



**JSC “Kazakh British Technical University”  
School of Mathematic and Cybernetics**

Analysis of Data Bases

**Laboratory Work #12  
(Python)**

**Prepared by: Maratuly Temirbolat**

**Almaty 2021**

## Short Description:

The main task of this Laboratory work is to investigate the interface with the connection of Database and opportunities to do defined activities for determined users.

The Laboratory work was done using OOP(Object Oriented Programming) principle. It means that we used classes (University System, Admin including User), methods (check password, create user, find user and so on) along with corresponded attributes (number id, name, surname, login, password etc.)

The program has to distinguish users and admins providing appropriate list of all possible activities respectively.

There must be appeared Errors if something was typed wrong (login of the user, password etc.) or requirements for the password or user may not exist.

## The code of the Classes:

```
class universitySystem:
    allTheUsers = []
    def __init__(self):
        self.allTheUsers = [Admin('1', 'Temirbolat', 'Maratuly', 't_maratuly', 'Timka2019!'), User('2', 'Assanali', 'Moldash', 'a_moldash', 'Timka2019!')]
    def checkPassword(self, newPassword):
        requiredCharacters = ['.', ',', ':', ';', '?', '!', '*', '+', '%', '-', '<', '>', '@', '[', ']', '{', '}', '/', '_', '$', '#']
        leghtPassword = False
        latinAlphabet = False
        arabicNumbers = False
        lowerCase = False
        upperCase = False
        isReqCharacter = False
        if (len(newPassword) >= 8 and len(newPassword) <= 14):
            leghtPassword = True
        for k in range(len(newPassword)):
            if ((ord(newPassword[k]) >= 65 and ord(newPassword[k]) <= 90 or (ord(newPassword[k]) >= 97 and ord(newPassword[k]) <= 122)):
                latinAlphabet = True
            elif (newPassword[k] in requiredCharacters):
                isReqCharacter = True
        for k in range(len(newPassword)):
            if (ord(newPassword[k]) >= 48 and ord(newPassword[k]) <= 57):
                arabicNumbers = True
            elif (newPassword[k].isupper() == True):
                upperCase = True
            elif (newPassword[k].islower() == True):
                lowerCase = True
        print(leghtPassword)
        print(latinAlphabet)
        print(arabicNumbers)
        print(lowerCase)
        print(upperCase)
        print(isReqCharacter)
        if (leghtPassword == True and latinAlphabet == True and arabicNumbers == True and lowerCase == True and upperCase == True and isReqCharacter == True):
            return True
        return False
    def checkLogin(self, login):
        isThere = False
        for user in self.allTheUsers:
            if (user.getLogin() == login):
                isThere = True
        print('Is There:', isThere)
        return isThere
    def createUser(self, university, numberId, name, surname, login, password):
        if (university.checkPassword(password) == True and university.checkLogin(login) == False):
            newUser = User(numberId, name, surname, login, password)
            self.allTheUsers.append(newUser)
        else:
            print('Sorry! You made a mistake in password format!')
    def findUser(self, login):
        user = None
        for curUser in self.allTheUsers:
            if (curUser.getLogin() == login):
                user = curUser
```

The code for User Class:

```
class User:
    numberId = ""
    name = ""
    surname = ""
    login = ""
    password = ""
    def __init__(self, numberId, name, surname, login, password):
        self.numberId = numberId
        self.name = name
        self.surname = surname
        self.login = login
        self.password = password

    def getName(self):
        return self.name
    def getSurname(self):
        return self.surname
    def setName(self, name):
        self.name = name
    def setSurname(self, surname):
        self.surname = surname
    def getLogin(self):
        return self.login
    def setLogin(self, login):
        self.login = login
    def getPassword(self):
        return self.password
    def setPassword(self, password):
        self.password = password
    def getNumberId(self):
        return self.numberId;

    def viewTable(self):
        cursor.execute("select * from users;")
        answer = ""
        for row in cursor:
            answer+=str(row)
            answer+='\n'
        return answer

    def isExistedLogin(self, login):
        if(university.findUser(login) == None):
            return False
        return True
```

The code for Admin class that has inheritance from User:

```
class Admin(User):
    def __init__(self,numberId,name,surname,login,password):
        super().__init__(numberId,name,surname,login,password)

    def insertUser(self,numberId,name,surname,login,password):
        cursor.execute("insert into users values('{}','{}','{}','user','{}');".format(numberId,name,surname,login))
        university.allTheUsers.append(User(numberId,name,surname,login,password))
        conn.commit()

    def updateUser(self,numberId,name,surname,login):
        cursor.execute("""UPDATE users
                        SET userName = '{}',userSurname = '{}'
                        where numberId = '{}';""".format(name,surname,numberId))
        university.findUser(login).setName(name)
        university.findUser(login).setSurname(surname)
        conn.commit()

    def deleteUser(self,numberId,user):
        cursor.execute("DELETE FROM users WHERE numberId = '{}';".format(user.getNumberId()))
        university.allTheUsers.remove(user)
        conn.commit()
```

