**System Implementations**

**Recommended System Requirements**

Processors: Intel® Core™ i3 processor 4300M at 2.60 GHz.

Disk space: 4 to 8 GB.

Operating systems: Windows® 10, MACOS, and UBUNTU.

Python Versions: 3.X.X or Higher.

**Minimum System Requirements**

Processors: Intel Atom® processor or Intel® Core™ i3 processor.

Disk space: 1 GB.

Operating systems: Windows 7 or later, MACOS, and UBUNTU.

Python Versions: 2.7.X, 3.9.X.

**ACKNOWLEDGEMENT**TTT

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

We would like to express my deep and sincere gratitude to my subject teacher, Mr. Amit Udiwal, for giving me the opportunity to do research and providing invaluable guidance throughout this research. His dynamism, vision, sincerity and motivation have deeply inspired me. He has taught me the methodology to carry out the research and to present the research works as clearly as and honour to work and study under his guidance. We are very much thankful to our Sr. Jasmin for giving valuable time and moral support to develop this software. We would like to take opportunity to extend my sincere thanks and gratitude to our parents for being a source of inspiration and providing time and freedom to develop this software project. We also feel indebted to my friends for the valuable suggestions during the project work.

Swastik Kasera

[Roll No.

Class XII

**CERTIFICATE**

This is to certify that the project on ‘School Management System’ is a work done by Shreyansh Jain fulfilment of CBSE’S AISSCE EXAMINATION 2022 and has been carried out under my direct supervision and guidance. This report or a similar report on the topic has not been submitted for any other examination and does not form any other examination and does not form any other course undergone by the candidate.

Name:

Shreyansh Jain[Roll No.

………………….

Signature of Teacher / Guide

Name: Mr. Amit Udiwal

Designation:

………………. ….………………

**REFERENCE**

The order to work on this project on ‘Institute Management System’ the following books & literature are referred by me during the various phrases of department of the project.

• http://www.python.org/.

• http://www.itsourcecode.org/.

• http://www.wikipedia.org/.

• Informatics Practices for Class XII

- By Sumita Arora

• Together with informatics practices.

Other than the above mentioned books, the suggestions and supervision of my teacher and my class experience also helped me to develop this software project.

**Introduction**

The Institute Management System In Python is a simple project developed using Python. The project contains only the admin side. It is an integrated console-based application that handles various academic activities of a College. It is a mini python project that is used to keep and maintain the complete record of Students.

The main advantage of Institute management system is the time that professors and other faculty can save by properly implementing it on campus. Attendance tracking, fee management, automated messages to parents and students, and more modules are accessible for everyday use.

**Objective and**

**Scope of The Project**

The main objective of the Institute Management System is to manage the details of Institute, Faculty, Course, Batch, Session.

It manages all the information about College, Student, Session, Institute. The project is totally built at administrative end and thus only the administrator is guaranteed the access.

This project deals with the various functioning in College management process. The main idea is to implement a proper process to system .

In our existing system contains a many operations registration, student search, fees, attendance, exam records, performance of the student etc.

**Institute**

**Management System**

import datetime

from tkinter import \*

import tkinter.messagebox as mb

from tkinter import ttk

from tkcalendar import DateEntry # pip install tkcalendar

import sqlite3

# Creating the universal font variables

headlabelfont = ("Noto Sans CJK TC", 15, 'bold')

labelfont = ('Garamond', 14)

entryfont = ('Garamond', 12)

# Connecting to the Database where all information will be stored

connector = sqlite3.connect('InstituteManagement.db')

cursor = connector.cursor()

connector.execute(

"CREATE TABLE IF NOT EXISTS Institute\_MANAGEMENT (Institute\_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, NAME TEXT, EMAIL TEXT, PHONE\_NO TEXT, GENDER TEXT, DOB TEXT, STREAM TEXT)"

)

