Project Report

On

Xattendance

(Modern Attendance Management System)

Submitted by

R170262, CHINTHA SIVAPRASAD

R170249, C PRASANTH

R170244, GUDDITI MANJUNATH

Under the guidance of

RAVI KUMAR PENUGONDA

M.Tech

Rajiv Gandhi University of Knowledge and Technologies(RGUKT),

R.K.Valley, Kadapa, Andhra Pradesh

Rajiv Gandhi University of Knowledge and Technologies(RGUKT),

Valley, Kadapa(Dist), Andhra Pradesh, 516330.

CERTIFICATE

This is to certify that the project work titled "**Xattendance (Modern Attendance Management System)**" is a bonafide project work submitted by **CHINTHA SIVAPRASAD, C PRASANTH, G MANJUNATH** in the department of COMPUTER SCIENCE AND ENGINEERING in partial fulfillment of requirements for the award of degree of Bachelor of Technology in Computer science and engineering for the year 2021-2022 carried out the work under the supervision

GUIDE

P RAVI KUMAR

HEAD OF THE DEPARTMENT

P HARINADHA

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success. We are incredibly grateful to our respected Director, Prof. K. SANDHYA RANI for fostering an excellent academic climate in our institution. We also express my sincere gratitude to our respected Head of the Department Mr. P HARINADHA for his encouragement, overall guidance in viewing this project as a good asset, and effort in bringing out this project. We would like to convey thanks to our guide at college Mr. P RAVI KUMAR for his guidance, encouragement, cooperation, and kindness during the entire duration of the course and academics. Our sincere thanks to all the members who helped us directly and indirectly in completing the project work. We express our profound gratitude to all our friends and family members for their encouragement.

INDEX

Abstract

Attendance Management is one of the most common processes that is done in many educational institutes, organizations, etc. Although there are traditional ways of managing attendance, it requires modern attention as these ways are cumbersome. With the development in technology and the massive increase in the establishment of organizations and educational institutes, it is important to manage the attendance of individuals efficiently and with integrity.

There are existing models that enhance attendance management using new technologies and computer systems. This project provides a novel approach to manage attendance efficiently with new features, especially for educational systems. Built-in python and UI frameworks like tkinter, pandas our system enhance and automate most of the attendance management tasks.

INTRODUCTION

What is Modern Attendance Management?

Traditionally attendance management is a cumbersome process that includes hard copies of the attendance and is managed by the individual manually. Modern Attendance Management systems tend to speed up the process usually by computerizing and making complex computations with modern computing power.

PURPOSE

The Purpose of Xattendance

The main objective of Xattendance is to simplify and automate attendance managing tasks. And also provides a novel way to store attendance, especially in soft copies reducing paper works.

SCOPE

The scope of our project extends to educational institutions and other organizations where attendance is mandatory. Educational institutions can download our software and use it for their attendance management system instead of using the traditional approach which requires a lot of human effort and paperwork. With our attendance management system, we can reduce the burden of storing physical copies of attendance. Our project is lightweight and can run on low memory and hardware

Configurations devices also all required is python.

Requirement Specification

Hardware Configuration

Ram	**512 MB**
Hard Disk	**10 GB**
Processor	**1.0 GHz**

Software Requirements

Front End	**python-Tkinter**
BackEnd	**python**
Operating System	**Any operating system with python installed**
Software	**Python, numpy, pandas, tkinter**

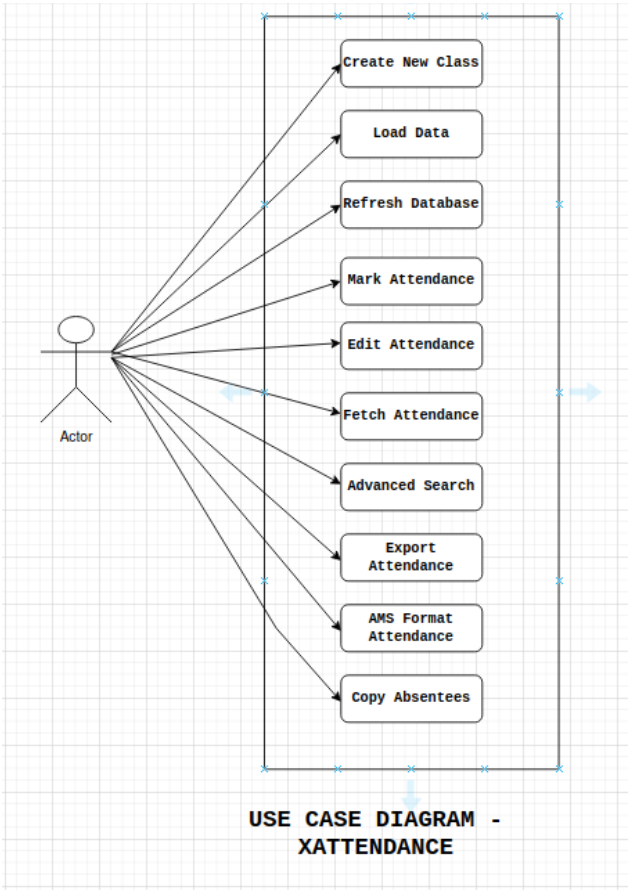
PYTHON:

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional.

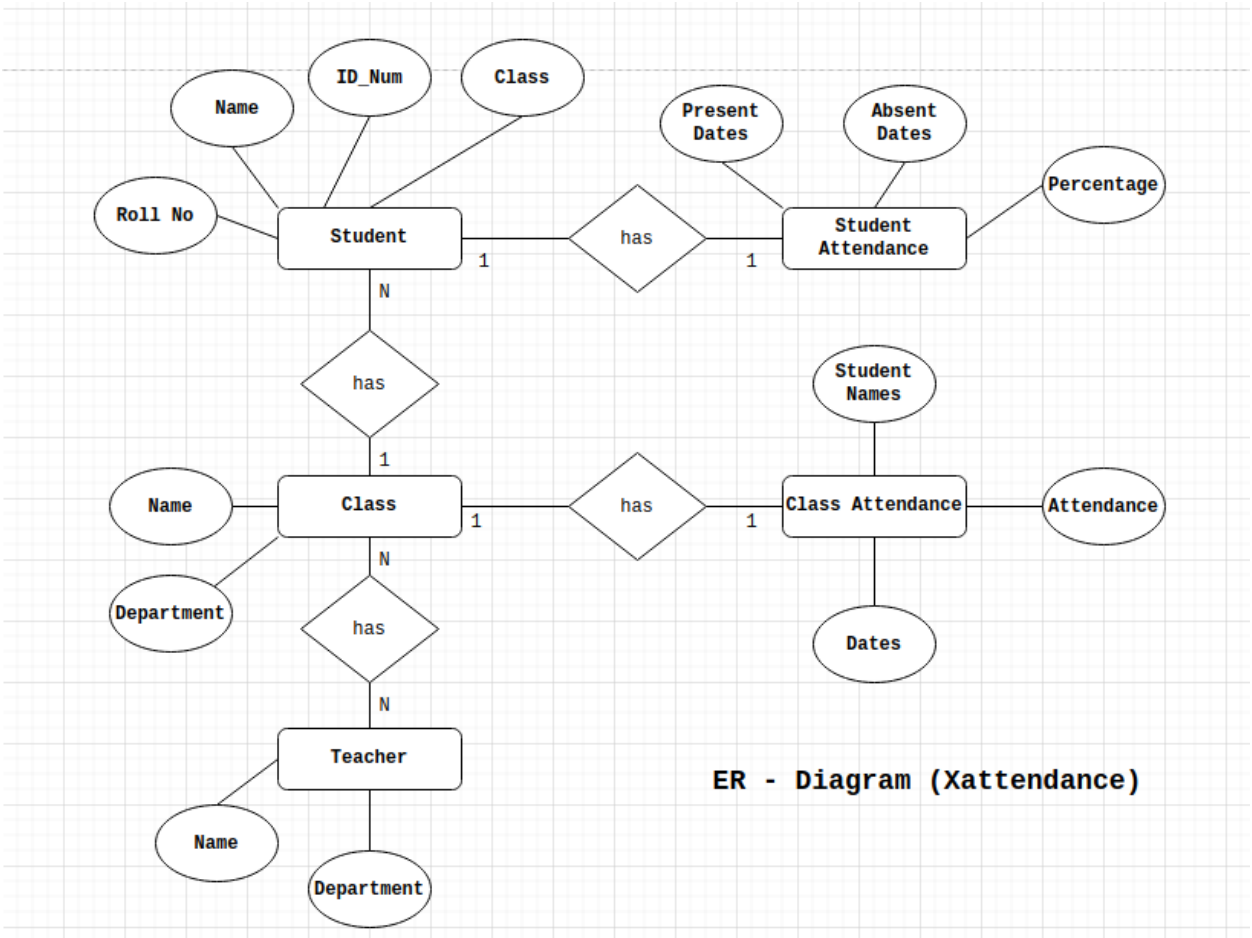
TKINTER:

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and macOS installs of Python. The name Tkinter comes from Tk interface

USECASE DIAGRAM:



