Computer Science Investigatory Project

Session 2020-21

Submitted By: Kandarp Pankaj Solanki

Class: XII - A

Roll No.:

Submitted To: Mrs. Vaishali Bhoite

CERTIFICATE

This is to certify that **Kandarp Pankaj Solanki** of Class XII-A, Bright Day School, Vadodara has successfully completed his project as per the guidelines issued by CBSE. He has taken proper care and shown utmost sincerity in the completion of the practical file.

Internal Examiner

External Examiner

Principal

ACKNOWLEDGEMENT

I, **Kandarp Pankaj Solanki** sincerely would like to thank my principal, Mrs. Neeta Sharma for her support.

I would like to extend my gratitude to Mrs. Vaishali Bhoite for guiding me through the project.

I am very grateful to my parents who were a constant source of inspiration and encouragement throughout the project. Without them this project would not have been a success.

Finally, I would like to thank CBSE for giving me this opportunity to undertake this project.

INDEX

SYSTEM REQUIREMENTS

- Hardware Requirements
    1. Keyboard & Mouse
    2. RAM - 4 GB or higher
    3. Dual Core - 2 GHz or higher

- Software Requirements
    1. Intel Atom Processor / Intel Core i3 Processor or higher
    2. MacOS, Windows 7 or later and Linux
    3. Python IDLE or any other IDE (Eg. VSCode by Microsoft, PyCharm, etc.)

FEASIBILITY STUDY

A **feasibility study** is an assessment of the practicality of a proposed project or system. A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing/proposed project.
Basically, feasibility study aims to find the **cost required** & the **value/result obtained**.

❖ Economical Feasibility

Economical analysis of any proposed project is the most frequently used method to evaluate the effectiveness of a project and thereby calculating the savings and the cost of the proposal.
This software is built with a total investment in time and at no other cost and requires just a machine to work upon. If you just have a display and Python installed (with a few more libraries) on your machine, you are all set to go. Thus, economically this saves a lots of money of papers, inks, etc since all records are now stored on your machine with just a click.

❖ Technical Feasibility

A **technical feasibility** study assesses the details of how you intend to deliver a product or service to customers. This also involves financial consideration to accommodate technical enhancements.

This software is technically feasible since whatever technology is needed to develop this software is easily available.

ERRORS & ITS TYPES

An error, sometimes called "**A BUG**" is anything in the code that prevents a program from compiling and running correctly. There are broadly three types of errors as follows:

1. Compile- time errors: Errors that occurs during compilation of a program is called compile time error. It has two types as follows:

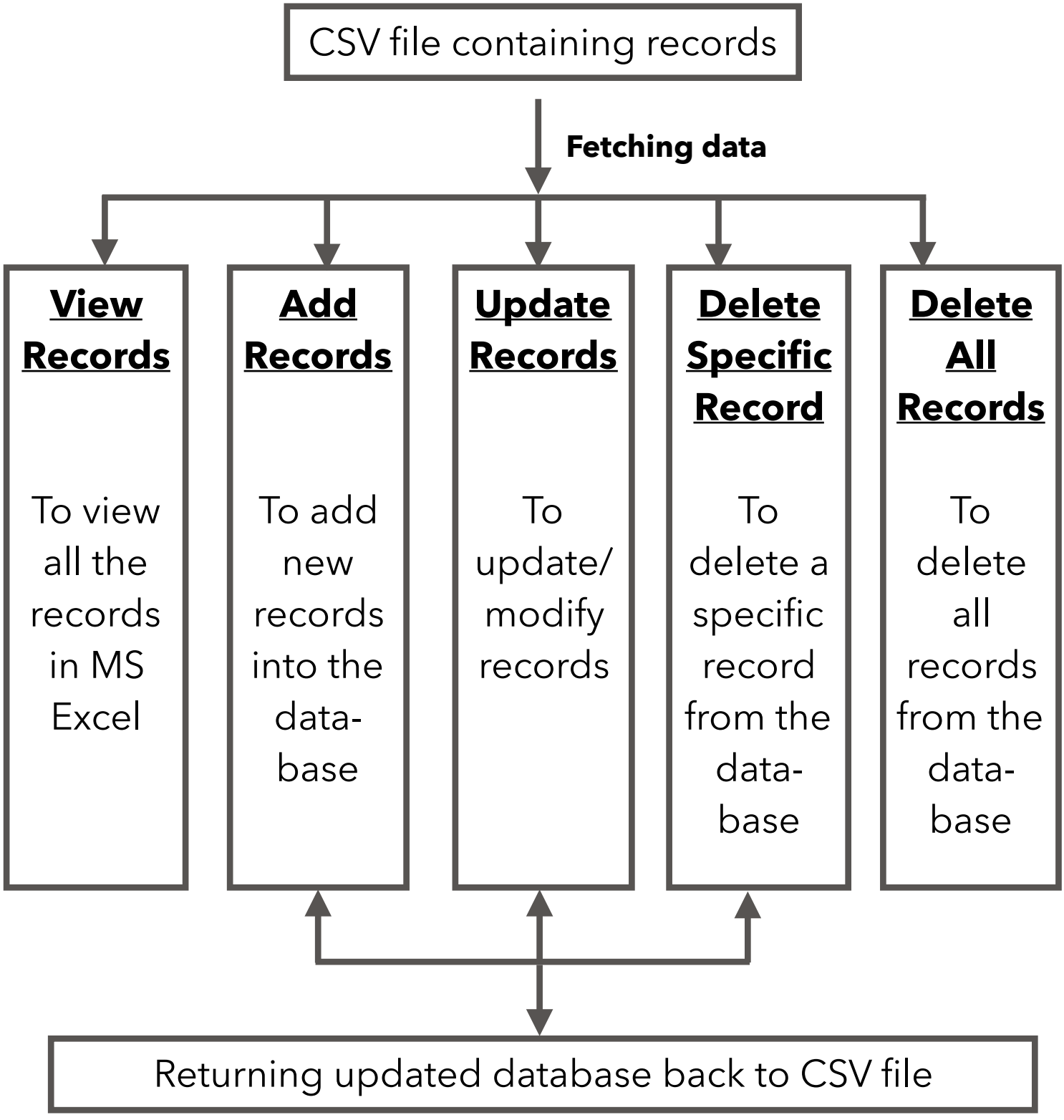
(i) **Syntax error**: It refers to formal rules governing the construction of valid statements in a language.

(ii)**Semantics error**: It refers to the set of rules which give the meaning of a statement.

2. Run time Errors: Errors that occur during the execution of program are run time errors. These are harder to detect errors. Some run-time error stop the execution of program which is then called program "**Crashed**".

3. Logical Errors: Sometimes, even if you don't encounter any error during compiling-time and runtime, your program does not provide the correct result. This is because of the programmer's mistaken analysis of the problem he or she is trying to solve. Such errors are called logical error.

FLOW CHART OF THE PROGRAM



CODE

```
# CS Project - Session - 2020-21
from tkinter import *
from tkinter import messagebox
from DBMSlibrary_CSV import addrec,searchrec,delrec,delparrec,updaterec,allrecs,info,info2,info3,lsttocsv
from PIL import ImageTk,Image
import pickle
import datetime
import time
from DBMSlibrary_CSV import csvtolst,lsttocsv
import subprocess

class GUI():


    # Authorised Login #
    def loginscreen(self):
        authname = 'admin'       #Authentification ADMIN_ID to be written here
        authpass = '123456'   #Authentification ADMIN_PASSWORD to be written here


        self.authlogin = Tk()
        self.authlogin.geometry('1000x320')
        self.authlogin.config(background = 'deeppink')
        self.authlogin.title('LOGIN')
        self.authlogin.minsize(650,420)
        self.authlogin.maxsize(650,420)


        passwordentry = StringVar()
        usernameentry = StringVar()


        imglogo = Image.open('/Users/kandarpsolanki/Desktop/Codings/logo.png')
        imglogo = imglogo.resize((250,110),Image.ANTIALIAS)
        photo = ImageTk.PhotoImage(imglogo)
        self.lab_pic = Label(self.authlogin,image = photo,borderwidth = 1,relief = 'sunken')
        self.lab_pic.image = photo
        self.lab_pic.config(image = photo)
        self.lab_pic.pack(fill=BOTH)
        def authorise():
            uid = self.username.get()
            upass = self.password.get()
            if uid == authname and upass == authpass:
                messagebox.showinfo('Welcome','You are successfully logged in !!')
                self.authlogin.destroy()
                self.homepge()
            else:
                messagebox.showwarning('Access Denied','Incorrect ID/Password !!')
                usernameentry.set(uid)
                passwordentry.set('')
```

```
51
52          def hide(event):
53              tempvar = self.password.get()
54              passwordentry.set(tempvar)
55              self.password = Entry(self.authlogin,textvariable = passwordentry,bg = 'grey',fg = 'white',\
56                              font = ('Operator Mono', 20 ,'bold'),show = '*',width = '15')
57              self.password.place(x=241,y=230)
58
59
60          def show(event):
61              tempvar = self.password.get()
62              passwordentry.set(tempvar)
63              self.password = Entry(self.authlogin,textvariable = passwordentry,bg = 'grey',fg = 'white',\
64                              font = ('Operator Mono', 20 ,'bold'),show = '',width = '15')
65              self.password.place(x=241,y=230)
66
67          self.usernamelab = Label(self.authlogin,text = 'Username :',bg = 'deeppink',font = ('Arial', 19))
68          self.usernamelab.place(x=91,y=164)
69
70          self.passwordlab = Label(self.authlogin,text = 'Password :',bg = 'deeppink',font = ('Arial', 19))
71          self.passwordlab.place(x=91,y=234)
72
73          self.username = Entry(self.authlogin,textvariable = usernameentry,bg = 'grey',fg = 'white',\
74                          font = ('Operator Mono', 20 ,'bold'),show = '',width = '15')
75          self.username.place(x=241,y=160)
76
77          self.password = Entry(self.authlogin,textvariable = passwordentry,bg = 'grey',fg = 'white',\
78                          font = ('Operator Mono', 20 ,'bold'),show = '*',width = '15')
79          self.password.place(x=241,y=230)
80
81          self.show = Button(self.authlogin,text = 'Show Password',command = hide,relief = 'sunken',width = 13,\
82                      height = 1,font = ('Arial',14))
83          self.show.place(x=450,y=237)
84          self.show.bind('<ButtonPress-1>',show)
85          self.show.bind('<ButtonRelease-1>',hide)
86
87          self.loginbut = Button(self.authlogin,text = 'Login',command = authorise,width = 7,height = 1,\
88                          font = ('Arial',16))
89          self.loginbut.place(x=295,y=300)
90
91          self.case = Label(self.authlogin,text='*Username/Password is case-sensitive.',fg='black',bg='deeppink',\
92                      font = ('Arial',17))
93          self.case.place(x=170,y=340)
94
95          self.labelackn = Label(self.authlogin,text='Designed and Developed by: KANDARP SOLANKI',fg='black',\
96                           bg='deeppink',font = ('Operator Mono Medium',13,'italic'))
97          self.labelackn.place(x=340,y=395)
98
99          self.authlogin.mainloop()
100
```

```
101
102     # Library App #
103     def library(self):
104         self.window = Toplevel(self.homepage)
105         self.window.geometry('1000x600')
106         self.window.config(background = 'deeppink')
107         self.window.title('Library Database Management System')
108         self.window.minsize(1000,600)
109         self.window.maxsize(1000,600)
110 
111         #Exitting Window
112         def exitout():
113             w = messagebox.askyesno('Quit','Are you sure you want to quit?')
114             if w:
115                 self.window.destroy()
116             else:
117                 pass
118 
119 
120         # Open document with default application in Python.
121         def opencsv():
122 
123             subprocess.call(['open', 'studentdatabase.csv'])
124 
125 
126         photo=ImageTk.PhotoImage(Image.open('logo.png'))
127         lab_pic = Label(self.window,image = photo)
128         lab_pic.image = photo
129         lab_pic.config(image = photo)
130         lab_pic.pack(fill=BOTH)
131 
132 
133         self.label_entry = Label(self.window,text='Welcome to Library Database Management System',\
134                                  fg='blue',bg='white',relief = ('solid'),font = ('Arial',21,'bold'))
135         self.label_entry.pack(fill=BOTH,padx=180,pady=30)
136 
137         self.button1 = Button(self.window,text='VIEW ALL RECORDS',fg='black',bg='white',relief = ('sunken'),\
138                               font = ('Operator Mono',20,'bold'),command=opencsv)
139         self.button1.pack(fill=BOTH,padx=220,pady=10)
140 
141 
142 
143         self.button2 = Button(self.window,text='ADD NEW RECORD',fg='black',bg='white',relief = ('sunken'),\
144                               font = ('Operator Mono',20,'bold'),command=self.adding)
145         self.button2.pack(fill=BOTH,padx=220,pady=10)
146 
147 
148         self.button4 = Button(self.window,text='UPDATE A RECORD',fg='black',bg='white',relief = ('sunken'),\
149                               font = ('Operator Mono',20,'bold'),command=self.updating)
150         self.button4.pack(fill=BOTH,padx=220,pady=10)
```

```
151
152
153         self.button5 = Button(self.window,text='RETURN A BOOK',fg='black',bg='white',relief = ('sunken'),\
154                         font = ('Operator Mono',20,'bold'),command=self.deletingparticular)
155         self.button5.pack(fill=BOTH,padx=220,pady=10)
156
157
158         self.button6 = Button(self.window,text='DELETE ALL RECORDS',fg='black',bg='white',relief = ('sunken'),\
159                         font = ('Operator Mono',20,'bold'),command=self.deletingall)
160         self.button6.pack(fill=BOTH,padx=220,pady=10)
161
162
163         self.button7 = Button(self.window,text='EXIT',fg='black',bg='white',relief = ('sunken'),\
164                         font = ('Operator Mono',20,'bold'),command=exitout)
165         self.button7.pack(fill=BOTH,padx=470,pady=10)
166
167
168         self.labelackn = Label(self.window,text='Designed and Developed by: KANDARP SOLANKI',fg='black',\
169                         bg='deeppink',font = ('Operator Mono Medium',13,'italic'))
170         self.labelackn.place(x=678,y=574)
171
172         self.label_caution = Label(self.window,text='NOTE: Open only one window at a time for best results.',\
173                         fg='black',bg='deeppink',relief = ('solid'),font = ('Arial 17 italic'))
174         self.label_caution.pack(fill=BOTH,padx=180,pady=30)
175
176
177     # School Homepage #
178     def homepge(self):
179         self.homepage = Tk()
180
181         self.homepage.config(background = 'deeppink')
182
183
184         self.homepage.geometry('1000x600')
185         self.homepage.minsize(1000,600)
186         self.homepage.maxsize(1000,600)
187         self.homepage.title('BRIGHT DAY SCHOOL')
188
189         def logout():
190             final = messagebox.askyesno('Confirm','Are you sure you want to logout ?')
191             if final:
192                 self.homepage.destroy()
193                 self.loginscreen()
194             else:
195                 pass
196
197         imglogo = Image.open('logo.png')
198         imglogo = imglogo.resize((250,110),Image.ANTIALIAS)
199         photo=ImageTk.PhotoImage(imglogo)
200         self.lab_pic = Label(self.homepage,image = photo,borderwidth = 1,relief = 'sunken')
```

```
        self.lab_pic.image = photo
        self.lab_pic.config(image = photo)
        self.lab_pic.pack(fill=BOTH)


        img=Image.open('library.jpg')
        img = img.resize((180,100), Image.ANTIALIAS)
        photo=ImageTk.PhotoImage(img)


        self.but_lib = Button(self.homepage,image = photo,command = self.library,borderwidth=25,relief='solid')
        self.but_lib.place(x=90,y=220)

        imgother2=Image.open('otherapp.png')
        imgother2 = imgother2.resize((145,100), Image.ANTIALIAS)
        photo2=ImageTk.PhotoImage(imgother2)


        self.but_app1 = Button(self.homepage,image = photo2,borderwidth=25,relief='solid')
        self.but_app1.place(x=425,y=220)

        imgother3=Image.open('otherapp.png')
        imgother3 = imgother3.resize((145,100), Image.ANTIALIAS)
        photo3=ImageTk.PhotoImage(imgother3)


        self.but_app2 = Button(self.homepage,image = photo3,borderwidth=25,relief='solid')
        self.but_app2.place(x=760,y=220)

        imgother4=Image.open('otherapp.png')
        imgother4 = imgother4.resize((145,100), Image.ANTIALIAS)
        photo4=ImageTk.PhotoImage(imgother4)


        self.but_app3 = Button(self.homepage,image = photo4,borderwidth=25,relief='solid')
        self.but_app3.place(x=760,y=420)

        imgother5=Image.open('otherapp.png')
        imgother5 = imgother5.resize((145,100), Image.ANTIALIAS)
        photo5=ImageTk.PhotoImage(imgother5)



        self.but_app4 = Button(self.homepage,image = photo5,borderwidth=25,relief='solid')
        self.but_app4.place(x=100,y=420)

        imgother6=Image.open('otherapp.png')
        imgother6 = imgother6.resize((145,100), Image.ANTIALIAS)
        photo6=ImageTk.PhotoImage(imgother6)

