D80 , rent

sys.stdout.write(f"Enter key pressed for {text\_input\_name}. Text Input string: {instance.text}")

if text\_input\_name == "Employee Name":

ename=instance.text

if text\_input\_name == "CTC":

CTC=instance.text

if text\_input\_name == "DOJ":

DOJ=instance.text

if text\_input\_name == "80C":

C80 = instance.text

if text\_input\_name == "80D":

D80 = instance.text

if text\_input\_name == "Rent":

rent= instance.text

def place\_button(self, x, y, button\_label, font\_size):

if font\_size == 60:

button = Button(

text=button\_label,

size\_hint=(None, None),

size=(667, 150),

pos\_hint={'x': x, 'y': y},

font\_size=font\_size,

background\_color=(1, 1, 1, 0),

background\_normal='bg\_button.png',

color=(0, 0, 0, 1) # Text color

)

button.bind(on\_press=lambda instance: self.on\_button\_press(instance, button\_label))

self.add\_widget(button)

else:

button = Button(

text=button\_label,

size\_hint=(None, None),

size=(667, 150),

pos\_hint={'x': x, 'y': y},

font\_size=font\_size,

background\_color=(1, 1, 1, 0), # Fully transparent background color

background\_normal='', # No background image

#color=(0, 0, 0, 1) # Text color

)

button.bind(on\_press=lambda instance: self.on\_button\_press(instance, button\_label))

self.add\_widget(button)

def delete\_widget\_by\_label(self):

text\_inputs\_to\_remove = [widget for widget in self.children if isinstance(widget, BoxLayout) and

isinstance(widget.children[0], TextInput)]

for box\_layout in text\_inputs\_to\_remove:

self.remove\_widget(box\_layout)

def on\_button\_press(self, instance, button\_label):

print(f"{button\_label} pressed!")

if button\_label == "Del":

self.Tost("User deleted")

User\_check.del\_employee(sid1)

sys.stdout.write("deleated my guy especiall: "+str(sid1))

if button\_label == "Calc":

self.Tost("Tax Calculated")

data = User\_check.info(sid1)

c=[]

for i in data:

for j in i:

c.append(j)

print(c)

c.pop(2)

c=[0]+c

print(c)

print(data)

#sys.stdout.write(data)

tax.tax\_calc(c,sid1)

if button\_label == "Up":

self.add\_styled\_text\_input(0.34, 0.6, 'Employee Name')

#self.add\_styled\_text\_input(0.34, 0.59, 'Password')

self.add\_styled\_text\_input(0.34, 0.48, 'DOJ')

self.add\_styled\_text\_input(0.74, 0.638, 'CTC')

self.add\_styled\_text\_input(0.74, 0.5, '80C')

self.add\_styled\_text\_input(0.74, 0.42, '80D')

self.add\_styled\_text\_input(0.74, 0.37, 'Rent')

self.place\_button(0.74, 0.10, 'UPDATE', 60)

if button\_label == "UPDATE":

sys.stdout.write(ename+"id:"+str(sid1)+"\ndoj:"+DOJ+"\n"+str(CTC)+"\n"+str(C80)+str(D80)+str(rent))

User\_check.update\_employee(ename,sid1,DOJ,CTC,C80,D80,rent)

if button\_label == "ADD PERSON":

User\_check.add\_employee(ename,length+1,DOJ,CTC,C80,D80,rent)

self.switch\_to\_tables()

def switch\_to\_tables(self, instance):

self.delete\_widget\_by\_label()

self.manager.current = 'next\_screen'

details\_screen = self.manager.get\_screen('details')

self.manager.remove\_widget(details\_screen)

**User\_check.py**

import sys

import mysql.connector as connection

con = connection.connect(host="localhost", user="root", database="test", password="password")

def execute(q):

cursor = con.cursor()

cursor.execute(q)

con.commit()

def test(q):

cursor = con.cursor()

cursor.execute(q)

data = cursor.fetchall()

for i in data:

pass

# print(i)

return data

def INSERT(username, password):

values\_query = '''

INSERT INTO login

values("%s","%s");''' % (username, password)

print(values\_query)

execute(values\_query)

#########################################################

'''initial\_query=

CREATE TABLE login

(username varchar(20) primary key,

password varchar(20));

initial\_query=

CREATE TABLE EMPLOYEE

(employee\_name varchar(20),

employee\_id int primary key,

doj date,

Cost\_to\_Company int,

80C int,

80D int,

rent int);