ER-DIAGRAM



IMPLEMENTATION

Filename:main.py

```
# -- Importing necessary modules
#!/usr/bin/env python3
from tkinter import *

import os
import pandas as pd
import datetime as dt
import re
import numpy as np
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import filedialog
import pyperclip
class Xattendance:
	def __init__(self):
		self.flag = 1
		self.gui()
		self.root.mainloop()
	def gui(self):
		self.mainDir = '/home/student/10.30.26.122:8000/Xattendance'
		self.root = Tk()
		####		self.root.attributes('-fullscreen', 'true')
		self.root.geometry('1365x767')
		img = PhotoImage(file = f'{self.mainDir}/Data/icon.png')
		self.root.tk.call('wm', 'iconphoto', self.root._w, img)
		with open(f'{self.mainDir}/Data/data.txt') as file:
			x = file.read()
			name, subject = x.split('\n')[0], x.split('\n')[1]
		self.faculty = name
		self.subject = subject
		self.root.title('Xattendance')
		self.root.geometry('1200x1200')
		self.root['bg'] = 'black'
		self.h1 = Label(self.root,
				text = 'Xattendance',
				font = ('TlwgTypist', 25, 'bold italic'),
				bg = 'black',
				fg = 'cyan')
		self.h1.pack()
		self.side_frame = Frame(self.root)
		self.side_frame.pack(side = LEFT, anchor = NW, padx = 10)
		self.c_search = Entry(self.side_frame,
				width = 17,
				font = ('TlwgTypist', 13, 'bold italic'))
		self.c_search.pack(side = TOP)
		self.id_frame = Frame(self.side_frame,
				bd = 5,
```

```
                    relief = 'ridge')
self.id_frame.pack()
self.id_list = Listbox(self.id_frame,
                           width = 18,
                           font = ('TlwgTypist', 10, 'bold'),
                           fg = 'lightblue',
                           bg = 'black',
                           height = 10)
self.scrollbar = Scrollbar(self.id_frame,
                           command = self.id_list.yview)
self.id_list.configure(yscrollcommand = self.scrollbar.set)
self.id_list.pack(side = LEFT, fill = X)
self.id_list.bind('<Return>', self.date_editor)
self.id_list.bind('<Double-Button-1>', self.date_editor)
self.id_list.bind('<Delete>', self.del_date)
self.scrollbar.pack(side = RIGHT, fill = Y)
self.but = Button(self.side_frame,
                    text = 'SEARCH',
                    font = ('URWGothic', 13, 'bold'),
                    fg = 'lightblue',
                    bg = 'black',
                    command = self.search_engine)
self.but.pack(fill = X)
self.c_search.bind('<Return>', self.search_engine)
self.c_search1 = Entry(self.side_frame,
                    width = 17,
                    font = ('TlwgTypist', 13, 'bold italic'),)
self.c_search1.pack(side = TOP)
self.c_search1.bind('<Return>', self.id_engine)
self.id_frame1 = Frame(self.side_frame,
                    bd = 5,
                    relief = 'ridge')
self.id_frame1.pack()
self.id_list1 = Listbox(self.id_frame1,
                           width = 15,
                           font = ('TlwgTypist', 13, 'bold italic'),
                           fg = 'lightgreen',
                           bg = 'black',
                           height = 9)
self.id_list1.bind('<Double-Button-1>', self.student_details)
self.id_list1.bind('<Return>', self.student_details)
self.scrollbar1 = Scrollbar(self.id_frame1,
                           command = self.id_list1.yview)
self.id_list1.configure(yscrollcommand = self.scrollbar1.set)
self.id_list1.pack(side = LEFT, fill = X)
self.scrollbar1.pack(side = RIGHT, fill = Y)
self.but1 = Button(self.side_frame,
```

```
                    text = 'SEARCH',
                    font = ('URWGothic', 13, 'bold'),
                    fg = 'lightblue',
                    bg = 'black',
                    command = self.id_engine)
self.but1.pack(fill = X)
but_frame = Frame(self.side_frame)
but_frame.pack()
b1 = Button(but_frame,
            bg = 'black',
            fg = 'white',
            text = 'RollNo',
            width = 4,
            command = self.fill_rolls)
b1.grid(row = 0, column = 1)
b2 = Button(but_frame,
            bg = 'black',
            fg = 'white',
            width = 3,
            text = 'IdNum',
            command = self.fill_ids)
b2.grid(row = 0, column = 2)
b3 = Button(but_frame,
            bg = 'black',
            fg = 'white',
            text = 'Name',
            command = self.fill_names)
b3.grid(row = 0, column = 3)
self.get_details = Button(self.side_frame,
                          text = 'Get Details',
                          font = ('URWGothic', 13, 'bold'),
                          fg = 'cyan',
                          bg = 'black',
                          command = self.getf_details)
self.get_details.pack(side = BOTTOM, fill = X)
self.d1 = Frame(self.root,
                    bd = 5,
                    width = 1190,
                    height = 700,
                    relief = 'ridge',
                    bg = 'lightgreen')
self.d1.pack(side = LEFT)
self.d1.pack_propagate(False)
self.s1_frame = Frame(self.d1,
                    bg = 'lightgreen',
                    bd = 5,
                    relief = 'groove')
```

```
self.s1_frame.pack(fill = X)
lab = Label(self.s1_frame,
             text = 'FacultyName:',
             font = ('TlwgTypist', 14, 'bold'),
             bg = 'lightgreen',
             fg = 'red')
lab.pack(side = LEFT, fill = Y)
lab1 = Label(self.s1_frame,
             text = self.faculty + '\t',
             font = ('TlwgTypist', 14, 'bold'),
             bg = 'lightgreen',
             fg = 'blue')
lab1.pack(side = LEFT, fill = Y)
lab = Label(self.s1_frame,
             text = '     Department:',
             font = ('TlwgTypist', 14, 'bold'),
             bg = 'lightgreen',
             fg = 'red')
lab.pack(side = LEFT, fill = Y)
lab1 = Label(self.s1_frame,
             text = self.subject,
             font = ('TlwgTypist', 14, 'bold'),
             bg = 'lightgreen',
             fg = 'blue')
lab1.pack(side = LEFT, fill = Y)
lab = Label(self.s1_frame,
             text = '\tClass:',
             font = ('TlwgTypist', 14, 'bold'),
             bg = 'lightgreen',
             fg = 'red')
lab.pack(side = LEFT, fill = Y)
self.load_data()
self.quit = Button(self.s1_frame,
                   text = 'QUIT',
                   bg = 'lightgreen',
                   font = ('TlwgTypist', 14, 'bold'),
                   command = self.quiting)
self.quit.pack(side = RIGHT, padx = 50)
self.add = Button(self.s1_frame,
                   text = 'New Class',
                   bg = 'lightgreen',
                   font = ('TlwgTypist', 14, 'bold'),
                   command = self.new_class)
self.add.pack(side = RIGHT, padx = 10)
self.but_frame = Frame(self.d1, bg ='lightgreen')
self.but_frame.pack(pady = 10)
self.load = Button(self.but_frame,
```

```
                    text = 'LOAD',
                    font = ('URWGothic', 12, 'bold'),
                    bg = 'lightblue',
                    activebackground = 'cyan',
                    command = self.main_frame)
self.load.pack(side  = LEFT, padx = 10)
self.refresh = Button(self.but_frame,
                    text = 'Refresh DataBase',
                    font = ('URWGothic', 12, 'bold'),
                    bg = 'lightblue',
                    activebackground = 'cyan',
                    command = self.load_data)
self.refresh.pack(side = LEFT, padx = 10)
def main_frame(self):
try:
      self.m_frame.destroy()
except AttributeError:
      pass
self.m_frame = Frame(self.d1,
                    width = 1190,
                    height = 600,
                    bd = 5,
                    highlightbackground = 'black',
                    highlightcolor = 'red',
                    highlightthickness = 3,
                    relief = 'ridge',
                    bg = 'lightgreen')
self.m_frame.pack()
self.m_frame.pack_propagate(False)
self.file = self.clas.get()
self.df = pd.read_csv(f'{self.mainDir}/Data/' + self.file + '.csv')
print(self.df)
self.dates = self.df.columns[3:]
self.ids = self.df['ID Num']
self.names = self.df['Name']
self.rolls = self.df['RollNo']
print(self.ids,self.names)
print(self.ids[0])
self.id_list.delete(0, END)
self.id_list1.delete(0, END)
for k in self.dates:
      self.id_list.insert(END, k)
for k in self.ids:
      self.id_list1.insert(END, k)
self.clas1, self.sub = self.file.split('_')
self.h3 = Label(self.m_frame,
                    text = '{}-{}  Attendance'.format(self.clas1.upper(),
```

```
self.sub.upper()),
                          font = ('TlwgTypist', 25, 'bold'),
                          bg = 'lightgreen',
                          fg = 'white')
        self.h3.pack()
        self.mf1 = Frame(self.m_frame,
                          bg = 'lightgreen')
        self.mf1.pack()
        self.mark_label = Label(self.mf1,
                                  text = 'Mark Attendance:  ',
                                  font = ('URWGothic', 12, 'bold'),
                                  fg = 'black',
                                  bg = 'lightgreen')
        self.mark_label.pack(side = LEFT)
        self.m_today = Button(self.mf1,
                          text = 'Today',
                          font = ('URWChanceryL', 14, 'bold'),
                          fg = 'red',
                          bg = 'lightgreen',
                          command = lambda :self.mark_today(str(dt.date.today()),
'mark'))
        self.m_today.pack(side = LEFT)
        self.m_oday = Button(self.mf1,
                                  text = 'Other',
                                  font = ('URWChanceryL', 14, 'bold'),
                                  fg = 'red',
                                  bg = 'lightgreen',
                          command = self.ask_date)
        self.m_oday.pack(side = LEFT)
        self.mf2 = Frame(self.m_frame,
                          bd = 5,
                          relief = 'groove')
        self.mf2.pack()
        self.s_frame = Frame(self.mf2,
                          width = 530,
                          bd = 5,
                          height = 430,
                          bg = 'lightgreen',
                          relief = 'groove')
        self.s_frame.pack(side = LEFT)
        self.s_frame.pack_propagate(False)
        self.d_frame = Frame(self.mf2,
                          width = 590,
                          height = 430,
                          bd = 5,
                          bg = 'lightgreen',
                          relief = 'groove')
```

```
        self.d_frame.pack(side = RIGHT)
        self.d_frame.pack_propagate(False)
        def getf_details(self, *args):
        if len(self.id_list.curselection()) == 0 and
len(self.id_list1.curselection()) != 0:
                self.student_details()
        elif len(self.id_list.curselection()) != 0 and
len(self.id_list1.curselection()) == 0:
                self.date_editor()
        else:
                pass
        def date_editor(self, *args):
        date = self.id_list.curselection()
        cur_date = self.id_list.get(date)
        print(cur_date)
        try:
                self.f1.destroy()
                self.l1.destroy()
                self.dl.destroy()
                self.ab_l1.destroy()
                self.ab_l2.destroy()
                self.edit.destroy()
                self.copy.destroy()
                self.frm.destroy()
        except AttributeError:
                pass
        self.f1 = Frame(self.s_frame, bg = 'lightgreen')
        self.f1.pack()
        self.l1 = Label(self.f1,
                text = 'Date:',
                font = ('TlwgTypist', 14, 'bold'),
                bg = 'lightgreen',
                fg = 'blue')
        self.l1.pack(side = LEFT)
        self.dl = Label(self.f1,
                            text = str(cur_date),
                            font = ('URWGothic', 14, 'bold'),
                            bg = 'lightgreen',
                            fg = 'red')
        self.dl.pack(side = RIGHT)
        abs_list = self.df[cur_date]
        abs_dict = dict(abs_list)
        self.abs_dict = [int(j)+1 for j in abs_dict if abs_dict[j] == 'A']
        self.ab_l1 = Label(self.s_frame,
                                  text = 'Absentees:',
                                  font = ('TlwgTypist', 13, 'bold'),
                                  fg = 'red',
```

```
                        bg = 'lightgreen')
self.ab_l1.pack(side = LEFT, anchor = NE)
self.ab_l2 = Message(self.s_frame,
                        text = self.abs_dict,
                        font = ('TlwgTypist', 13, 'bold'),
                        fg = 'blue',
                        bg = 'lightgreen')
self.ab_l2.pack(side = LEFT, anchor = NE)
self.edit = Button(self.s_frame,
                   text = 'Edit Attendance',
                   font = ('Times', 16, 'bold italic'),
                   fg = 'red',
                   bg = 'lightgreen',
                   command = lambda:self.mark_today(cur_date, 'edit'))
self.edit.place(relx = 0.38, rely = 0.6)
self.frm = Frame(self.s_frame)
self.frm.place(relx = 0, rely = 0.9)
self.format = Button(self.frm,
                   text = 'AMS Format',
                   font = ('TlwgTypist', 13, 'bold'),
                         fg = 'cyan',
                         bg = 'black',
                   command = self.format_func)
self.format.pack(side = LEFT)
self.copy = Button(self.frm,
                   text = 'Copy Absentees',
                   font = ('TlwgTypist', 13, 'bold'),
                         fg = 'cyan',
                         bg = 'black',
                   command = self.copy_func)
self.copy.pack(side = RIGHT)
def copy_func(self):
stri = ''
for j in self.abs_dict:
      stri += str(j) + ','
stri = stri[:-1]
print(stri)
pyperclip.copy(stri)
print('COPY DONE')
def format_func(self):
new = self.df.T.iloc[3:]
boo = (new == 'A')
dic = {}
for k in self.dates:
      dic[k] = []
      for j in self.rolls:
      if boo.loc[k, j-1]:
```

```
                dic[k].append(j)
        print(dic)
        with open(f'{self.mainDir}/Data/{self.file}_f.csv', 'w') as out2:
                for l in dic:
                string2 = l.replace(',', '-') + ','
                for k in dic[l]:
                        string2 += str(k) + ','
                out2.write(string2 + '\n')
        self.pop_up("Your file is created in the application's working directory in
the format of filename_f.csv")
        def mark_today(self, date, flag):
        self.n_wind = Toplevel()
        self.n_wind.config(bg = 'black')
        self.n_wind.title('Attendance Marker')
        self.value = IntVar()
        Label(self.n_wind,text = '{} Attendance'.format(date),
                font = ('URWGothic', 17, 'bold'),
                fg = 'cyan',
                bg = 'black').pack()
        f4 = Frame(self.n_wind, bg = 'black')
        f4.pack()
        Label(f4,
                text = 'Period:   ',
                font = ('TlwgTypist', 14, 'bold'),
                fg = 'blue',
                bg = 'black').pack(side = LEFT)
        self.per = Entry(f4,
                font = ('TlwgTypist', 14, 'bold'),
                fg = 'orange',
                bg = 'black')
        self.per.pack(side = LEFT)
        Label(self.n_wind,text = 'Enter roll numbers seperated by commas(\',\'):',
                font = ('Times', 14),
                fg = 'red',
                bg = 'black').pack()
        self.a_entry = Entry(self.n_wind,
                            font = ('TlwgTypist', 14, 'bold'),
                            bg = 'lightgreen',
                            width = 50,
                            fg = 'black')
        self.a_entry.pack()
        if flag == 'edit':
                self.per.insert(END, date.split('-')[-1][1:-1])
                self.a_entry.focus()
        f3 = Frame(self.n_wind, bg = 'black')
        f3.pack()
        Label(f3, text = 'Choose Filter: ', font = ('TlwgTypist', 10, 'bold'), bg =
```

```
'black', fg = 'blue').pack(side = LEFT)
        self.r1 = Radiobutton(f3,
                              text = 'Absentees',
                              font = ('TlwgTypist', 10, 'bold'),
                              value = 1,
                              variable = self.value)
        self.r1.pack(side = LEFT)
        self.r2 = Radiobutton(f3,
                              text = 'Attendees',
                              font = ('TlwgTypist', 10, 'bold'),
                              value = 2,
                              variable = self.value)
        self.r2.pack(side = LEFT)
        self.mark_but = Button(self.n_wind,
                                    text = 'Mark Attendance',
                                    font = ('TlwgTypist', 13, 'bold'),
                                    fg = 'lightgreen',
                                    bg = 'black',
                                    command = lambda:self.mark_func(date, flag))
        self.mark_but.pack()
        self.value.set(1)
        def ask_date(self):
        self.new = Toplevel()
        self.new.title('Custom Date')
        Label(self.new,
              text = 'Choose Your Custom Date:',
              font = ('TlwgTypist', 13, 'bold'),
              fg = 'lightgreen',bg = 'black').pack()
        self.year = StringVar()
        self.month = StringVar()
        self.day = StringVar()
        frame = Frame(self.new)
        frame.pack()
        year_lis = [j for j in range(2019, 2026)]
        month_lis = [str(j).zfill(2) for j in range(1, 13)]
        day_lis = [j for j in range(1, 32)]
        o1 = OptionMenu(frame,self.year,*year_lis)
        o1.pack(side = LEFT)
        o2 = OptionMenu(frame,self.month,*month_lis)
        o2.pack(side = LEFT)
        o3 = OptionMenu(frame,self.day,*day_lis)
        o3.pack(side = LEFT)
        ok = Button(self.new,
                    text = 'OK',
                    font = ('TlwgTypist', 10, 'bold'),
                    fg = 'black',
                    command = self.get)
```

```
        ok.pack()
        def get(self):
        date = self.year.get() + '-' + self.month.get() + '-' + self.day.get()
        self.new.destroy()
        self.mark_today(date, 'mark')
        def mark_func(self, date, flag):
        print(date)
        filt = self.value.get()
        period = self.per.get()
        matter1 = self.a_entry.get()
        m_lis = []
        if matter1 == '' and filt == 1:
               for x in self.df.index:
               m_lis.append('A')
        elif matter1 == '' and filt == 2:
               for x in self.df.index:
               m_lis.append('P')
        else:
               matter1 = matter1.split(',')
               matter = [int(j) - 1 for j in matter1]
               for x in self.df.index:
               if (x in matter) and filt == 1:
                      m_lis.append('A')
               elif (x not in matter) and filt == 1:
                      m_lis.append('P')
               elif (x in matter) and filt == 2:
                      m_lis.append('P')
               else:
                      m_lis.append('A')
        if flag == 'mark':
