A

Project Report

On

"VIRTUAL PHONE APPLICATION"

By

Audusiddha Gurdehalli

SAVITRIBAI PHULE PUNE UNIVERSITY

MASTER IN COMPUTER APPLICATION

MAHARASHTRA EDUCATION SOCIETY's

INSTITUTE OF MANAGEMENT AND CAREER COURSES

(IMCC), PUNE-411038

2022-23

Maharashtra Education Society's

Institute of Management & Career Courses (IMCC)

Approved by AICTE and Recognized by Savitribai Phule Pune University, Pune

NBA Accredited MCA Program (Valid up to 30.06.2022)

IMCC Campus, 131, Mayur Colony, Kothrud, Pune 411038, Maharashtra, India | Ph.: 020-2546 3453 / 6271 / 73 | e-mail: info.imcc@mespune.in

CERTIFICATE

This is to certify that the Project Report entitled

“VIRTUAL PHONE APPLICATION””

is prepared by

Audusiddha Gurdehalli

M.C.A. Semester III Course for the Academic Year 2022-23 at M.E. Society's Institute of Management & Career Courses (IMCC), Pune - 411038.

M.C.A Course is affiliated to Savitribai Phule Pune University.

To the best of our knowledge, this is original study done by the said student and important sources used by him/her have been duly acknowledged in this report.

The report is submitted in partial fulfillment of M.C.A Course for the Academic Year 2022-23 as per the rules and prescribed guidelines of Savitribai Phule Pune University.

[illegible]

Dr. Ravikant Zirmite
Head, Dept of MCA
MES IMCC

[illegible]

Dr. Santosh Deshpande
Director,
MES IMCC

Chapter 1

1. Introduction

This project ‘**VIRTUAL PHONE**’ helps us to perform few of the basic functionalities of a mobile phone. It can mainly be used as a substitute of a mobile phone for specific modules.

Over the internet, if we analyze, we will not get an application like this which performs the basic tasks of a phone in desktop environment. Keeping this in mind, we are trying to implement an application which can do few of the basic tasks very easily.

It is a simple desktop application without any ads and uses very trial API’s for the successful working of the modules. The designing of the UI’s have been done with the Qt Designer and the coding part with Python3 and the PyQt5 framework for Python.

1.1 Scope of the Project

In our application we can do the following tasks:

- ❖ Send free voice messages
- ❖ Store and Backup contacts
- ❖ Browse the internet using the Browser
- ❖ Write and save notes
- ❖ Make basic calculations using the Calculator
- ❖ Have access to the Calendar
- ❖ Get to know the present weather conditions using the Weather app.

1.2 Module wise Functionalities

In Virtual phone app, there is a main module called **Homescreen**. In the homescreen module, there are different applications.

1. Phonebook

2. Browser

3. Calendar

4. Calculator

5. Notepad

6. Weather Widget

1. **Phonebook**: In the Phonebook Module, there are the following

   Functionalities in the form of Tabs.

   - Add contacts
   - Backup Contacts
   - Voice call or audio message.

   i. **Add contacts**: We can add anyone's name, phone number, contact icon, address, email id and date of birth. Add contacts adds a contact and its details to the Contacts Tab reading from a CSV file.

   ii. **Voice call or Audio message**: Audio message functionality is limited/restricted due to the trial account API of twilio. We can select the audio from predefined list of audios and audio message can be sent only to verified contacts list. It is a type of simplex communication.

2.**Browser**:It is a single tabbed browser with navigation buttons and home button. Icons for valid SSL certificates are also present. There is also reload and cancel buttons.

3.Calender: It displays the present day, current date and time as well the calendar for the month of the current year.

4.Calculator: It is a simple calculator where addition, subtraction, multiplication and division can be done. Values can be whole numbers, integers and decimal numbers. There are also two buttons for backspace and all clear functionality of the display screen.

5.Notepad: It is a simple text editor with some special features. We can **save** the written text to a **.txt** file and can even **open** a .txt file into the text area. We have also the function to write a new file. There is also a quit option to close the notepad. Another special feature to change the font type, size and style is also present in the Edit section.

6.Weatherwidget: It displays the weather data of the current location based on the location ID of openweathermap.org . Weather data includes temperature, sky condition, humidity and wind speed. All the data are fetched using the python openweathermap API. Weather icons also are showed on the side of the widget depending on the sky conditions.

1.3 Software requirements

Platform (OS) used: Windows 10 Pro 64-bit.

IDE used: VS code

Core Technologies Used : Python3 and PyQt5

Supporting Technologies Used: Git and GitHub

1.4 Hardware Requirements

Processor: 1 Gigahertz(GHz) or faster.

RAM: Minimum 2 GB

Hard Disk: Minimum 20 GB

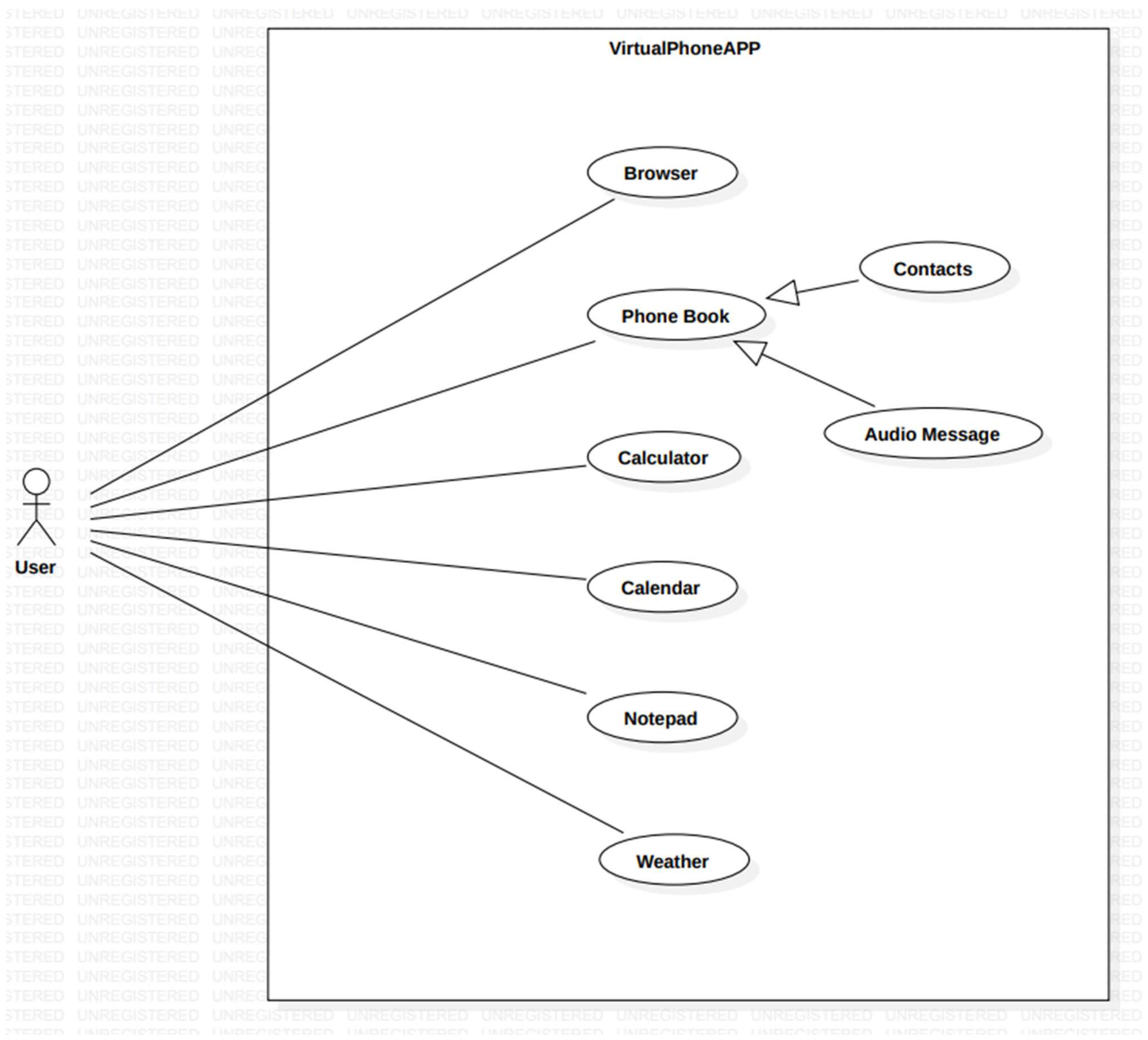
Graphics Card: DirectX 9 or later with WDDM 1.0 driver.