CREATE TABLE Tax

(employee\_name varchar(20),

employee\_id int primary key,

Basic int,

HRA int,

Sp\_Allow int,

Total\_Earning int,

Net\_pay int,

PF int,

IT int,

Tot\_Detuct int);

'''

#execute(initial\_query)

def create\_new\_user(username, password):

# Y=check(username)

# if Y == "access "

INSERT(username, password)

def delete\_account(username):

query='''DELETE FROM LOGIN WHERE USERNAME = "%s";'''%username

execute(query)

def Update\_user(username,new\_password):

q="""UPDATE LOGIN SET password = "%s" where username = "%s" ;"""%(new\_password,username)

execute(q)

def check(USERNAME, passw):

q = "SELECT PASSWORD FROM LOGIN WHERE USERNAME = '%s';" % USERNAME

y = test(q)

# print(y)

if [(passw,)] == y:

return "access granted"

else:

return "access denied"

# return q

# INSERT("a","p")

def add\_employee(name,id,doj,ctc,C\_80Exemption,D\_80Exemption,rent):

values\_query = '''

INSERT INTO EMPLOYEE

values("%s","%s","%s","%s","%s","%s","%s");''' % (name,id,doj,ctc,C\_80Exemption,D\_80Exemption,rent)

execute(values\_query)

def del\_employee(sid):

q='''delete from employee where employee\_id = %s;'''%(sid)

execute(q)

def update\_employee(name,id,doj,ctc,C\_80Exemption,D\_80Exemption,rent):

q="""UPDATE employee SET employee\_name = "%s",

doj = "%s",

Cost\_to\_company = %s,

80C = %s,

80D = %s,

rent = %s

WHERE employee\_id = %s;

"""%(name,doj,ctc,C\_80Exemption,D\_80Exemption,rent,id)

sys.stdout.write(q)

execute(q)

def info(id):

q = "Select \* from EMPLOYEE where employee\_id = '%s';" % id

y = test(q)

return y

print(info(1))

def tax\_add(name,id,b,hra,sa,te,np,pf,it,td):

q="""INSERT INTO TAX

values("%s",%s,%s,%s,%s,%s,%s,%s,%s,%s);"""%(name,id,b,hra,sa,te,np,pf,it,td)

#add\_employee("p3",8,"2023-04-22",400000,4040,404,204)

**test.py**

from kivy.uix.screenmanager import Screen

from kivy.uix.image import Image

from kivy.uix.button import Button

from kivy.uix.label import Label

from kivy.clock import Clock

import User\_check

from details import Details

from kivy.uix.boxlayout import BoxLayout

import tax

data =User\_check.test("select \* from employee;")

length = len(data)

while len(data) % 6 !=0:

data.append(("","","","","",""))

done=True

global p

p=0

import sys

class NextScreen(Screen):

shared\_id = 1

def \_\_init\_\_(self, \*\*kwargs):

super(NextScreen, self).\_\_init\_\_(\*\*kwargs)

self.add\_widget(Image(source='newnew3.png', allow\_stretch=True)) # Use your next screen background image

self.add\_widget(Button(text='Go to Home Screen', size=(400, 100), size\_hint=(None, None), pos=(300, 200),

on\_press=self.switch\_to\_home\_screen))

self.place\_button(0.2, 0.473, 'Details', 40, "d1")

self.place\_button(0.28, 0.473, 'Select', 40, "s1")

self.place\_button(0.434, 0.473, 'Details', 40, "d2")

self.place\_button(0.514, 0.473, 'Select', 40, "s2")

self.place\_button(0.664, 0.473, 'Details', 40, "d3")

self.place\_button(0.744, 0.473, 'Select', 40, "s3")

self.place\_button(0.2, 0.186, 'Details', 40, "d4")

self.place\_button(0.28, 0.186, 'Select', 40, "s4")

self.place\_button(0.434, 0.186, 'Details', 40, "d5")

self.place\_button(0.514, 0.186, 'Select', 40, "s5")

self.place\_button(0.664, 0.186, 'Details', 40, "d6")

self.place\_button(0.744, 0.186, 'Select', 40, "s6")

self.place\_button(0.82, 0.2, 'next', 40, "nxt")