The code for connection with Microsoft Server DBMS:

```
conn = pyodbc.connect('Driver={SQL Server};'
                      'Server=DESKTOP-9VV7AM5;'
                      'Database=usersdb;'
                      'Trusted_Connection=yes;')
cursor = conn.cursor()
```

Some methods(functions) for interface realization:

To Enter the System:

```
def enterSystem():
    enteredLogin = ent_l.get()
    enteredPassword = ent_p.get()
    if (university.findUser(enteredLogin) == None):
        mesBox.showerror('Error','There is no such User')

    else:
        myUser = university.findUser(enteredLogin)
        if(myUser.getPassword() != enteredPassword):
            if(type(myUser) == Admin):
                mesBox.showerror('Error','You can not enter Admin Account which is not your!')
            else:
                mesBox.showerror('Error','You can not enter User Account which is not your!')
        else:
            if(type(myUser) == Admin):
                adminInterface(myUser)
            else:
                userInterface(myUser)
```

To view all the existed Users:

```
def viewTable(myUser):
    global grateFullOutput,outputLabel
    myOutput = myUser.viewTable()
    grateFullOutput = Label(window,text = 'All the Existed Users are : ',bg = "#856ff8",fg='#000000',font = ("Arial",11))
    outputLabel = Label(window,text = myOutput,bg = "#856ff8",fg='#000000',font = ("Arial",11))
    grateFullOutput.place(x = 60,y=270)
    outputLabel.place(x = 150,y = 290)
```

## To Delete any Existed User:

```
def deleteUserFinally(myAdmin):
    typedLogin = deleteUserLoginEntry.get()
    global showSuccessfulDeletion
    if (university.findUser(typedLogin) == None):
        mesBox.showerror('Error!', 'There is no such user!')
    else:
        foundUser = university.findUser(typedLogin)
        showSuccessfulDeletion = Label(window, text = 'The user {} with Id = {} and Login = {} is deleted successfully'.format(foundUser.getName(), foundUser.getNumberId(), foundUser.getLogin()))
        showSuccessfulDeletion.pack()
        myAdmin.deleteUser(foundUser.getNumberId(), foundUser)
        showSuccessfulDeletion.destroy()
        infoLabel.destroy()
        deleteUserLoginEntry.destroy()
        deleteUserB.destroy()
        adminInterface(myAdmin)

def deleteUser(myAdmin):
    global infoLabel, deleteUserLoginEntry, deleteUserB
    introLabel.destroy()
    viewPeopleButton.destroy()
    deleteButton.destroy()
    insertButton.destroy()
    updateButton.destroy()
    grateFullOutput.destroy()
    outputLabel.destroy()
    infoLabel = Label(window, text = 'Please, type the Login of the User which you want to delete:', font = ("Arial", 12))
    deleteUserLoginEntry = Entry(window)
    deleteUserB = Button(window, text = 'Delete User', bg = "#ba181b", fg = "#000000", font = ("Arial", 12), command = Lambda:deleteUserFinally(myAdmin))
    infoLabel.place(x = 40, y = 170)
    deleteUserLoginEntry.place(x = 180, y = 200)
    deleteUserB.place(x = 195, y = 230)
```

## To insert a user into the System By Admin:

```
def insertFinally(myAdmin, id, name, surname, login, password):
    if (university.checkPassword(password) == False):
        mesBox.showerror('Error!', 'The password is not Suitable!')
    else:
        foundUser = university.findUser(login)
        if (foundUser != None):
            mesBox.showerror('Error!', 'You can not add existed user!')
        else:
            myAdmin.insertUser(id, name, surname, login, password)
            infoLabelInsert.destroy()
            infoId.destroy()
            idEntry.destroy()
            infoName.destroy()
            nameEntry.destroy()
            infoSurname.destroy()
            surnameEntry.destroy()
            infoLogin.destroy()
            loginEntry.destroy()
            infoPassword.destroy()
            passwordEntry.destroy()
            insertB.destroy()
            adminInterface(myAdmin)

def insertUser(myAdmin):
    global infoLabelInsert, infoId, idEntry, infoName, nameEntry, infoSurname, surnameEntry, infoLogin, loginEntry, infoPassword, passwordEntry, insertB
    introLabel.destroy()
    viewPeopleButton.destroy()
    deleteButton.destroy()
    insertButton.destroy()
    updateButton.destroy()
    grateFullOutput.destroy()
    outputLabel.destroy()
    infoLabelInsert = Label(window, text = 'Please, type the User Id, User Name, User Surname, User Login, User Password:', font = ("Arial", 10))
    infoId = Label(window, text = 'Id', font = ("Arial", 10))
    idEntry = Entry(window)
    infoName = Label(window, text = 'Name', font = ("Arial", 10))
    nameEntry = Entry(window)
    infoSurname = Label(window, text = 'Surname', font = ("Arial", 10))
    surnameEntry = Entry(window)
    infoLogin = Label(window, text = 'Login', font = ("Arial", 10))
    loginEntry = Entry(window)
    infoPassword = Label(window, text = 'Password', font = ("Arial", 10))
    passwordEntry = Entry(window)
    insertB = Button(window, text = 'Insert', bg = "#43aa8b", fg = "#000000", font = ("Arial", 12), command = Lambda:insertFinally(myAdmin, idEntry.get(), nameEntry.get(), surnameEntry.get(), loginEntry.get(), passwordEntry.get()))
    infoLabelInsert.place(x = 10, y = 100)
    infoId.place(x = 244, y = 130)
    idEntry.place(x = 190, y = 150)
    infoName.place(x = 231, y = 180)
    nameEntry.place(x = 190, y = 200)
    infoSurname.place(x = 222, y = 230)
    surnameEntry.place(x = 190, y = 250)
    infoLogin.place(x = 234, y = 280)
    loginEntry.place(x = 190, y = 300)
    infoPassword.place(x = 222, y = 330)
```