Display Resolution: 1920 x 1080

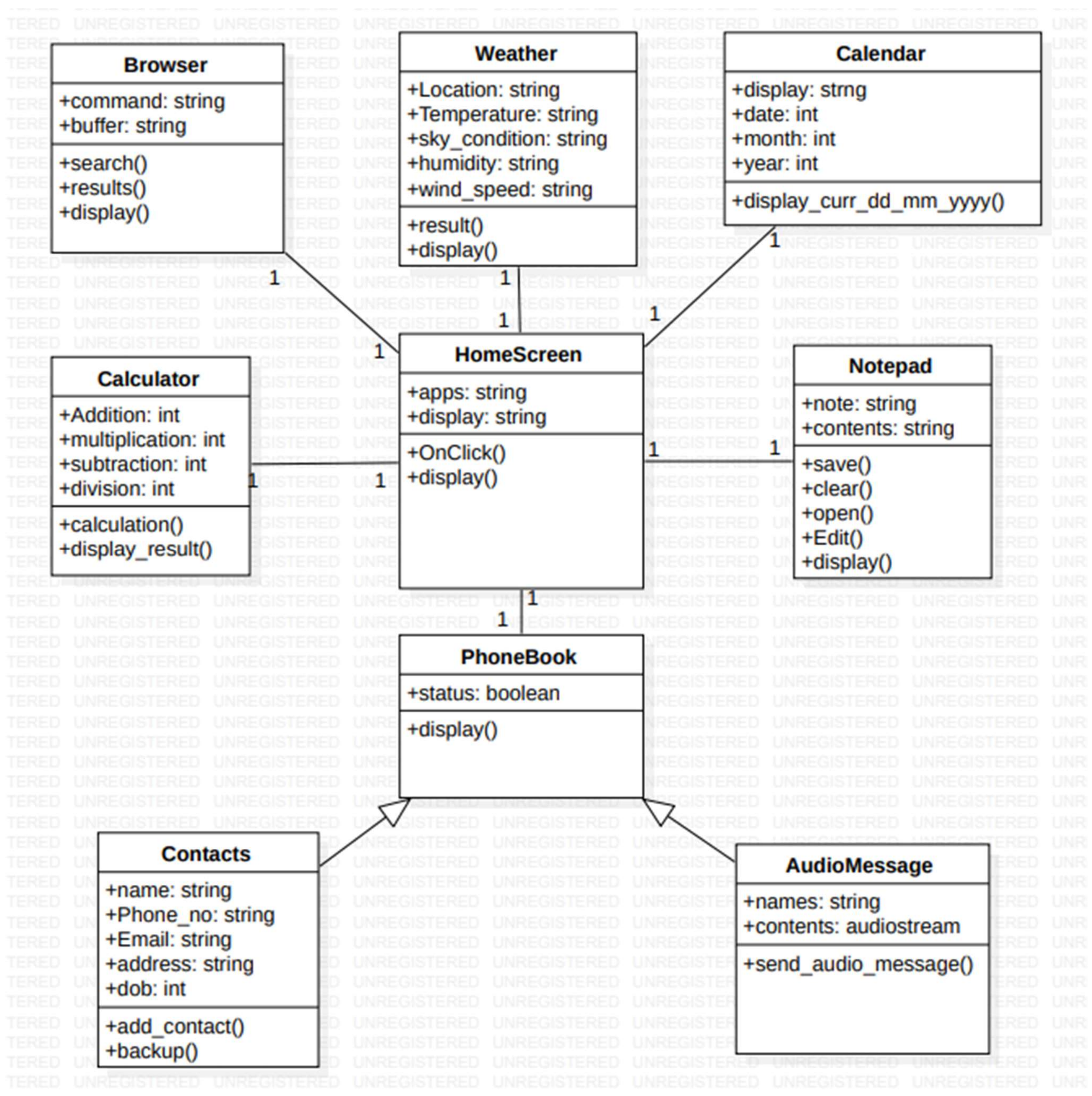
Chapter 2

Diagrams

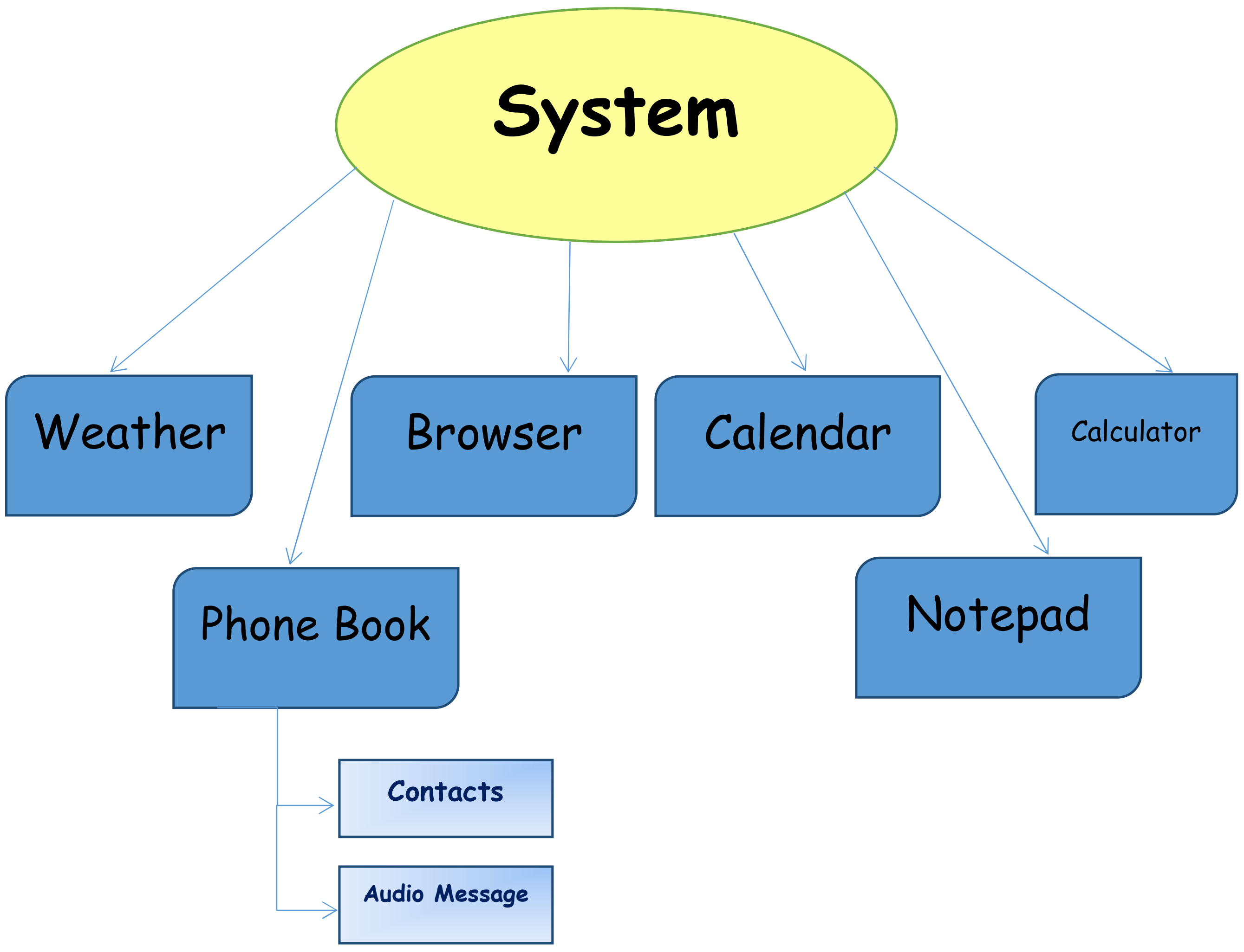
2.1 Business Use Case Diagram



2.2 Class Diagram



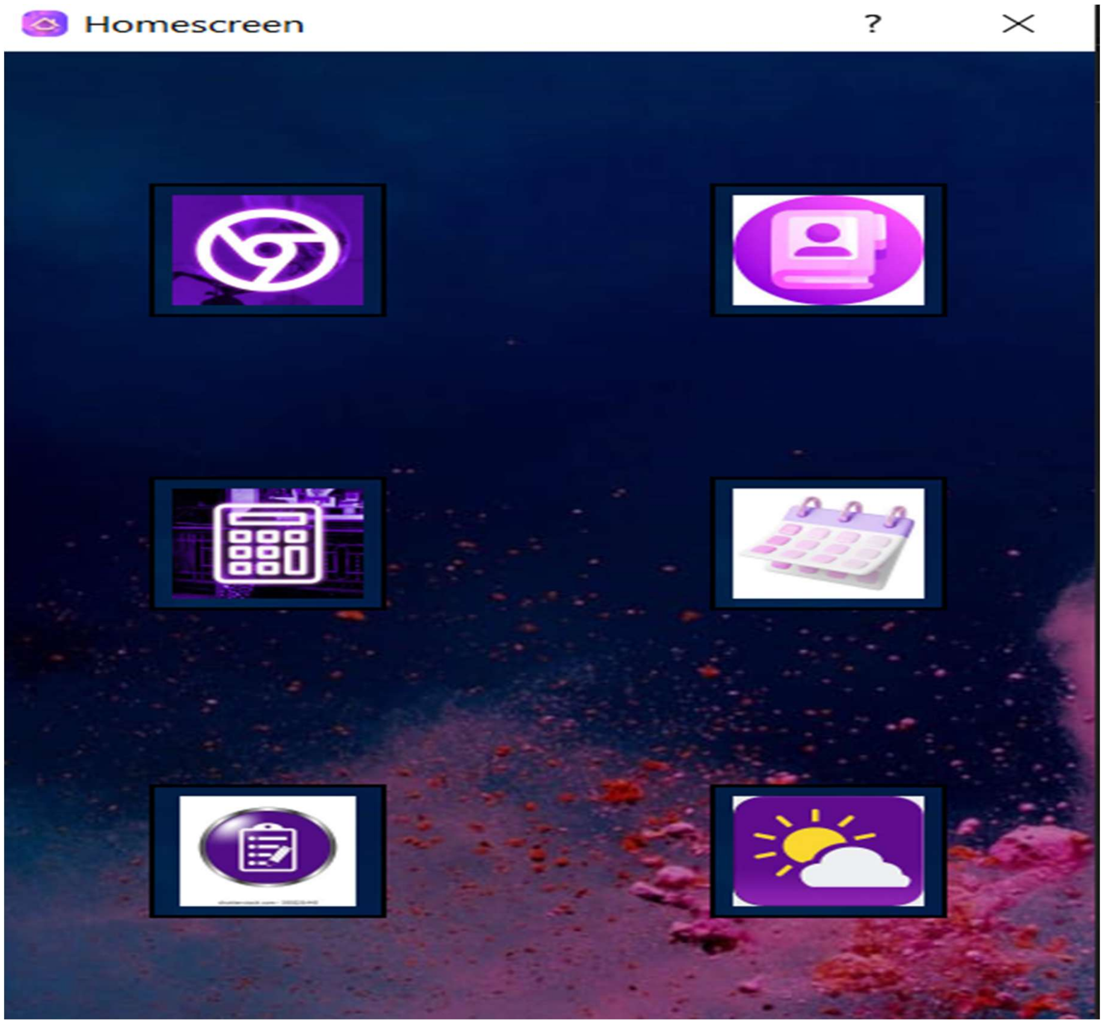
2.3 Block Diagrams



Chapter 3

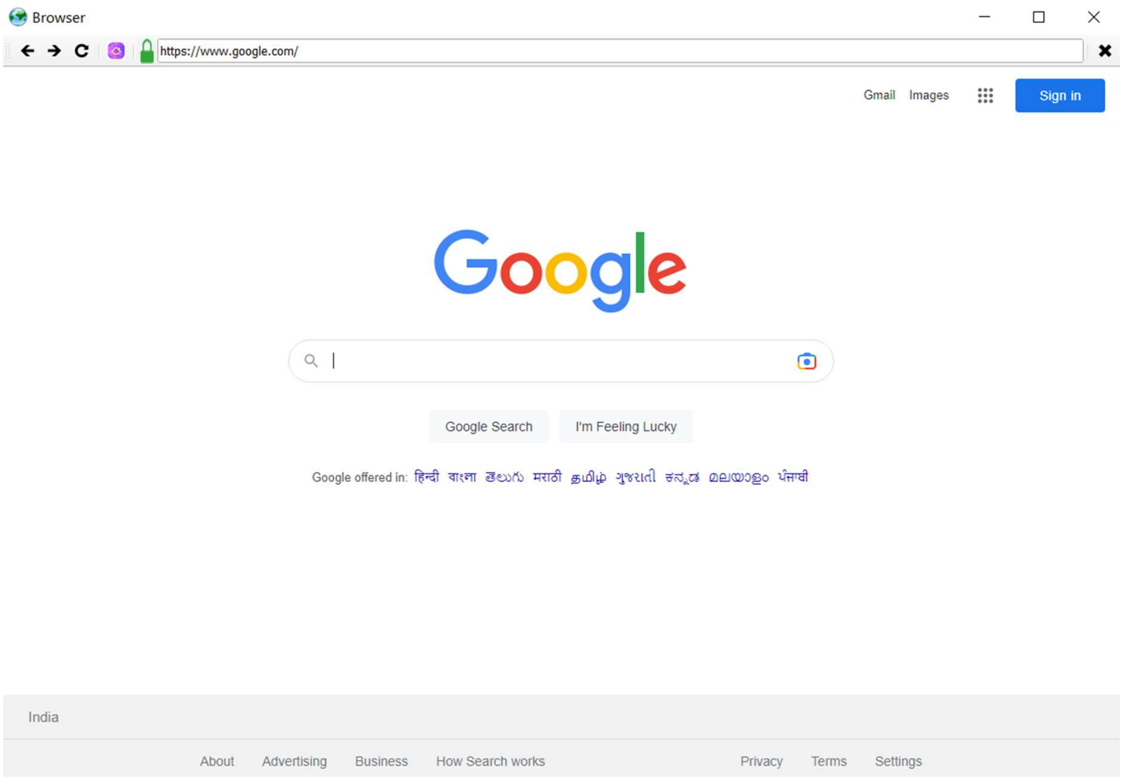
3 .User Interface

3.1 Input Design



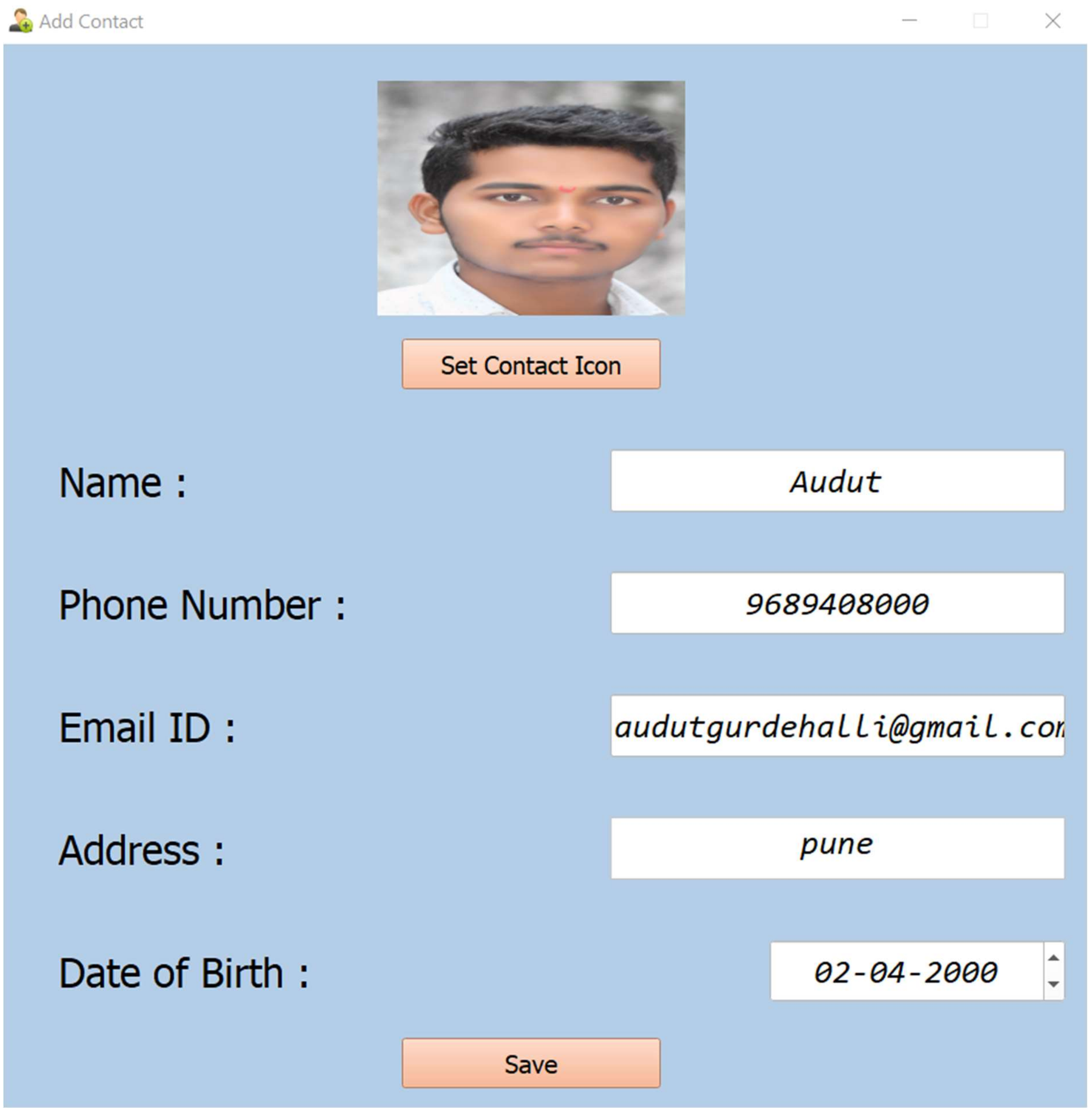
3.2 Output Design & Discussions

1. **One-Tabbed Browser**



One Tabbed browser having navigation buttons, reload, home and cancel button.It is also having an icon for “**https**” and “**http**” SSL certifications depending on the website. It is also having a URL bar where we can search a particular website manually.

2. **Add Contacts** (name, phone number, email id, address, DOB)



Name : Audut

Phone Number : 9689408000

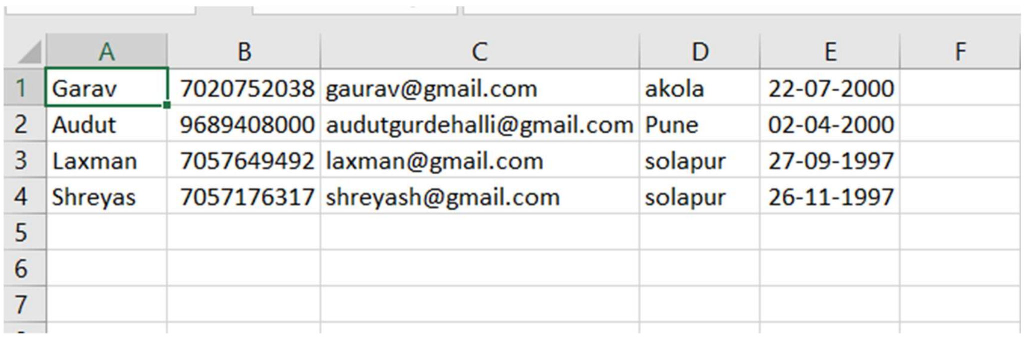
Email ID : audutgurdehalli@gmail.co

Address : pune

Date of Birth : 02-04-2000

The “add contact” functionality allows us to add a new contact to the contacts table. Contact details include name, phone number, email id, address and date of birth. We can even set a contact icon to the contact.