```

```
        self.but_app5 = Button(self.homepage,image = photo6,borderwidth=25,relief='solid')
        self.but_app5.place(x=425,y=420)

        self.logout = Button(self.homepage,text = 'Logout',bg = 'deeppink',command = logout,relief='sunken',\
                    borderwidth=5,font = ('Arial',20))
        self.logout.place(x=930,y=114)

        self.labelackn = Label(self.homepage,text='Designed and Developed by: KANDARP SOLANKI',fg='black',\
                        bg='deeppink',font = ('Operator Mono Medium',13,'italic'))
        self.labelackn.place(x=678,y=574)

        self.homepage.mainloop()


    # Add Record ~ Library App #
    def adding(self):
        name = StringVar()
        roll = None
        isbn = StringVar()
        book = StringVar()


        self.add = Toplevel(self.window)
        self.add.geometry('1000x600')
        self.add.config(background = 'deeppink')
        self.add.title('NEW RECORD')
        self.add.minsize(1000,600)
        self.add.maxsize(1000,600)

        imglogo = Image.open('/Users/kandarpsolanki/Desktop/Codings/logo.png')
        imglogo = imglogo.resize((250,110),Image.ANTIALIAS)
        photo = ImageTk.PhotoImage(imglogo)
        self.lab_pic = Label(self.add,image = photo,borderwidth = 1,relief = 'sunken')
        self.lab_pic.image = photo
        self.lab_pic.config(image = photo)
        self.lab_pic.pack(fill=BOTH)

        self.namelabel = Label(self.add,text = 'Student Name :',bg = 'deeppink',font = ('Arial 20 bold'))
        self.namelabel.place(x=200,y=200)
        self.stu_name = Entry(self.add,textvariable = name,font = ('Arial 20 italic'))
        self.stu_name.place(x=500,y=200)


        self.rolllabel = Label(self.add,text = 'Examination Seat No :',bg = 'deeppink',font = ('Arial 20 bold'))
        self.rolllabel.place(x=200,y=240)
        self.stu_roll = Entry(self.add,textvariable = roll,font = ('Arial 20 italic'))
        self.stu_roll.place(x=500,y=240)

        self.isbnlabel = Label(self.add,text = 'ISBN No. :',bg = 'deeppink',font = ('Arial 20 bold'))
        self.isbnlabel.place(x=200,y=280)
```

```
301         self.stu_isbn = Entry(self.add,textvariable = isbn,font = ('Arial 20 italic'))
302         self.stu_isbn.place(x=500,y=280)
303 
304         self.booklabel = Label(self.add,text = 'Book Name :',bg = 'deeppink',font = ('Arial 20 bold'))
305         self.booklabel.place(x=200,y=320)
306         self.stu_book = Entry(self.add,textvariable = book,font = ('Arial 20 italic'))
307         self.stu_book.place(x=500,y=320)
308 
309         self.labelackn = Label(self.add,text='Designed and Developed by: KANDARP SOLANKI',fg='black',\
310                                bg='deeppink',font = ('Operator Mono Medium',13,'italic'))
311         self.labelackn.place(x=678,y=574)
312 
313         def restore():
314             isbn.set('')
315             book.set('')
316 
317         try:
318             open_bin=open('dbmscsv','rb')  # Regular use
319             add_temp=pickle.load(open_bin)
320             open_bin.close()
321         except:
322             open_bin=open('dbmscsv','wb') # If initialising for 1st time
323             lst=[]
324             pickle.dump(lst,open_bin)
325             open_bin.close()
326             open_bin=open('dbmscsv','rb')
327             add_temp=pickle.load(open_bin)
328             open_bin.close()
329 
330 
331 
332 
333         def printt():
334             letpass1 = True
335             letpass2 = False
336             tryother = True
337             userallow = True
338             open_bin=open('dbmscsv','wb')
339             try:
340                 name = self.stu_name.get()
341                 roll = self.stu_roll.get()
342                 isbn = self.stu_isbn.get()
343                 book = self.stu_book.get()
344             except:
345                 pass
346             try:
347                 name = int(name)
348                 letpass1 = False
349             except:
350                 pass
```

```
            try:
                roll = int(roll)
                letpass2 = True
            except:
                pass
            if name == '' or roll == ''  or book == '' or isbn == '':
                messagebox.showerror('Error Message','Fill all the information !!')
                tryother = False
                userallow = False

            if (letpass1 == False or letpass2 == False) and tryother == True:
                messagebox.showerror('Error Message','Enter correct datatype !!')
                userallow = False


            stu_dateissue=datetime.date.today()
            stu_datedue=datetime.date.today()+datetime.timedelta(days=7)
            book_count = 0
            isbn_count = 1
            book_count2 = 0
            condition = 'match'


            try:
                for i in add_temp:
                    if int(i[1]) == roll and i[0] == name:
                        condition = 'match'
                        for i in add_temp:
                            if int(i[1]) == roll and i[0] == name:
                                book_count += 1
                        break
                    else:
                        condition = 'mismatch'
                for i in add_temp:
                    if i[2] == isbn:
                        isbn_count += 1
                curr_rolls = []
                for i in add_temp:
                    curr_rolls.append(str(i[1]))
                if str(roll) not in curr_rolls:
                    condition = 'match'
                    book_count2 += 1


                if (book_count < 2 and isbn_count == 1 and condition == 'match' and userallow == True) or \
                   (book_count2 == 1 and isbn_count == 1 and condition == 'match' and userallow == True):
                    lst_temp = [name,roll,isbn,book,stu_dateissue,stu_datedue]
                    add_temp.append(lst_temp)
                    pickle.dump(add_temp,open_bin)
                    lsttocsv(add_temp)
```

```
                    messagebox.showinfo('Message','Record added successfully.')
                    restore()
                elif book_count >= 2:
                    messagebox.showinfo('Error','Student has already issued 2 books !!')
                    info3(add_temp,stu_roll,1)
                    pickle.dump(add_temp,open_bin)
                    restore()
                elif isbn_count > 1:
                    messagebox.showinfo('Error','The book with ISBN number already issued.')
                    info3(add_temp,stu_isbn,2)
                    pickle.dump(add_temp,open_bin)
                    restore()
                elif condition=='mismatch':
                    messagebox.showinfo('Error','Details( Name & Unique ID ) don\'t match !!')
                    restore()
                    pickle.dump(add_temp,open_bin)
                else:
                    pickle.dump(add_temp,open_bin)
            except:
                pickle.dump(add_temp,open_bin)
                pass
            open_bin.close()

        self.tbut = Button(self.add,text = 'Add',command = printt,width = 25,height = 2,font = ('Arial 20 bold'))
        self.tbut.place(x=350,y=400)


    # Update Record ~ Library App #
    def updating(self):

        exseat = ''
        updainfo = ''
        name = 'N'
        exam = 'E'
        isbn = 'I'
        book = 'B'



        self.updat = Toplevel(self.window)
        self.updat.geometry('1000x600')
        self.updat.config(background ='deeppink')
        self.updat.title('UPDATE RECORD')
        self.updat.minsize(1000,600)
        self.updat.maxsize(1000,600)

        def updateahead():

            open_bin=open('dbmscsv','rb')
            records=pickle.load(open_bin)
```

```
451             open_bin.close()
452
453             exseat = self.stu_es.get()
454
455             updatedinfo = self.newlabelentry.get()
456
457             user_choose = v.get()
458             user_can = 'yes'
459             user_can2 = 'yes'
460             user_can3 = 'yes'
461
462             for i in records:
463                 if updatedinfo == '':
464                     user_can3 = 'no'
465                     break
466                 elif user_choose == 'E':
467                     user_can='yes'
468                     allow = True
469                     try:
470                         updatedinfo = int(updatedinfo)
471                     except:
472                         allow = False
473                     if updatedinfo == (int(i[1])) and allow == True and updatedinfo !='':
474                         messagebox.showinfo('Error','Can\'t update. Student already exist with the unique ID !!')
475                         user_can2 = 'no'
476                         break
477                     elif allow == False and updatedinfo != '':
478                         messagebox.showinfo('Error','Enter only numeric type !!')
479                         user_can2 = 'no'
480                         user_can3 = 'no'
481                         break
482                     elif updatedinfo == '' and allow == False:
483                         messagebox.showinfo('Error','Can\'t keep the field empty !!')
484                         user_can2='no'
485                         break
486                 elif user_choose == 'I':
487                     user_can2='yes'
488                     if updatedinfo == i[2] and updatedinfo != '':
489                         messagebox.showinfo('Error','Book with ISBN No.already issued.')
490                         user_can='no'
491                         break
492                     elif updatedinfo == '':
493                         messagebox.showinfo('Error','Can\'t keep the field empty !!')
494                         user_can='no'
495                         user_can3 = 'no'
496                         break
497                     else:
498                         user_can='yes'
499                         pass
500                 else:
```

```
                pass
            if user_can != 'no' and user_can2 != 'no' and user_can3 != 'no':
                if user_choose == 'N':
                    stat=''
                    stat2=''
                    for i in records:
                        if int(i[1]) == int(exseat):
                            i[0] = updatedinfo
                            messagebox.showinfo('Message','Record updated successfully !!')
                            stat2='found'
                            break
                        else:
                            stat='notfound'
                    if stat=='notfound' and stat2=='':
                        messagebox.showinfo('Error','No such record !!')
                elif user_choose == 'E':
                    stat=''
                    stat2=''
                    for i in records:
                        if int(i[1]) == int(exseat):
                            i[1]=updatedinfo
                            messagebox.showinfo('Message','Record updated successfully !!')
                            stat2='found'
                            break
                        else:
                            stat='notfound'
                    if stat=='notfound' and stat2=='':
                        messagebox.showinfo('Error','No such record !!')
                elif user_choose.upper()=='I':
                    stat=''
                    stat2=''
                    for i in records:
                        if int(i[1]) == int(exseat):
                            i[2]=updatedinfo
                            messagebox.showinfo('Message','Record updated successfully !!')
                            stat2='found'
                            break
                        else:
                            stat='notfound'
                    if stat=='notfound' and stat2=='':
                        messagebox.showinfo('Error','No such record !!')
                elif user_choose == 'B':
                    stat=''
                    stat2=''
                    for i in records:
                        if int(i[1]) == int(exseat):
                            i[3]=updatedinfo
                            messagebox.showinfo('Message','Record updated successfully !!')
                            stat2='found'
                            break
```

```
                    else:
                        stat='notfound'
                if stat=='notfound' and stat2=='':
                    messagebox.showinfo('Error','No such record !!')
            elif user_choose == 'IS':
                stat=''
                stat2=''
                n=-1
                for i in records:
                    if int(i[1])==user_find:
                        i[4]=user_change
                        i[5]=user_change+datetime.timedelta(days=7)
                        print('\nRecord updated successfully !!')
                        print('Issue date set to current date.')
                        info(records,user_find,1)
                        stat2='found'
                        n=4
                    else:
                        stat='notfound'
                if stat=='notfound' and stat2=='':
                    print('No such record !!')

            open_bin=open('dbmscsv','wb')
            pickle.dump(records,open_bin)
            lsttocsv(records)
        elif updatedinfo == '' and user_can == 'yes' and user_can2 == 'yes':
            messagebox.showinfo('Error','Can\'t keep the updated information field empty !!')
            open_bin=open('dbmscsv','wb')
            pickle.dump(records,open_bin)
            lsttocsv(records)

        open_bin.close()



    def updatt():
        self.clearlabel = Label(self.updat,text = '
        self.clearlabel.place(x=150,y=450)
        exseatno = self.stu_es.get()
        user_choose = v.get()

        letpass = True
        try:
            exseatno = int(exseatno)
        except:
            letpass = False
        open_bin=open('dbmscsv','rb')
        records=pickle.load(open_bin)
        open_bin.close()
        usersearch='E'
```

```
601             if usersearch.upper()=='E' and letpass == True:
602                 list_checknames=[]
603                 for i in records:
604                     list_checknames.append(int(i[1]))
605 
606 
607                 if exseatno not in list_checknames:
608                         messagebox.showinfo('Error','Unique ID doesn\'t exist !!')
609                 elif exseatno in list_checknames:
610                     if user_choose == 'N':
611                         self.newlabel = Label(self.updat, text = 'Enter updated name :                ',\
612                                       bg = 'deeppink',font = ('Arial 20 bold'))
613                         self.newlabel.place(x=150,y=450)
614                         self.newlabelentry = Entry(self.updat,textvariable = updainfo,font = ('Arial 20 italic'))
615                         self.newlabelentry.place(x=500,y=450)
616                         self.tbutupd = Button(self.updat,text = 'Update',command = updateahead,width = 25,\
617                                       height = 2,font = ('Arial 20 bold') )
618                         self.tbutupd.place(x=350,y=500)
619                     elif user_choose == 'E':
620                         self.newlabel = Label(self.updat, text = 'Enter updated Exam Seat No. :',\
621                                       bg = 'deeppink',font = ('Arial 20 bold'))
622                         self.newlabel.place(x=150,y=450)
623                         self.newlabelentry = Entry(self.updat,textvariable = updainfo,\
624                                           font = ('Arial 20 italic'))
625                         self.newlabelentry.place(x=500,y=450)
626                         self.tbutupd = Button(self.updat,text = 'Update',command = updateahead,width = 25,\
627                                       height = 2,font = ('Arial 20 bold') )
628                         self.tbutupd.place(x=350,y=500)
629                     elif user_choose == 'I':
630                         self.newlabel = Label(self.updat, text = 'Enter updated ISBN No. :          ',\
631                                       bg = 'deeppink',font = ('Arial 20 bold'))
632                         self.newlabel.place(x=150,y=450)
633                         self.newlabelentry = Entry(self.updat,textvariable = updainfo,font = ('Arial 20 italic'))
634                         self.newlabelentry.place(x=500,y=450)
635                         self.tbutupd = Button(self.updat,text = 'Update',command = updateahead,width = 25,\
636                                       height = 2,font = ('Arial 20 bold') )
637                         self.tbutupd.place(x=350,y=500)
638                     elif user_choose == 'B':
639                         self.newlabel = Label(self.updat, text = 'Enter updated Book Name :    ',bg = 'deeppink'\
640                                       ,font = ('Arial 20 bold'))
641                         self.newlabel.place(x=150,y=450)
642                         self.newlabelentry = Entry(self.updat,textvariable = updainfo,font = ('Arial 20 italic'))
643                         self.newlabelentry.place(x=500,y=450)
644                         self.tbutupd = Button(self.updat,text = 'Update',command = updateahead,width = 25,\
645                                       height = 2,font = ('Arial 20 bold') )
646                         self.tbutupd.place(x=350,y=500)
647                     elif user_choose.upper() == 'IS':
648                         user_change=datetime.date.today()
649                 else:
650                     pass
```

```
651             elif letpass == False and exseatno == '':
652                 messagebox.showinfo('Error','Can\'t keep the \'Seat No.\' field empty !!')
653             else:
654                 messagebox.showinfo('Error','Enter only numeric type !!')
655 
656 
657 
658         imglogo = Image.open('/Users/kandarpsolanki/Desktop/Codings/logo.png')
659         imglogo = imglogo.resize((250,110),Image.ANTIALIAS)
660         photo = ImageTk.PhotoImage(imglogo)
661         self.lab_pic = Label(self.updat,image = photo,borderwidth = 1,relief = 'sunken')
662         self.lab_pic.image = photo
663         self.lab_pic.config(image = photo)
664         self.lab_pic.pack(fill=BOTH)
665 
666 
667         v = StringVar()
668         v.set(name)
669 
670 
671         self.uplab = Label(self.updat,text = 'What to Update ?',bg = 'deeppink',font = ('Arial 20 bold'))
672         self.uplab.place(x=377,y=265)
673 
674 
675         self.eslabel = Label(self.updat,text = 'Exam Seat no. of student (presently in records) :',\
676                         bg = 'deeppink',font = ('Arial 20 bold'))
677         self.eslabel.place(x=51,y=200)
678 
679         self.stu_es = Entry(self.updat,textvariable = exseat,font = ('Arial 20 bold'))
680         self.stu_es.place(x=600,y=200)
681 
682 
683         self.r1 = Radiobutton(self.updat,text = 'Name',variable = v,value = name,bg = 'deeppink',\
684                 font = ('Arial 20 bold'))
685         self.r1.place(x=180,y=300)
686 
687 
688         self.r2 = Radiobutton(self.updat,text = 'Exam Seat No.',variable = v,value = exam,bg = 'deeppink',\
689                 font = ('Arial 20 bold'))
690         self.r2.place(x=287,y=300)
691 
692 
693         self.r3 = Radiobutton(self.updat,text = 'ISBN No.',variable = v,value = isbn,bg = 'deeppink',\
694                 font = ('Arial 20 bold'))
695         self.r3.place(x=475,y=300)
696 
697         self.r4 = Radiobutton(self.updat,text = 'Book Name',variable = v,value = book,bg = 'deeppink',\
698                 font = ('Arial 20 bold'))
699         self.r4.place(x=610,y=300)
700 
```

```
        self.tbutchoose = Button(self.updat,text = 'Choose',command = updatt,width = 7,height = 1,\
                                 font = ('Arial 20 bold'))
        self.tbutchoose.place(x=430,y=350)

        self.infolabel = Label(self.updat,text = '*Press CHOOSE to enter the updated information',\
                               bg = 'deeppink',font = ('Arial 18 italic'))
        self.infolabel.place(x=280,y=385)

        self.labelackn = Label(self.updat,text='Designed and Developed by: KANDARP SOLANKI',fg='black',\
                               bg='deeppink',font = ('Operator Mono Medium',13,'italic'))
        self.labelackn.place(x=678,y=574)



    # Delete All Records ~ Library App #
    def deletingall(self):
        open_bin=open('dbmscsv','rb')
        record=pickle.load(open_bin)
        open_bin.close()

        if record != []:
            final = messagebox.askyesno('Warning','Are you sure you want to delete all records ? (Y/N)')
            if final:
                open_bin=open('dbmscsv','wb')
                lst_std=[]
                pickle.dump(lst_std,open_bin)
                open_bin.close()
                makenew = [['','','','','','']]
                lsttocsv(makenew)
                messagebox.showinfo('Status','All records deleted successfully !!')
            else:
                messagebox.showinfo('Status','Records are safe !!')
        elif record==[]:
            messagebox.showinfo('Error','No records present !!')



    # Delete Particular Record ~ Library App #
    def deletingparticular(self):

        isbnreturn = StringVar()
        isbnreturn.set('')


        def delparcmd():
            open_bin=open('dbmscsv','rb')
            records=pickle.load(open_bin)
            open_bin.close()
            open_bin=open('dbmscsv','wb')
            usersearch = isbnreturn.get()
```

```
751             lst_temp=[]
752             stud_count=0
753 
754             if usersearch != '':
755                 confirm = messagebox.askyesno('Confirm','Are you sure you want to delete record ?')
756                 confirm2 = False
757                 if confirm:
758                     for i in records:
759                         if i[2] == usersearch:
760                             stud_count+=1
761                             labeltext = ('+------------------+\n'+'| Name                          |-->'+\
762                             i[0]+'\n+------------------+'+'\n| Exam Seat No.         |-->'+str(i[1])+\
763                             '\n+------------------+'+'\n| ISBN No.                 |-->'+i[2]+\
764                             '\n+------------------+'+'\n| Book Name              |-->'+i[3]+\
765                             '\n+------------------+'+'\n| Issue Date               |-->'+str(i[4])+\
766                             '\n+------------------+'+'\n| Due Date                 |-->'+str(i[5])+\
767                             '\n+------------------+\n')
768                             messagebox.showinfo('Student Record',labeltext)
769                             confirm2 = messagebox.askyesno('Attention','Once deleted, it can\'t be restored !!\
770                             \nAre you sure you want to permanently delete the record ?')
771                             if confirm2:
772                                 pass
773                             else:
774                                 lst_temp.append(i)
775                         else:
776                             lst_temp.append(i)
777                     pickle.dump(lst_temp,open_bin)
778                     lsttocsv(lst_temp)
779                 else:
780                     pickle.dump(records,open_bin)
781                     lsttocsv(records)
782 
783                 if confirm and confirm2:
784                     show = '\nRecord under ISBN \''+usersearch+'\' deleted successfully.'
785                     messagebox.showinfo('Status',show)
786                 elif stud_count == 0 and confirm == True:
787                     show = 'No record found under ISBN No. \''+usersearch+'\' !!'
788                     messagebox.showerror('Status',show)
789                 else:
790                     pass
791             else:
792                 messagenew = 'Can\'t keep the ISBN No. field empty !!'
793                 messagebox.showerror('Error',messagenew)
794                 pickle.dump(records,open_bin)
795                 lsttocsv(records)
796             open_bin.close()
797 
798 
799         self.delpar = Toplevel(self.window)
800         self.delpar.geometry('1000x600')
```

```
801         self.delpar.config(background ='deeppink')
802         self.delpar.title('DELETE PARTICULAR RECORD')
803         self.delpar.minsize(1000,600)
804         self.delpar.maxsize(1000,600)
805
806
807
808         photo=ImageTk.PhotoImage(Image.open('logo.png'))
809         lab_pic = Label(self.delpar,image = photo)
810         lab_pic.image = photo
811         lab_pic.config(image = photo)
812         lab_pic.pack(fill=BOTH)
813
814         self.isbnlab = Label(self.delpar,text = 'Enter ISBN No. of book being returned :',bg = 'deeppink',\
815                              font = ('Arial 20 bold'))
816         self.isbnlab.place(x=100,y=200)
817
818
819         self.isbnentry = Entry(self.delpar,textvariable = isbnreturn,font = ('Arial 20 italic'))
820         self.isbnentry.place(x=600,y=200)
821
822         self.butdel = Button(self.delpar,text = 'DELETE',command = delparcmd,width = 9,height = 2,\
823                              font = ('Arial 20 bold') )
824         self.butdel.place(x=430,y=350)
825
826         self.labelackn = Label(self.delpar,text='Designed and Developed by: KANDARP SOLANKI',fg='black',\
827                                bg='deeppink',font = ('Operator Mono Medium',13,'italic'))
828         self.labelackn.place(x=678,y=574)
829
830
831
832     # Initialising of Class #
833     def __init__(self):
834         self.loginscreen()
835
836 GUI()
```