               print(period)
               self.df[date + '-(' + period + ')'] = m_lis
        elif flag == 'edit':
               print(period)
               self.df[date] = m_lis
        self.df.to_csv(f'{self.mainDir}/Data/' + self.file + '.csv', index = False)
        self.n_wind.destroy()
        self.n_wind = Toplevel()
        self.n_wind.config(bg = 'black')
        self.n_wind.title('Status')
        data = 'Class:{0}  Subject:{1}\nNumber of
{2}:{3}\n{2}:{4}'.format(self.clas1, self.sub, 'Absentees:' if filt == 1 else
'Attendees', len(matter1), matter1)
        Label(self.n_wind,
               text = 'Attendance Marked Successfully.',
               font = ('URWGothic', 17, 'bold'),
               fg = 'cyan',
```

```
                bg = 'black').pack()
        Message(self.n_wind,
                text = data,
                font = ('TlwgTypist', 13, 'bold'),
                fg = 'blue',
                bg = 'black').pack()
        Button(self.n_wind,
                text = 'OK',
                font = ('TlwgTypist', 13, 'bold'),
                bg = 'black',
                fg = 'cyan',
                command = self.n_wind.destroy).pack()
        self.main_frame()
        def quiting(self):
        reply = messagebox.askquestion('Quit in progress...', 'Are you sure?')
        if reply == 'yes':
                self.root.destroy()
        def del_date(self, *args):
        date = self.id_list.curselection()
        cur_date = self.dates[date]
        reply = messagebox.askquestion(f'Deleting {cur_date}......', 'Are you
sure???')
        if reply == 'yes':
                del self.df[cur_date]
                self.id_list.delete(date)
                self.df.to_csv(f'{self.mainDir}/Data/' + self.file + '.csv', index =
False)
                self.main_frame()
        def load_data(self):
        try:
                self.s1.destroy()
        except AttributeError:
                pass
        files = os.listdir(f'{self.mainDir}/Data/')
        files = [j[:-4] for j in files]
        self.clas = StringVar()
        self.s1 = OptionMenu(self.s1_frame, self.clas, *files)
        self.s1.place(relx = 0.58, rely = 0.01)
        self.s1.config(font = ('TlwgTypist', 13, 'bold'), bg = 'lightgreen')
        def new_class(self):
        self.new_wind = Toplevel()
        self.new_wind.bg = 'black'
        self.new_wind.config(bg = 'black')
        self.new_wind.title('New Class Creator')
        self.h2 = Label(self.new_wind,
                        text = 'New Class Creator',
                        font = ('TlwgTypist', 25, 'bold italic'),
```

```
                    bg = 'black',
                    fg = 'cyan')
self.h2.pack()
self.frame = Frame(self.new_wind,bg = 'black')
self.frame.pack()
self.l1 = Label(self.frame,
                    text = 'Name:  ',
                    font = ('TlwgTypist', 15, 'bold italic'),
                    bg = 'black',
                    fg = 'red')
self.l1.pack(side = LEFT)
self.name = Entry(self.frame,
                    fg = 'blue',
                    font = ('URWGothic', 12, 'bold'))
self.name.pack(side = LEFT)
self.l2 = Label(self.frame,
                    text = '     Subject:  ',
                    font = ('TlwgTypist', 15, 'bold italic'),
                    bg = 'black',
                    fg = 'red')
self.l2.pack(side = LEFT)
self.sub = Entry(self.frame,
                    fg = 'blue',
                    font = ('URWGothic', 12, 'bold'))
self.sub.pack(side = LEFT)
self.csv = Button(self.new_wind,
                    text = 'CSV file',
                    font = ('TlwgTypist', 15, 'bold italic'),
                    bg = 'black',
                    fg = 'red',
                    command = self.add_csv)
self.csv.pack()
self.f2 = Frame(self.new_wind)
self.f2.pack(pady = 10)
self.create = Button(self.f2,
                    text = 'Create',
                    font = ('Times', 15, 'bold italic'),
                    bg = 'black',
                    fg = 'cyan',
                    command = self.create_csv)
self.create.pack(side = LEFT)
self.close = Button(self.f2,
                    text = 'Cancel',
                    font = ('Times', 15, 'bold italic'),
                    bg = 'black',
                    fg = 'cyan',
                    command = self.new_wind.destroy)
```

```
        self.close.pack(side = LEFT)
        def create_csv(self):
        cla = self.name.get()
        sub = self.sub.get()
        os.system(f'cp {self.csv}
/home/xand/Desktop/Xattendance/Data/{cla}_{sub}.csv')
        self.new_wind.destroy()
        self.pop_up('Successfully Created new class.')
        def add_csv(self):
        self.csv = filedialog.askopenfilename(initialdir = '/', title = 'Select CSV
file', filetypes = [['CSV files', '*.csv'], ['All Files', '*.*']])
        try:
                self.lab.destroy()
        except AttributeError:
                pass
        self.lab = Label(self.new_wind,
                                text = 'File:' + self.csv,
                                font = ('Times', 14, 'italic'),
                        bg = 'black',
                        fg = 'white')
        self.lab.pack()
        def cancel_func(self):
        self.window.quit()
        self.window.destroy()
        def search_engine(self, *args):
        days = ['january', 'february', 'march', 'april', 'may', 'june', 'july',
'august', 'september', 'october', 'november', 'december']
        shorts = ['jan', 'feb', 'march', 'april', 'may', 'june', 'july', 'august',
'sept', 'oct', 'nov', 'dec']
        d1 = {days[x-1]:x for x in range(1, 13)}
        d2 = {shorts[x-1]:x for x in range(1, 13)}
        pattern = self.c_search.get().upper()
        self.id_list.delete(0, END)
        print(self.dates)
        for j in self.dates:
                if re.search(pattern, j):
                self.id_list.insert(END, j)
        pattern = pattern.lower()
        if pattern in days:
                for j in self.dates:
                if int(d1[pattern]) == int(j[5:7]):
                        self.id_list.insert(END, j)
        if pattern in shorts:
                for j in self.dates:
                if int(d2[pattern]) == int(j[5:7]):
                        self.id_list.insert(END, j)
        # FROM HERE WE ARE WRITING  THE STUDENT DETAILS
```

```
        def student_details(self, *args):
        sel = self.id_list1.curselection()
        if self.flag == 1:
               id_num = self.id_list1.get(sel[0])
               ind = list(self.ids).index(id_num)
               name = self.names[ind]
               roll = self.rolls[ind]
        elif self.flag == 2:
               name = self.id_list1.get(sel[0])
               ind = list(self.names).index(name)
               id_num = self.ids[ind]
               roll = self.rolls[ind]
        else:
               roll = self.id_list1.get(sel[0])
               ind = list(self.rolls).index(int(roll))
               name = self.names[ind]
               id_num = self.ids[ind]
        self.common_function(id_num, name, ind, roll)
        def common_function(self, id_num, name, ind, roll):
        try:
               self.sub_frame.destroy()
               self.s_edit.destroy()
               self.img_frame.destroy()
        except AttributeError:
               pass
        data1 = 'RollNo:\nID_Num:\nName:'
        print(roll)
        data = str(roll) + '\n' + id_num + '\n' + name
        # d1_label ----> Headings
        # d2_label ----> Details Data
        data1 += '\nTotal Periods:\nNo. of periods present:\nNo. of periods
absent:\nPercentage:\nAbsent Dates:'
        series = self.df.loc[ind][3:]
        self.atte_dict = dict(zip(self.dates, series))
        print('Atte_dict:', self.atte_dict)
        ab = list(series).count('A')
        pe = list(series).count('P')
        percent = round(((pe/len(series)) * 100), 3)
        data += f'\n{len(series)}\n{pe}\n{ab}\n{percent} %'
        abs_dates = [j for j in series.index if series[j] == 'A']
        print('Absent',abs_dates)
        self.sub_frame = Frame(self.d_frame, bg = 'lightgreen')
        self.sub_frame.pack(side = LEFT, anchor = NW)
        self.d1_label = Message(self.sub_frame,
                            text = data1,
                            font = ('URWGothic', 11, 'bold'),
                            fg = 'black',
```

```
                    bg = 'lightgreen')
self.d1_label.pack(side = LEFT, anchor = NW, pady = 10)
self.d2_label = Message(self.sub_frame,
                    text = data,
                    font = ('URWGothic', 11, 'bold'),
                    fg = 'blue',
                    bg = 'lightgreen')
self.d2_label.pack(anchor = N, pady = 10)
self.l_frame = Frame(self.sub_frame, bg = 'lightgreen')
self.l_frame.pack(side = LEFT)
self.lis_frame = Frame(self.l_frame)
self.lis_frame.pack()
self.l_box = Listbox(self.lis_frame)
self.s_bar = Scrollbar(self.lis_frame,
                           command = self.l_box.yview)
self.l_box.config(bg = 'black',
                    yscrollcommand = self.s_bar.set,
                    font = ('TlwgTypist',10,'bold'),
                    fg = 'white',
                    height = 5)
self.l_box.pack(side = LEFT)
self.s_bar.pack(side = RIGHT, fill = Y)
if ab != 0:
      for i in abs_dates:
      self.l_box.insert(END, i)
else:
      self.l_box.insert(END, 'NONE')
self.s_edit = Button(self.l_frame,
                    text = 'Edit Attendance',
                    font = ('Times', 16, 'bold italic'),
                    fg = 'red',
                    bg = 'lightgreen',
                    command = lambda:self.student_edit(roll - 1))
self.s_edit.pack(side = BOTTOM, pady = 20)
self.img_frame = Button(self.d_frame,
                           bd = 5,
                           relief = 'groove',
                           bg = 'black',
                           activebackground = 'red')
self.img_frame.pack(side = RIGHT, anchor = NE)
try:
      img = PhotoImage(file = f'{self.mainDir}/image/{id_num}.png')
      self.img_frame['image'] = img
      self.img_frame.image = img
except:
      pass
def pop_up(self, message):
```

```
self.pop_wind = Toplevel(self.root)
Label(self.pop_wind,
      text = message,
      font = ('Verdana', 14)).pack()
Button(self.pop_wind,
      text = 'OKay',
      font = ('URWGOthic', 14),
      command = self.pop_wind.destroy).pack()
def student_edit(self, roll):
self.sdit = Toplevel()
self.sdit.config(bg = 'lightgreen')
h1 = Label(self.sdit,
            text = f'{self.ids[roll]}-{self.names[roll]}',
            font = ('URWGothic', 15, 'italic'),
            bg = 'lightgreen',
            fg = 'black')
h1.pack()
frame = Frame(self.sdit,
            bg = 'lightgreen')
frame.pack(fill = X)
f2 = Frame(frame, bg = 'lightgreen')
f2.pack(fill = X)
self.s_eng = Entry(f2,
                   bg = 'black',
                   fg = 'white',
                   width = 30,
                   font = ('TlwgTypist', 14, 'bold'))
self.s_eng.pack(side = LEFT, fill = X)
self.s_eng.bind('<Return>', lambda x:self.search_engine1(roll))
self.sea = Button(f2,
                   bg = 'black',
                   text = 'Search',
                   fg = 'lightgreen',
                   font = ('TlwgTypist', 12, 'bold'),
                   command = lambda :self.search_engine1(roll))
self.sea.pack(side = LEFT, fill = X)
self.d1_box = Listbox(frame,
                   font = ('Uroob', 20, 'bold'),
                   height = 4)
self.d1_box.pack(side = LEFT)
self.d2_box = Listbox(frame,
                   font = ('Uroob', 20, 'bold'),
                   height = 4)
self.d2_box.pack(side = LEFT)
s_bar = Scrollbar(frame,
                   command = self.double_scroll)
s_bar.pack(side = RIGHT, fill = Y)
```

```
        self.d1_box.config(yscrollcommand = s_bar.set)
        self.d2_box.config(yscrollcommand = s_bar.set)
        for j in self.dates:
                self.d1_box.insert(END, j)
        l = list(self.df.loc[roll][3:])
        self.d1_box.bind('<Button-4>',self.list_scroll)
        self.d2_box.bind('<Button-5>',self.list_scroll)
        self.d1_box.bind('<Button-5>',self.list_scroll)
        self.d2_box.bind('<Button-4>',self.list_scroll)
        self.d1_box.bind('<Down>', self.ar_down)
        self.d1_box.bind('<Up>', self.ar_up)
        self.d2_box.bind('<Down>', self.ar_down1)
        self.d2_box.bind('<Up>', self.ar_up1)
        self.d2_box.bind('<Double-Button-1>', lambda
one:self.change1(self.d2_box.curselection()))
        self.d2_box.bind('<Return>', lambda
one:self.change1(self.d2_box.curselection()))
        for k in l:
                if k == 'P':
                self.d2_box.insert(END, 'Present')
                self.d2_box.itemconfig(END, {'fg' : 'green'})
                else:                self.d2_box.insert(END, 'Absent')
                self.d2_box.itemconfig(END, {'fg' : 'red'})
        self.up_but = Button(self.sdit,
                             text = 'UPDATE',
                             font = ('TlwgTypist', 15),
                             command = lambda:self.update1(roll))
        self.up_but.pack()
        self.sdit.mainloop()
        def double_scroll(self, *args):
        self.d1_box.yview(*args)
        self.d2_box.yview(*args)
        def list_scroll(self, event):
        if event.num == 4:
                delta = -1
        else:
                delta = 1
        self.d1_box.yview('scroll', delta, 'units')
        self.d2_box.yview('scroll', delta, 'units')
        return 'break'
        def ar_down(self, *args):
        cur = self.d1_box.curselection()[0]
        self.d2_box.selection_set(cur + 1)
        self.d1_box.selection_set(cur + 1)
        self.d1_box.yview('scroll', 1, 'units')
        self.d2_box.yview('scroll', 1, 'units')
        return 'break'
```

```
def ar_up(self, *args):
cur = self.d1_box.curselection()[0]
self.d2_box.selection_set(cur - 1)
self.d1_box.selection_set(cur - 1)
self.d1_box.yview('scroll', -1, 'units')
self.d2_box.yview('scroll', -1, 'units')
return 'break'
def ar_down1(self, *args):
cur = self.d2_box.curselection()[0]
self.d1_box.selection_set(cur + 1)
self.d2_box.selection_set(cur + 1)
self.d2_box.yview('scroll', 1, 'units')
self.d1_box.yview('scroll', 1, 'units')
return 'break'
def ar_up1(self, *args):
cur = self.d2_box.curselection()[0]
self.d1_box.selection_set(cur - 1)
self.d2_box.selection_set(cur - 1)
self.d2_box.yview('scroll', -1, 'units')
self.d1_box.yview('scroll', -1, 'units')
return 'break'
def change1(self, cur, *args):
print('Yes')
it = self.d2_box.get(cur[0])
self.d2_box.insert(cur[0], 'Present' if it[0] == 'A' else 'Absent')
self.d2_box.itemconfig(cur[0], {'fg' : 'green' if it[0] == 'A' else 'red'})
self.d2_box.delete(cur[0] + 1)
self.d2_box.selection_set(cur[0])
if it[0] == 'A':
      self.atte_dict[self.d1_box.get(cur[0])] = 'P'
else:
      self.atte_dict[self.d1_box.get(cur[0])] = 'A'
def update1(self, roll, *args):
self.s_eng.delete(0, END)
self.search_engine1(roll)
n_lis = [str(roll + 1), self.ids[roll], self.names[roll]]
print('Dict:', self.atte_dict)
n_lis = n_lis + list(self.atte_dict.values())
print('N_lis', n_lis)
index = ['RollNo', 'ID Num', 'Name'] + list(self.atte_dict.keys())
print('Index', index)
series = pd.Series(n_lis, index = index)
print('Series', series)
print('Iloc', self.df.iloc[roll])
self.df.iloc[roll] = series
print(self.df.iloc[roll])
self.df.to_csv(f'{self.mainDir}/Data/' + self.file + '.csv', index = False)
```

```
        self.sdit.destroy()
        self.pop_up('Successfully Updated Attendance...')
        self.main_frame()
        self.id_list1.selection_set(roll)
        self.student_details()
        def search_engine1(self, roll, *args):
        days = ['january', 'february', 'march', 'april', 'may', 'june', 'july',
'august', 'september', 'october', 'november', 'december']
        shorts = ['jan', 'feb', 'march', 'april', 'may', 'june', 'july', 'august',
'sept', 'oct', 'nov', 'dec']
        d1 = {days[x-1]:x for x in range(1, 13)}
        d2 = {shorts[x-1]:x for x in range(1, 13)}
        pattern = self.s_eng.get().upper()
        self.d1_box.delete(0, END)
        self.d2_box.delete(0, END)
        print(self.dates)
        for j in self.dates:
                if re.search(pattern, j):
                self.d1_box.insert(END, j)
        for k in self.d1_box.get(0, END):
                self.d2_box.insert(END, 'Absent' if self.atte_dict[k] == 'A' else
'Present')
                self.d2_box.itemconfig(END, {'fg':'green' if self.atte_dict[k] == 'A'
else 'red'})
        pattern = pattern.lower()
        if pattern in days:
                for j in self.dates:
                if int(d1[pattern]) == int(j[5:7]):
                        self.d1_box.insert(END, j)
                for k in self.d1_box.get(0, END):
                self.d2_box.insert(END, 'Absent' if self.atte_dict[k] == 'A' else
'Present')
                self.d2_box.itemconfig(END, {'fg':'green' if self.atte_dict[k] == 'A'
else 'red'})
        if pattern in shorts:
                for j in self.dates:
                if int(d2[pattern]) == int(j[5:7]):
                        self.d1_box.insert(END, j)
                for k in self.d1_box.get(0, END):
                self.d2_box.insert(END, 'Absent' if self.atte_dict[k] == 'A' else
'Present')
                self.d2_box.itemconfig(END, {'fg':'green' if self.atte_dict[k] == 'A'
else 'red'})
        def id_engine(self, *args):
        pattern = self.c_search1.get().upper()
        self.id_list1.delete(0, END)
        self.id_list1.delete(0, END)
```

```
        print(self.dates)
        if self.flag == 1:
                for j in self.ids:
                if re.search(pattern, j):
                        self.id_list1.insert(END, j)
        elif self.flag == 2:
                for j in self.names:
                if re.search(pattern, j):
                        self.id_list1.insert(END, j)
        else:
                for j in self.rolls:
                if re.search(pattern, str(j)):
                        self.id_list1.insert(END, j)
        def fill_ids(self):
        self.flag = 1
        self.id_list1.config(font = ('TlwgTypist', 12, 'bold italic'),
                                fg = 'lightgreen',
                            width = 15)
        self.id_list1.delete(0, END)
        for j in self.ids:
                self.id_list1.insert(END, j)
        def fill_names(self):
        self.flag = 2
        self.id_list1.config(font = ('TlwgTypist', 10, 'bold'),fg =
'lightblue',width = 18)
        self.id_list1.delete(0, END)
        for j in self.names:
                self.id_list1.insert(END, j)
        def fill_rolls(self):
        self.flag = 3
        self.id_list1.config(font = ('TlwgTypist', 12, 'bold italic'),
                                fg = 'lightgreen',
                            width = 15)
        self.id_list1.delete(0, END)
        for j in self.rolls:
                self.id_list1.insert(END, str(j))
if __name__ == '__main__':
        obj = Xattendance()
```