## The view for User's interface:

```
def userInterface(myUser):
    global introLabel, viewPeopleButton
    if (type(auth_l) != int):
        auth_l.destroy()
    if (type(auth_p) != int):
        auth_p.destroy()
    if (type(ent_l) != int):
        ent_l.destroy()
    if (type(ent_p) != int):
        ent_p.destroy()
    if (type(ent_b) != int):
        ent_b.destroy()
    introLabel = Label(window, text = 'Good Afternoon {}! Here is the list of possible options:'.format(myUser.getName()), font = ("Arial", 12))
    viewPeopleButton = Button(window, text = 'View table', width = 50, bg = "#f7cad0", fg = "#000000", font = ("Arial", 12), command = Lambda:viewTable(myUser))
    introLabel.place(x = 30, y = 150)
    viewPeopleButton.place(x = 19, y = 180)
```

## To Update information about User:

```
def finalUpdate(myAdmin, login, newName, newSurname):
    if (university.findUser(login) == None):
        mesBox.showerror('Error!', 'There is no such user!')
    else:
        myUser = university.findUser(login)
        myAdmin.updateUser(myUser.getNumberId(), newName, newSurname, login)
        loginOfNeededUser.destroy()
        loOfUser.destroy()
        userEntry.destroy()
        newNameLabel.destroy()
        newNameEntry.destroy()
        newSurnameLabel.destroy()
        newSurnameEntry.destroy()
        finalButtonUpdate.destroy()
        adminInterface(myAdmin)

def updateUser(myAdmin):
    global loginOfNeededUser, loOfUser, userEntry, newNameLabel, newNameEntry, newSurnameLabel, newSurnameEntry, finalButtonUpdate
    introLabel.destroy()
    viewPeopleButton.destroy()
    deleteButton.destroy()
    insertButton.destroy()
    updateButton.destroy()
    grateFullOutput.destroy()
    outputLabel.destroy()

    loginOfNeededUser = Label(window, text = 'Please, type the login of the User that you want to change and fill new Data below:', font = ("Arial", 10))
    loOfUser = Label(window, text= 'User login', font = ("Arial", 10))
    userEntry = Entry(window)
    newNameLabel = Label(window, text = 'New Name', font = ("Arial", 10))
    newNameEntry = Entry(window)
    newSurnameLabel = Label(window, text = 'New Surname', font = ("Arial", 10))
    newSurnameEntry = Entry(window)
    finalButtonUpdate = Button(window, text= 'Update User', bg = "#fcf6bd", fg="#000000", font = ("Arial", 10), command = Lambda:finalUpdate(myAdmin, userEntry.get(), newNameEntry.get(), newSurnameEntry.get()))
    loginOfNeededUser.place(x = 9, y = 150 )
    loOfUser.place(x = 215, y = 180 )
    userEntry.place(x = 184, y = 200 )
    newNameLabel.place(x = 213, y = 230)
    newNameEntry.place(x = 183, y = 250 )
    newSurnameLabel.place(x = 285, y = 280 )
    newSurnameEntry.place(x = 183, y = 300 )
    finalButtonUpdate.place(x = 208, y = 330)
```

## Admin's Interface:

```
def adminInterface(myAdmin):
    userInterface(myAdmin)
    global deleteButton, insertButton, updateButton
    deleteButton = Button(window, text = 'Kill User(Delete)', bg = "#ef476f", fg='#000000', font = ("Arial", 12), command = Lambda:deleteUser(myAdmin))
    insertButton = Button(window, text = 'Add User(Insert)', bg = "#06d6a0", fg='#000000', font = ("Arial", 12), command = Lambda:insertUser(myAdmin))
    updateButton = Button(window, text = 'Refresh User\'s Data(Update)', bg = "#ffd166", fg='#000000', font = ("Arial", 12), command = Lambda:updateUser(myAdmin))
    deleteButton.place(x=10, y=225)
    insertButton.place(x=140, y=225)
    updateButton.place(x=270, y=225)
```