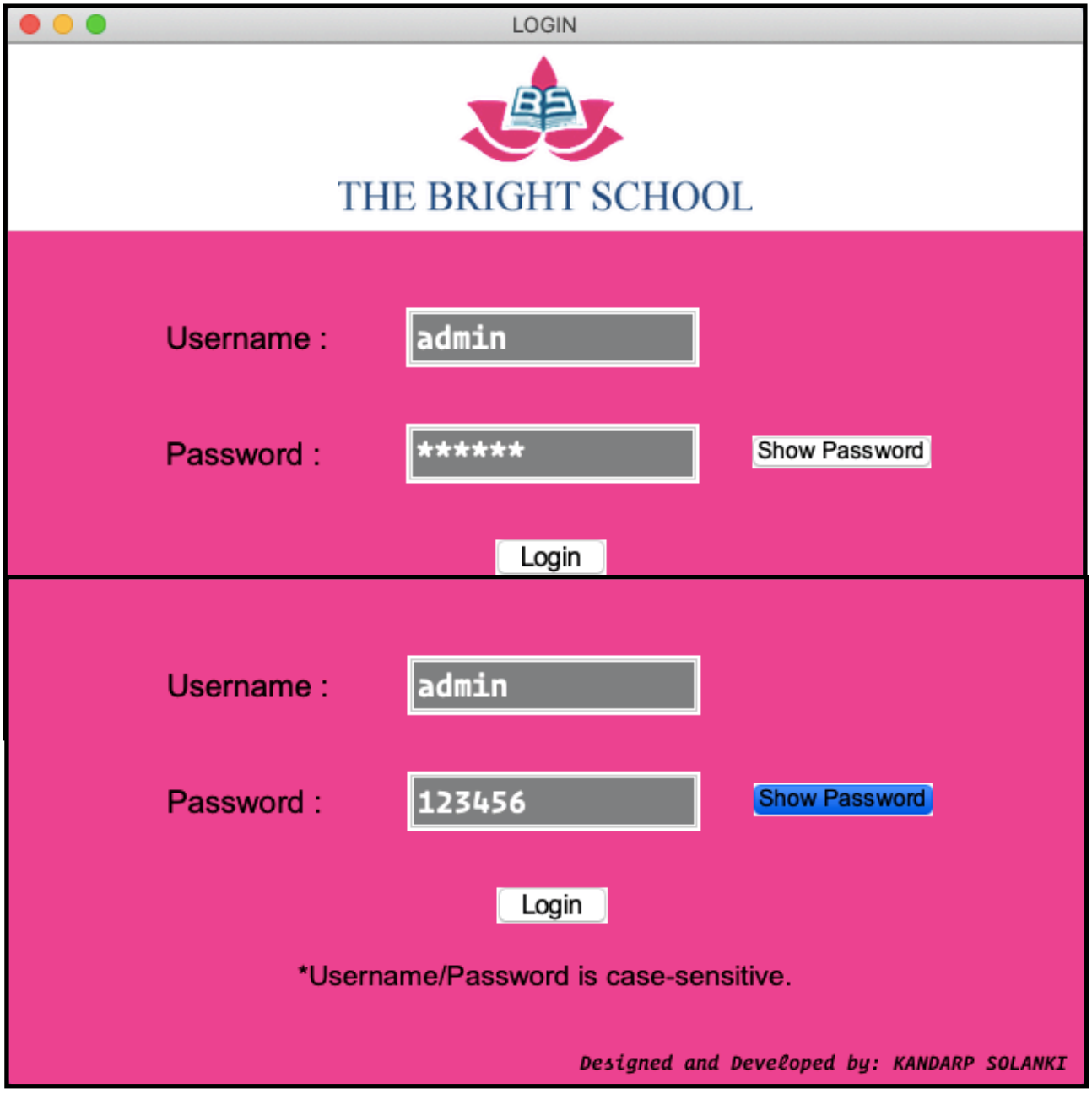
The following modules should be installed before executing the programs:

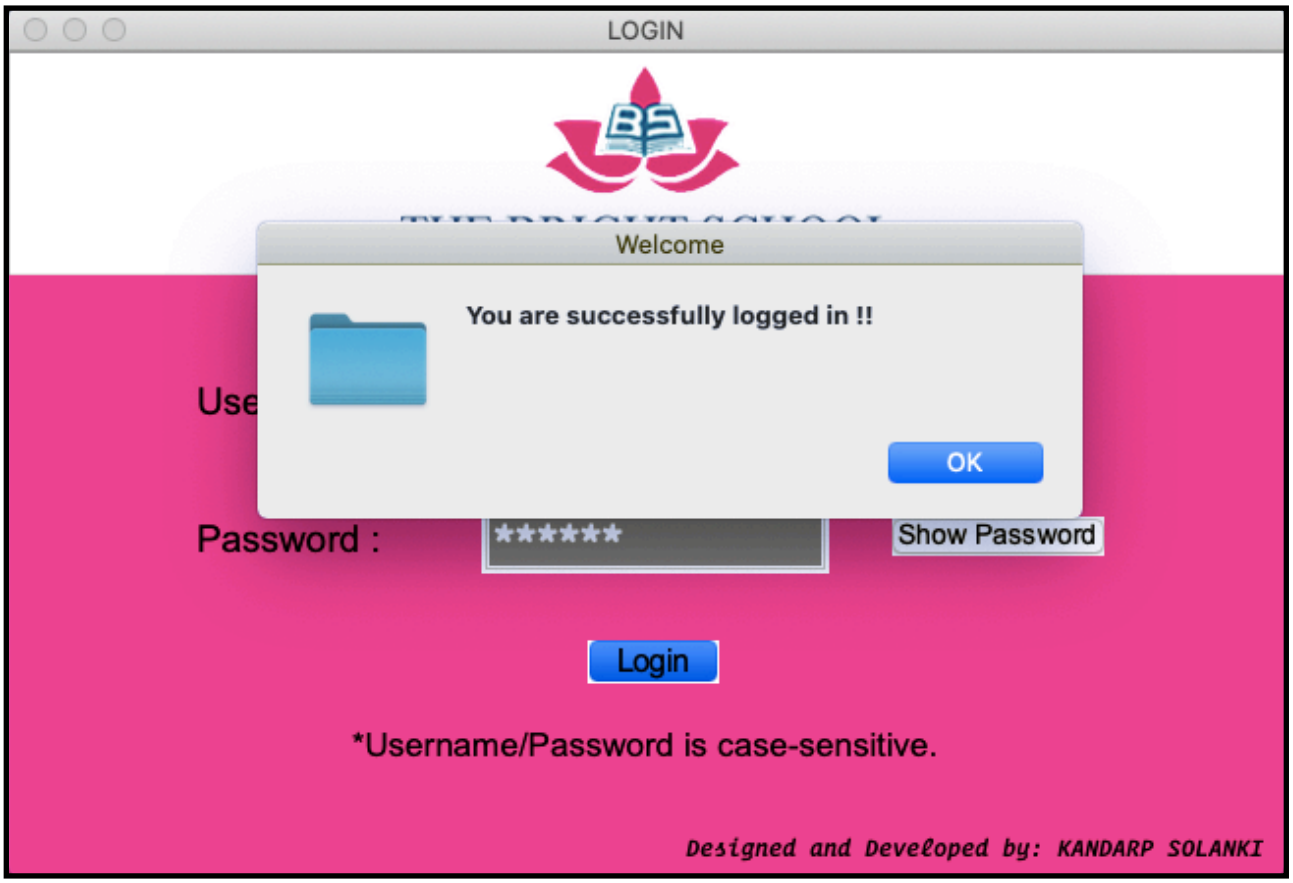
- **PIL** - It adds support for opening, manipulating, and saving many different image file formats.
- **Pickle** - To convert a Python object into a byte stream to store it in a file/database.
- **Datetime & Time** - It supplies classes to work with date and time.
- **Subprocess** - To open an application (here CSV file)
- **DBMSlibrary_CSV** - It is a user-defined module created to support some functions of adding/modifying data being added and fetched.

NOTE - *Tkinter* is a framework (a collection of packages or modules) and its the only one that's built into the Python standard library.

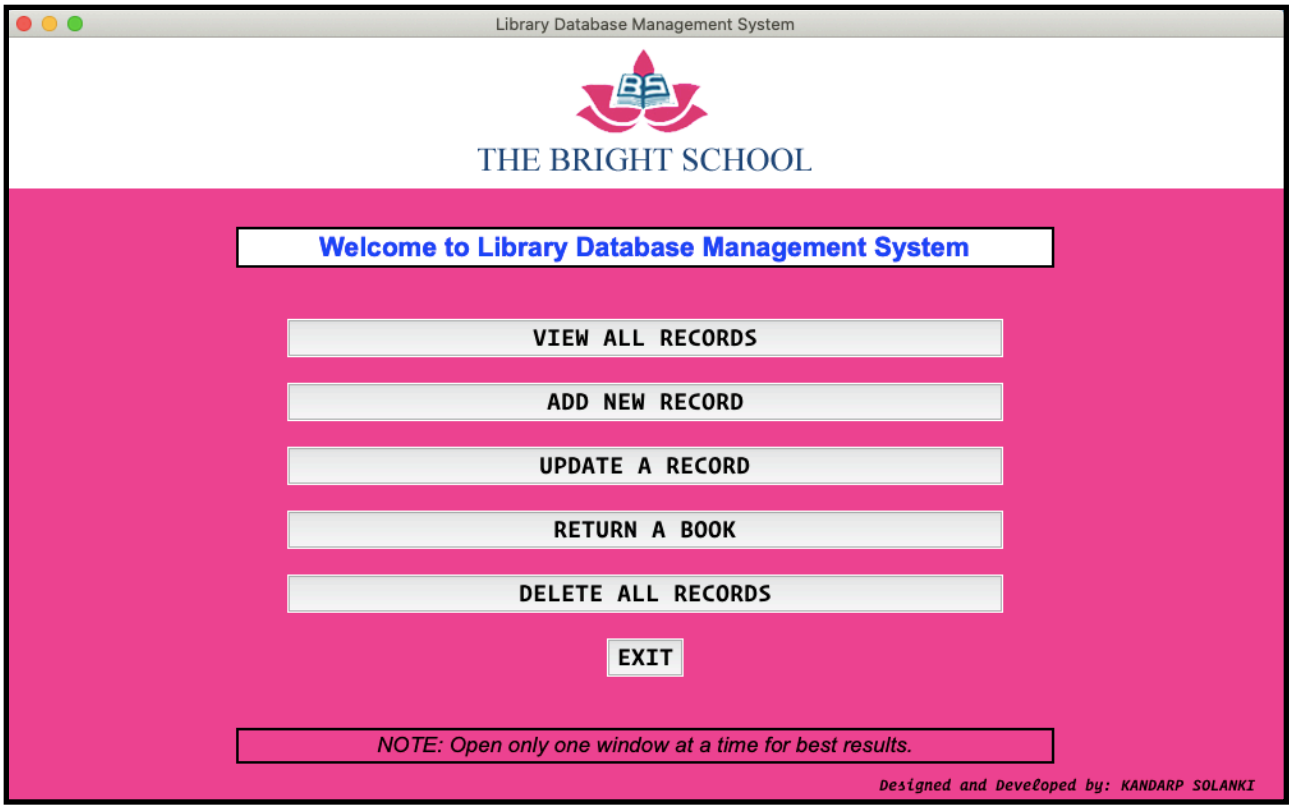
OUTPUT

LOGIN PAGE




LOGIN
Welcome
You are successfully logged in !!
OK
Password :
******
Show Password
Login
*Username/Password is case-sensitive.
Designed and Developed by: KANDARP SOLANKI