FILENAME : INSTALLER.PY

```
from tkinter import *
from threading import *
import os
```

```
from tkinter import ttk
from tkinter import scrolledtext as srctext
from tkinter import PhotoImage
import subprocess
class Installer:
      def __init__(self):
      self.host = ''
      self.port = ''
      self.user = ''
      self.pwd =  ''
      self.gui()
      self.root.mainloop()
      def gui(self):
      self.root = Tk()
      self.root.config(bg = 'white')
      self.root.title('Xattendance Installer')
      self.root.geometry('730x450')
      self.root.resizable(False, False)
      self.f1 = Frame(self.root,
                      bg = 'white')
      self.f1.pack(side = LEFT)
      self.img = PhotoImage(file = 'icon.png')
      self.imf = Label(self.f1,
                      image = self.img,
                      bg = 'white')
      self.imf.pack(side = LEFT)
      self.imf.image = self.img
      self.f2 = Frame(self.root,
                      bg = 'white')
      self.f2.pack(side = BOTTOM)
      self.back = Button(self.f2,
                           width = 15,
                           text = '<Back')
      self.back.pack(side = LEFT, fill = X)
      self.next = Button(self.f2,
                           width = 15,
                           text = 'Next>')
      self.next.pack(side = LEFT, fill = X)
      self.cancel = Button(self.f2,
```

```
                              width = 15,
                              text = 'Cancel')
        self.cancel.pack(side = RIGHT, fill = X)
        self.f3 = Frame(self.root,
                        bg = 'white')
        self.f3.pack()
        self.start_permit()
        def start_permit(self):
        self.f3.destroy()
        self.f3 = Frame(self.root,
                        bg = 'white')
        self.f3.pack()
        self.l1 = Label(self.f3,
                        text = 'Welcome to Installation Wizard of
Xattendance',
                        font = ('TlwgTypist', 12, 'bold'),
                        bg = 'white')
        self.l1.pack(side = TOP, pady = 10)
        self.l2 = Message(self.f3,
                        text = 'This will install Offline Attendance
Management System(Xattendance) on your PC.Click "Next" to
continue....',
                        bg = 'white',
                        width = 515,
                        font = ('aakar', 12))
        self.l2.pack(pady = 120, padx = 10)
        self.l3 = Message(self.f3,
                        text = 'NOTE:This installation requires
internet access unless you already installed the dependencies.',
                        bg = 'white',
                        width = 525,
                        font = ('aakar', 12))
        self.l3.pack()
        self.next['command'] = self.network_access
        self.back['command'] = None
        def network_access(self):
        self.f3.destroy()
        self.f3 = Frame(self.root,
                        bg = 'white')
```

```
self.f3.pack()
self.r1 = Label(self.f3,
                text = 'Network Configuration:',
                font = ('TlwgTypist', 12, 'bold'),
                bg = 'white')
self.r1.pack()
self.f4 = Frame(self.f3,
                bg = 'white')
self.f4.pack(side = LEFT, anchor = NW, pady = 30, fill = X)
self.flag = IntVar()
self.r2 = Radiobutton(self.f4,
                text = 'No Proxy',
                bg = 'white',
                font = ('URWGothic', 12),
                activebackground = 'white',
                highlightthickness = 0,
                value = 0,
                variable = self.flag)
self.r2.pack(side = TOP, anchor = NW)
self.r3 = Radiobutton(self.f4,
                text = 'Manual Proxy',
                bg = 'white',
                activebackground = 'white',
                highlightthickness = 0,
                value = 1,
                variable = self.flag,
                font = ('URWGothic', 12))
self.r3.pack(side = TOP, anchor = NW)
self.flag.set(0)
self.manual_proxy()
self.r2.bind('<Button-1>', self.e_deactive)
self.r3.bind('<Button-1>', self.e_active)
self.next['command'] = self.details_prompt
self.back['command'] = self.start_permit
def manual_proxy(self):
f4 = Frame(self.f4,
           bg = 'white')
f4.pack(fill = X)
l1 = Label(f4,
```

```
            text = 'Proxy Host:',
            bg = 'white',
            font = ('URWGothic', 12))
l1.pack(side = TOP, anchor = NW)
self.e1 = Entry(f4,
            bg = 'white',
            font = ('TlwgTypist', 12))
self.e1.pack(side = TOP, anchor = NW)
l2 = Label(f4,
            text = 'Port:',
            bg = 'white',
            font = ('URWGothic', 12))
l2.pack(side = TOP, anchor = NW)
self.e2 = Entry(f4,
            bg = 'white',
            font = ('TlwgTypist', 12))
self.e2.pack(side = TOP, anchor = NW)
l3 = Label(f4,
            text = 'UserName:',
            bg = 'white',
            font = ('URWGothic', 12))
l3.pack(side = TOP, anchor = NW)
self.e3 = Entry(f4,
            bg = 'white',
            font = ('TlwgTypist', 12))
self.e3.pack(side = TOP, anchor = NW)
f5 = Frame(f4,
            bg = 'white')
f5.pack(side = TOP, anchor = NW)
l4 = Label(f5,
            text = 'Password:',
            bg = 'white',
            font = ('URWGothic', 12))
l4.pack(side = TOP, anchor = NW)
self.e4 = Entry(f5,
            bg = 'white',
            font = ('TlwgTypist', 12),
            show = '*')
self.e4.pack(side = LEFT)
```