self.place\_button(0, 0.2, 'back', 40, "bk")

self.add\_widget(self.create\_label("%s" %(data[0+p][0]), 0.3, 0.63,widget= "label1"))

self.add\_widget(self.create\_label("%s" %(data[0+p][1]), 0.3, 0.594, widget="label2"))

self.add\_widget(self.create\_label("%s" %(data[0+p][2]), 0.3, 0.558, widget="label3"))

self.add\_widget(self.create\_label("%s"%(data[1+p][0]), 0.53, 0.63,widget= "label4"))

self.add\_widget(self.create\_label("%s"%(data[1+p][1]), 0.53, 0.594,widget= "label5"))

self.add\_widget(self.create\_label("%s"%(data[1+p][2]), 0.53, 0.558, widget="label6"))

self.add\_widget(self.create\_label("%s"%(data[2+p][0]), 0.76, 0.63, widget="label7"))

self.add\_widget(self.create\_label("%s"%(data[2+p][1]), 0.76, 0.594,widget= "label8"))

self.add\_widget(self.create\_label("%s"%(data[2+p][2]), 0.76, 0.558,widget= "label9"))

self.add\_widget(self.create\_label("%s"%(data[3+p][0]), 0.3, 0.34, widget="label10"))

self.add\_widget(self.create\_label("%s"%(data[3+p][1]), 0.3, 0.302, widget="label11"))

self.add\_widget(self.create\_label("%s"%(data[3+p][2]), 0.3, 0.267, widget="label12"))

self.add\_widget(self.create\_label("%s"%(data[4+p][0]), 0.53, 0.34, widget="label13"))

self.add\_widget(self.create\_label("%s"%(data[4+p][1]), 0.53, 0.302,widget="label14"))

self.add\_widget(self.create\_label("%s"%(data[4+p][2]), 0.53, 0.267, widget="label15"))

self.add\_widget(self.create\_label("%s"%(data[5+p][0]), 0.76, 0.34, widget="label16"))

self.add\_widget(self.create\_label("%s"%(data[5+p][1]), 0.76, 0.302,widget= "label17"))

self.add\_widget(self.create\_label("%s"%(data[5+p][2]), 0.76, 0.267, widget="label18"))

def create\_label(self, message, x\_rel, y\_rel, font\_size=50, halign='center', valign='middle', widget=None):

label\_pos\_rel = {'center\_x': x\_rel, 'center\_y': y\_rel}

label = Label(text=message, font\_size=font\_size, halign=halign, valign=valign, pos\_hint=label\_pos\_rel,

color=(0, 0, 0, 1))

label.bind(on\_press=lambda instance: self.on\_widget\_press(instance, widget))

return label

def place\_button(self, x, y, button\_label, font\_size, widget\_name):

if button\_label == "next" or button\_label == "back":

button = Button(

text=button\_label,

size\_hint=(None, None),

size=(667, 150),

pos\_hint={'x': x, 'y': y},

font\_size=font\_size,

background\_color=(1, 1, 1, 0),

background\_normal='',

color=(0, 0, 0, 1)

)

if button\_label == "next":

button.bind(on\_release=lambda instance: self.on\_widget\_press(instance, 'nxt'))

self.add\_widget(button)

else:

button.bind(on\_release=lambda instance: self.on\_widget\_press(instance, 'bk'))

self.add\_widget(button)

else:

button = Button(text=button\_label, size\_hint=(None, None), size=(200, 50),

pos\_hint={'x': x, 'y': y}, font\_size=font\_size, background\_normal='bg\_button.png',

color=(0, 0, 0, 1))

button.bind(on\_release=lambda instance: self.on\_widget\_press(instance, widget\_name))

setattr(self, widget\_name, button) # Set the attribute dynamically

self.add\_widget(button)

def show\_tost\_msg(self,text): # shows tost messages

background\_layout = BoxLayout(

orientation='horizontal',

pos\_hint={'center\_x': 0.5, 'top': 0.93}, # Centered at the top of the screen

size\_hint=(None, None),

size=(500, 60),

)

background\_layout.background = 'button\_bg.png'

invalid\_label = Label(

text=text,

font\_size=100,

color=(0, 0, 0, 1), # Black text color

)

background\_layout.add\_widget(invalid\_label)

self.add\_widget(background\_layout)