HOME PAGE

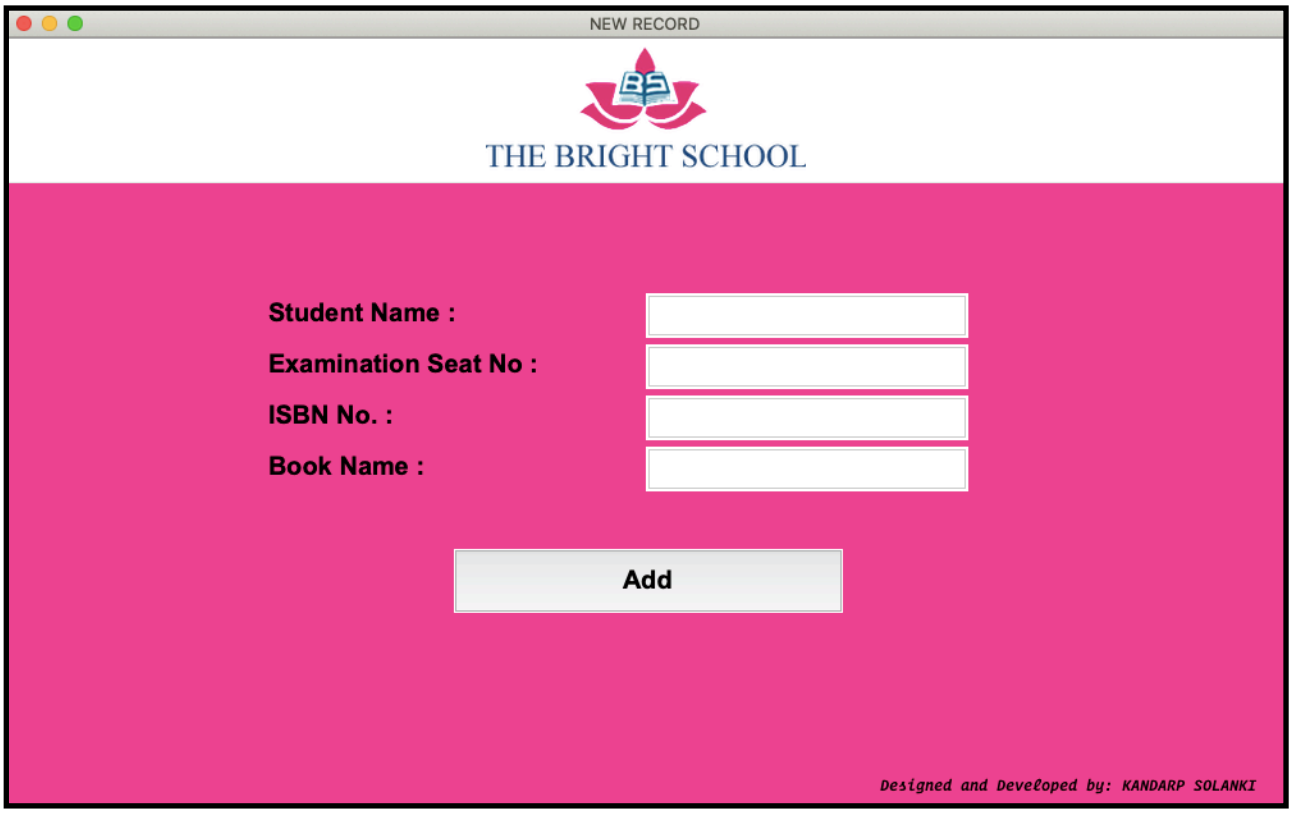

Library Database Management System
THE BRIGHT SCHOOL
Welcome to Library Database Management System
VIEW ALL RECORDS
ADD NEW RECORD
UPDATE A RECORD
RETURN A BOOK
DELETE ALL RECORDS
EXIT
NOTE: Open only one window at a time for best results.
Designed and Developed by: KANDARP SOLANKI

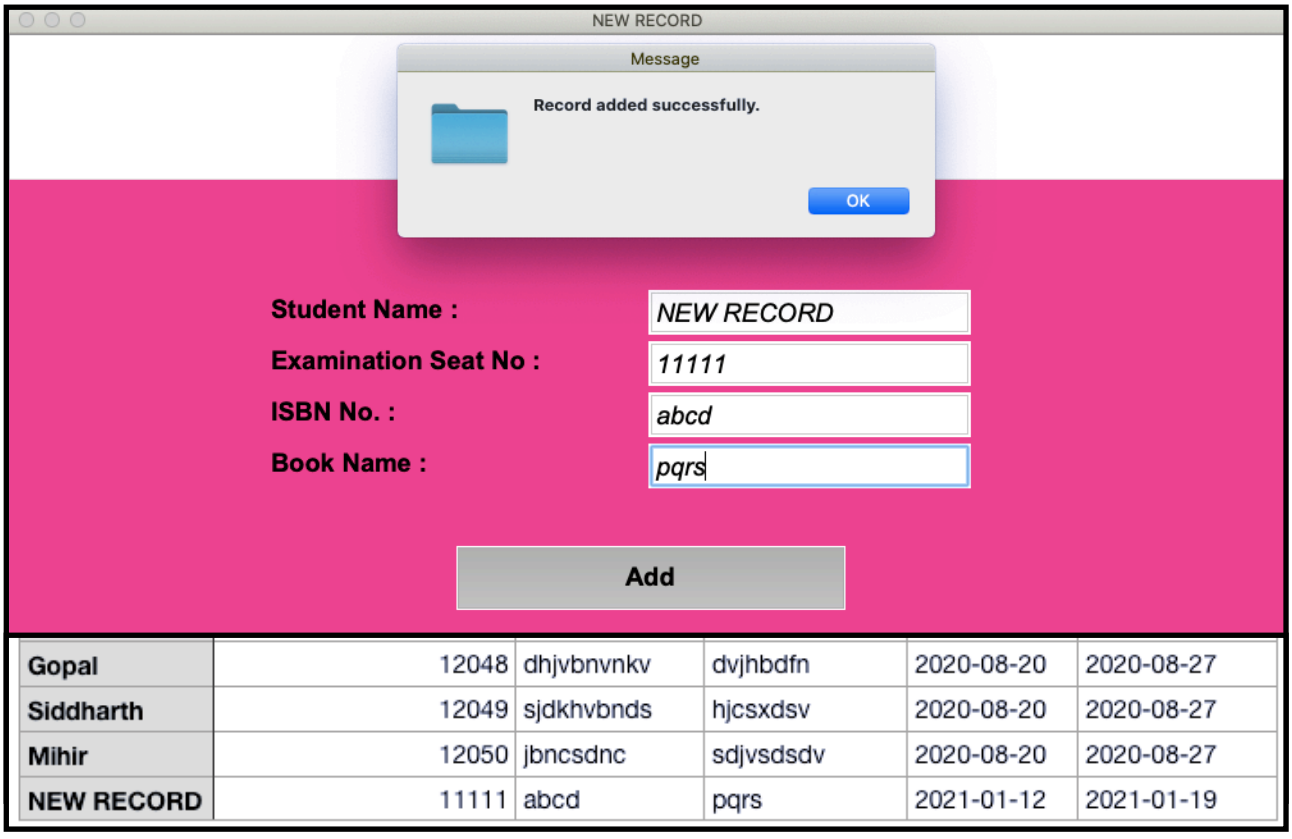
Opening of CSV File using 'VIEW ALL RECORDS'

studentdatabase

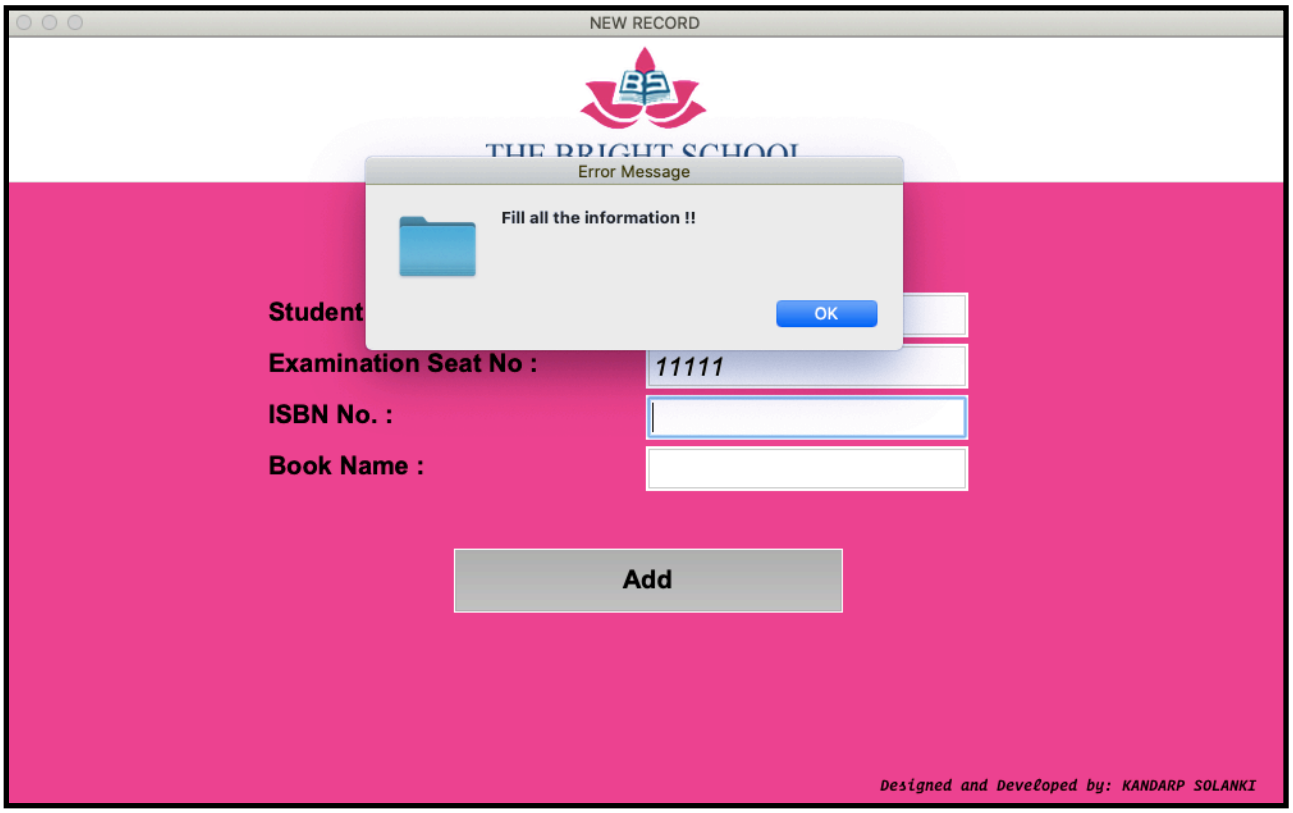
Name	Examination Seat No.	ISBN No.	Book Name	Issue Date	Due Date
Kandarp	12001	dkjhschj	sdbcksdb	2020-08-20	2020-08-27
Rahul	12002	ndkja	flsnfkjsd	2020-08-20	2020-08-27
Mehul	12003	dbfhjs	webkfjwe	2020-08-20	2020-08-27
Sohan	12004	bkfhds	wbkehdf	2020-08-20	2020-08-27
Mohan	12005	snjk	sbkdjvd	2020-08-20	2020-08-27
Rohan	12006	dfjkbshj	bhvkf	2020-08-20	2020-08-27
Karan	12007	sdjfsdkfs	skhdvf	2020-08-20	2020-08-27
Ranveer	12008	svkhd	sbdvj	2020-08-20	2020-08-27
Akshay	12009	dhvbdjfhb	dhfvb	2020-08-20	2020-08-27
Suresh	12010	dshcjsd bc	sdchs	2020-08-20	2020-08-27
Ravindra	12011	sfkdchbd	sdvnb	2020-08-20	2020-08-27
Pradeep	12012	bsc sd	shdbjhsdv	2020-08-20	2020-08-27
Rajesh	12013	hscjdh	shdv	2020-08-20	2020-08-27
Mahesh	12014	vbdfhjfgbf	vhsjdhfd	2020-08-20	2020-08-27
Nikul	12015	ndjkfbv	dvhfbd	2020-08-20	2020-08-27
Kiran	12016	vjshkdv	hdvbfsdlv	2020-08-20	2020-08-27
Suman	12017	sjhdsd	sjdnvjks	2020-08-20	2020-08-27
Garima	12018	schjd	dvhdjvbd	2020-08-20	2020-08-27
Manish	12019	bsfbvgbf	djnfvdk	2020-08-20	2020-08-27
Ganesh	12025	svdjkch	jscuwkjsdc	2020-08-20	2020-08-27
Reshma	12031	sdvnasdv	djksnvcjdknsv	2020-08-20	2020-08-27
Sunita	12032	dsjn	sd	2020-08-20	2020-08-27
Rajeev	12033	sdbn kjndc	jkdnscmkjd	2020-08-20	2020-08-27
Akhilesh	12034	vdn njkfdnjfekd	jkbfdkf	2020-08-20	2020-08-27
Ranjan	12035	hjhjfdhjfdhjf	sdbnnbdsk	2020-08-20	2020-08-27
Manav	12036	fdbgfn	gfndghndgh	2020-08-20	2020-08-27
Bhavya	12037	hjdshjhdjfv	djhdsvhjdkf	2020-08-20	2020-08-27
Raman	12038	behedhjehjfd	vhjrevdhrjv	2020-08-20	2020-08-27
Sangita	12039	bnvfskjxsjhdv	svbdskjv	2020-08-20	2020-08-27
Isha	12040	shdvjsdkj	dvkbnd	2020-08-20	2020-08-27
Naman	12041	jsdfbjshdn	kvnjkdfmv	2020-08-20	2020-08-27
Chaitanya	12042	fvdmdfb	djhjvdkfj	2020-08-20	2020-08-27
Parth	12043	gfbfgnfg	mghjdf	2020-08-20	2020-08-27
Yatharth	12044	everger	ergerger	2020-08-20	2020-08-27
Swayam	12045	efbrtgesd	wefercds	2020-08-20	2020-08-27
Aditya	12046	vhdjbdjhfv	dhvbnvnsmdv	2020-08-20	2020-08-27
Deep	12047	dsjvkhsjbdv	vsdvs	2020-08-20	2020-08-27
Gopal	12048	dhjvbnvnkv	dvjhbdfn	2020-08-20	2020-08-27
Siddharth	12049	sjdkhvbnds	hjcsxdsv	2020-08-20	2020-08-27
Mihir	12050	jbncsdnc	sdjvsdsdv	2020-08-20	2020-08-27

Adding new records using 'ADD NEW RECORD'