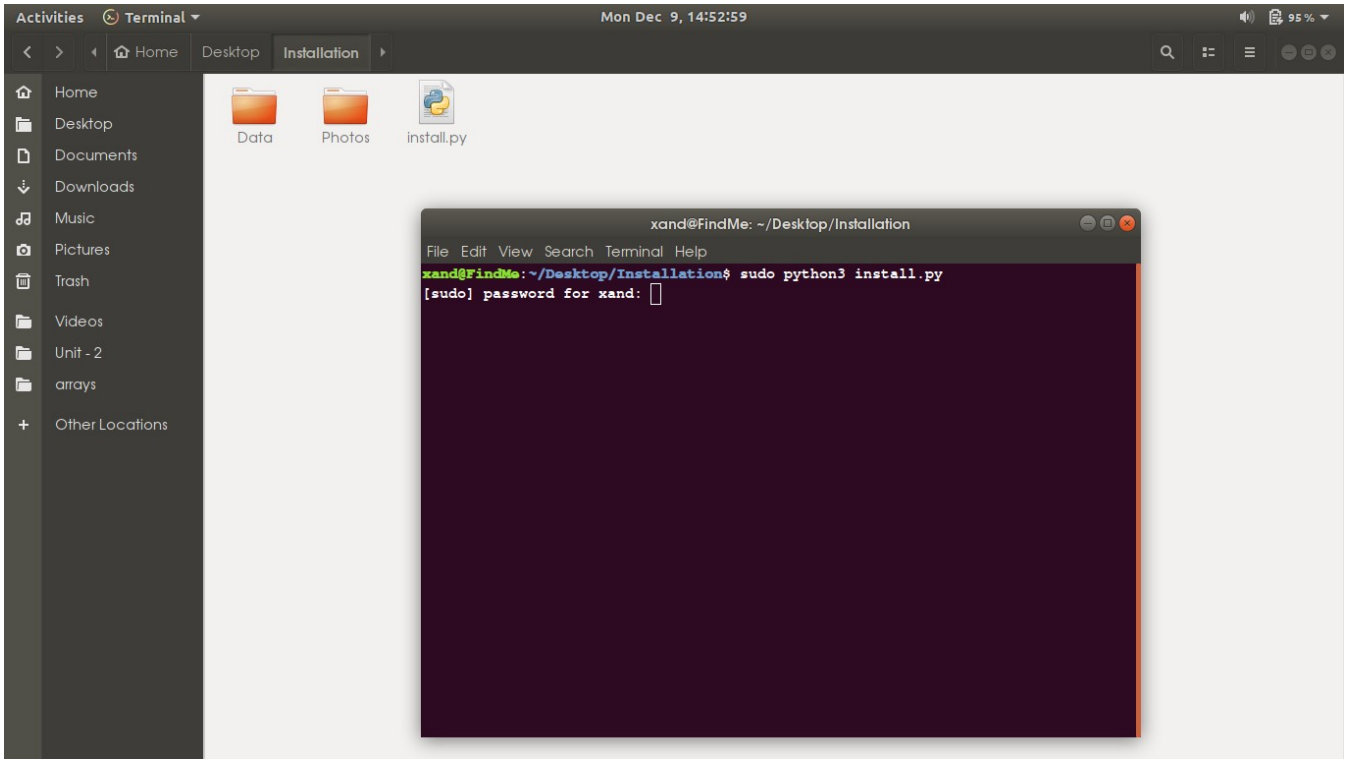
```
      self.show_var = 0
      self.show = Button(f5,
                         text = 'Show',
                         bg = 'white',
                         command = self.show_func,
                         font = ('URWGothic', 12))
      self.show.pack(side = LEFT)
      self.e_deactive()
      self.update = Button(f4,
                         text = 'UPDATE',
                         font = ('TlwgTypist', 11, 'bold'),
                         highlightthicknes = 0,
                         command = self.proxy_update)
      self.update.pack(side = TOP, anchor = NW, pady = 3)
      def show_func(self):
      if self.show_var == 0:
            self.e4['show'] = ''
            self.show['text'] = 'Hide'
            print('Entered Hide Block')
            self.show_var = 1
      else:
            self.e4['show'] = '*'
            print('Enter the show bloack')
            self.show['text'] = 'Show'
            self.show_var = 0
      def e_active(self, *args):
      self.e1['state'] = self.e2['state'] = self.e3['state'] =
self.e4['state'] = ['normal']
      def e_deactive(self, *args):
      self.e1['state'] = self.e2['state'] = self.e3['state'] =
self.e4['state'] = ['disabled']
      def proxy_update(self):
      try:
            with open('/etc/apt/apt.conf', 'w') as file:
                  if self.flag == 0:
                  os.system("notify-send 'Proxy Configuration updated
to No Proxy.'")
                  else:
                  self.host = self.e1.get()
```

```
                self.port = self.e2.get()
                self.user = self.e3.get()
                self.pwd =  self.e4.get()
                file.write(f'Acquire::http::proxy
"http://{self.user}:{self.pwd}@{self.user}:{self.pwd}/";')
                file.write(f'Acquire::https::proxy
"https://{self.user}:{self.pwd}@{self.user}:{self.pwd}/";')
                os.system("notify-send 'Proxy COnfiguration Updated
Successfully.'")
      except:
            print('Entered')
            os.system("notify-send 'Proxy Configuration Not
Updated.'")
      def details_prompt(self):
      self.f3.destroy()
      self.f3 = Frame(self.root,
                      bg = 'white')
      self.f3.pack()
      self.r1 = Label(self.f3,
                      text = 'USER DETAILS',
                      font = ('TlwgTypist', 12, 'bold'),
                      bg = 'white')
      self.r1.pack(fill = X)
      self.f4 = Frame(self.f3,
                      bg = 'white')
      self.f4.pack(side = LEFT, anchor = NW, pady = 40, fill = X)
      self.r2 = Label(self.f4,
                      text = 'Name:',
                      bg = 'white',
                      font = ('URWGothic', 12))
      self.r2.pack(side = TOP, anchor = NW)
      self.name = Entry(self.f4,
                      font = ('TlwgTypist', 12))
      self.name.pack(side = TOP, anchor = NW)
      self.r2 = Label(self.f4,
                      text = 'Department:',
                      bg = 'white',
                      font = ('URWGothic', 12))
      self.r2.pack(side = TOP, anchor = NW)
```