# removal of the label and background after 3 seconds

Clock.schedule\_once(lambda dt: self.remove\_widget(background\_layout), 3)

def on\_widget\_press(self, instance, widget):

if widget is not None:

if widget == "nxt":

#self.manager.current = 'home\_screen'

self.manager.current= "next\_screen"

global p

p+=6

self.clear\_labels()

self.add\_widget(self.create\_label("%s" % (data[0 + p][0]), 0.3, 0.63, widget="label1"))

self.add\_widget(self.create\_label("%s" % (data[0 + p][1]), 0.3, 0.594, widget="label2"))

self.add\_widget(self.create\_label("%s" % (data[0 + p][2]), 0.3, 0.558, widget="label3"))

self.add\_widget(self.create\_label("%s" % (data[1 + p][0]), 0.53, 0.63, widget="label4"))

self.add\_widget(self.create\_label("%s" % (data[1 + p][1]), 0.53, 0.594, widget="label5"))

self.add\_widget(self.create\_label("%s" % (data[1 + p][2]), 0.53, 0.558, widget="label6"))

self.add\_widget(self.create\_label("%s" % (data[2 + p][0]), 0.76, 0.63, widget="label7"))

self.add\_widget(self.create\_label("%s" % (data[2 + p][1]), 0.76, 0.594, widget="label8"))

self.add\_widget(self.create\_label("%s" % (data[2 + p][2]), 0.76, 0.558, widget="label9"))

self.add\_widget(self.create\_label("%s" % (data[3 + p][0]), 0.3, 0.34, widget="label10"))

self.add\_widget(self.create\_label("%s" % (data[3 + p][1]), 0.3, 0.302, widget="label11"))

self.add\_widget(self.create\_label("%s" % (data[3 + p][2]), 0.3, 0.267, widget="label12"))

self.add\_widget(self.create\_label("%s" % (data[4 + p][0]), 0.53, 0.34, widget="label13"))

self.add\_widget(self.create\_label("%s" % (data[4 + p][1]), 0.53, 0.302, widget="label14"))

self.add\_widget(self.create\_label("%s" % (data[4 + p][2]), 0.53, 0.267, widget="label15"))

self.add\_widget(self.create\_label("%s" % (data[5 + p][0]), 0.76, 0.34, widget="label16"))

self.add\_widget(self.create\_label("%s" % (data[5 + p][1]), 0.76, 0.302, widget="label17"))

self.add\_widget(self.create\_label("%s" % (data[5 + p][2]), 0.76, 0.267, widget="label18"))

self.add\_widget(

Button(text='Go to Home Screen', size=(400, 100), size\_hint=(None, None), pos=(300, 200),

on\_press=self.switch\_to\_home\_screen))

self.place\_button(0.2, 0.473, 'Details', 40, "d1")

self.place\_button(0.28, 0.473, 'Select', 40, "s1")

self.place\_button(0.434, 0.473, 'Details', 40, "d2")

self.place\_button(0.514, 0.473, 'Select', 40, "s2")

self.place\_button(0.664, 0.473, 'Details', 40, "d3")

self.place\_button(0.744, 0.473, 'Select', 40, "s3")

self.place\_button(0.2, 0.186, 'Details', 40, "d4")

self.place\_button(0.28, 0.186, 'Select', 40, "s4")

self.place\_button(0.434, 0.186, 'Details', 40, "d5")

self.place\_button(0.514, 0.186, 'Select', 40, "s5")

self.place\_button(0.664, 0.186, 'Details', 40, "d6")

self.place\_button(0.744, 0.186, 'Select', 40, "s6")

self.place\_button(0, 0.2, 'back', 40, "bk")

if done == False:

self.place\_button(0.82, 0.2, 'next', 40, "nxt")

# Do something for Button 12

if widget in ["s1", "s2", "s3", "s4", "s5", "s6"]:

ssid=int(widget[1])+p

if ssid > length:

self.shared\_id = 9

sys.stdout.write("OOOOOOOOOOOOKKKKKKKK")

try:

details\_screen = self.manager.get\_screen('details')

pass

except:

details = Details(name='details')

self.manager.add\_widget(details)