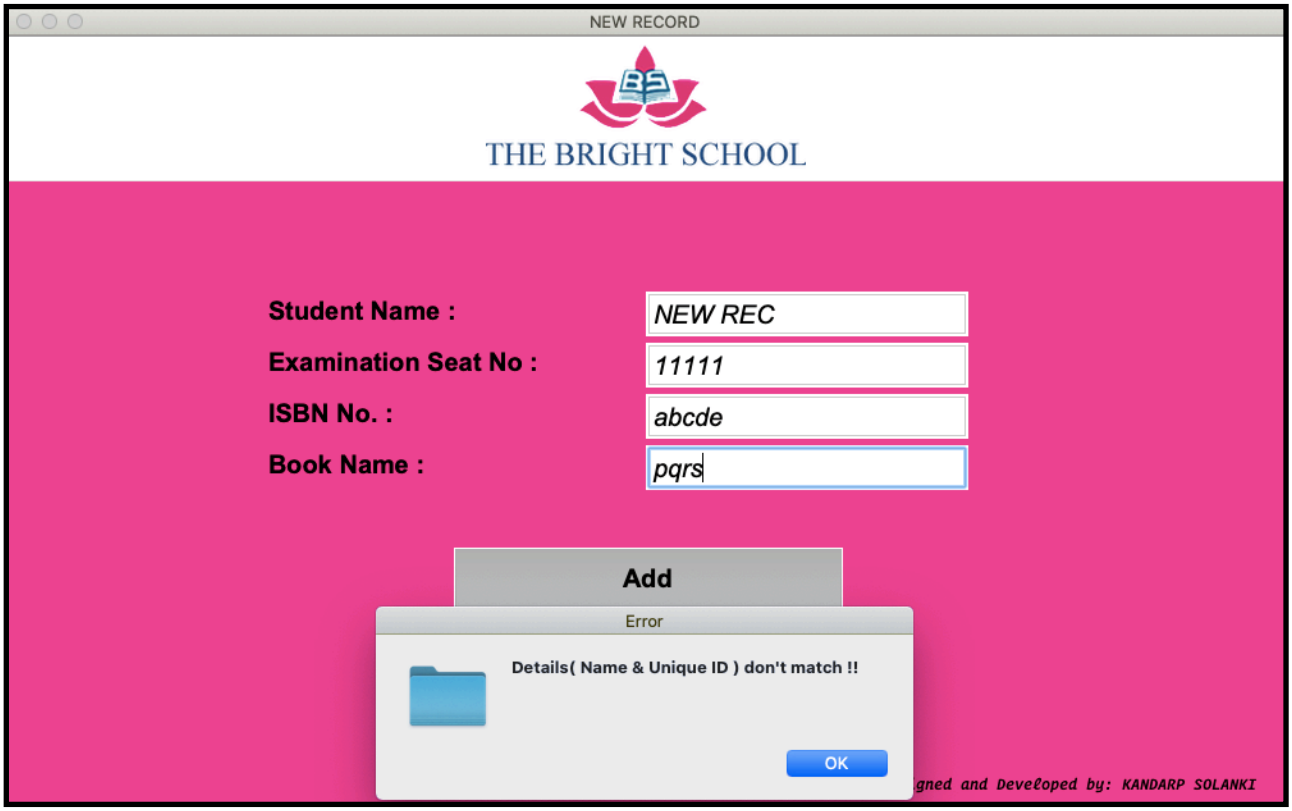


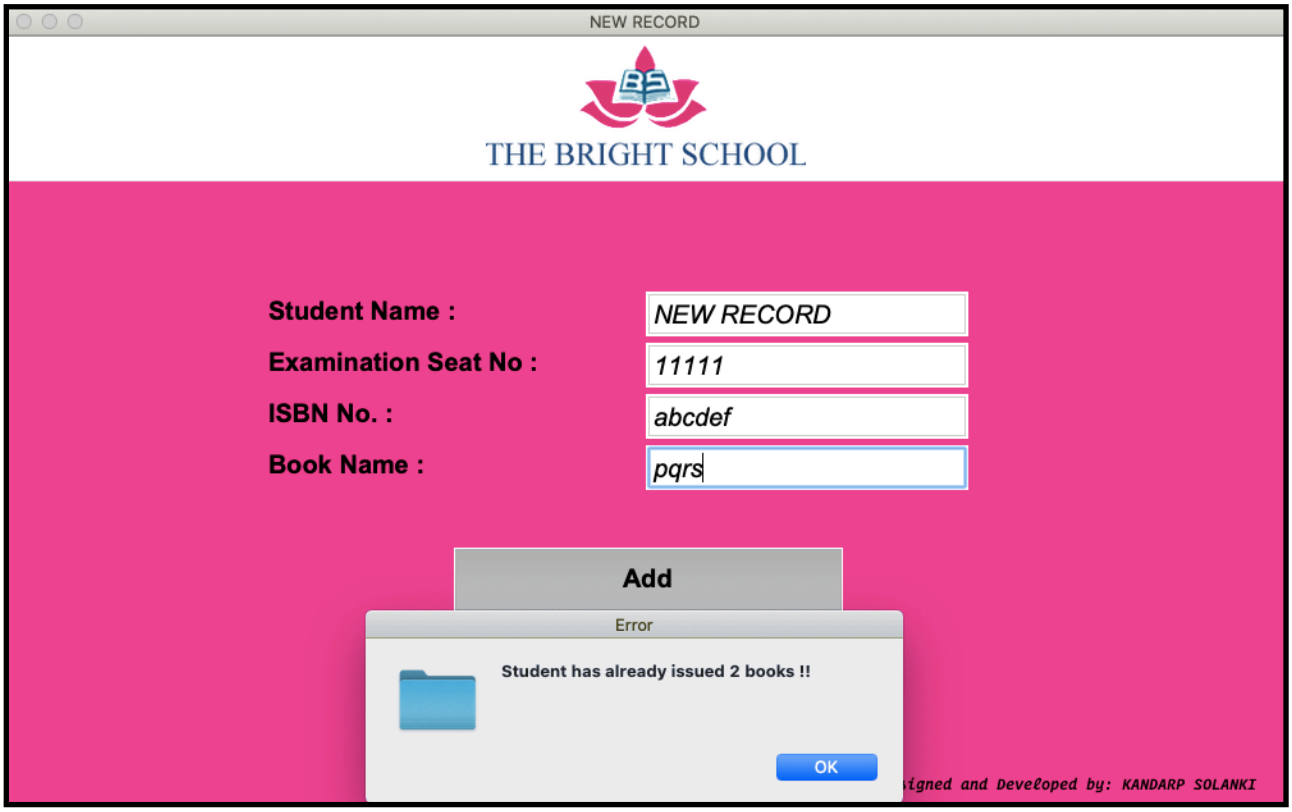


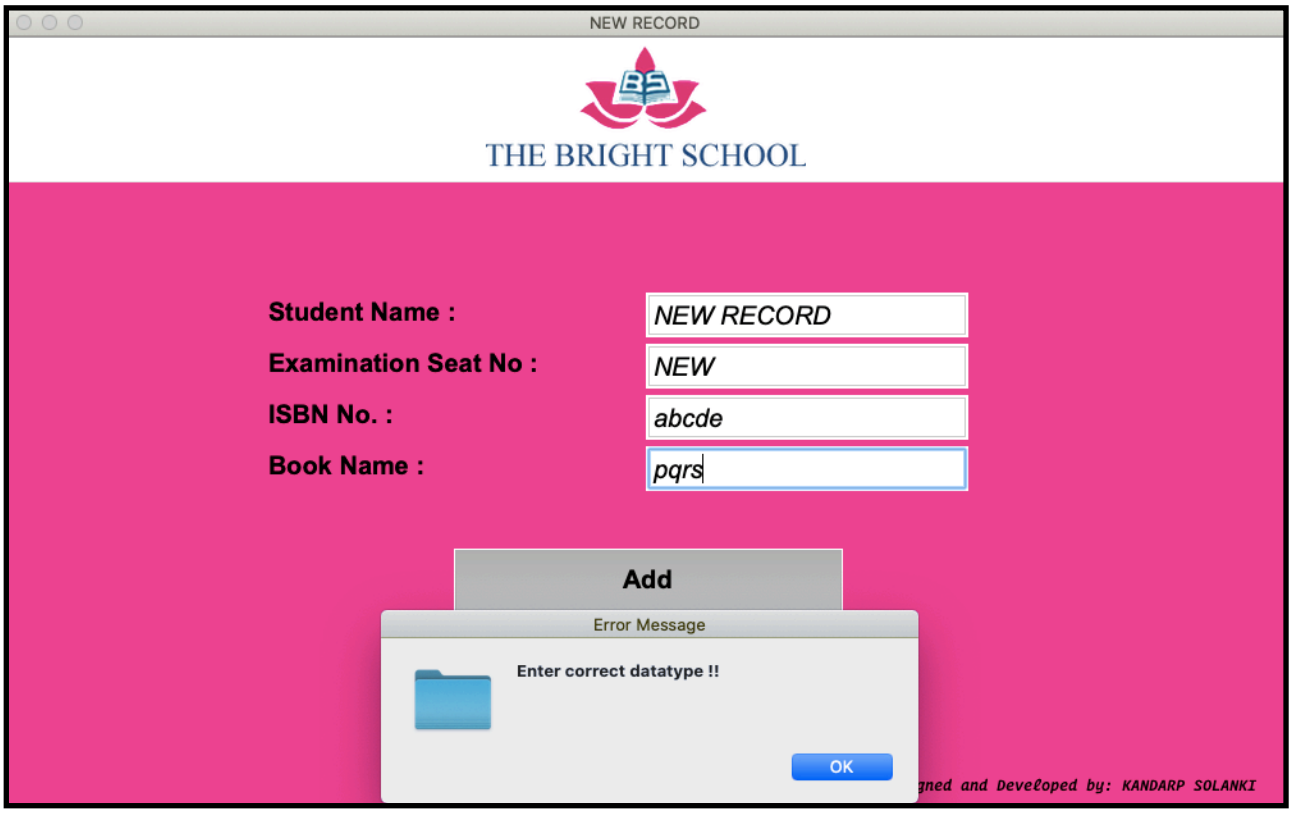
Gopal	12048	dhjvbnvnkv	dvjhbdfn	2020-08-20	2020-08-27
Siddharth	12049	sjdkhvbnds	hjcsxdsv	2020-08-20	2020-08-27
Mihir	12050	jbncsdnc	sdjvsdsdv	2020-08-20	2020-08-27
NEW RECORD	11111	abcd	pqrs	2021-01-12	2021-01-19


NEW RECORD
Error Message
Fill all the information !!
OK
Student
Examination Seat No :
11111
ISBN No. :
Book Name :
Add
Designed and Developed by: KANDARP SOLANKI

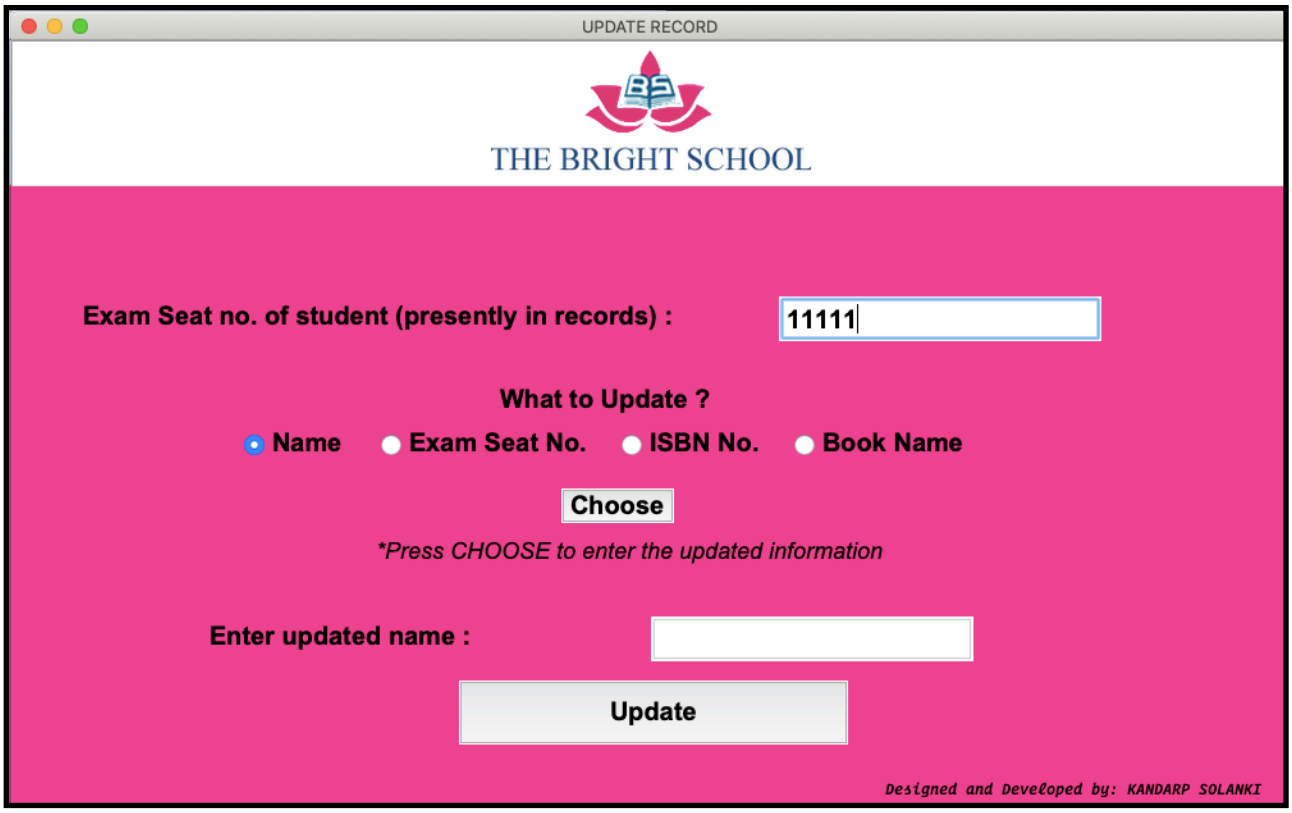

NEW RECORD
Error
The book with ISBN number already issued.
OK
Student Name :
NEW RECORD 2
Examination Seat No :
11112
ISBN No. :
abcd
Book Name :
pqrs
Add
Designed and Developed by: KANDARP SOLANKI

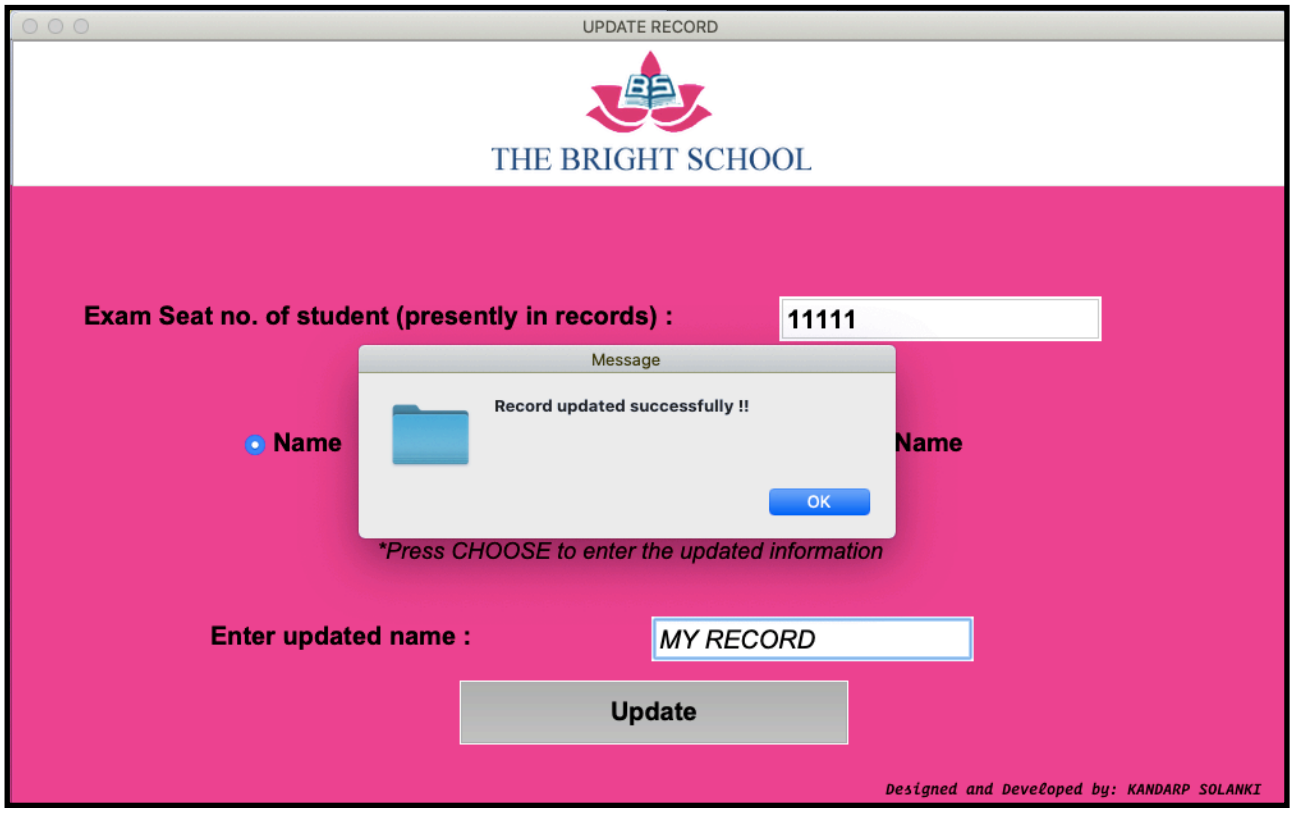

NEW RECORD
THE BRIGHT SCHOOL
Student Name :
NEW REC
Examination Seat No :
11111
ISBN No. :
abcde
Book Name :
pqrs
Add
Error
Details(Name & Unique ID) don't match !!
OK
gned and Developed by: KANDARP SOLANKI