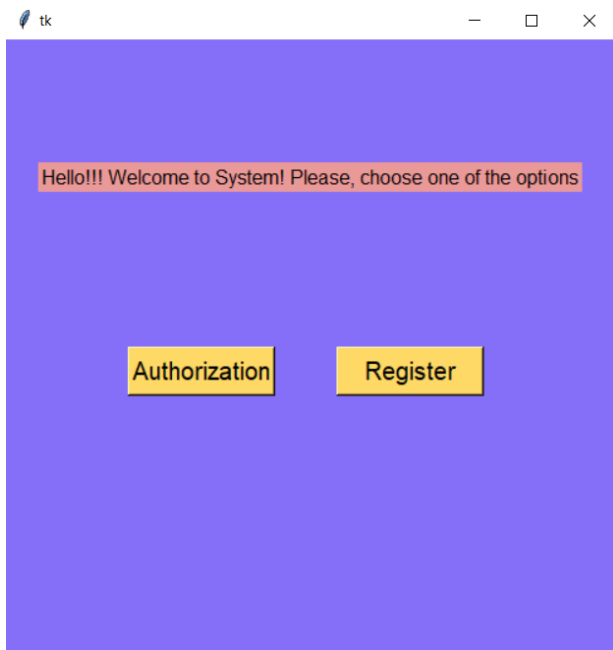
## To log in the System:

```
def authorization():
    global auth_l, auth_p, ent_l, ent_p, ent_b
    l1.destroy()
    b1.destroy()
    b2.destroy()
    auth_l = Label(window, text='login', font = ("Arial", 12))
    auth_p = Label(window, text='password', font = ("Arial", 12))
    ent_l = Entry(window)
    ent_p = Entry(window)
    ent_b = Button(window, text='Enter', font = ("Arial", 12), command=enterSystem)
    auth_l.place(x=200, y=120)
    auth_p.place(x=200, y=180)
    ent_l.place(x=200, y=150)
    ent_p.place(x=200, y=210)
    ent_b.place(x=200, y=240)
```

## Working Principle of the program:

There are 2 included users(Admin- Temirbolat, User – Asanali) in advanced

### 1) Main Menu:



Here we can choose one of 2 options:

1) Authorization 2) Registration

Let's firstly Try to register a new User:

## 2) User's Registration by himself:

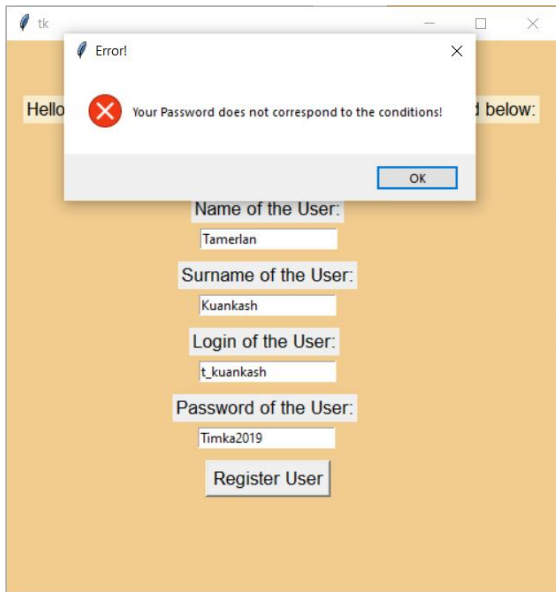
Here we see another opened window which meets us with message and asks to fill the gaps with required information. Let's type the information for new User but instead of new Login we write already existed. The corresponded Error must appear.

## Attempt to register with existed user's login:

The login 't\_maratuly' belongs to Admin Temirbolat → we can not add a new user taking the same unique part.

Let's try to type a good login, but with not appropriate password (Password requirements were provided in the task)

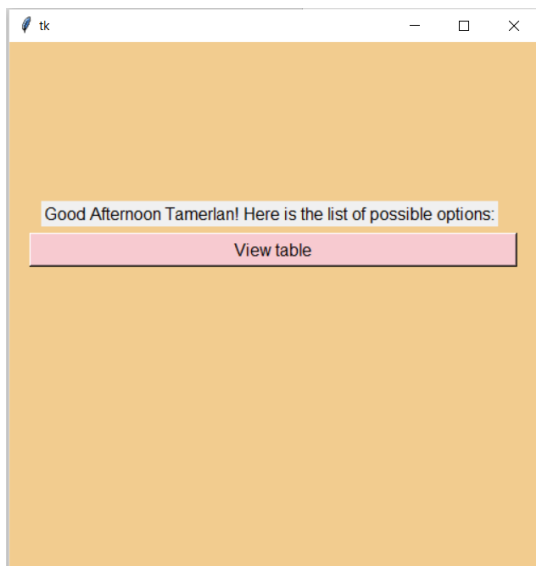
### Attempt to register with not suitable password:



The screenshot shows a registration form titled 'tk'. It has five input fields: 'Name of the User:' (Tamerlan), 'Surname of the User:' (Kuankash), 'Login of the User:' (t\_kuankash), and 'Password of the User:' (Timka2019). Below these fields is a 'Register User' button. An error dialog box is overlaid on the form, titled 'Error!' with a red 'X' icon. The message in the dialog says: 'Your Password does not correspond to the conditions!'. There is an 'OK' button in the dialog.