3. **Backup Contacts** (write contacts data in .csv file)

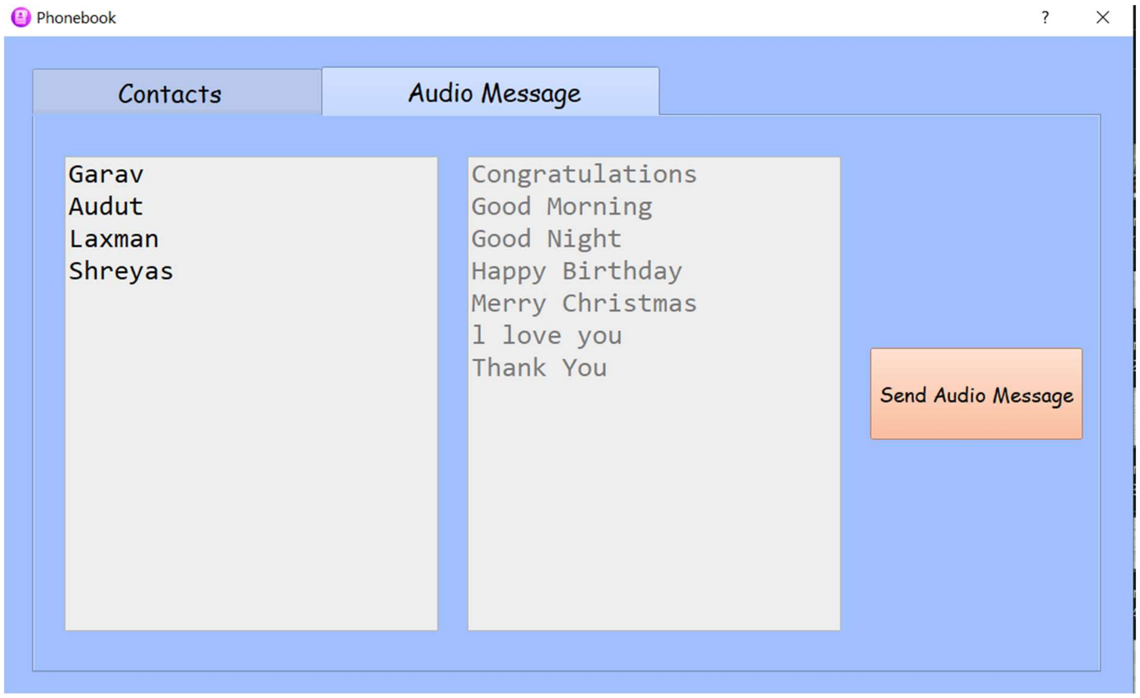


	A	B	C	D	E	F
1	Garav	7020752038	gaurav@gmail.com	akola	22-07-2000	
2	Audut	9689408000	audutgurdehalli@gmail.com	Pune	02-04-2000	
3	Laxman	7057649492	laxman@gmail.com	solapur	27-09-1997	
4	Shreyas	7057176317	shreyash@gmail.com	solapur	26-11-1997	
5						
6						
7						

We can take backup of the contacts in csv format.

4. **Audio Message** from Predefined list of audios and recipients

   (Trial Account restrictions)

