# Main page setting

image = Image.open('autoaudit\_t.png')

st.set\_page\_config(page\_title="AutoAudit", page\_icon=":white\_check\_mark:", layout="wide")

#st.title(":white\_check\_mark: AutoAudit")

st.image(image,width=250)

st.set\_page\_config(

    page\_title="CRUD",

    page\_icon="🧊",

    menu\_items={

        'Get Help': 'https://www.extremelycoolapp.com/help',

        'Report a bug': "https://www.extremelycoolapp.com/bug",

        'About': "# This is a header. This is an \*extremely\* cool app!"

    },layout="wide"

)

### convert text to date

datetime.strptime(exp\_date, '%Y-%m-%d').date()

## add minutes in datetime

from datetime import datetime, timedelta

now = datetime.now()

now\_plus\_10m = now + datetime.timedelta(minutes = 10)

## check email in format

import re

regex = r'\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,7}\b'

if(re.fullmatch(regex, userName))

# Look& Feel

## insert line for line

st.markdown("""---""")

## CSS

### Change Label of Text Input

st.markdown(".stTextInput > label {font-size:105%; font-weight:bold; color:blue;} ",unsafe\_allow\_html=True) #for all text-input label sections

### Change Label of multi-select

st.markdown(".stMultiSelect > label {font-size:105%; font-weight:bold; color:blue;} ",unsafe\_allow\_html=True) #for all multi-select label sections

div.stSelectbox

### Change Label of Radio

div[class\*="stRadio"] > label > div[data-testid="stMarkdownContainer"] > p {

font-size: 32px;

## Error/ other messages

st.toast(f"Error:-{addrecord}", icon="👎")

st.toast("Record Added Successfully...Continue to Add more", icon="👍")

# Secretes

Create folder in root- .stremli & in thar creat file secrets.toml file

In secrets.TOML file

*# Everything in this section will be available as an environment variable*

db\_username = "Jane"

db\_password = "12345qwerty"

*# You can also add other sections if you like.*

*# The contents of sections as shown below will not become environment*

*# variables, but they'll be easily accessible from within Streamlit anyway*

*# as we show later in this doc.*

[my\_cool\_secrets]

things\_i\_like = ["Streamlit", "Python"]

Access your secrets as environment variables or by querying the st.secrets dict.

import streamlit as st

import os

*# Everything is accessible via the st.secrets dict:*

st.write("DB username:", st.secrets["db\_username"])

st.write("DB password:", st.secrets["db\_password"])

st.write("My cool secrets:", st.secrets["my\_cool\_secrets"]["things\_i\_like"])

*# And the root-level secrets are also accessible as environment variables:*

st.write(

"Has environment variables been set:",

os.environ["db\_username"] == st.secrets["db\_username"],

)

# Create Containers

headerSection = st.container()

mainSection = st.container()

loginSection = st.container()

createcompanysection=st.container()

logOutSection = st.container()

# session state

## change disable / enable

if button\_a:

st.session\_state.disabled = False

if button\_b:

st.session\_state.disabled = True

button\_c = st.button('c', key='but\_c', disabled=st.session\_state.disabled)

# Add records

Use Tabs instead of Menu , that will maintain flow:

## mark text input as Mandatory \* red

names = st.text\_input(f"Enter your name below: :red[\*]", key="namest",)

## Check all Required fields are entered before saving

placeholder = st.empty()

    def newlice(name,Expiry\_Date,time\_zone,email):

        #1st check email

        if(re.fullmatch(regex, email)):

             clearfields=["tnamelic","licemail"]

             allfields= False

             for i in clearfields:

                  if len(st.session\_state[f'{i}'])<1:

                       allfields=True

             #if (len(names)<1 or len(names1)<1 or len(names2)<1):

             if allfields:

                st.toast("Enter All Manadtory Fields \*")

             else:

                #now add record

                addrecord= add\_new\_license(name,Expiry\_Date,time\_zone,email)

                #placeholder.empty()

                if addrecord==True:

                    st.toast("Record Added Successfully...Continue to Add more", icon="👍")

                else:

                    st.toast(f"Error:-{addrecord}", icon="👎")

                    st.toast("Try Again")

                #clear text fields on form

                for i in clearfields:

                        #st.write(st.session\_state[f'{i}'])

                        st.session\_state[f'{i}']=""

        else:

            #st.toast(email)

            st.toast("email not in Proper format...")

    with placeholder.container(border=True):

        st.subheader("Add New License")

        name=st.text\_input(f"Enter Name :red[\*]",key="tnamelic")