Here we can see appeared Error because the password is wrong. It contains the Capital letter as well as small ones, numbers, good length, however, it doesn't have a specific sign(symbol) that doesn't allow to continue. Let's type a good password by adding '!' in the end.

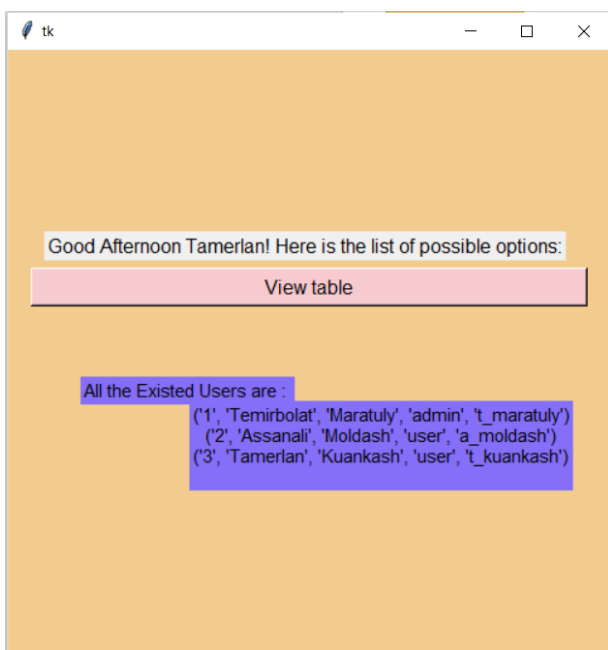
### Good Register Example:



The screenshot shows the same registration form as before, but now it displays a message: 'Good Afternoon Tamerlan! Here is the list of possible options:'. Below this message is a pink button labeled 'View table'.

When data were provided correctly, new user is entered to the system automatically and he/she sees all the possible buttons which he/she can use.

### 3) View Table Option for All the Users:



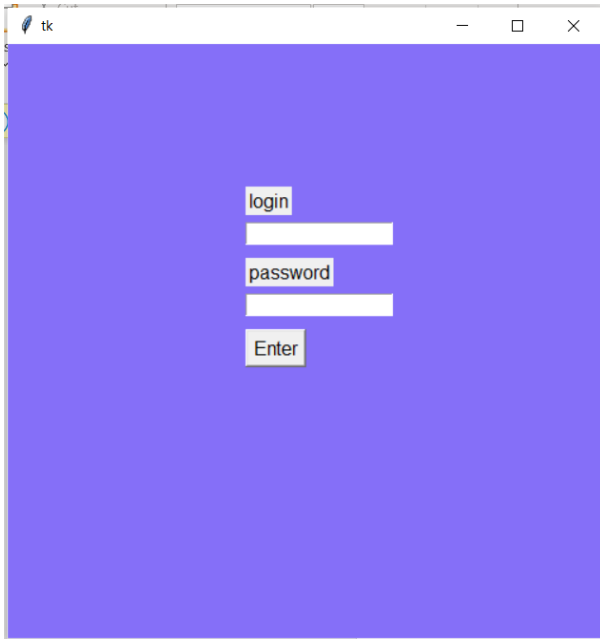
The screenshot shows the same registration form as before, but now it displays a message: 'Good Afternoon Tamerlan! Here is the list of possible options:'. Below this message is a pink button labeled 'View table'. Below the button, there is a text area that says 'All the Existed Users are :'. Below this text, there is a list of users in a table format:

1	Temirbolat	Maratuly	admin	t_maratuly
2	Assanali	Moldash	user	a_moldash
3	Tamerlan	Kuankash	user	t_kuankash

If we type the button "View table" we can see all the existed users in our database from Microsoft Server DBMS. It can be clearly seen that our created user Tamerlan is already here 😊