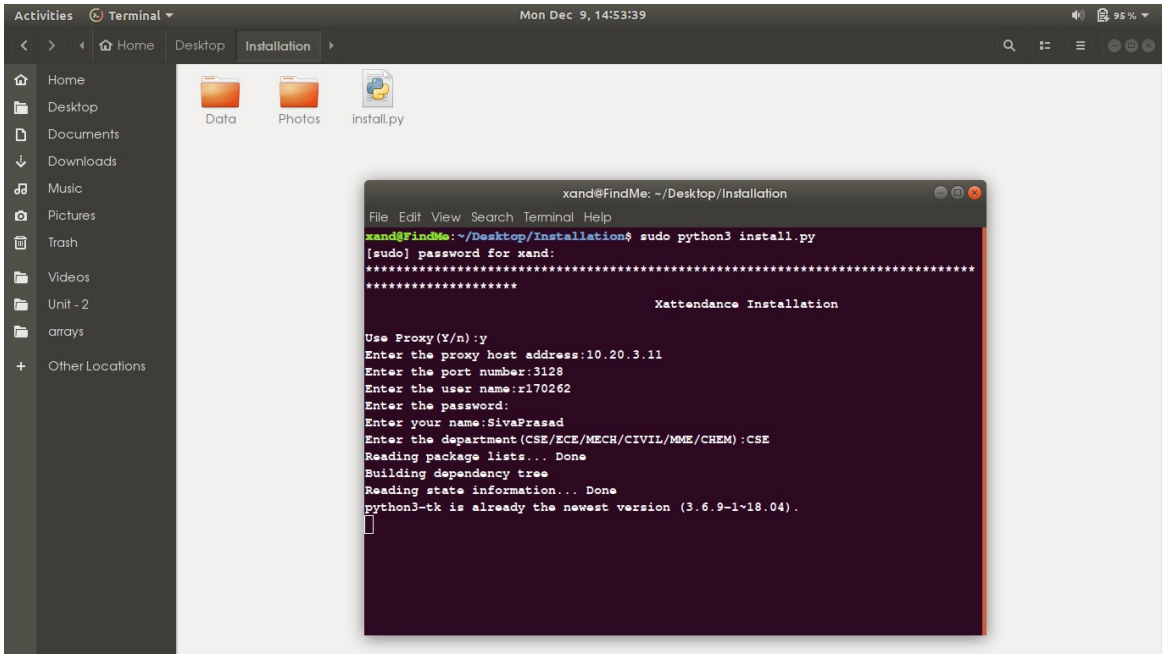
```
      self.dept = Entry(self.f4,
                        font = ('TlwgTypist', 12))
      self.dept.pack(side = TOP, anchor = NW)
      self.next['command'] = self.package_installer
      self.back['command'] = self.network_access
      def package_installer(self):
      self.next['text'] = 'Start'
      self.f3.destroy()
      self.f3 = Frame(self.root,
                        bg = 'white')
      self.f3.pack(side = LEFT,anchor = NW)
      Message(self.f3,
                  bg = 'white',
                  text = '''After this process the following packges
will be installed in your PC:
1.Python3-tk
2.Python3-numpy
3.Python3-pandas
4.Python3-pyperclip
And so 1.2 GB disk space is required to continue the installation.
Press "Start" to continue the installtion.........''',
                  font = ('URWGothic', 11, 'italic')).pack(side =
LEFT)
      self.progressbar = ttk.Progressbar(self.root)
      self.progressbar.place(relx = 0.289, rely = 0.5)
      self.progressbar['maximum'] = 100
      self.progressbar['length'] = 500
      self.progressbar['value'] = 100
      self.progress_text = srctext.ScrolledText(self.root,
                                                width = 60,
                                                height = 8)
      self.progress_text.place(relx = 0.289, rely = 0.55)
      self.next['command'] = None
      self.back['command'] = self.details_prompt
      def dependency_installation(self):
      os.system('apt install python3-tk')
      os.system('apt install python3-numpy')
      os.system('apt install python3-pandas')
      os.system('apt install python3-pyperclip')
```

```
if __name__ == '__main__':
      obj = Installer()
```