# Switch to the "details" screen

# self.manager.current = "details"

# details = Details(name='details')

# self.manager.add\_widget(details)

self.manager.current = "details"

else:

data1=User\_check.info(ssid)

c = []

for i in data1:

for j in i:

c.append(j)

print(c)

c.pop(2)

c = [0] + c

tax.tax\_calc(c,ssid)

self.show\_tost\_msg("Tax calculated for: "+str(ssid))

if widget in ["d1","d2","d3","d4","d5","d6"]:

NextScreen.shared\_id = int(widget[1])+p

#sys.stdout.write(str(shared\_id)+" "+widget[1]+str(p)+"helooooooooooo\n\n")

try:

details\_screen = self.manager.get\_screen('details')

pass

except:

details = Details(name='details')

self.manager.add\_widget(details)

# Switch to the "details" screen

#self.manager.current = "details"

#details = Details(name='details')

#self.manager.add\_widget(details)

self.manager.current = "details"

if widget == "bk" :

# self.manager.current = 'home\_screen'

self.manager.current = "next\_screen"

p -= 6

self.clear\_labels()

self.add\_widget(self.create\_label("%s" % (data[0 + p][0]), 0.3, 0.63, widget="label1"))

self.add\_widget(self.create\_label("%s" % (data[0 + p][1]), 0.3, 0.594, widget="label2"))

self.add\_widget(self.create\_label("%s" % (data[0 + p][2]), 0.3, 0.558, widget="label3"))

self.add\_widget(self.create\_label("%s" % (data[1 + p][0]), 0.53, 0.63, widget="label4"))

self.add\_widget(self.create\_label("%s" % (data[1 + p][1]), 0.53, 0.594, widget="label5"))

self.add\_widget(self.create\_label("%s" % (data[1 + p][2]), 0.53, 0.558, widget="label6"))

self.add\_widget(self.create\_label("%s" % (data[2 + p][0]), 0.76, 0.63, widget="label7"))

self.add\_widget(self.create\_label("%s" % (data[2 + p][1]), 0.76, 0.594, widget="label8"))

self.add\_widget(self.create\_label("%s" % (data[2 + p][2]), 0.76, 0.558, widget="label9"))

self.add\_widget(self.create\_label("%s" % (data[3 + p][0]), 0.3, 0.34, widget="label10"))

self.add\_widget(self.create\_label("%s" % (data[3 + p][1]), 0.3, 0.302, widget="label11"))

self.add\_widget(self.create\_label("%s" % (data[3 + p][2]), 0.3, 0.267, widget="label12"))

self.add\_widget(self.create\_label("%s" % (data[4 + p][0]), 0.53, 0.34, widget="label13"))

self.add\_widget(self.create\_label("%s" % (data[4 + p][1]), 0.53, 0.302, widget="label14"))

self.add\_widget(self.create\_label("%s" % (data[4 + p][2]), 0.53, 0.267, widget="label15"))

self.add\_widget(self.create\_label("%s" % (data[5 + p][0]), 0.76, 0.34, widget="label16"))

self.add\_widget(self.create\_label("%s" % (data[5 + p][1]), 0.76, 0.302, widget="label17"))

self.add\_widget(self.create\_label("%s" % (data[5 + p][2]), 0.76, 0.267, widget="label18"))

self.add\_widget(

Button(text='Go to Home Screen', size=(400, 100), size\_hint=(None, None), pos=(300, 200),

on\_press=self.switch\_to\_home\_screen))

self.place\_button(0.2, 0.473, 'Details', 40, "d1")

self.place\_button(0.28, 0.473, 'Select', 40, "s1")

self.place\_button(0.434, 0.473, 'Details', 40, "d2")

self.place\_button(0.514, 0.473, 'Select', 40, "s2")

self.place\_button(0.664, 0.473, 'Details', 40, "d3")

self.place\_button(0.744, 0.473, 'Select', 40, "s3")

self.place\_button(0.2, 0.186, 'Details', 40, "d4")

self.place\_button(0.28, 0.186, 'Select', 40, "s4")

self.place\_button(0.434, 0.186, 'Details', 40, "d5")

self.place\_button(0.514, 0.186, 'Select', 40, "s5")

self.place\_button(0.664, 0.186, 'Details', 40, "d6")

self.place\_button(0.744, 0.186, 'Select', 40, "s6")

self.place\_button(0, 0.2, 'back', 40, "bk")

self.place\_button(0.82, 0.2, 'next', 40, "nxt")

def clear\_labels(self):