        Expiry\_Date=st.date\_input("Set Expiry Date",value=today,min\_value=today,key="dexpdate")

        time\_zone=st.number\_input("Enter Time Difference",min\_value=-720,max\_value=840,value=0,key="ntime")

        email=st.text\_input(f"email :red[\*]",key="licemail")

        st.button("Submit",on\_click=newlice,

                                args=[st.session\_state.tnamelic,st.session\_state.dexpdate,st.session\_state.ntime,st.session\_state.licemail])

## Clear form after submitting

See above example

## Show Selected rows of Dataframe

def dataframe\_with\_selections(df):

        df\_with\_selections = df.copy()

        df\_with\_selections.insert(0, "Select", False)

        # Get dataframe row-selections from user with st.data\_editor

        edited\_df = st.data\_editor(

            df\_with\_selections,

            hide\_index=True,

            column\_config={"Select": st.column\_config.CheckboxColumn(required=True)},

            disabled=df.columns,

        )

        # Filter the dataframe using the temporary column, then drop the column

        selected\_rows = edited\_df[edited\_df.Select]

        return selected\_rows.drop('Select', axis=1)

    selection = dataframe\_with\_selections(df)

    st.write(selection)

<https://discuss.streamlit.io/t/how-to-delete-several-widgets-programmatically-by-clicking-button/45453/2>

# Add / Del rows widget

cont = st.empty()

    #for enabling  button default true

    if 'b1\_enable' not in st.session\_state:

            st.session\_state.b1\_enable=True

            #st.write(st.session\_state[f'delet\_b{k}'])

     #for total check

    if "b1\_total" not in st.session\_state:

            st.session\_state.b1\_total=0

    with cont.container(border=True):

        def decrease\_rows(k):

            #st.session\_state['rows'] -= 1

            del st.session\_state[f'first\_{k}']

            del st.session\_state[f'middle\_{k}']

            del st.session\_state[f'last\_{k}']

            del st.session\_state[f'delet\_b{k}']

            st.session\_state['rows'].remove(k)

            #st.success(st.session\_state['rows'])

            #if no rows set total to 0

            if len(st.session\_state['rows'])==0:

                 st.session\_state.b1\_total=0

        left, middle, right ,delet\_b= st.columns(4)

        left.success("Account")

        middle.success("Amount")

        right.success("Dr/CR")

        delet\_b.error("Remove")

        if 'rows' not in st.session\_state:

            st.session\_state['rows'] = [0]

        def display\_input\_row(index):

                left, middle, right ,delet\_b= st.columns(4)

                left.selectbox('',("A","B","C"), key=f'first\_{index}',label\_visibility="collapsed",)

                middle.number\_input('',step=1.00, key=f'middle\_{index}',label\_visibility="collapsed")

                right.selectbox('',options=["Dr","Cr"], key=f'last\_{index}',label\_visibility="collapsed")

                delet\_b.button(":red[\*\*X\*\*]",key=f'delet\_b{index}',on\_click=decrease\_rows,args=(index,))

        def increase\_rows():

            if len(st.session\_state['rows'])>0:

                nrow=1+max(st.session\_state['rows'])

            else:

                 nrow=0

            st.session\_state['rows'].append(nrow)

            #checktotal()

        #list all row

        for i in st.session\_state['rows']:

            display\_input\_row(i)

        #check if all form is proper then only Enable button by changing session state

        total=0

        for i in st.session\_state['rows']:

            total=total+st.session\_state[f'middle\_{i}']

            #st.write(st.session\_state[f'middle\_{i}'])

            #st.write(total)

        st.session\_state.b1\_total=total

        #enable Button

        if st.session\_state.b1\_total==0:

            st.session\_state.b1\_enable=False

            st.write(f':blue[Total: {st.session\_state.b1\_total}]')

        else:

            st.session\_state.b1\_enable=True

            st.write(f':red[Total must be 0. Current Total is: {st.session\_state.b1\_total}]')

        st.button('Add person', on\_click=increase\_rows,key='bincre',disabled=st.session\_state.b1\_enable)

    # Show the results

    st.subheader('Data')

    for i in st.session\_state['rows']:

        st.write(

            f'Record {i+1}:',

            st.session\_state[f'first\_{i}'],

            st.session\_state[f'middle\_{i}'],

            st.session\_state[f'last\_{i}'],

            st.session\_state.b1\_total,

            st.session\_state.b1\_enable

        )