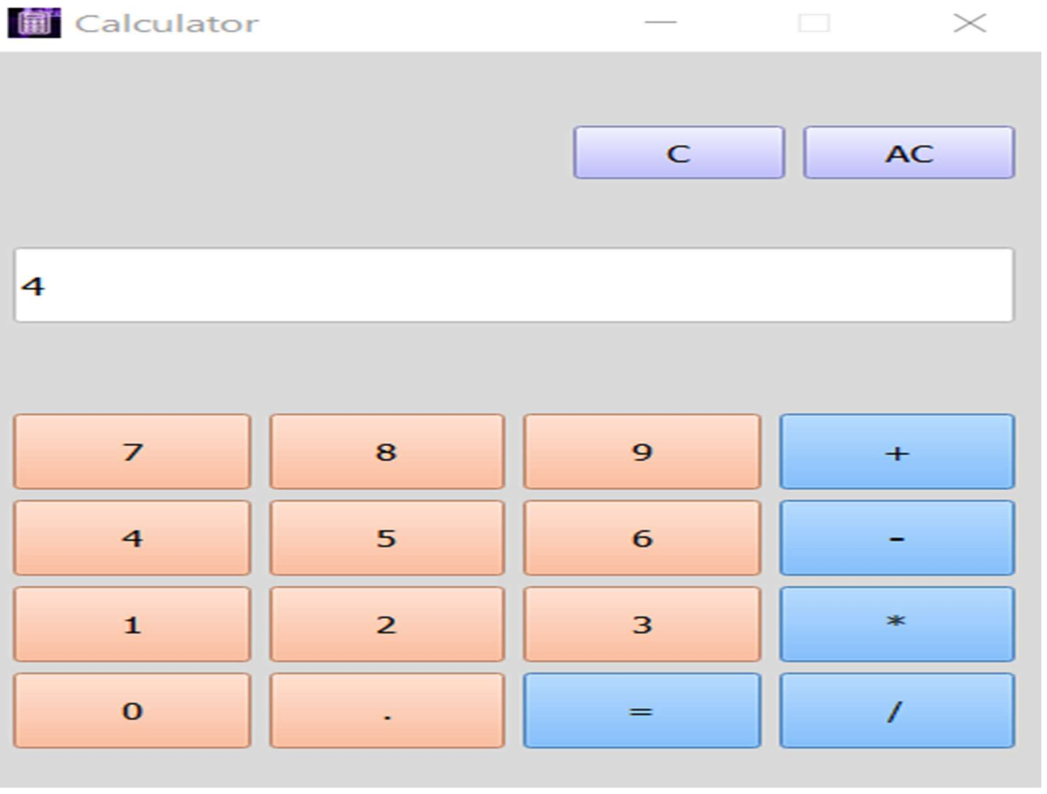


Garav
Audut
Laxman
Shreyas

Congratulations
Good Morning
Good Night
Happy Birthday
Merry Christmas
l love you
Thank You

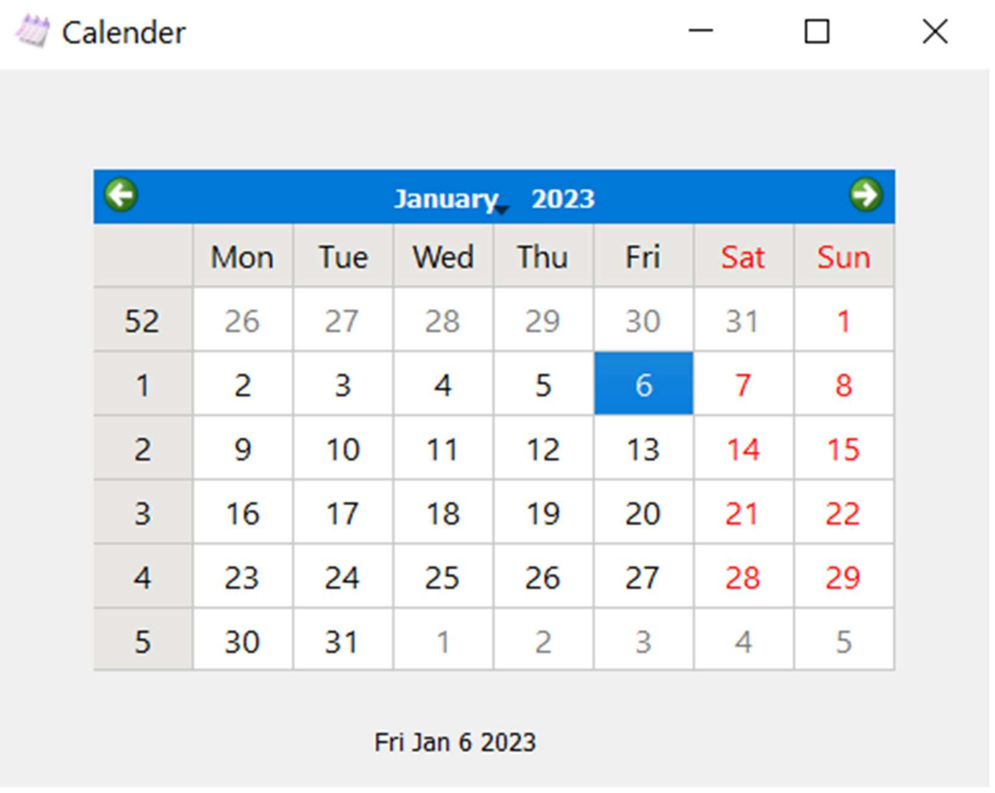
In the audio message section, we have to select the contact and the audio message from the contact and audio lists respectively. On clicking the "send audio message" button, audio message in the form of a call will be sent to the recipient. Trial account of the API allows limited recipients.

1. **Simple Calculator**



The simple calculator allows us to make basic arithmetic calculations like addition, subtraction, multiplication and division operations of whole number and decimal numbers. There are also two buttons for full clear and backspace clear of the display screen. The display screen is set to read only so that nobody can alter the values displayed there.

1.Simple Calendar



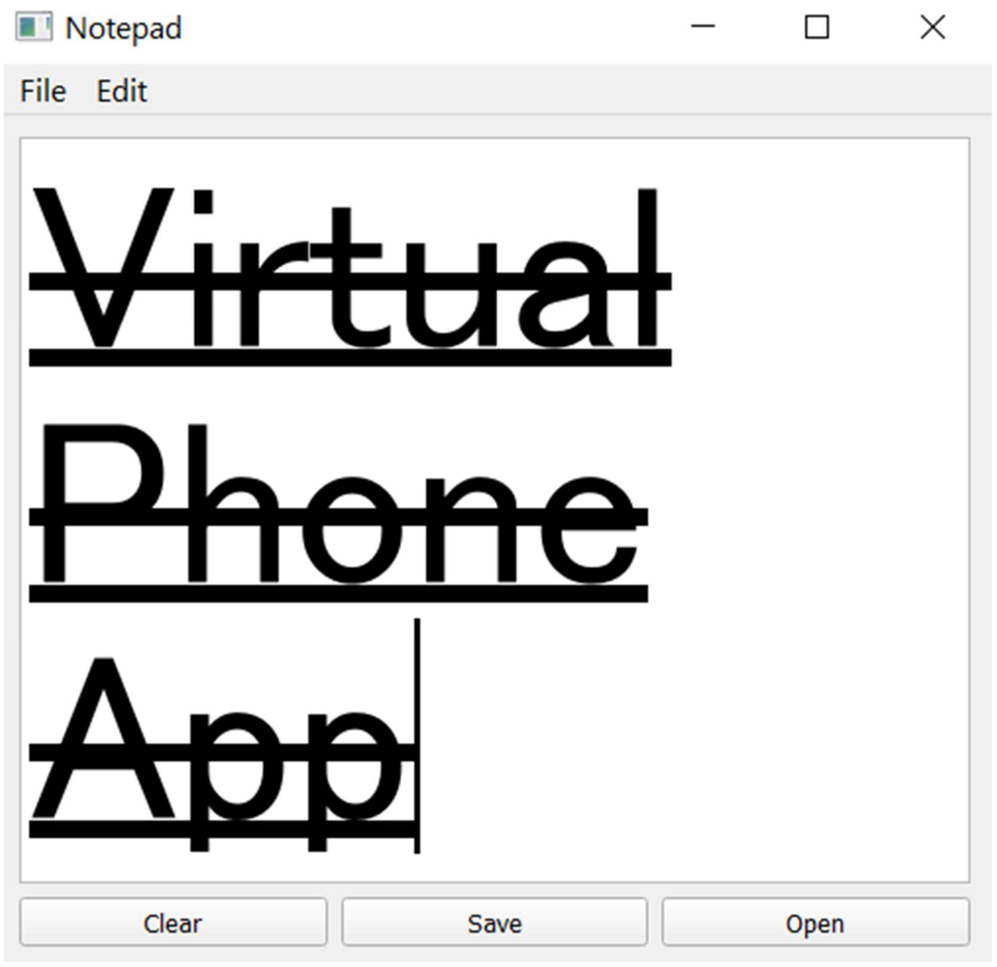
January 2023

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
52	26	27	28	29	30	31	1
1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22
4	23	24	25	26	27	28	29
5	30	31	1	2	3	4	5

Fri Jan 6 2023

The simple calendar displays the calendar for the current month of the year and also shows the present day and date. The present day is highlighted in the calendar.