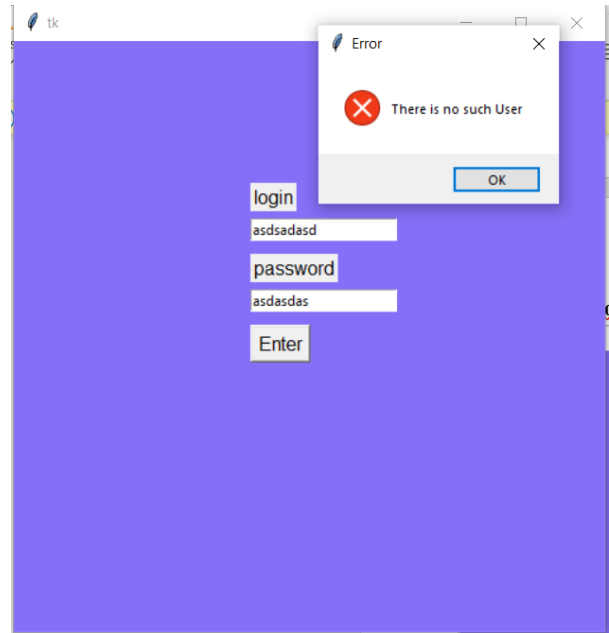
#### 4) Authorization part:



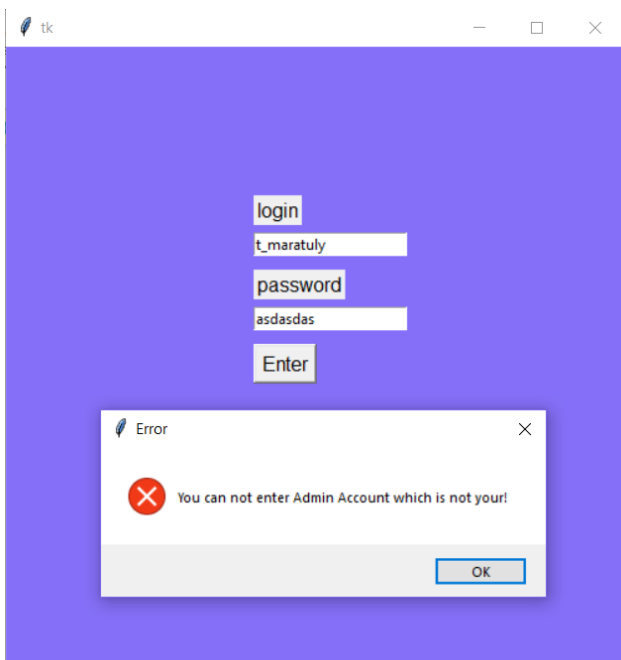
If we choose Authorization button in the beginning, we come to the page with filling Login and Password.

Let's try to fill in wrong login.

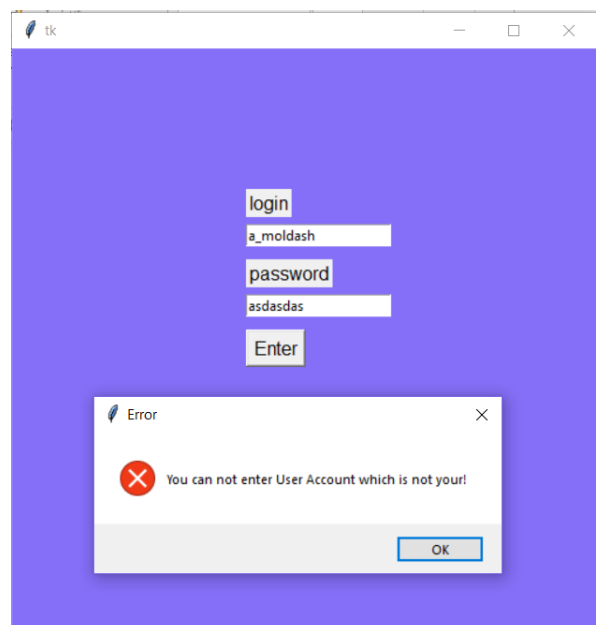
Here we can see that there is no such user in system with this Login. Therefore, we obtain an error. Let's write an existed login but among with wrong password.



Here a person tries to login in Admin's account without password knowledge. He obtains the corresponded error. DO the same for User.



Here we see the same problem but here the person tries to enter a user account without correct password. He gets about the same Error, although there is written User Account.



## 5) Entering the System By Admin:



If we type the correct Admin's login and password correctly we come to this page and see all the available options for Temirbolat (Admin)