# Creating the functions

def reset\_fields():

global name\_strvar, email\_strvar, contact\_strvar, gender\_strvar, dob, stream\_strvar

for i in ['name\_strvar', 'email\_strvar', 'contact\_strvar', 'gender\_strvar', 'stream\_strvar']:

exec(f"{i}.set('')")

dob.set\_date(datetime.datetime.now().date())

def reset\_form():

global tree

tree.delete(\*tree.get\_children())

reset\_fields()

def display\_records():

tree.delete(\*tree.get\_children())

curr = connector.execute('SELECT \* FROM Institute\_MANAGEMENT')

data = curr.fetchall()

for records in data:

tree.insert('', END, values=records)

def add\_record():

global name\_strvar, email\_strvar, contact\_strvar, gender\_strvar, dob, stream\_strvar

name = name\_strvar.get()

email = email\_strvar.get()

contact = contact\_strvar.get()

gender = gender\_strvar.get()

DOB = dob.get\_date()

stream = stream\_strvar.get()

if not name or not email or not contact or not gender or not DOB or not stream:

mb.showerror('Error!', "Please fill all the missing fields!!")

else:

try:

connector.execute(

'INSERT INTO Institute\_MANAGEMENT (NAME, EMAIL, PHONE\_NO, GENDER, DOB, STREAM) VALUES (?,?,?,?,?,?)', (name, email, contact, gender, DOB, stream)

)

connector.commit()

mb.showinfo('Record added', f"Record of {name} was successfully added")

reset\_fields()

display\_records()

except:

mb.showerror('Wrong type', 'The type of the values entered is not accurate. Pls note that the contact field can only contain numbers')

def remove\_record():

if not tree.selection():

mb.showerror('Error!', 'Please select an item from the database')

else:

current\_item = tree.focus()

values = tree.item(current\_item)

selection = values["values"]

tree.delete(current\_item)

connector.execute('DELETE FROM Institute\_MANAGEMENT WHERE Institute\_ID=%d' % selection[0])

connector.commit()

mb.showinfo('Done', 'The record you wanted deleted was successfully deleted.')

display\_records()

def view\_record():

global name\_strvar, email\_strvar, contact\_strvar, gender\_strvar, dob, stream\_strvar

if not tree.selection():

mb.showerror('Error!', 'Please select a record to view')

else:

current\_item = tree.focus()

values = tree.item(current\_item)

selection = values["values"]

name\_strvar.set(selection[1]); email\_strvar.set(selection[2])

contact\_strvar.set(selection[3]); gender\_strvar.set(selection[4])

date = datetime.date(int(selection[5][:4]), int(selection[5][5:7]), int(selection[5][8:]))

dob.set\_date(date);stream\_strvar.set(selection[6])

# Initializing the GUI window

main = Tk()

main.title('DataFlair Institute Management System')

main.geometry('1000x600')

main.resizable(0, 0)

# Creating the background and foreground color variables

lf\_bg = 'MediumSpringGreen' # bg color for the left\_frame

cf\_bg = 'PaleGreen' # bg color for the center\_frame

# Creating the StringVar or IntVar variables

name\_strvar = StringVar()

email\_strvar = StringVar()

contact\_strvar = StringVar()

gender\_strvar = StringVar()

stream\_strvar = StringVar()

# Placing the components in the main window

Label(main, text="Institute MANAGEMENT SYSTEM", font=headlabelfont, bg='SpringGreen').pack(side=TOP, fill=X)

left\_frame = Frame(main, bg=lf\_bg)

left\_frame.place(x=0, y=30, relheight=1, relwidth=0.2)

center\_frame = Frame(main, bg=cf\_bg)

center\_frame.place(relx=0.2, y=30, relheight=1, relwidth=0.2)

right\_frame = Frame(main, bg="Gray35")

right\_frame.place(relx=0.4, y=30, relheight=1, relwidth=0.6)

# Placing components in the left frame

Label(left\_frame, text="Name", font=labelfont, bg=lf\_bg).place(relx=0.375, rely=0.05)