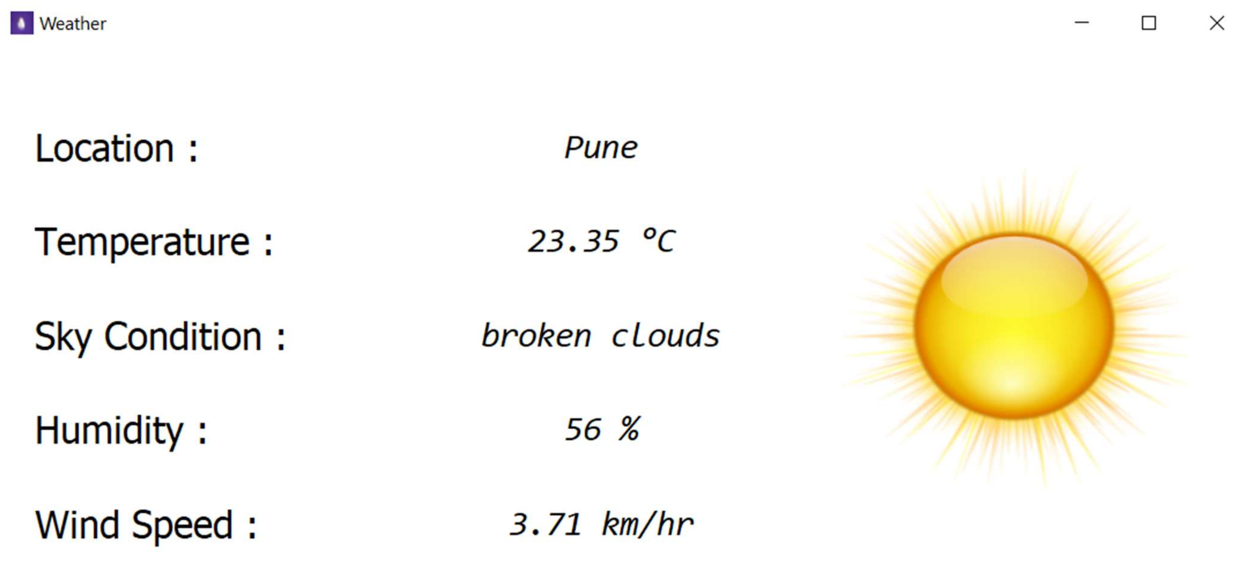
1.Notepad / Text Editor



~~Virtual Phone App~~

The notepad/text editor allows writing text files in “.txt” format and reading “.txt” files. There is also a clear button for clearing the text screen. It also has a menu bar with functionalities like **creating new file**, **quit** operation and the most important **changing the font style and size** functionality.

1.**Weather Widget** showing weather data



Location : Pune

Temperature : 23.35 °C

Sky Condition : broken clouds

Humidity : 56 %

Wind Speed : 3.71 km/hr

The weather widget shows us the current weather data of the present location based on a API call. The weather data set includes present location, current temperature (in °C), sky condition, humidity (in %) and wind speed (in km/hr.). There's also an icon beside depending on the sky condition. It uses the python openweathermap API.

Chapter 4

4.Code

4.1 HomeScree

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import QSize
import os

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(431, 668)
        Dialog.setStyleSheet("background-image:
url(homescreen_background3.jpg);")


        self.pushButton = QtWidgets.QPushButton(Dialog)
        self.pushButton.setGeometry(QtCore.QRect(60, 90, 93, 91))
        self.pushButton.setIcon(QtGui.QIcon('browser.png'))
        self.pushButton.setIconSize(QSize(75,75))
        self.pushButton.setStyleSheet('background-color: indica')
        self.pushButton.setText("")
        self.pushButton.setObjectName("pushButton")
        self.pushButton.clicked.connect(self.run_browser)


        self.pushButton_2 = QtWidgets.QPushButton(Dialog)
        self.pushButton_2.setGeometry(QtCore.QRect(280, 90, 93, 91))
        self.pushButton_2.setIcon(QtGui.QIcon('phonebook.png'))
        self.pushButton_2.setIconSize(QSize(75, 75))
        self.pushButton_2.setStyleSheet('background-color: indica')
        self.pushButton_2.setText("")
        self.pushButton_2.setObjectName("pushButton_2")
        self.pushButton_2.clicked.connect(self.run_phonebook)


        self.pushButton_3 = QtWidgets.QPushButton(Dialog)
        self.pushButton_3.setGeometry(QtCore.QRect(280, 290, 93, 91))
        self.pushButton_3.setIcon(QtGui.QIcon('calender.png'))
        self.pushButton_3.setIconSize(QSize(75, 75))
        self.pushButton_3.setStyleSheet('background-color: indica')
        self.pushButton_3.setText("")
        self.pushButton_3.setObjectName("pushButton_3")
        self.pushButton_3.clicked.connect(self.run_calender)
```

```python
        self.pushButton_4 = QtWidgets.QPushButton(Dialog)
        self.pushButton_4.setGeometry(QtCore.QRect(60, 290, 93, 91))
        self.pushButton_4.setIcon(QtGui.QIcon('calculator.png'))
        self.pushButton_4.setIconSize(QSize(75, 75))
        self.pushButton_4.setStyleSheet('background-color: indica')
        self.pushButton_4.setText("")
        self.pushButton_4.setObjectName("pushButton_4")
        self.pushButton_4.clicked.connect(self.run_calculator)


        self.pushButton_5 = QtWidgets.QPushButton(Dialog)
        self.pushButton_5.setGeometry(QtCore.QRect(280, 500, 93, 91))
        self.pushButton_5.setIcon(QtGui.QIcon('cloud.png'))
        self.pushButton_5.setIconSize(QSize(75, 75))
        self.pushButton_5.setStyleSheet('background-color: indica')
        self.pushButton_5.setText("")
        self.pushButton_5.setObjectName("pushButton_5")
        self.pushButton_5.clicked.connect(self.run_weather)


        self.pushButton_6 = QtWidgets.QPushButton(Dialog)
        self.pushButton_6.setGeometry(QtCore.QRect(60, 500, 93, 91))
        self.pushButton_6.setIcon(QtGui.QIcon('notepad.png'))
        self.pushButton_6.setIconSize(QSize(75, 75))
        self.pushButton_6.setStyleSheet('background-color: indica')
        self.pushButton_6.setText("")
        self.pushButton_6.setObjectName("pushButton_6")
        self.pushButton_6.clicked.connect(self.run_notepad)


        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Homescreen"))
        Dialog.setWindowIcon(QtGui.QIcon('home.png'))


    def run_browser(self):
        print("browser is run")
        Dialog.hide()
        os.system('python browser.py')
        Dialog.show()


    def run_phonebook(self):
        print("Phonebook is run")
        Dialog.hide()
        os.system('python phonebook_app.py')
        Dialog.show()


    def run_calender(self):
        print("Calender is Run")
        Dialog.hide()
        os.system('python calender.py')
        Dialog.show()


    def run_calculator(self):
        print("Calculator is run")
        Dialog.hide()
        os.system('python calculator_app.py')
        Dialog.show()

    def run_notepad(self):
        print("Notepad is run")
        Dialog.hide()
        os.system('python Notepad.py')
        Dialog.show()

    def run_weather(self):
        print("Weather is run")
        Dialog.hide()
        os.system('python weather_app.py')
        Dialog.show()


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    app.setStyle(QtWidgets.QStyleFactory.create('Fusion'))
    Dialog = QtWidgets.QDialog()
    ui = Ui_Dialog()
    ui.setupUi(Dialog)
    Dialog.show()
    sys.exit(app.exec_())
```

4.2 Browser