## 6) Add New User

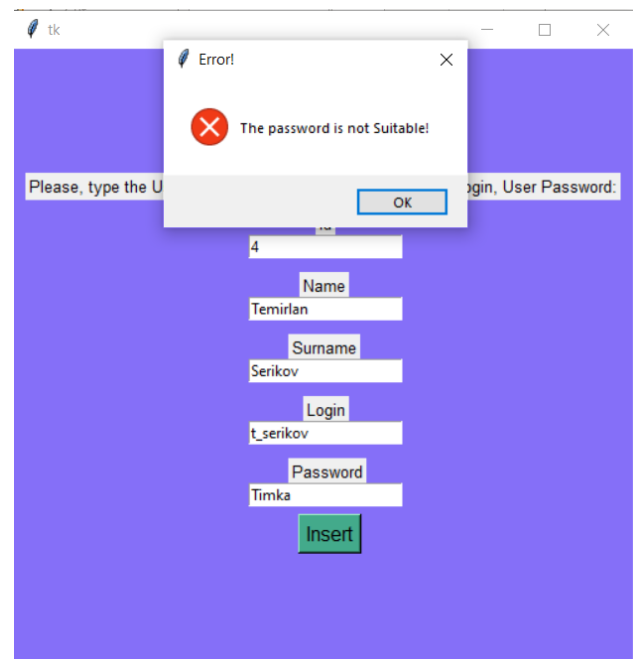
If we decide to press the button 'Add User(Insert)' we see this page →

Here we are asked to write all the required data to add new User into the system

A screenshot of a Tkinter window titled 'tk' showing a form for adding a new user. The window has a purple background. At the top, a white text box contains the instruction 'Please, type the User Id, User Name, User Surname, User Login, User Password:'. Below this, there are five input fields labeled 'Id', 'Name', 'Surname', 'Login', and 'Password'. At the bottom, there is a green button labeled 'Insert'.A screenshot of a Tkinter window titled 'tk' showing the same 'Add New User' form as before, but with data entered into the input fields. The 'Id' field contains '4', the 'Name' field contains 'Temirlan', the 'Surname' field contains 'Serikov', the 'Login' field contains 't\_serikov', and the 'Password' field contains 'Timka'. The green 'Insert' button is still at the bottom.

Here we typed the data for new User. Id = 4, Name = Temirlan, Surname = Serikov, Login = t\_serikov, Password = Timka. (Which is not suitable)

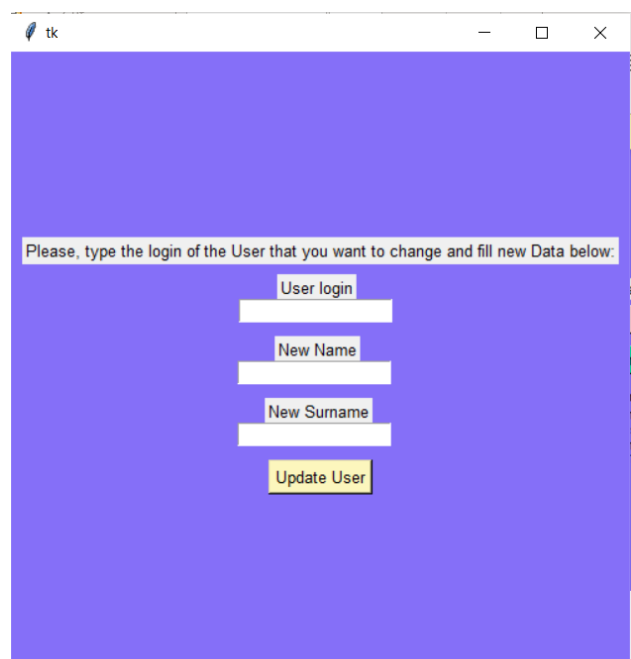
Here the Admins obtains the Error, since the typed password doesn't fit the requirements of the password. Here it doesn't have any number and specific sign.

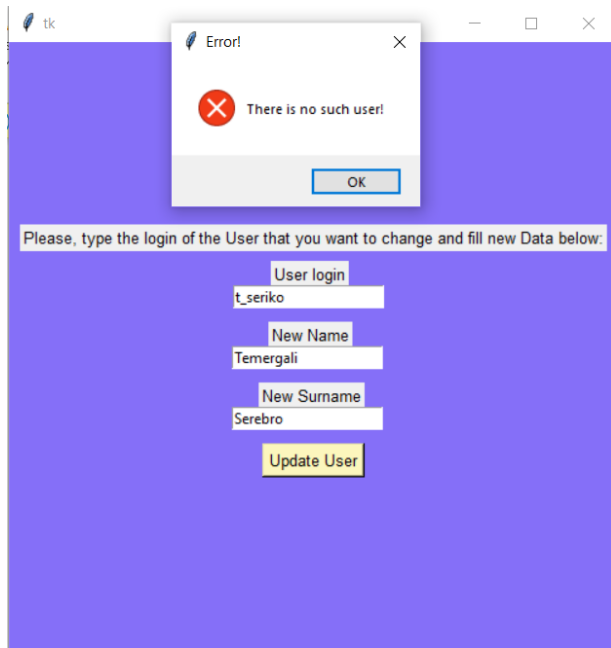


Now, when we typed the suitable password, the user is instantly created and added to the system. Here we can ← see it