INPUT AND OUTPUT

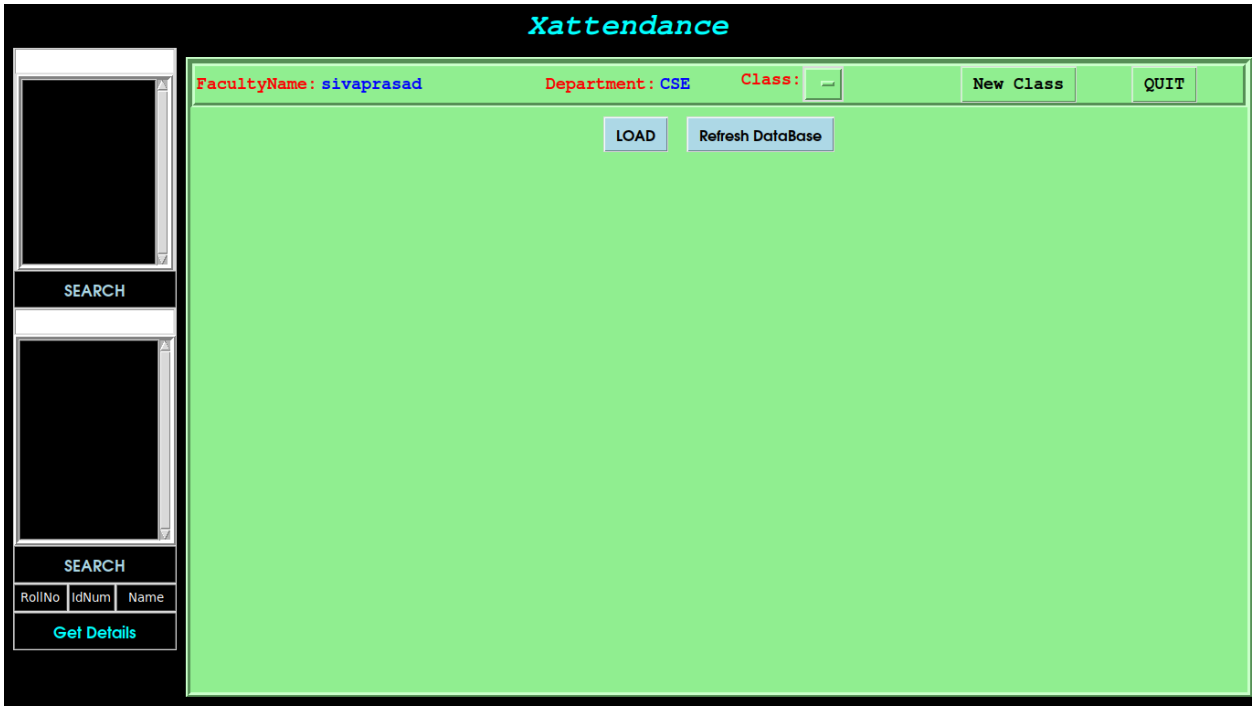


INSTALLING SOFTWARE

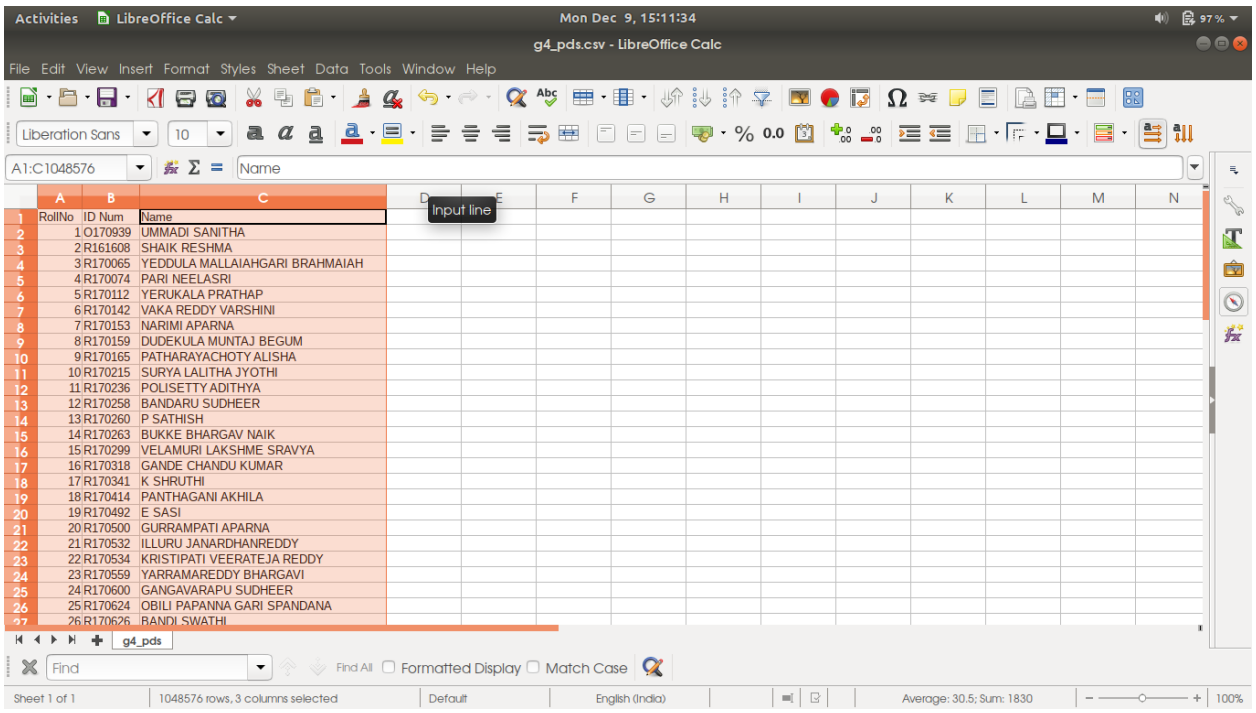


INSTALLATION PROCESS

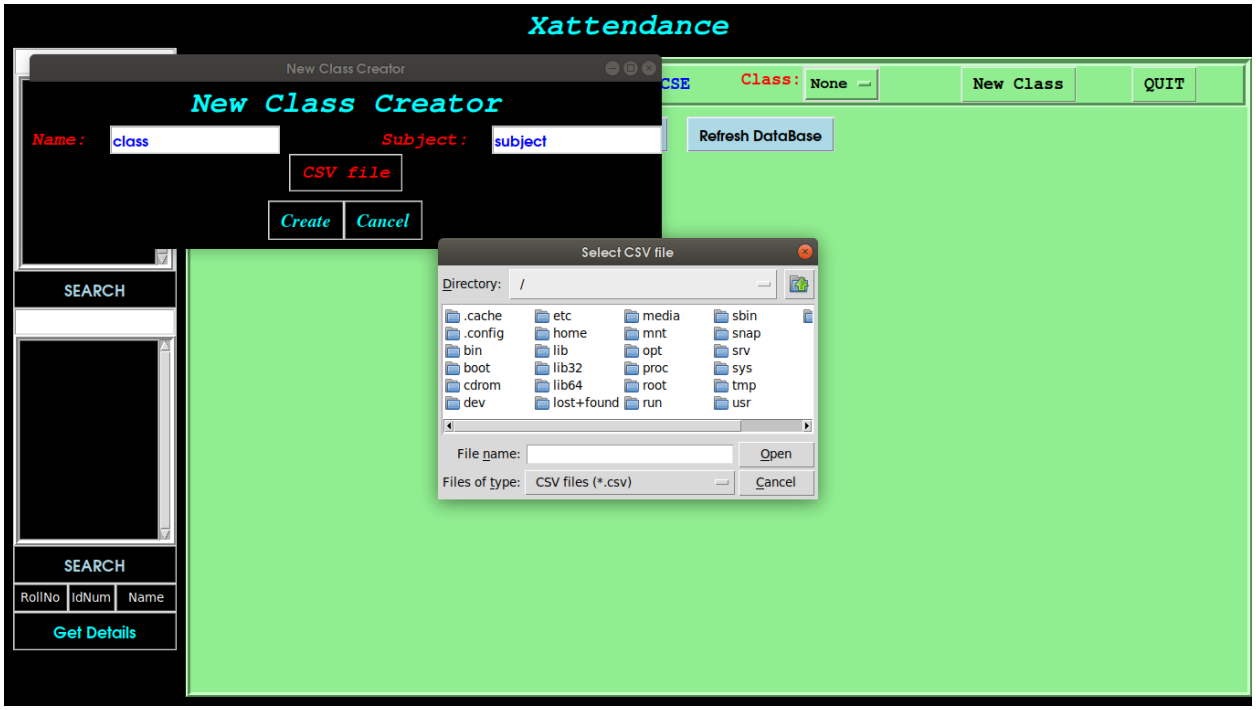
VERIFYING APP INSTALLED OR NOT IN SYSTEM



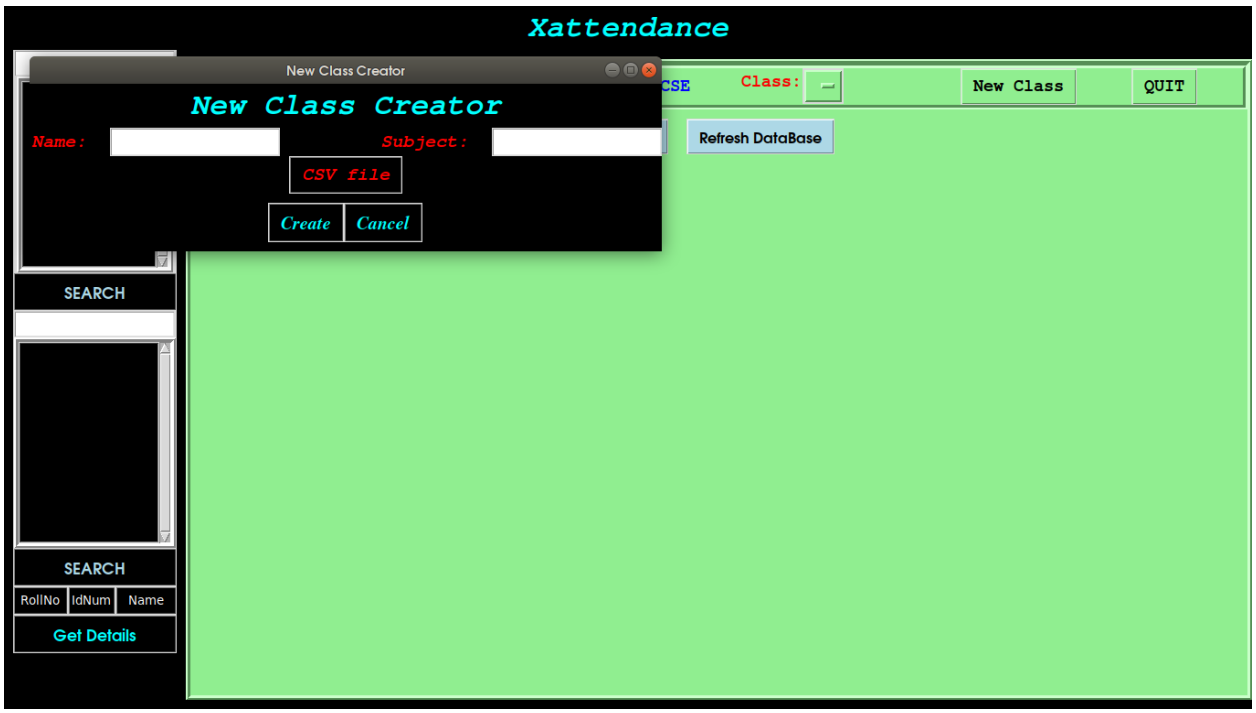
HOME SCREEN OF APP



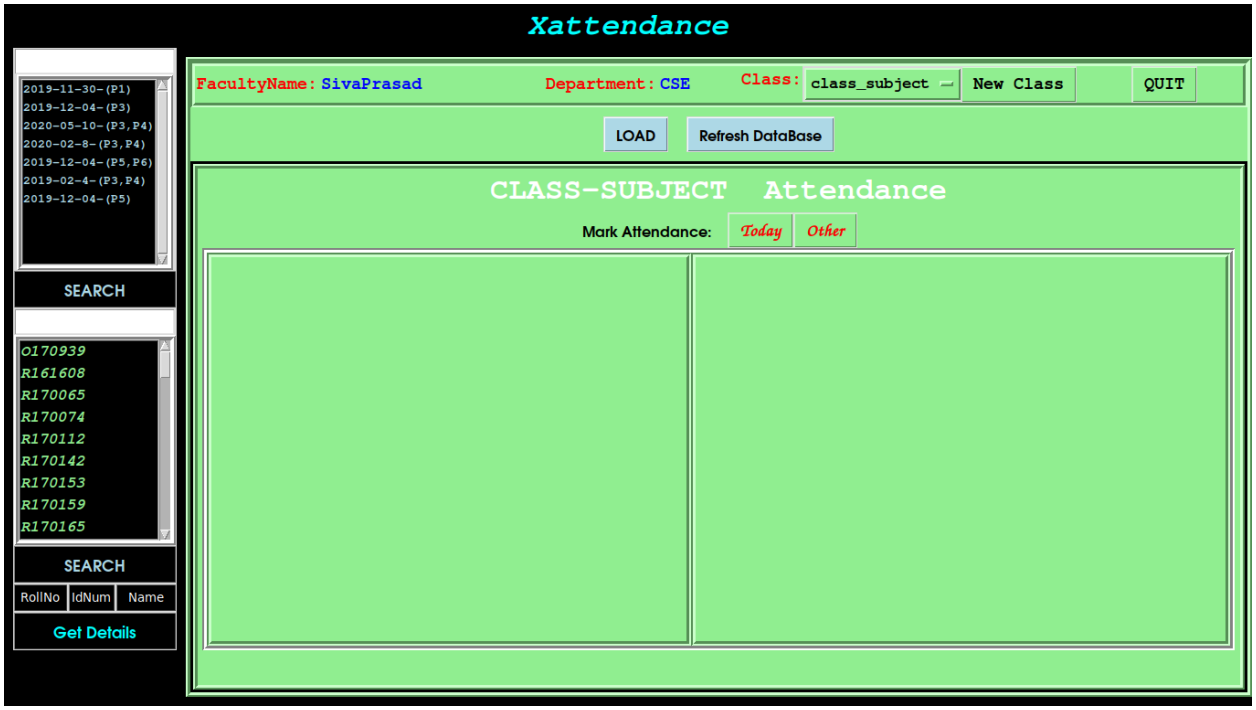
STUDENTS DETAILS SHEET



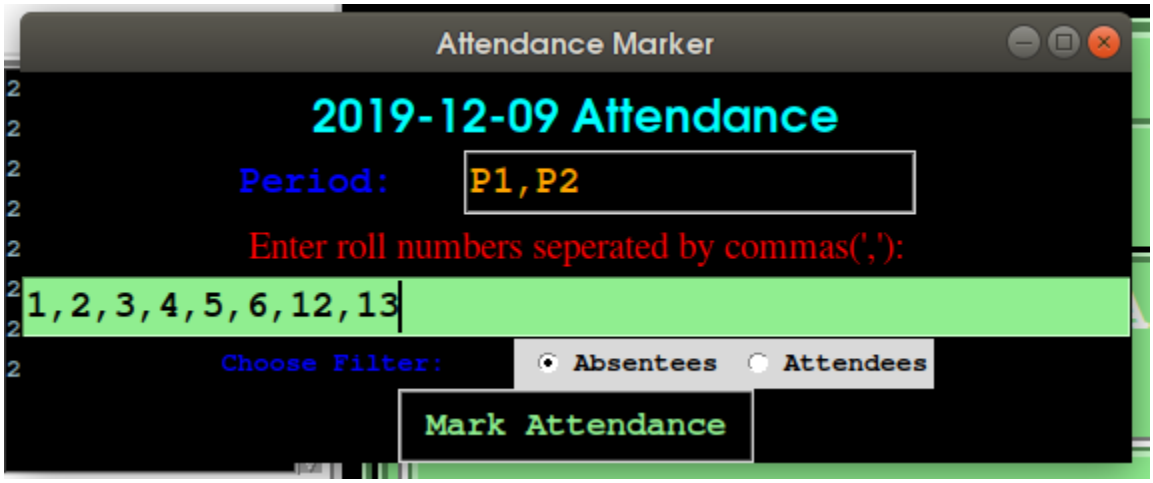
UPLOADING EXCEL SHEET TO SOFTWARE



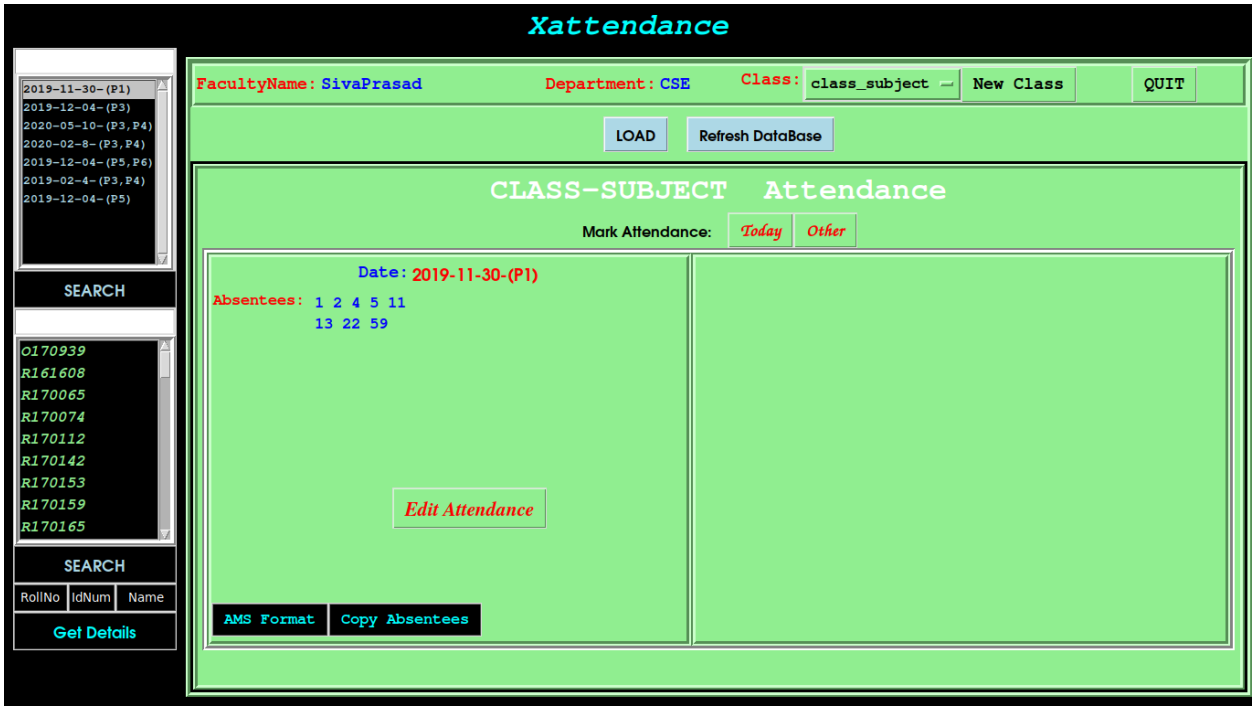
ENTER DETAILS OF CLASS



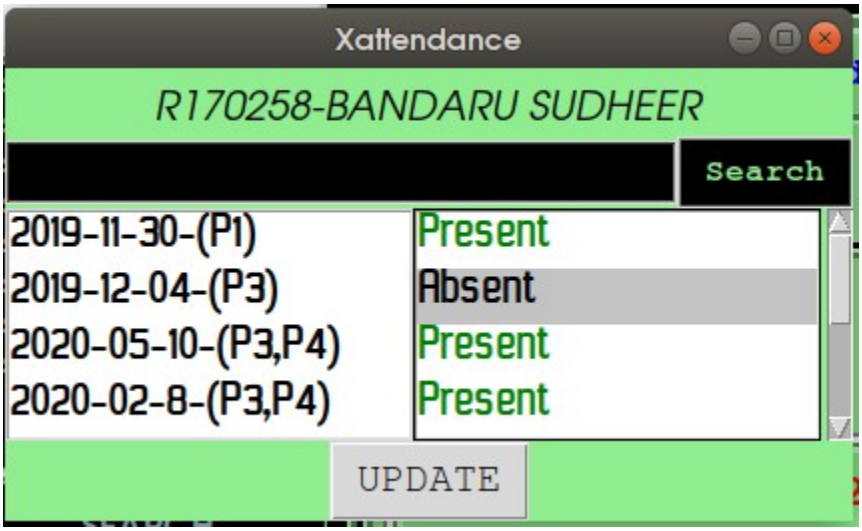
FULLY LOADED CLASS



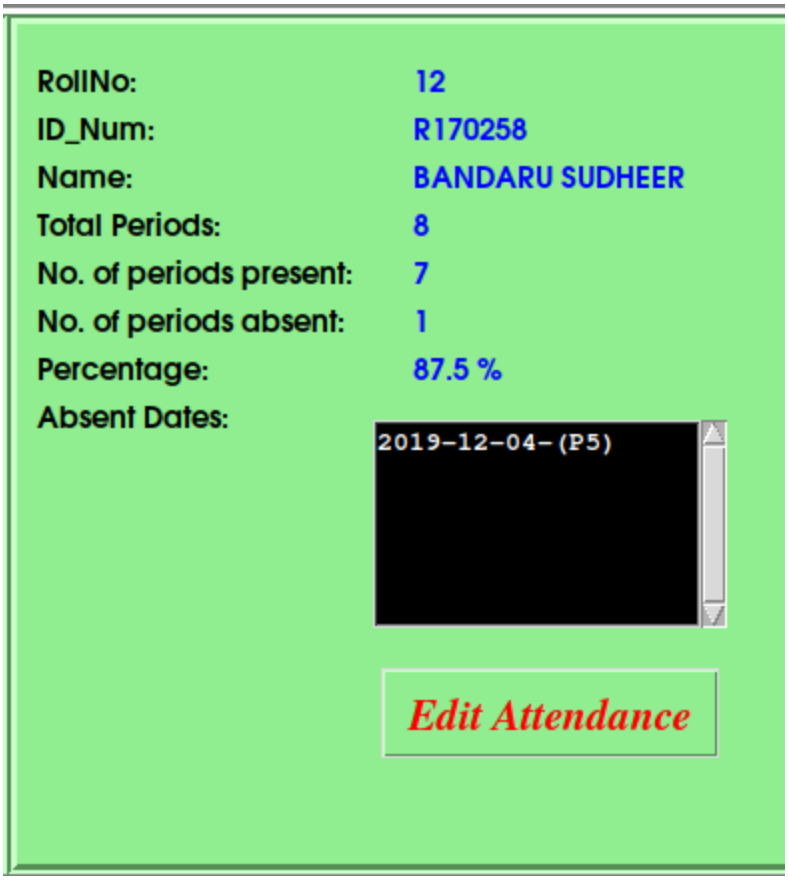
MARKING ATTENDANCE



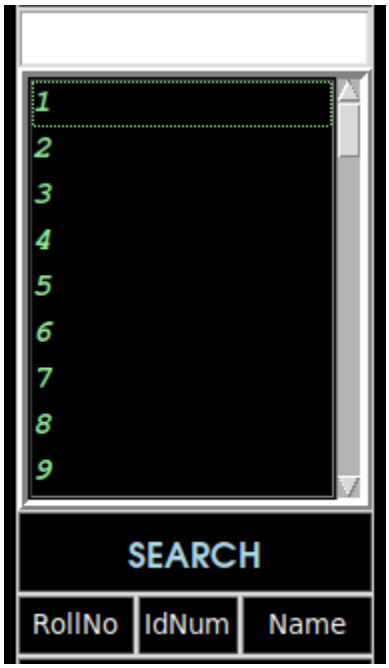
VIEWING ATTENDANCE FOR PARTICULAR DATE



EDITING ATTENDANCE



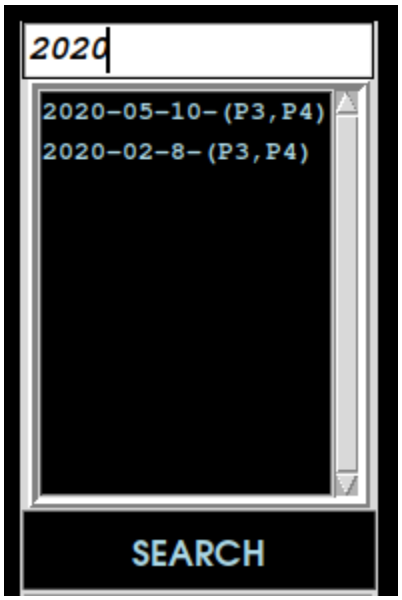
VIEWING DETAILS OF A STUDENT



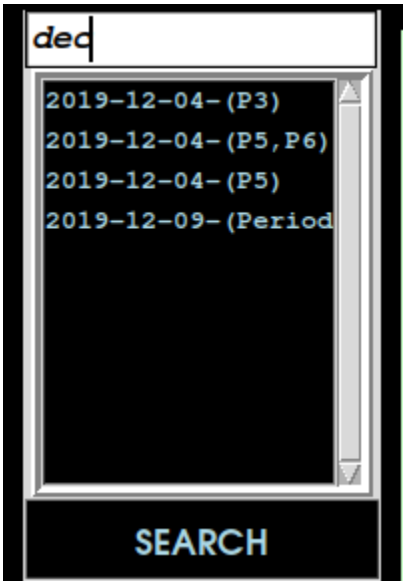
SEARCHING A STUDENT USING ROLL NO

SEARCH STUDENT USING ID NO

SEARCH STUDENT USING NAME