```python
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
from PyQt5.QtWebEngineWidgets import *
from PyQt5.QtPrintSupport import *
import sys
import os

class MainWindow(QMainWindow):

    def __init__(self, *args, **kwargs):
        super(MainWindow, self).__init__(*args, **kwargs)

        self.show()
        self.setWindowTitle("Browser")
        self.setWindowIcon(QIcon('browser_icon.png'))
        self.setGeometry(50,50,1200,800)
        self.browser = QWebEngineView()
        self.browser.setUrl(QUrl("http://www.google.com"))

        self.setCentralWidget(self.browser)

        navtab = QToolBar("Navigation")
        navtab.setIconSize(QSize(20,20))
        self.addToolBar(navtab)

        back_btn = QAction(QIcon("nav_back.png"), "Back", self)
        back_btn.setStatusTip("Back to the Previous Page")
        back_btn.setShortcut("Ctrl+z")
        back_btn.triggered.connect(self.browser.back)
        navtab.addAction(back_btn)

        next_btn = QAction(QIcon('nav_forward.png'), "Forward", self)
        next_btn.setStatusTip("Forward to the next page")
        next_btn.setShortcut("Ctrl+x")
        next_btn.triggered.connect(self.browser.forward)
        navtab.addAction(next_btn)

        reload_btn = QAction(QIcon('reload.png'), "Reload", self)
        reload_btn.setStatusTip("Reload")
        reload_btn.setShortcut("Ctrl+r")
        reload_btn.triggered.connect(self.browser.reload)
        navtab.addAction(reload_btn)

        navtab.addSeparator()

        home_btn = QAction(QIcon('home.png'), "Home", self)
        home_btn.setStatusTip("Home")
        home_btn.triggered.connect(self.navigate_home)
        navtab.addAction(home_btn)

        navtab.addSeparator()

        self.httpsicon = QLabel()
        self.httpsicon.resize(5,5)
        self.httpsicon.setPixmap(QPixmap('https.png'))
        self.httpsicon.setScaledContents(True)

        navtab.addWidget(self.httpsicon)

        self.urlbar = QLineEdit()
        self.urlbar.returnPressed.connect(self.navigate_to_url)
        navtab.addWidget(self.urlbar)
        self.browser.urlChanged.connect(self.update_urlbar)

        navtab.addSeparator()

        stop_btn = QAction(QIcon('cancel.png'), "Stop", self)
        stop_btn.setStatusTip("Stop loading the current page")
        stop_btn.triggered.connect(self.browser.stop)
        navtab.addAction(stop_btn)

    def navigate_to_url(self):
        q = QUrl(self.urlbar.text())
        #print(q.scheme())
        if q.scheme()=="":
            q.setScheme("http")

        self.browser.setUrl(q)


    def navigate_home(self):
        self.browser.setUrl(QUrl("http://www.google.com"))

    def update_urlbar(self, q):

        if(q.scheme()=='https'):
            #Secure lock icon
            self.httpsicon.setPixmap(QPixmap('https.png'))

        else:
            self.httpsicon.setPixmap(QPixmap('http.png'))
            #Unsecure lock icon

        self.urlbar.setText(q.toString())
        self.urlbar.setCursorPosition(0)

app = QApplication(sys.argv)
app.setApplicationName("Blazing Browser")
app.setStyle(QStyleFactory.create('Fusion'))
window = MainWindow()
window.show()
```

```
        app.exec_()

```

4.3 Phonebook

```
import sys
import os
import csv
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog
import addcontact_non_exec
class addcontact_class(addcontact_non_exec.Ui_Form, QtWidgets.QWidget):

    def __init__(self):
        super(addcontact_class, self).__init__()
        self.setupUi(self)


        regex_name = QtCore.QRegExp("[a-z-A-Z_]+")
        name_validator = QtGui.QRegExpValidator(regex_name)
        self.name_lineEdit.setValidator(name_validator)

        self.phone_lineEdit.setMaxLength(10)
        regex_phone = QtCore.QRegExp("[0-9_]+")
        phone_validator = QtGui.QRegExpValidator(regex_phone)
        self.phone_lineEdit.setValidator(phone_validator)
        regex_email = QtCore.QRegExp('^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-
]+(\.[a-z0-9-]+)*(\.[a-z]{2,4})$')
        email_validator = QtGui.QRegExpValidator(regex_email)
        self.email_lineEdit.setValidator(email_validator)


        self.dob_dateEdit.setDisplayFormat('dd-MM-yyyy')

        self.dob_dateEdit.setDate(QtCore.QDate.currentDate())
        temp_dob = self.dob_dateEdit.date()
        # DOB = temp_dob.toPyDate()
        DOB =temp_dob.toString('dd/MM/yyyy')
        print(DOB)

        self.btn_set_icon.clicked.connect(self.openFileNameDialog)
        self.btn_save.clicked.connect(self.save_action)
    def save_action(self):
        print("Save button clicked")
        with open('contacts.csv', 'a', newline='') as f:
            writer = csv.writer(f)
            writer.writerow([self.name_lineEdit.text(),
self.phone_lineEdit.text(), self.email_lineEdit.text(),
self.address_textEdit.toPlainText(), self.dob_dateEdit.date().toString('dd-MM-
yyyy')])
```

```
        QtCore.QCoreApplication.processEvents()

        addcontact_obj.hide()
        os._exit(0)
    def openFileNameDialog(self):
        print("set icon clicked")
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        fileName, _ = QFileDialog.getOpenFileName(self, "Choose Contact Icon",
"", "Image Files (*.jpg *.png)", options=options)
        if fileName:
            print(fileName)
            pixmap = QtGui.QPixmap(fileName)
            self.contact_icon.setPixmap(pixmap)
            self.contact_icon.setScaledContents(True)

if __name__ == '__main__':
    qapp = QtWidgets.QApplication(sys.argv)
    qapp.setStyle(QtWidgets.QStyleFactory.create('Fusion'))
    addcontact_obj = addcontact_class()
    addcontact_obj.show()
    qapp.exec_()
```

4.3 Calculator

```
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
import calculator_ui
class Calculator_class(calculator_ui.Ui_MainWindow, QtWidgets.QMainWindow):

    def __init__(self):
        super(Calculator_class, self).__init__()
        self.setupUi(self)
        #self.display_screen('25662')

        self.btn_0.clicked.connect(lambda: self.display_screen('0'))
        self.btn_1.clicked.connect(lambda: self.display_screen('1'))
        self.btn_2.clicked.connect(lambda: self.display_screen('2'))
        self.btn_3.clicked.connect(lambda: self.display_screen('3'))
        self.btn_4.clicked.connect(lambda: self.display_screen('4'))
        self.btn_5.clicked.connect(lambda: self.display_screen('5'))
        self.btn_6.clicked.connect(lambda: self.display_screen('6'))
        self.btn_7.clicked.connect(lambda: self.display_screen('7'))
        self.btn_8.clicked.connect(lambda: self.display_screen('8'))
        self.btn_9.clicked.connect(lambda: self.display_screen('9'))

        self.btn_add.clicked.connect(lambda: self.display_screen(' + '))
        self.btn_sub.clicked.connect(lambda: self.display_screen(' - '))
        self.btn_multiply.clicked.connect(lambda: self.display_screen(' * '))
        self.btn_divide.clicked.connect(lambda: self.display_screen(' / '))
        self.btn_decimal.clicked.connect(lambda: self.display_screen('.'))

        self.btn_0.setShortcut("0")
        self.btn_1.setShortcut("1")
        self.btn_2.setShortcut("2")
        self.btn_3.setShortcut("3")
        self.btn_4.setShortcut("4")
        self.btn_5.setShortcut("5")
        self.btn_6.setShortcut("6")
        self.btn_7.setShortcut("7")
        self.btn_8.setShortcut("8")
        self.btn_9.setShortcut("9")
        self.btn_add.setShortcut("+")
        self.btn_sub.setShortcut("-")
        self.btn_multiply.setShortcut("*")
        self.btn_divide.setShortcut("/")
        self.btn_decimal.setShortcut(".")
        self.btn_clear.setShortcut("Backspace")
        self.btn_equals.setShortcut("Enter")
        self.btn_allclear.setShortcut("Delete")

        self.btn_allclear.clicked.connect(self.screen.clear)
        self.btn_clear.clicked.connect(self.screen.backspace)
        self.btn_equals.clicked.connect(self.calculation)
        self.screen.setReadOnly(True)

    def display_screen(self, value):
        """
        this function will insert data to screen
        :param value:
        :return:
        """
        self.screen.insert(value)


    def calculation(self):

        """
        This is a calculation function that will take values from screen
        and pass the values to maths function
        :return:
        """

        screen_value = str(self.screen.text()).split(' ')
        screen_text = str(self.screen.text())
        #x = screen_value.split(' ')
        x = (eval(str(screen_text)))
        self.screen.setText(str(x))
if __name__ == '__main__':
    qapp = QtWidgets.QApplication(sys.argv)
    qapp.setStyle(QtWidgets.QStyleFactory.create('Fusion'))
    calc = Calculator_class()
    calc.show()
    qapp.exec_()
```

4.4 Calendar