NEW RECORD
THE BRIGHT SCHOOL
Student Name :
NEW RECORD
Examination Seat No :
11111
ISBN No. :
abcdef
Book Name :
pqrs
Add
Error
Student has already issued 2 books !!
OK
igned and Developed by: KANDARP SOLANKI

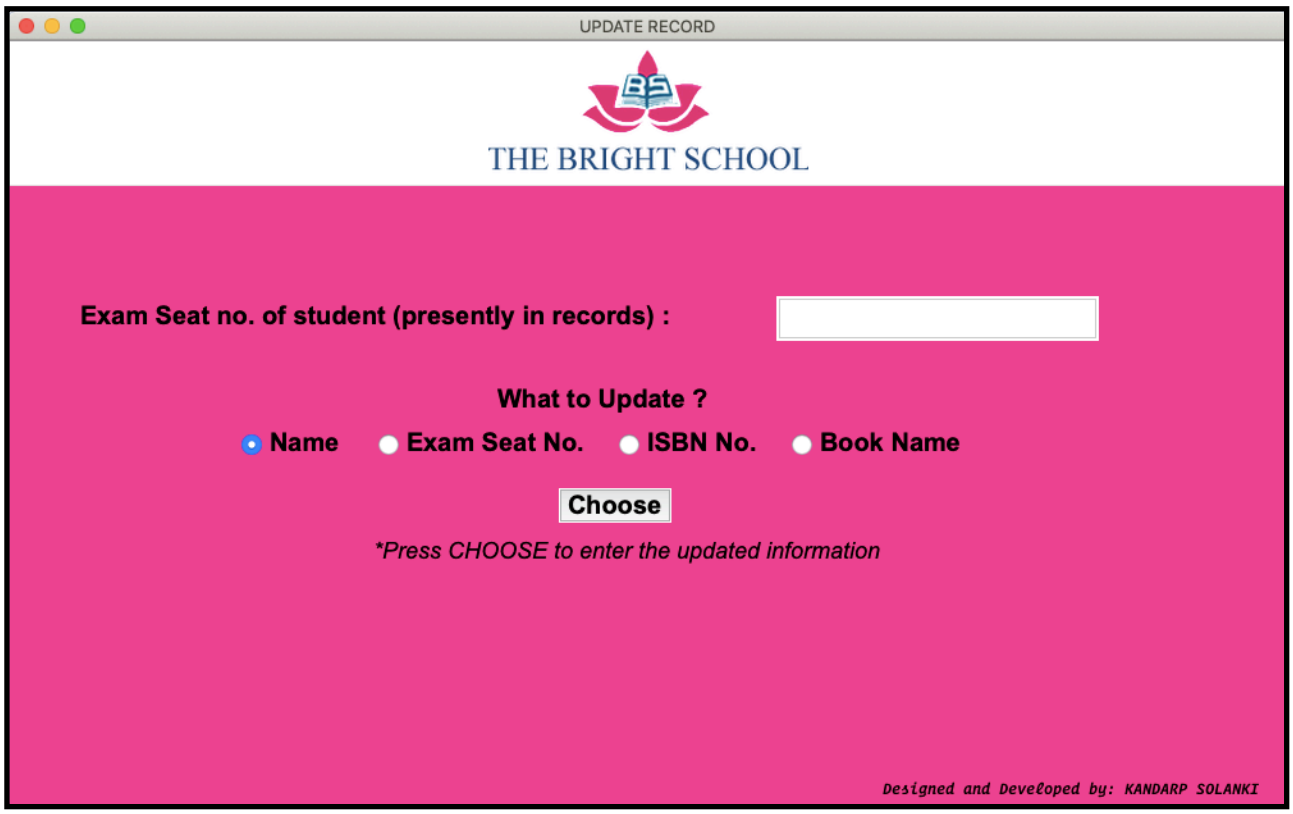


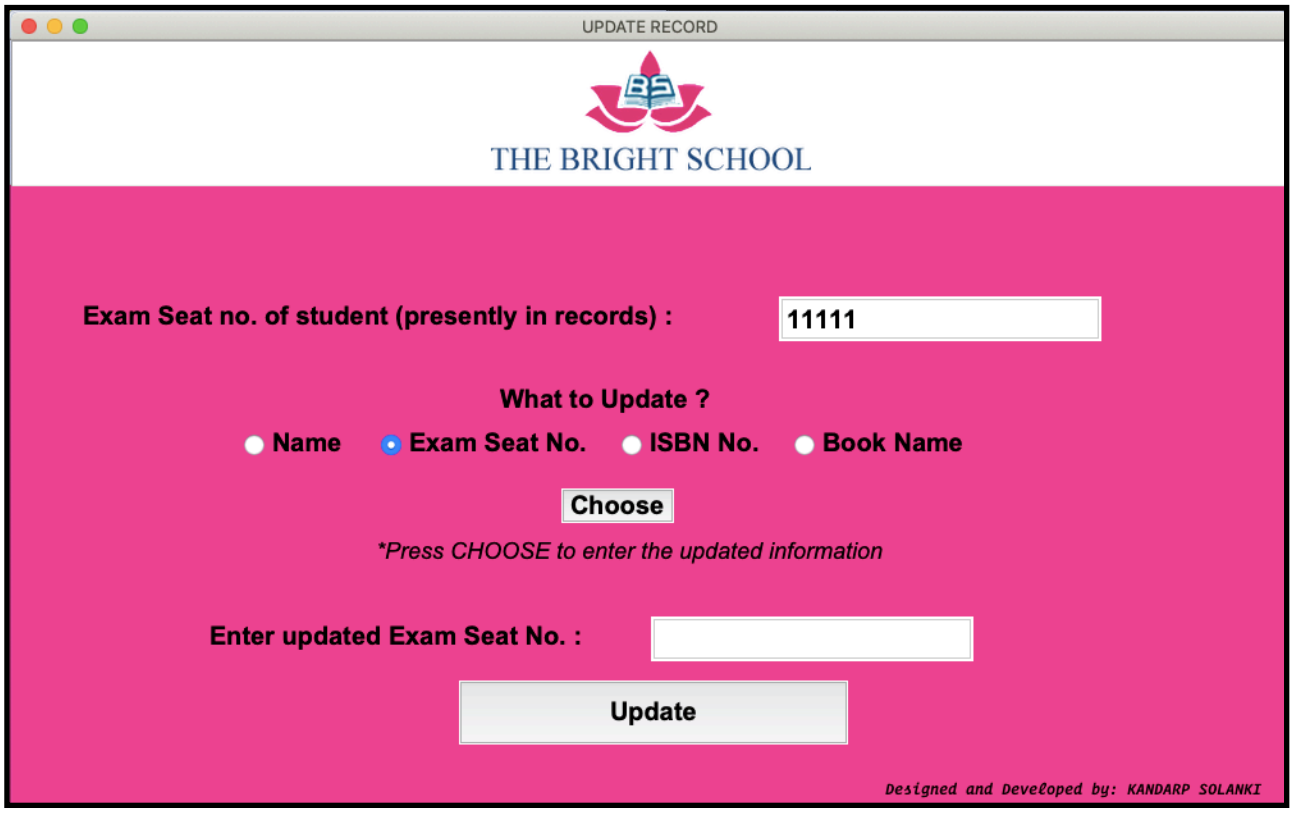
Updating existing records using 'UPDATE A RECORD'

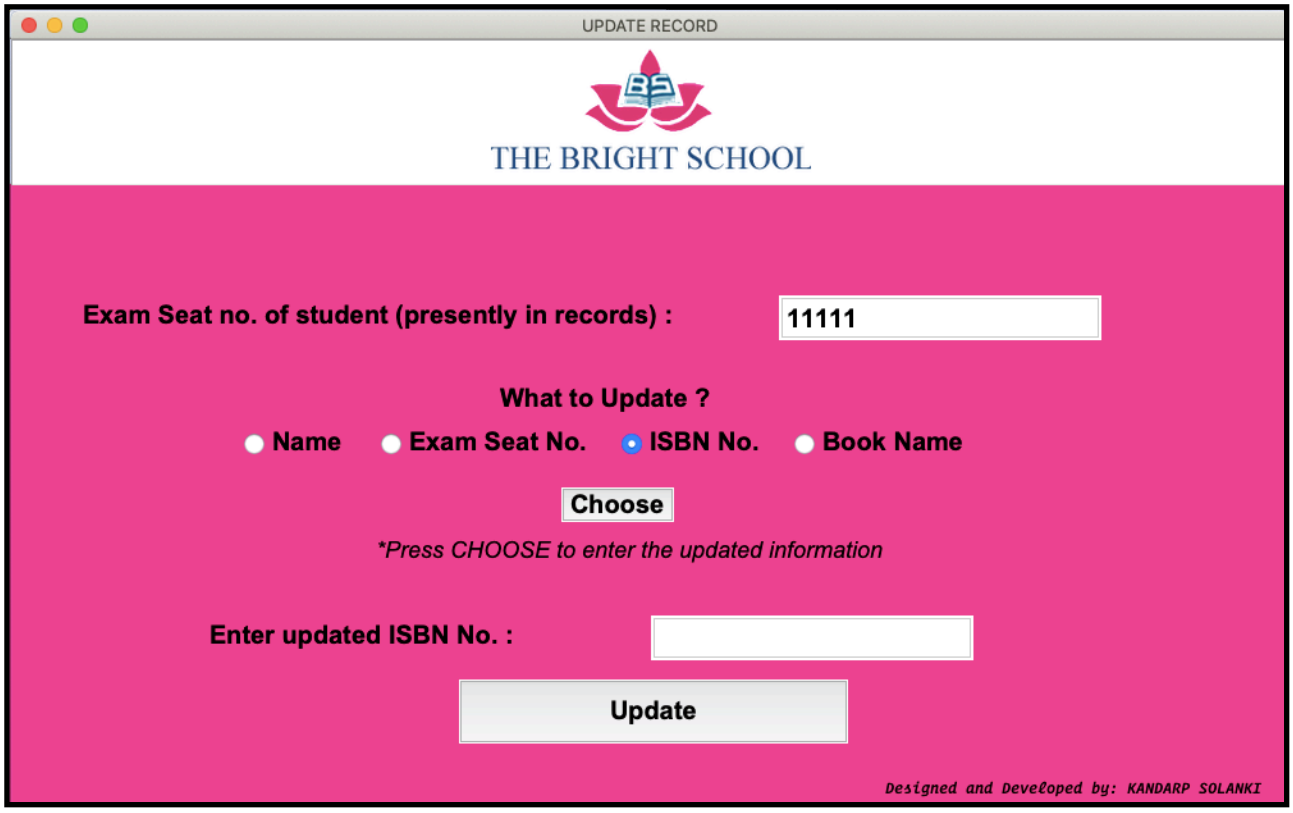


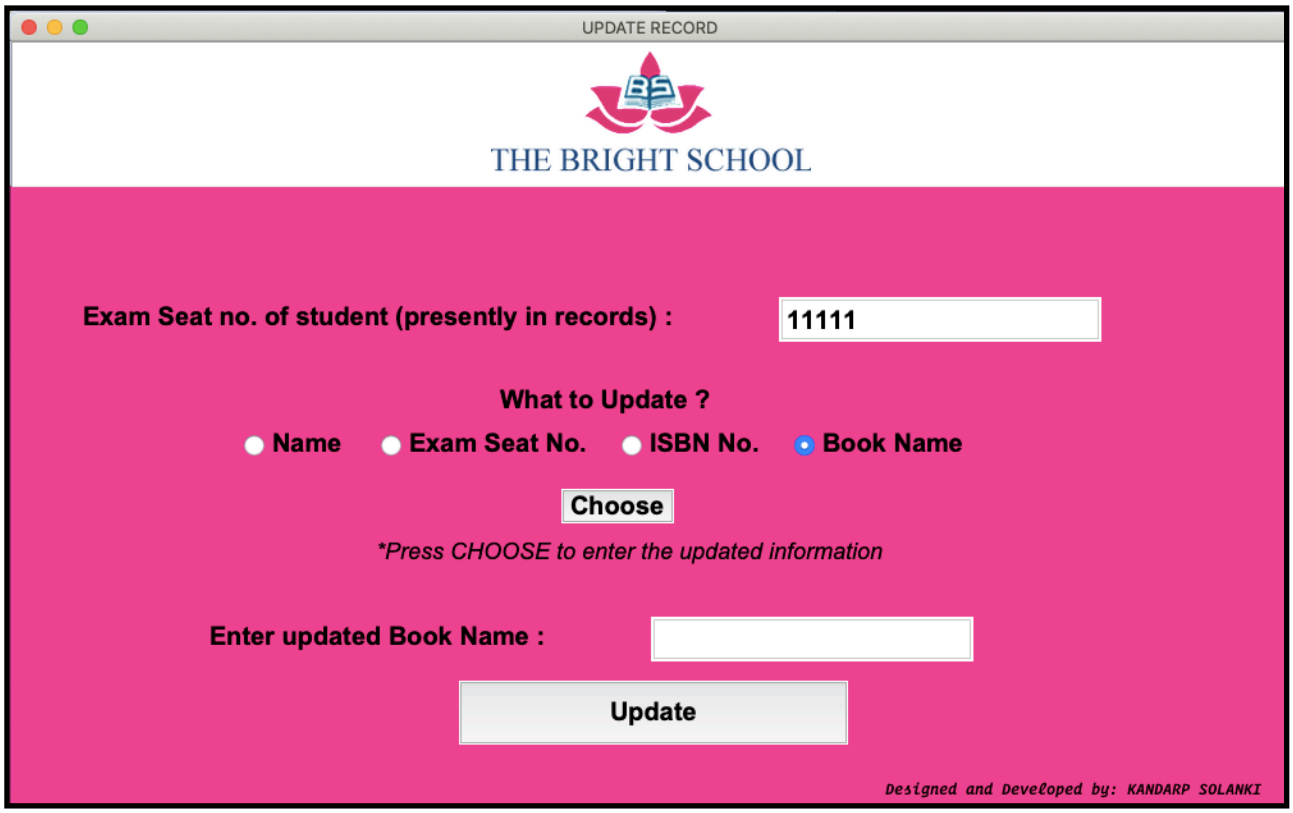


Sangita	12039	bnvfskjxsjhdv	svbdskjv	2020-08-20	2020-08-27
Isha	12040	shdvjsdkj	dvkbnd	2020-08-20	2020-08-27
Naman	12041	jsdfbjshdn	kvnjkdfmv	2020-08-20	2020-08-27
Chaitanya	12042	fvdmdfb	djhjvdkfj	2020-08-20	2020-08-27
Parth	12043	gfbfgnfg	mghjdf	2020-08-20	2020-08-27
Yatharth	12044	everger	ergerger	2020-08-20	2020-08-27
Swayam	12045	efbrtgesd	wefercds	2020-08-20	2020-08-27
Aditya	12046	vhdjbdjhfv	dhvbnvnsmdv	2020-08-20	2020-08-27
Deep	12047	dsjvkhsjbdv	vsdvs	2020-08-20	2020-08-27
Gopal	12048	dhjvbnvnkv	dvjhbdfn	2020-08-20	2020-08-27
Siddharth	12049	sjdkhvbnds	hjcsxdsv	2020-08-20	2020-08-27
Mihir	12050	jbncsdnc	sdjvsdsdv	2020-08-20	2020-08-27
MY RECORD	11111	abcd	pqrs	2021-01-11	2021-01-18