## 7) Updating User's Information:

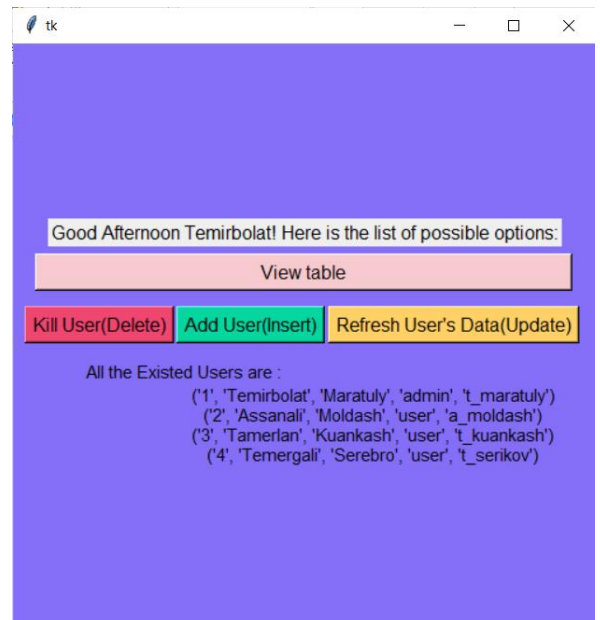
If a Admin desires to Update User's Information and press the button 'Refresh User's Data (Update)'. He finds out the window (where he needs to fill the corresponded login as well as new Name including Surname of the User) →



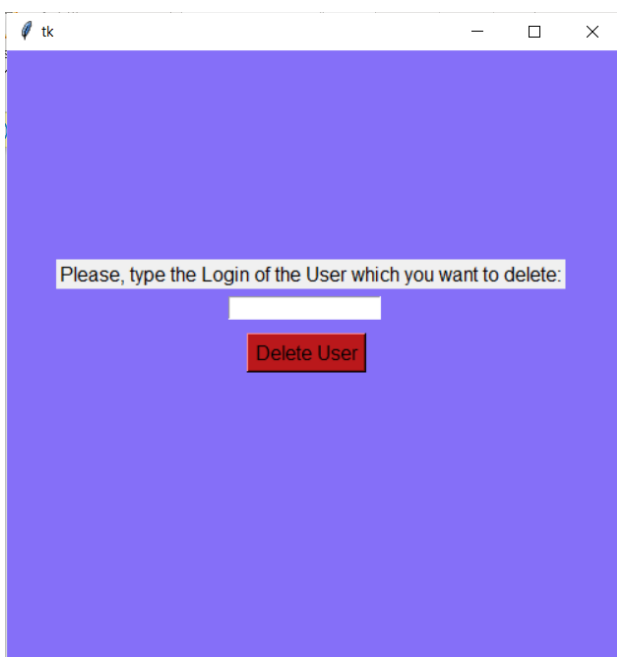


If we type wrong login we get an error since the existed user doesn't have this written unexisted login.

When we typed the login correctly (t\_serikov) and click on 'Update User' we get to the initial page where we see that this user is successfully changed.

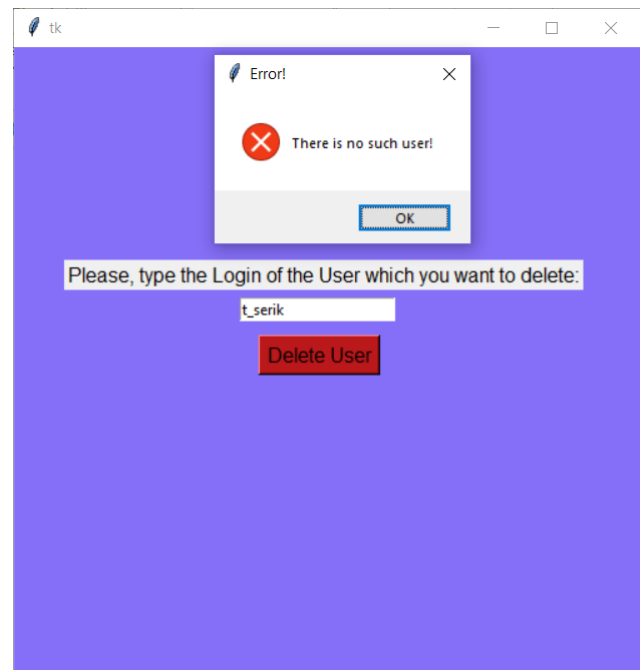


## 8) Delete the User



If we click on 'Kill User(Delete)' we get to another page which asks us to type user's login which we want to delete (Since it is unique)

If we remain the login incorrectly we get the Error message that says ‘You made a mistake’.



When you finished the previous step with correct login you can find out that this user is successfully deleted and now it is not listed below.



## Final Result:

### Table in SQL:

	numberId	userName	userSurname	userType	userLogin
1	1	Temirbolat	Maratuly	admin	t_maratuly
2	2	Assanali	Moldash	user	a_moldash
3	3	Tamerlan	Kuankash	user	t_kuankash

We see that the result is exactly the same as it was given above. It means that our Program works correctly and it is connected to our DBMS.