```
import sys
import datetime
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.setGeometry(50,50,500,360)
        self.calendarWidget = QtWidgets.QCalendarWidget(Form)
        self.calendarWidget.setGridVisible(True)


        self.calendarWidget.setGeometry(QtCore.QRect(50, 50, 400, 250))
        self.calendarWidget.setObjectName("calendarWidget")


        self.label = QtWidgets.QLabel(Form)
        date = self.calendarWidget.selectedDate()
        self.label.setText(date.toString())
        self.label.setGeometry(QtCore.QRect(190, 200, 350, 270))
        self.label.setObjectName("label")

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)



    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Calender"))
        Form.setWindowIcon(QtGui.QIcon('calender.png'))

        self.calendarWidget.setWhatsThis(_translate("Form", "<!DOCTYPE HTML
PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style
type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:8pt;
font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-
right:0px; -qt-block-indent:0; text-
indent:0px;\">currentdate=bold</p></body></html>"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    app.setStyle(QtWidgets.QStyleFactory.create('Fusion'))
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

4.5 Notepad

```
import os
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QFileDialog, QWidget,
QFontDialog, QStyleFactory
from PyQt5.QtWidgets import QTextEdit, QPushButton, QVBoxLayout, QHBoxLayout,
QAction, qApp
from PyQt5.QtGui import QFont, QIcon


class Notepad(QWidget):

    def __init__(self):
        super().__init__()
        self.text = QTextEdit(self)
        self.clr_btn = QPushButton('Clear')
        self.sav_btn = QPushButton('Save')
        self.opn_btn = QPushButton('Open')
        self.init_ui()

    def init_ui(self):
        v_layout = QVBoxLayout()
        h_layout = QHBoxLayout()

        h_layout.addWidget(self.clr_btn)
        h_layout.addWidget(self.sav_btn)
        h_layout.addWidget(self.opn_btn)
        v_layout.addWidget(self.text)
        v_layout.addLayout(h_layout)

        self.sav_btn.clicked.connect(self.save_text)
        self.clr_btn.clicked.connect(self.clear_text)
        self.opn_btn.clicked.connect(self.open_text)

        self.setLayout(v_layout)
        #self.setWindowTitle("notepad")
        #self.setWindowIcon(QIcon('notepad.png'))



        self.show()

    def save_text(self):
        filename = QFileDialog.getSaveFileName(self, 'Save File',
os.getenv('HOME'))
        with open(filename[0], 'w') as f:
            my_text = self.text.toPlainText()
            f.write(my_text)

    def open_text(self):
        filename = QFileDialog.getOpenFileName(self, 'Open File',
os.getenv('HOME'))
        with open(filename[0], 'r') as f:
            file_text = f.read()
            self.text.setText(file_text)

    def clear_text(self):
        self.text.clear()


class Writer(QMainWindow):
    def __init__(self):
        super().__init__()

        self.form_widget = Notepad()
        self.setCentralWidget(self.form_widget)

        self.init_ui()

    def init_ui(self):
        bar = self.menuBar()
        file = bar.addMenu('File')
        edit = bar.addMenu('Edit')

        new_action = QAction('New', self)
        new_action.setShortcut('Ctrl+N')

        save_action = QAction('&Save', self)
        save_action.setShortcut('Ctrl+S')

        open_action = QAction('&Open', self)

        quit_action = QAction('&Quit', self)

        font_action = QAction('&Font', self)

        file.addAction(new_action)
        file.addAction(save_action)
        file.addAction(open_action)
        file.addAction(quit_action)
        edit.addAction(font_action)

        quit_action.triggered.connect(self.quit_trigger)
        file.triggered.connect(self.respond)
        font_action.triggered.connect(self.font_changer1)
        self.setWindowTitle("Notepad")
        self.setWindowIcon(QIcon('pnotepad.png'))
        self.show()

    def quit_trigger(self):
        qApp.quit()

    def respond(self, q):
        signal = q.text()

        if signal == 'New':
            self.form_widget.clear_text()
        elif signal == '&Open':
            self.form_widget.open_text()
        elif signal == '&Save':
            self.form_widget.save_text()


    def font_changer1(self):
        font, ok = QFontDialog.getFont()
        if ok:
            self.form_widget.text.setFont(font)




app = QApplication(sys.argv)
app.setStyle(QStyleFactory.create('Fusion'))
writer = Writer()
```

```
sys.exit(app.exec_())
```

4.6 Weather

```
import weather_non_exec
from PyQt5 import QtWidgets, QtGui, QtCore
import sys
import pyowm
import json
import requests




class Weather_class(weather_non_exec.Ui_weather_screen, QtWidgets.QWidget):

    def __init__(self):
        super(Weather_class, self).__init__()
        self.setupUi(self)
        self.display_weather()

    def display_weather(self):
api_address='http://api.openweathermap.org/data/2.5/weather?appid=cd763d1cc74b
d82bdbe1381ad4af4f6b&q=pune'

        url = api_address

        json_data = requests.get(url).json()
        location = json_data['name']
        print ('Location:',location)

        temperature = (json_data['main'] ['temp'])-273
        temperature = round(temperature,2)
        print ('Temperature:',temperature,'°C')

        sky_condition = json_data['weather'] [0] ['description']
        print ('Sky Condition:',sky_condition)

        humidity = json_data['main'] ['humidity']
        print ('Humidity:',humidity)

        wind_speed = json_data['wind'] ['speed']
        print ('Wind Speed:',wind_speed)




        if str(sky_condition) == 'haze':
            pixmap = QtGui.QPixmap('haze.png')
            self.weather_image.setPixmap(pixmap)
            self.weather_image.setScaledContents(True)
        elif str(sky_condition) == 'cloudy':
            pixmap = QtGui.QPixmap('cloudy.png')
            self.weather_image.setPixmap(pixmap)
            self.weather_image.setScaledContents(True)
        elif str(sky_condition) == 'light rain':
            pixmap = QtGui.QPixmap('rainy.png')
            self.weather_image.setPixmap(pixmap)
            self.weather_image.setScaledContents(True)
        else:
            pixmap = QtGui.QPixmap('sunny.png')
            self.weather_image.setPixmap(pixmap)
            self.weather_image.setScaledContents(True)
        self.value_location.setText(location)
        self.value_temperature.setText(str(temperature) + str(' °C'))
        self.value_humidity.setText(str(humidity) + str(' %'))
        self.value_skycondition.setText(sky_condition)
        self.value_windspeed.setText(str(wind_speed) + str(" km/hr"))
if __name__ == '__main__':
    qapp = QtWidgets.QApplication(sys.argv)
    qapp.setStyle(QtWidgets.QStyleFactory.create('Fusion'))
    weather_obj = Weather_class()
    weather_obj.show()
    qapp.exec_()
```

5.Bibliography

www.youtube.com
www.geeksforgeeks.com
www.github.com

6. Future Scope

1. Browser to be made multi-tabbed with additional features like downloading files, viewing page source and saving and opening html files.
2. Contact icons to be added in the contacts table section beside the contacts in each row.
3. Contacts to be taken backup in **.vcf** format instead of **.csv** format.
4. Recorder functionality to be added in audio message section for sending of own customized audio after recording them.
5. Adding subject to the email to be implemented.
6. Scientific Calculator to be implemented.
7. Birthday reminder alarm functionality to be added of the contacts using the calendar widget.
8. ***Find & Replace*** option, ***Spell Checker*** and ***help*** option to be added to the text editor module.
9. Converting the whole application into a **“.exe**” file and implementing the same project in **Android platform** in the form of a app or ROM (adding additional modules to it).

7. Conclusion

In this project, **Python3**, which is one the popular programming language of recent times was used as a core technology along with **PyQt5** which is also a very popular framework of python. The **Virtual Phone App** have been presented and all the modules have been discussed in details. It was a wonderful learning experience for us while working on this project. This project took us through the various phases of **software development** and gave us a real insight of the world of concept of SDLC. The effort of working together as a team and tackling the various problems and challenges in the due course gave us a feel of a development teams of the corporate world. Working on this project we came to know about various new concepts related to programming. However, this project is further extendable and has a vast list of future scope. Overall, we enjoyed each and every bit of work done for the successful completion of this project.