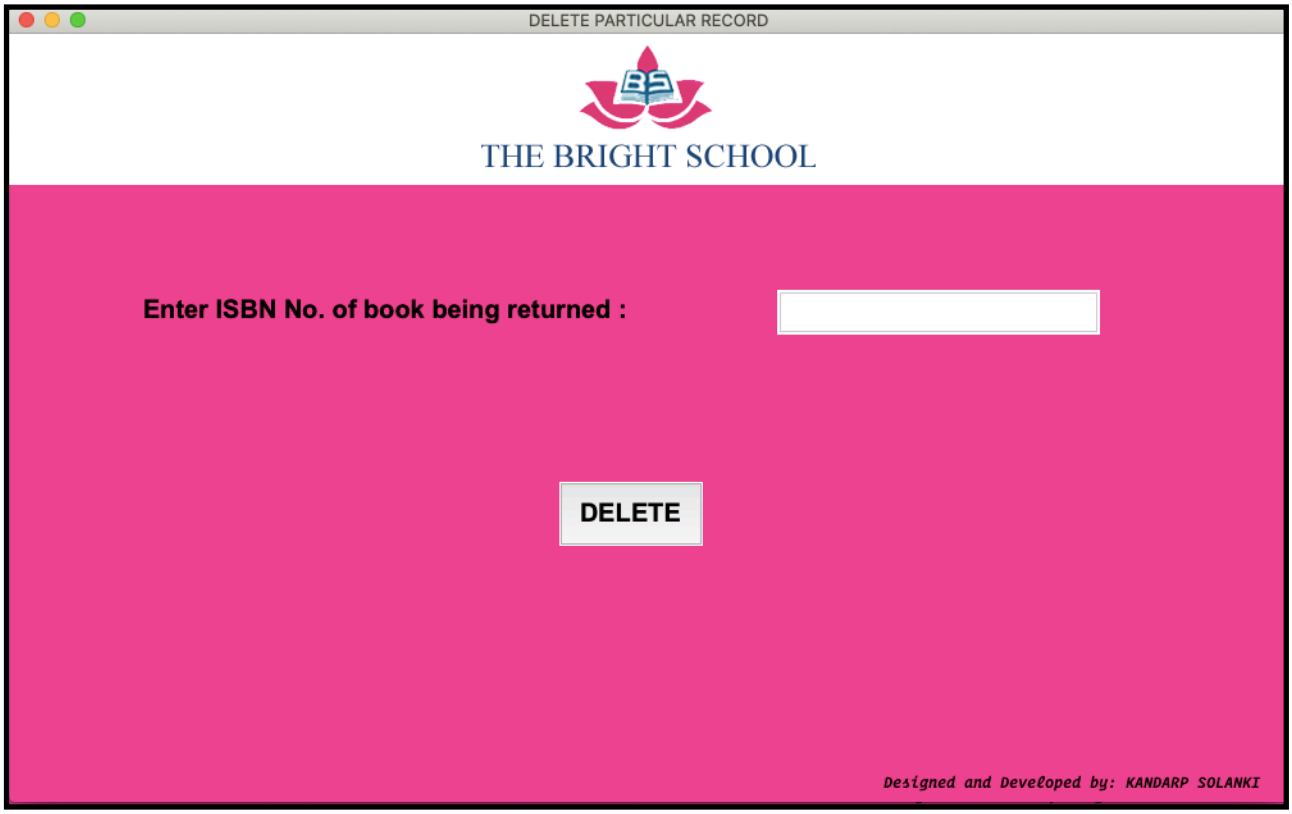

UPDATE RECORD
THE BRIGHT SCHOOL
Exam Seat no. of student (presently in records) :
What to Update ?
Name
Exam Seat No.
ISBN No.
Book Name
Choose
*Press CHOOSE to enter the updated information
Designed and Developed by: KANDARP SOLANKI

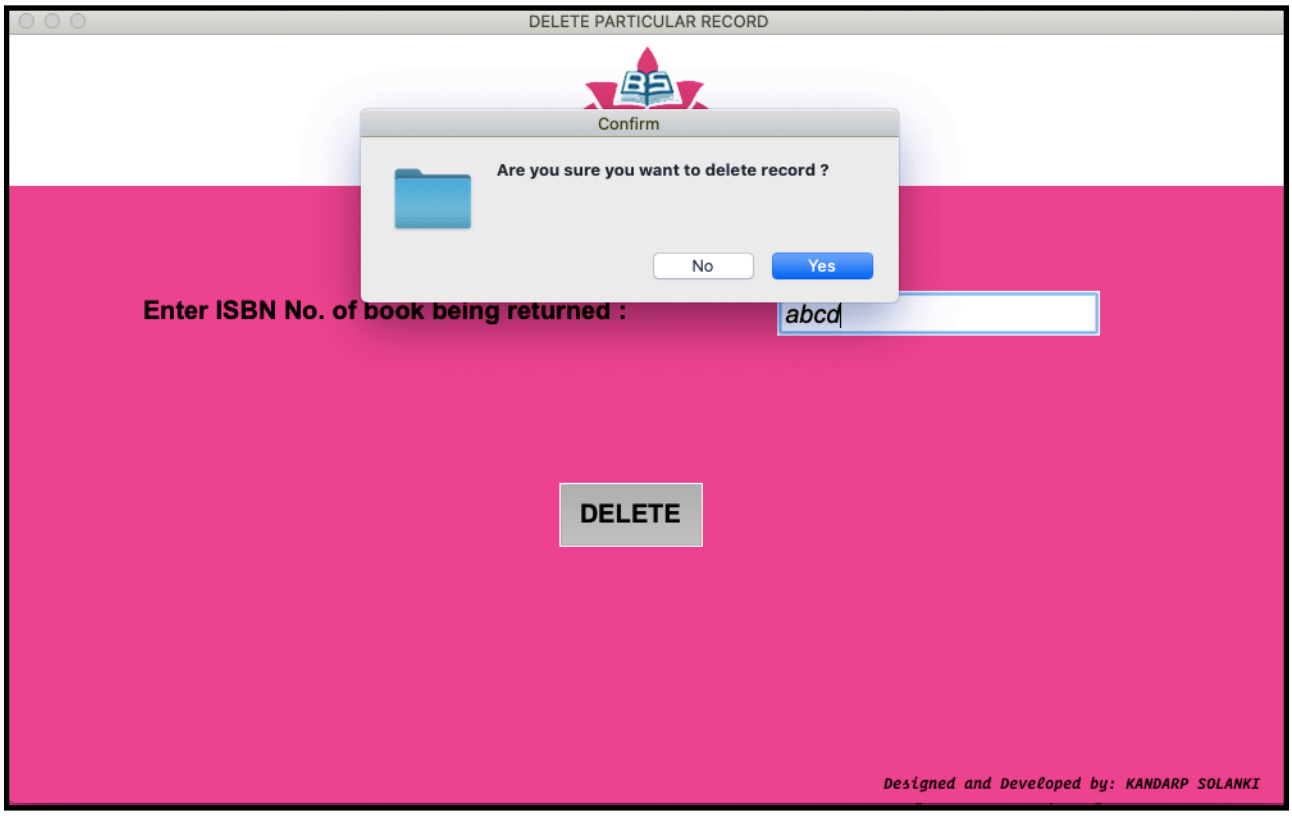

UPDATE RECORD
THE BRIGHT SCHOOL
Exam Seat no. of student (presently in records) :
11111
What to Update ?
Name
Exam Seat No.
ISBN No.
Book Name
Choose
*Press CHOOSE to enter the updated information
Enter updated Exam Seat No. :
Update
Designed and Developed by: KANDARP SOLANKI

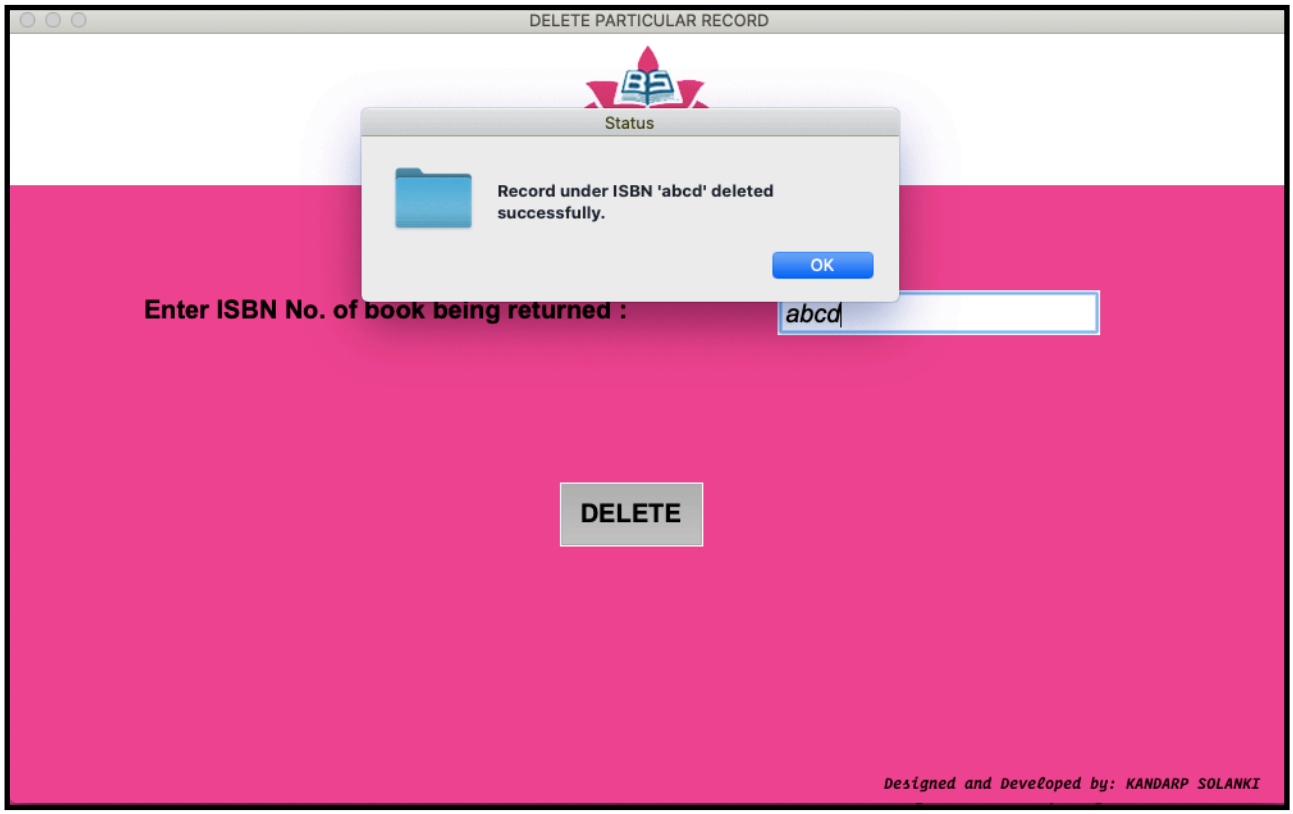

UPDATE RECORD
THE BRIGHT SCHOOL
Exam Seat no. of student (presently in records) :
11111
What to Update ?
Name
Exam Seat No.
ISBN No.
Book Name
Choose
*Press CHOOSE to enter the updated information
Enter updated ISBN No. :
Update
Designed and Developed by: KANDARP SOLANKI


UPDATE RECORD
THE BRIGHT SCHOOL
Exam Seat no. of student (presently in records) :
11111
What to Update ?
Name
Exam Seat No.
ISBN No.
Book Name
Choose
*Press CHOOSE to enter the updated information
Enter updated Book Name :
Update
Designed and Developed by: KANDARP SOLANKI

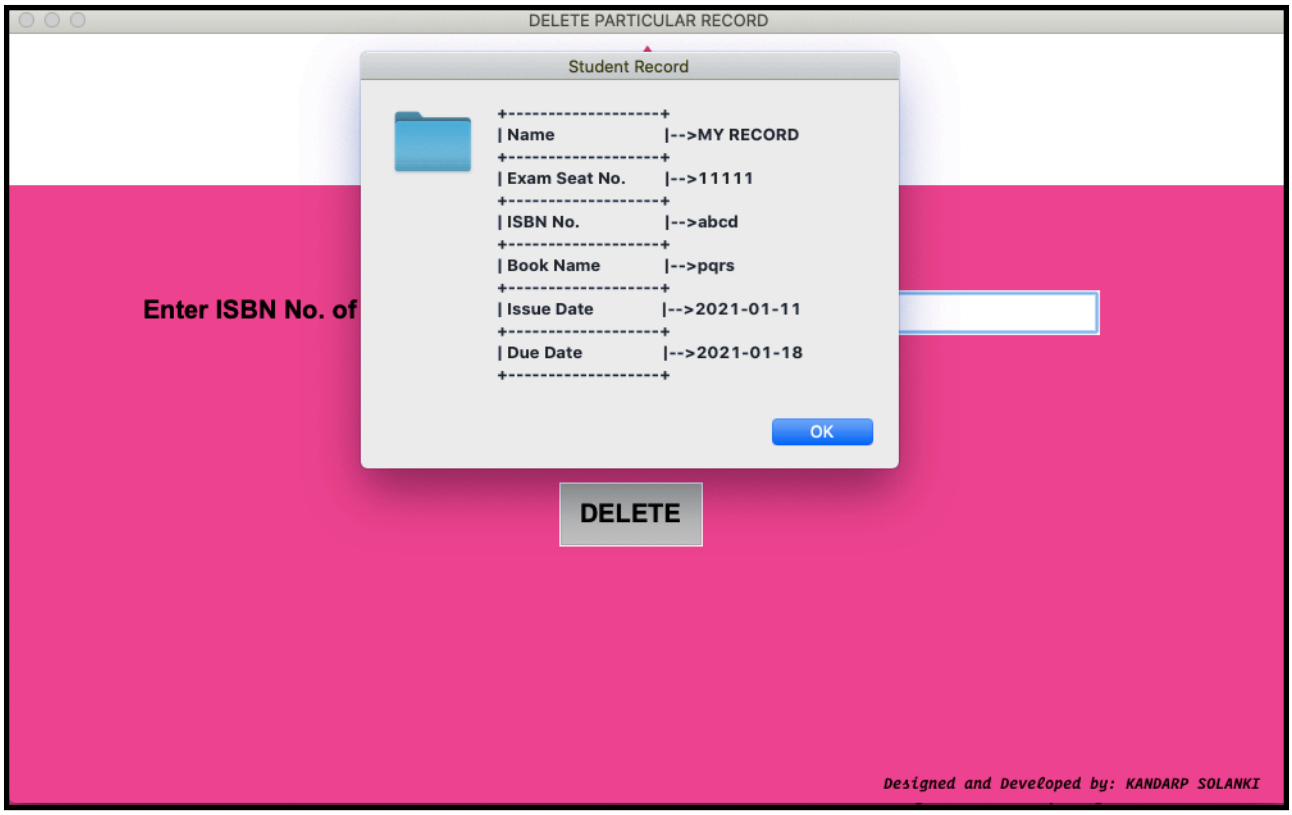
Deleting existing record using 'RETURN A BOOK'