# Clear only label widgets from the screen

widgets\_to\_remove = [widget for widget in self.children if isinstance(widget, Label)]

for widget in widgets\_to\_remove:

self.remove\_widget(widget)

def switch\_to\_home\_screen(self, instance):

# Switch to the "HomeScreen"

self.manager.current = 'home\_screen'

details\_screen = self.manager.get\_screen('next\_screen')

self.manager.remove\_widget(details\_screen)

#sys.stdout.write("deleted next screen\n")

**tax.py**

def tax\_calc(data,sid):

j=data

print("working")

ctc = int(j[3])

C\_80Exemption = int(j[4])

D\_80Exemption = int(j[5])

rent = int(j[6])

print("please:mm",ctc,C\_80Exemption)

basic = (50 / 100) \* ctc

hra = (40 / 100) \* basic

pf = (12 / 100) \* basic

spal = ctc - (basic + hra + pf)

# income tax calc

if C\_80Exemption > 150000:

print("Invalid")

if D\_80Exemption > 25000:

print("invalid")

Anual\_basic = basic # date

Anual\_hra = hra

Anual\_sp = spal

total\_ = Anual\_sp + Anual\_hra + Anual\_basic

cal = rent - ((10 / 100) \* basic)

hra\_detuction = min([hra, cal])

sec80 = C\_80Exemption + D\_80Exemption

total\_detuction = hra\_detuction + sec80

def old(gross, less\_dens, s1, s2, s3, su):

less\_dens = -less\_dens

Net\_inc = gross + less\_dens

print(Net\_inc)

if Net\_inc < 250000:

tax\_rate = 0

if Net\_inc > 1000000:

tax = 5 / 100

s1 = tax \* (500000 - 250000)

tax = 20 / 100

s2 = tax \* (1000000 - 500000)

tax = 30 / 100

s3 = tax \* (Net\_inc - 1000000)

su = s1 + s2 + s3

if Net\_inc > 500000 and Net\_inc < 1000000:

tax = 5 / 100

s1 = tax \* (500000 - 250000)

tax = 20 / 100

s2 = tax \* (Net\_inc - 500000)

su = s1 + s2

if Net\_inc > 250000 and Net\_inc < 500000:

tax = 5 / 100

su = tax \* (Net\_inc - 250000)

Tax\_cess = su

cess = (4 / 100) \* Tax\_cess

IT = Tax\_cess + cess

recoverO = IT / 12

print(recoverO) # date

return recoverO

def new(gross, s1, s2, s3, s4, s5, s6, su):

Net\_inc = gross

if Net\_inc > 1500000:

tax = 5 / 100

s1 = tax \* (500000 - 250000)

tax = 10 / 100

s2 = tax \* (250000)

tax = 15 / 100

s3 = tax \* (250000)

tax = 20 / 100

s4 = tax \* (250000)

tax = 25 / 100

s5 = tax \* (250000)

tax = 30 / 100

s6 = tax \* (Net\_inc - 1500000)

su = s1 + s2 + s3 + s4 + s5 + s6

if Net\_inc > 500000 and Net\_inc < 750000:

tax = 5 / 100

s1 = tax \* (500000 - 250000)

tax = 10 / 100

s2 = tax \* (Net\_inc - 500000)

su = s1 + s2

if Net\_inc > 750000 and Net\_inc < 1000000:

tax = 5 / 100

s1 = tax \* (500000 - 250000)

tax = 10 / 100

s2 = tax \* (250000)

tax = 15 / 100

s3 = tax \* (Net\_inc - 750000)

su = s1 + s2 + s3

if Net\_inc > 1000000 and Net\_inc < 1250000:

tax = 5 / 100

s1 = tax \* (500000 - 250000)

tax = 10 / 100

s2 = tax \* (250000)

tax = 15 / 100

s3 = tax \* (250000)

tax = 20 / 100

s4 = tax \* (Net\_inc - 1000000)

su = s1 + s2 + s3 + s4

if Net\_inc > 1250000 and Net\_inc < 1500000:

tax = 5 / 100

s1 = tax \* (500000 - 250000)

tax = 10 / 100

s2 = tax \* (250000)

tax = 15 / 100

s3 = tax \* (250000)

tax = 20 / 100

s4 = tax \* (250000)

tax = 25 / 100

s5 = tax \* (Net\_inc - 1250000)

su = s1 + s2 + s3 + s4 + s5

if Net\_inc > 250000 and Net\_inc < 500000:

tax = 5 / 100

su = tax \* (Net\_inc - 250000)