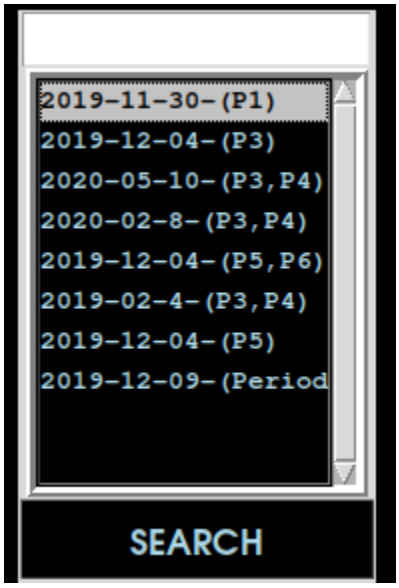


SEARCHING ATTENDACNE USING YEAR



SEARCH ATTENDANCE USING SHORT MONTH NAME

SEARCHING ATTENDACE USING FULL MONTH NAME



VIEWING ALL ATTENDANCE

CONCLUSION

Xattendance(Modern Attendance Management System) provides us one of the most optimal way to manage attendance in huge organizations like educational institutes by lowering the usage of paper and using computing power optimally to its limits.

- Store attendance in spreadsheets automatically.
- It can also generate in different formats depending upon the use cases.
- Advanced search features are implemented to make the process simple.
- Images of the students are also provided in the interface for better recognition.
- Attendance can be editied efficiently and robustly.
-

REFERENCES

For tkinter

https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.html

For pandas and numpy

https://pandas.pydata.org/docs/index.html

https://numpy.org/doc/