Label(left\_frame, text="Contact Number", font=labelfont, bg=lf\_bg).place(relx=0.175, rely=0.18)

Label(left\_frame, text="Email Address", font=labelfont, bg=lf\_bg).place(relx=0.2, rely=0.31)

Label(left\_frame, text="Gender", font=labelfont, bg=lf\_bg).place(relx=0.3, rely=0.44)

Label(left\_frame, text="Date of Birth (DOB)", font=labelfont, bg=lf\_bg).place(relx=0.1, rely=0.57)

Label(left\_frame, text="Stream", font=labelfont, bg=lf\_bg).place(relx=0.3, rely=0.7)

Entry(left\_frame, width=19, textvariable=name\_strvar, font=entryfont).place(x=20, rely=0.1)

Entry(left\_frame, width=19, textvariable=contact\_strvar, font=entryfont).place(x=20, rely=0.23)

Entry(left\_frame, width=19, textvariable=email\_strvar, font=entryfont).place(x=20, rely=0.36)

Entry(left\_frame, width=19, textvariable=stream\_strvar, font=entryfont).place(x=20, rely=0.75)

OptionMenu(left\_frame, gender\_strvar, 'Male', "Female").place(x=45, rely=0.49, relwidth=0.5)

dob = DateEntry(left\_frame, font=("Arial", 12), width=15)

dob.place(x=20, rely=0.62)

Button(left\_frame, text='Submit and Add Record', font=labelfont, command=add\_record, width=18).place(relx=0.025, rely=0.85)

# Placing components in the center frame

Button(center\_frame, text='Delete Record', font=labelfont, command=remove\_record, width=15).place(relx=0.1, rely=0.25)

Button(center\_frame, text='View Record', font=labelfont, command=view\_record, width=15).place(relx=0.1, rely=0.35)

Button(center\_frame, text='Reset Fields', font=labelfont, command=reset\_fields, width=15).place(relx=0.1, rely=0.45)

Button(center\_frame, text='Delete database', font=labelfont, command=reset\_form, width=15).place(relx=0.1, rely=0.55)

# Placing components in the right frame

Label(right\_frame, text='Students Records', font=headlabelfont, bg='DarkGreen', fg='LightCyan').pack(side=TOP, fill=X)

tree = ttk.Treeview(right\_frame, height=100, selectmode=BROWSE,

columns=('Student ID', "Name", "Email Address", "Contact Number", "Gender", "Date of Birth", "Stream"))

X\_scroller = Scrollbar(tree, orient=HORIZONTAL, command=tree.xview)

Y\_scroller = Scrollbar(tree, orient=VERTICAL, command=tree.yview)

X\_scroller.pack(side=BOTTOM, fill=X)

Y\_scroller.pack(side=RIGHT, fill=Y)

tree.config(yscrollcommand=Y\_scroller.set, xscrollcommand=X\_scroller.set)

tree.heading('Student ID', text='ID', anchor=CENTER)

tree.heading('Name', text='Name', anchor=CENTER)

tree.heading('Email Address', text='Email ID', anchor=CENTER)

tree.heading('Contact Number', text='Phone No', anchor=CENTER)

tree.heading('Gender', text='Gender', anchor=CENTER)

tree.heading('Date of Birth', text='DOB', anchor=CENTER)

tree.heading('Stream', text='Stream', anchor=CENTER)

tree.column('#0', width=0, stretch=NO)

tree.column('#1', width=40, stretch=NO)

tree.column('#2', width=140, stretch=NO)

tree.column('#3', width=200, stretch=NO)

tree.column('#4', width=80, stretch=NO)

tree.column('#5', width=80, stretch=NO)

tree.column('#6', width=80, stretch=NO)

tree.column('#7', width=150, stretch=NO)

tree.place(y=30, relwidth=1, relheight=0.9, relx=0)

display\_records()

# Finalizing the GUI window

main.update()

main.mainloop()