Tax\_cess = su

cess = (4 / 100) \* Tax\_cess

IT = Tax\_cess + cess

recoverN = IT / 12

print(recoverN) # date

return recoverN

o = old(total\_, total\_detuction, 0, 0, 0, 0)

n = new(total\_, 0, 0, 0, 0, 0, 0, 0)

if o > n:

used = "new"

IT = n

else:

IT = o

used = "old"

te = (hra / 12 + basic / 12 + spal / 12)

print(hra, basic, spal, te)

print(pf, IT)

td = pf / 12 + IT

print("net:", te - td)

print("used:", used)

# exel conversion

from openpyxl import Workbook

from openpyxl.utils.dataframe import dataframe\_to\_rows

from openpyxl.styles import Alignment

wb = Workbook()

ws = wb.active

ws.merge\_cells('A1:D1')

cell = ws.cell(row=1, column=1)

cell.alignment = Alignment(horizontal='center', vertical='center')

ws['A1'] = "ABC Company Private Limited"

ws.merge\_cells('F3:J3')

cell = ws.cell(row=3, column=6)

cell.alignment = Alignment(horizontal='center', vertical='center')

ws['F1'] = "Payroll of month"

ws['F4'] = "Earnings"

ws['G4'] = "Amount"

ws['I4'] = "Deduction"

ws['J4'] = "Amount"

ws['I5'] = "PF"

ws['J5'] = pf / 12

ws['I6'] = "IT"

ws['J6'] = IT / 12

ws['I8'] = "Total Deductions"

ws['J8'] = td

ws['F9'] = "NET PAY"

ws['G9'] = te - td

ws['G8'] = te

ws['F8'] = "Total Earnings"

ws['G7'] = spal / 12

ws['F7'] = "Special Allow"

ws['G6'] = hra / 12

ws['F6'] = "HRA"

ws['G5'] = basic / 12

ws['F5'] = "Basic"

tex= "Tax of %s"%(sid)

wb.save(tex+".xlsx")

# **Log of project**

**LOG-1:30/6/2023**

* Ideation
* Discussion of topic – Tax calculator

**LOG-2:3/7/2023**

* Searched for various modules available to access mySql and Excel

**LOG-3:19/7/2023**

* Learning to work with kivy

**LOG-4:26/7/2023**

* Created SQL table Login

**LOG-5:2/8/2023**

* Worked on Home screen

**LOG-6:8/8/2023**

* Created SQL table User

**LOG-7:20/8/2023**

* Worked on sign up and login options using Python-SQL Connectivity

**LOG-8:24/8/2023**

* Worked on individual tax calculations

**LOG-9:29/8/2023**

* Worked on debugging code

**LOG-10:6/9/2023**

* Worked on designing for GUI

**LOG-11:13/9/2023**

* Worked on python connectivity with MySQL.

**LOG-12:21/9/2023**

* Worked on Home Screen GUI

**LOG-13:26/9/2023**

* Worked on designing on tabular system

**LOG-14:1/9/2023**

* Worked on debugging code

**LOG-15:6/9/2023**

* Worked on python connectivity with Excel

**LOG-16:11/9/2023**

* Finished individual tax calculations

**LOG-17:18/9/2023**

* Finished GUI

**LOG-18:30/10/2023**

* Worked on integrating all the program files together

**LOG-19:7/11/2023**

* Worked on various bug fixes and improvements

**LOG-20:16/11/2023**

* Improvements in the design of GUI interface and layout

**LOG-21:21/11/2023**

* Worked on optimizing functions

**LOG-22:25/11/2023**

* Worked on displaying data in Excel

**LOG-23:30/11/2023**

* Worked on final bug fixes and optimizations

**LOG-24:11/12/2023**

* Completion of Project and Submission

# **Sample Output**

A screenshot of a login screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Song Screen

User Home Screen

Search Result Display

Search Screen

Add Song

Admin Home Screen

# **Bibliography**

* <https://openpyxl.readthedocs.io>
* <https://kivy.org>
* <https://www.w3schools.com/python/>