**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.

Ahmad Nagori

[Roll No.

Class XII

**CERTIFICATE**

This is to certify that the project on ‘Account Management System’ is a work done by Ahmad Nagori fulfilment of CBSE’S AISSCE EXAMINATION 2020¢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:

Ahmad Nagori [Roll No.

………………….

Signature of Teacher / Guide

Name: Mr. Amit Udiwal

Designation:

………………. ….………………

**REFERENCE**

The order to work on this project on ‘Account 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**

Account Management System project is written in Python. The project file contains a database file and python script. This is a simple GUI based project which is very easy to understand and use. Talking about the system, it contains all the required functions which include adding, viewing, deleting, updating and searching user account lists. While adding the account details of a person, he/she has to provide name, username/email, password, category, and date. The user can also update the account list if he/she wants to. For this, the user has to view all records, click on a record that he/she wishes to edit and press the update button. The system shows the account details in a list view. And also the user easily delete any account details.

**Objective and**

**Scope of The Project**

The [Account Management System](https://code-projects.org/account-management-system-in-python-with-source-code/) is created using python programming language. Account Management System file contains a python script.

Talking approximately the system, it includes all of the required features which consist of adding, viewing, deleting, updating, searching and looking a person account lists. While including the account information of a person, he/she has to offer first name, last name, username, password, position, and date.

***Functions:***

* Add accounts
* List accounts
* Update accounts
* Delete accounts
* Search accounts

**Account Management System**

from tkinter import \*

from tkinter import messagebox

import database\_account

window = Tk()

window.title("Account Management System")

window.config(bg="navajo white")

window.geometry('1350x750')

def view():

lb.delete(0,END)

for row in database\_account.viewall():

lb.insert(END, row)

def search():

lb.delete(0, END)

for row in database\_account.search(firstname=firstname.get(), lastname=lastname.get(), username=username.get(), password=password.get(), position=position.get()):

lb.insert(END, row)

def add():

database\_account.add(firstname.get(), lastname.get(), username.get(), password.get(), position.get(), date.get())

messagebox.showinfo("Add", "New Account Added Successfully")

lb.delete(0, END)

lb.insert(END, firstname.get(), lastname.get(), username.get(), password.get(), position.get(), date.get())

def get\_selected\_row(event):

try:

global selected\_tuple

index = lb.curselection()[0]

selected\_tuple = lb.get(index)

EntryFirstName.delete(0,END)

EntryFirstName.insert(END,selected\_tuple[1])

EntryLastName.delete(0,END)

EntryLastName.insert(END,selected\_tuple[2])

EntryUsername.delete(0,END)

EntryUsername.insert(END,selected\_tuple[3])

EntryPassword.delete(0,END)

EntryPassword.insert(END,selected\_tuple[4])

EntryPosition.delete(0,END)

EntryPosition.insert(END,selected\_tuple[5])

EntryDate.delete(0, END)

EntryDate.insert(END, selected\_tuple[6])

except IndexError:

pass

def update():

database\_account.update(selected\_tuple[0], firstname.get(), lastname.get(), username.get(), password.get(), position.get(), date.get())

messagebox.showinfo("Update", "Account Has Been Updated Successfully")

view()

def delete():

database\_account.delete(selected\_tuple[0])

messagebox.showinfo("Delete Account", 'Account Has Been Deleted Successfully')

view()

#lb.delete(END,get\_selected\_row.selected\_tuple)

def clear():

lb.delete(0,END)

EntryFirstName.delete(0,END)

EntryLastName.delete(0,END)

EntryUsername.delete(0,END)

EntryPassword.delete(0,END)

EntryPosition.delete(0,END)

EntryDate.delete(0, END)

lblfirstname = Label(window, text="First Name", font=("Calibri", 14, "bold"), fg="black", bg="navajo white")

lblfirstname.grid(row=0, column=0, columnspan=2)

lbllastname = Label(window,text="Last Name", font=("Calibri", 14, "bold"), fg="black", bg="navajo white")

lbllastname.grid(row=1,column=0,columnspan=2)

lblusername = Label(window,text="Username", font=("Calibri", 14, "bold"), fg="black", bg="navajo white")

lblusername.grid(row=2,column=0,columnspan=2)

lblpassword = Label(window, text="Password", font=("Calibri", 14, "bold"), fg="black", bg="navajo white")

lblpassword.grid(row=3, column=0, columnspan=2)

lblposition = Label(window,text="Position", font=("Calibri", 14, "bold"), fg="black", bg="navajo white")

lblposition.grid(row=4,column=0,columnspan=2)

lbldate = Label(window,text="Date", font=("Calibri", 14, "bold"), fg="black", bg="navajo white")

lbldate.grid(row=5,column=0,columnspan=2)

firstname=StringVar()

EntryFirstName = Entry(window,textvariable=firstname, font=("Calibri", 14, "italic"), width=30)

EntryFirstName.grid(row=0,column=0,columnspan=10)

lastname=StringVar()

EntryLastName = Entry(window,textvariable=lastname, font=("Calibri", 14, "italic"), width=30)

EntryLastName.grid(row=1,column=0,columnspan=10)

username=StringVar()

EntryUsername = Entry(window,textvariable=username, font=("Calibri", 14, "italic"), width=30)

EntryUsername.grid(row=2,column=0,columnspan=10)

password=StringVar()

EntryPassword = Entry(window,textvariable=password, font=("Calibri", 14, "italic"), width=30)

EntryPassword.grid(row=3,column=0,columnspan=10)

position=StringVar()

EntryPosition = Entry(window,textvariable=position, font=("Calibri", 14, "italic"), width=30)

EntryPosition.grid(row=4,column=0,columnspan=10)

date = StringVar()

EntryDate = Entry(window,textvariable=date, font=("Calibri", 14, "italic"), width=30)

EntryDate.grid(row=5,column=0,columnspan=10)

AddButton = Button(window,text="Add",width=12,command=add, font=("Calibri", 10, "italic"), fg="black", bg="navajo white", relief=RIDGE, bd=10)

AddButton.grid(row=6,column=0)

UpdateButton = Button(window,text="Update",width=12,command=update, font=("Calibri", 10, "italic"), fg="black", bg="navajo white", relief=RIDGE, bd=10)

UpdateButton.grid(row=6,column=1)

SearchButton = Button(window,text="Search",width=12,command=search, font=("Calibri", 10, "italic"), fg="black", bg="navajo white", relief=RIDGE, bd=10)

SearchButton.grid(row=6,column=2)

ViewAllButton = Button(window,text="View All",width=12,command=view, font=("Calibri", 10, "italic"), fg="black", bg="navajo white", relief=RIDGE, bd=10)

ViewAllButton.grid(row=6,column=3)

DeleteButton = Button(window,text="Delete",width=12,command=delete, font=("Calibri", 10, "italic"), fg="black", bg="navajo white", relief=RIDGE, bd=10)

DeleteButton.grid(row=6,column=4)

ClearAllButton = Button(window,text="Clear All",width=12,command=clear, font=("Calibri", 10, "italic"), fg="black", bg="navajo white", relief=RIDGE, bd=10)

ClearAllButton.grid(row=6,column=5)

lb=Listbox(window,height=20,width=94)

lb.grid(row=7,column=0,columnspan=6)

sb=Scrollbar(window)

sb.grid(row=7,column=6,rowspan=6)

lb.configure(yscrollcommand=sb.set)

sb.configure(command=lb.yview)

lb.bind('<<ListboxSelect>>',get\_selected\_row)

window.mainloop()