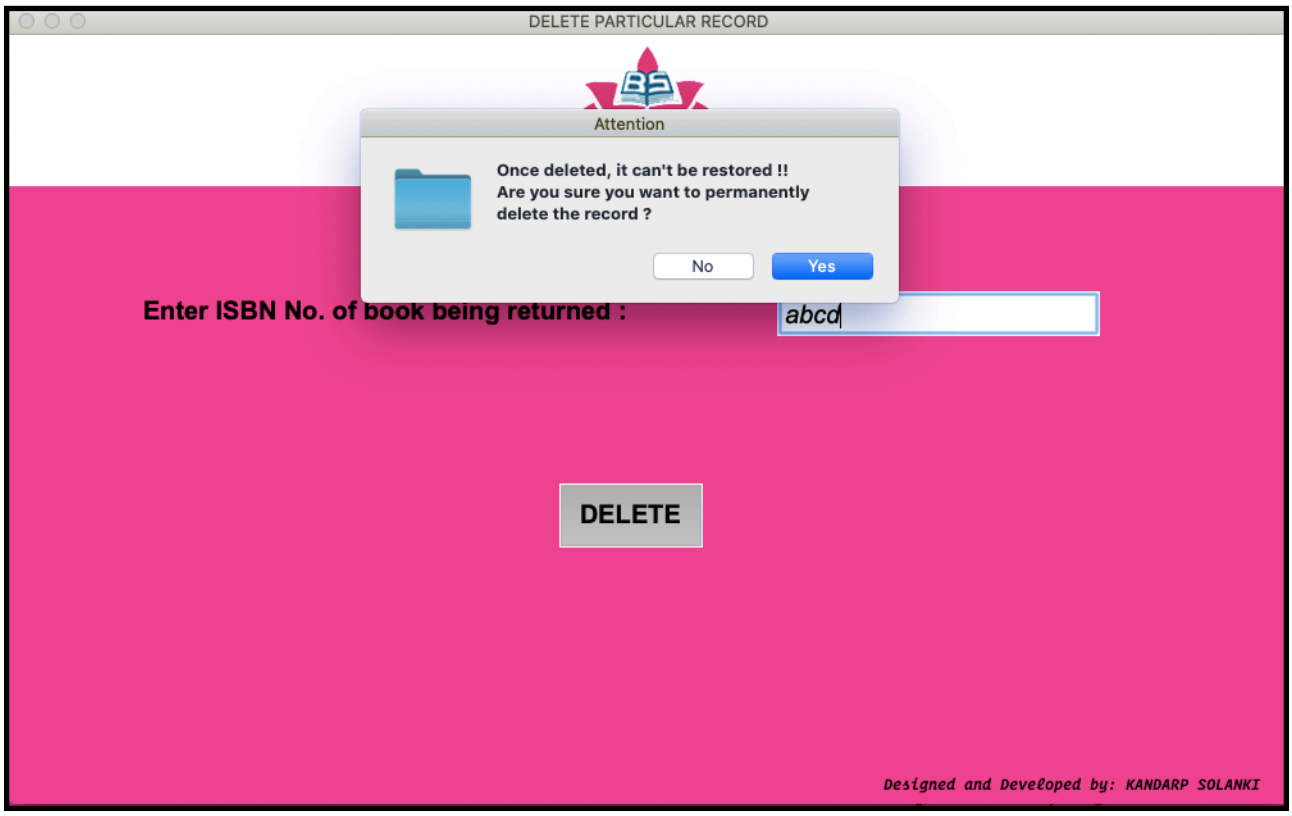




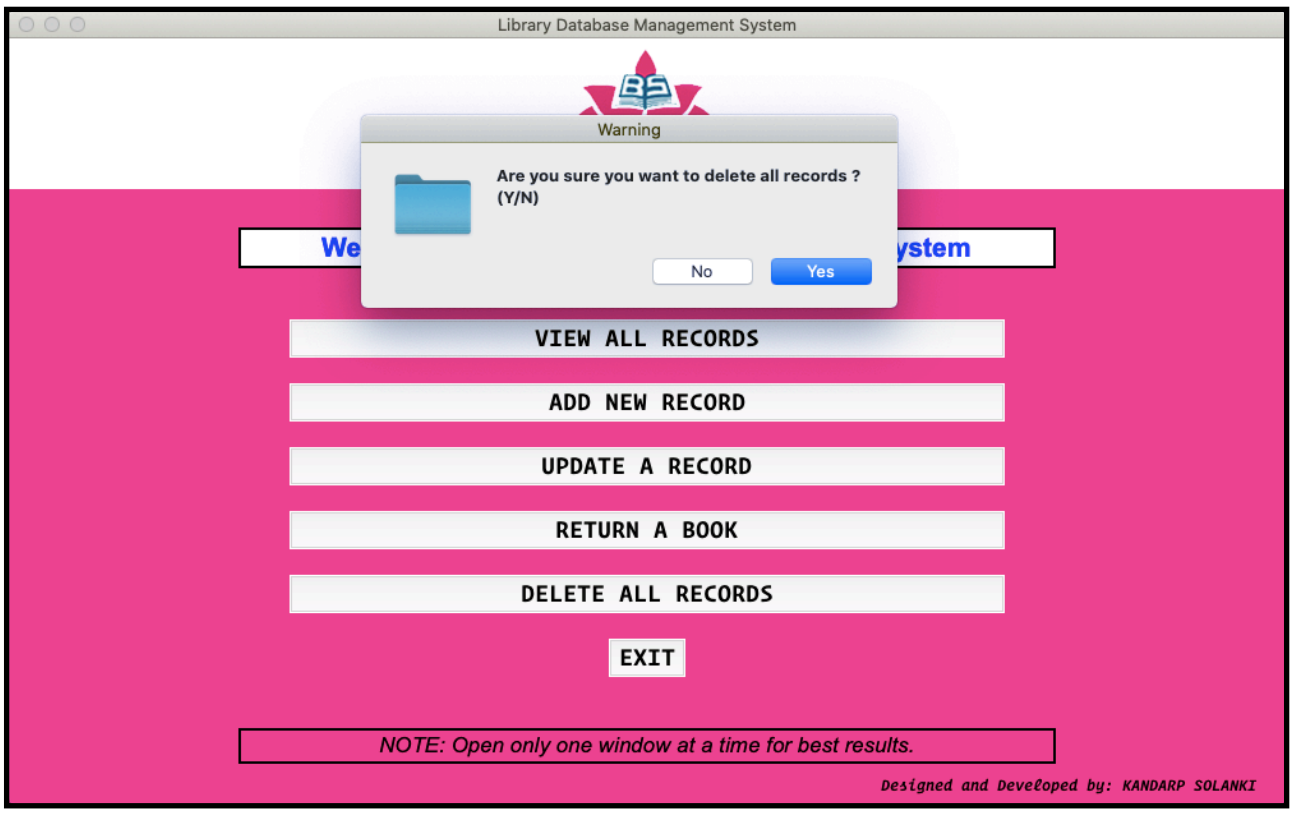


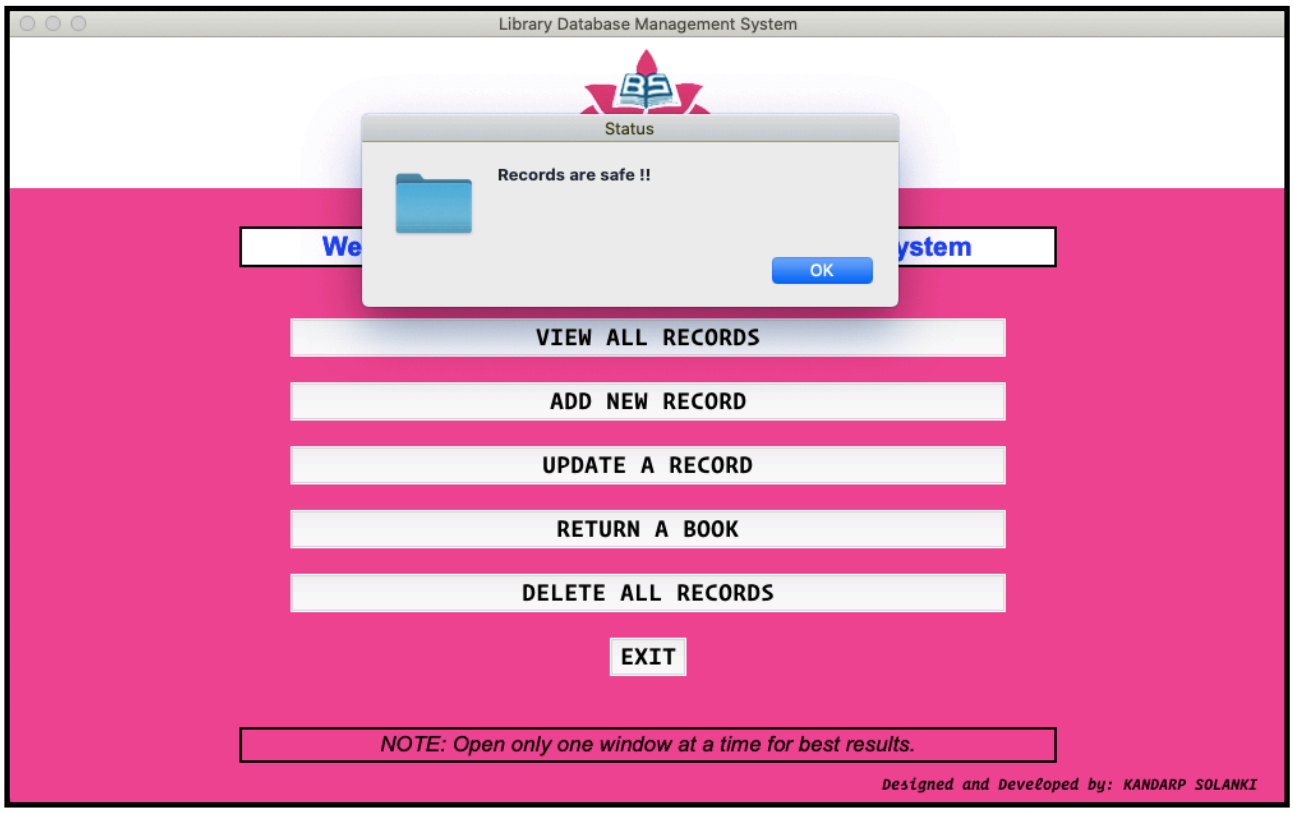
Manav	12036	fdbgfn	gfndghndgh	2020-08-20	2020-08-27
Bhavya	12037	hjdshjhdjfv	djhdsvhjdkf	2020-08-20	2020-08-27
Raman	12038	behedhjehjfd	vhjrevdhrjv	2020-08-20	2020-08-27
Sangita	12039	bnvfskjxsjhdv	svbdskjv	2020-08-20	2020-08-27
Isha	12040	shdvjsdkj	dvkbnd	2020-08-20	2020-08-27
Naman	12041	jsdfbjshdn	kvnjkdfmv	2020-08-20	2020-08-27
Chaitanya	12042	fvdmdfb	djhjvdkfj	2020-08-20	2020-08-27
Parth	12043	gfbfgnfg	mghjdf	2020-08-20	2020-08-27
Yatharth	12044	everger	ergerger	2020-08-20	2020-08-27
Swayam	12045	efbrtgesd	wefercds	2020-08-20	2020-08-27
Aditya	12046	vhdjbdjhfv	dhvbnvnsmdv	2020-08-20	2020-08-27
Deep	12047	dsjvkhsjbdv	vsdvs	2020-08-20	2020-08-27
Gopal	12048	dhjvbnvnkv	dvjhbdfn	2020-08-20	2020-08-27
Siddharth	12049	sjdkhvbnds	hjcsxdsv	2020-08-20	2020-08-27
Mihir	12050	jbncsdnc	sdjvsdsdv	2020-08-20	2020-08-27

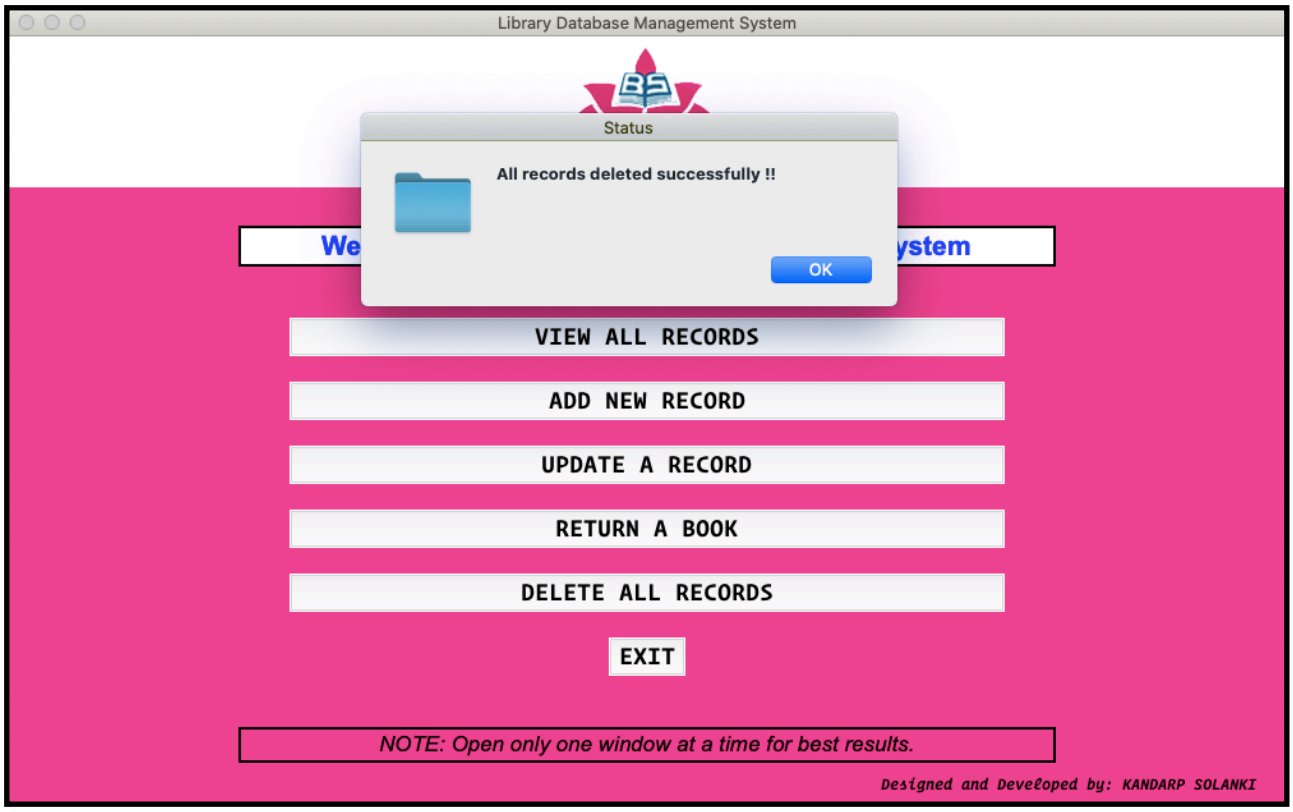

DELETE PARTICULAR RECORD
Student Record
+--------------------+
| Name |-->MY RECORD
+--------------------+
| Exam Seat No. |-->11111
+--------------------+
| ISBN No. |-->abcd
+--------------------+
| Book Name |-->pqrs
+--------------------+
| Issue Date |-->2021-01-11
+--------------------+
| Due Date |-->2021-01-18
+--------------------+
OK
Enter ISBN No. of
DELETE
Designed and Developed by: KANDARP SOLANKI


DELETE PARTICULAR RECORD
Attention
Once deleted, it can't be restored !!
Are you sure you want to permanently
delete the record ?
No
Yes
Enter ISBN No. of book being returned :
abcd
DELETE
Designed and Developed by: KANDARP SOLANKI

Deleting all records using 'DELETE ALL RECORDS'

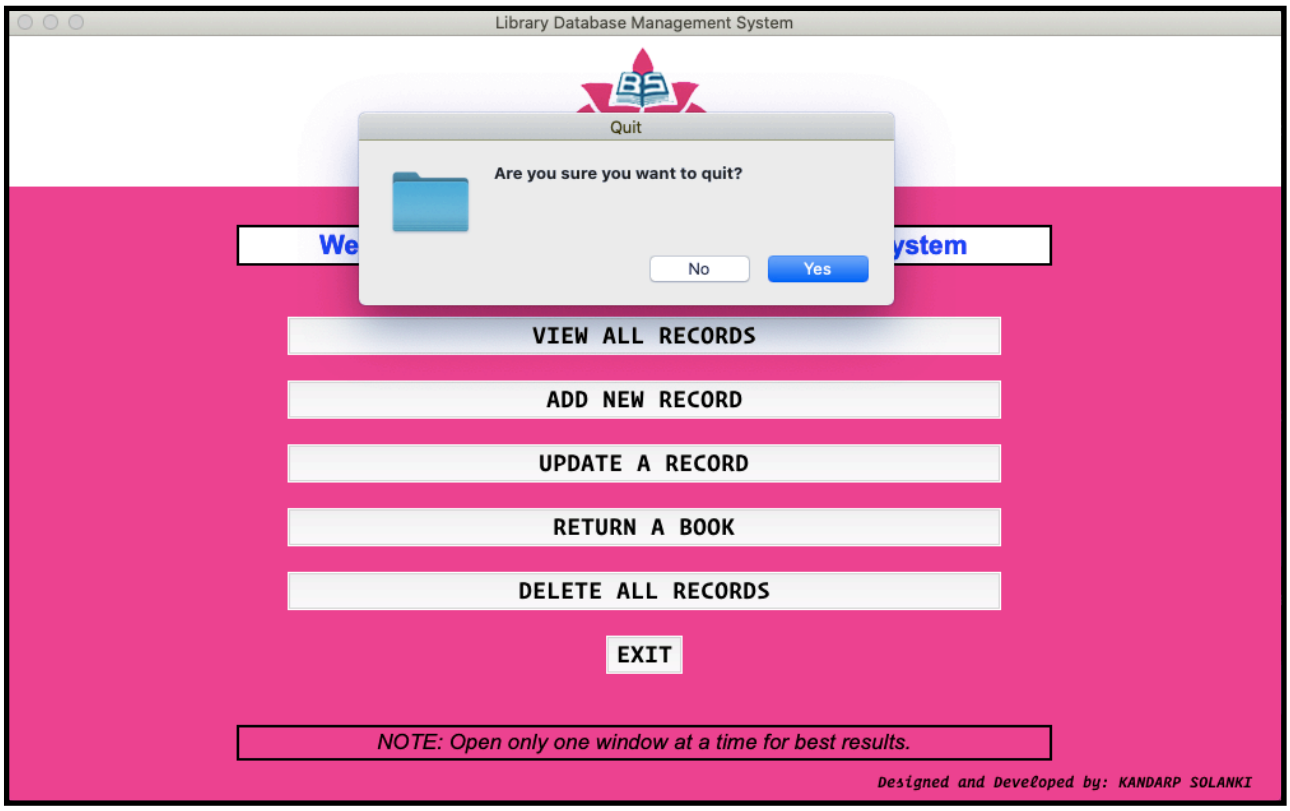


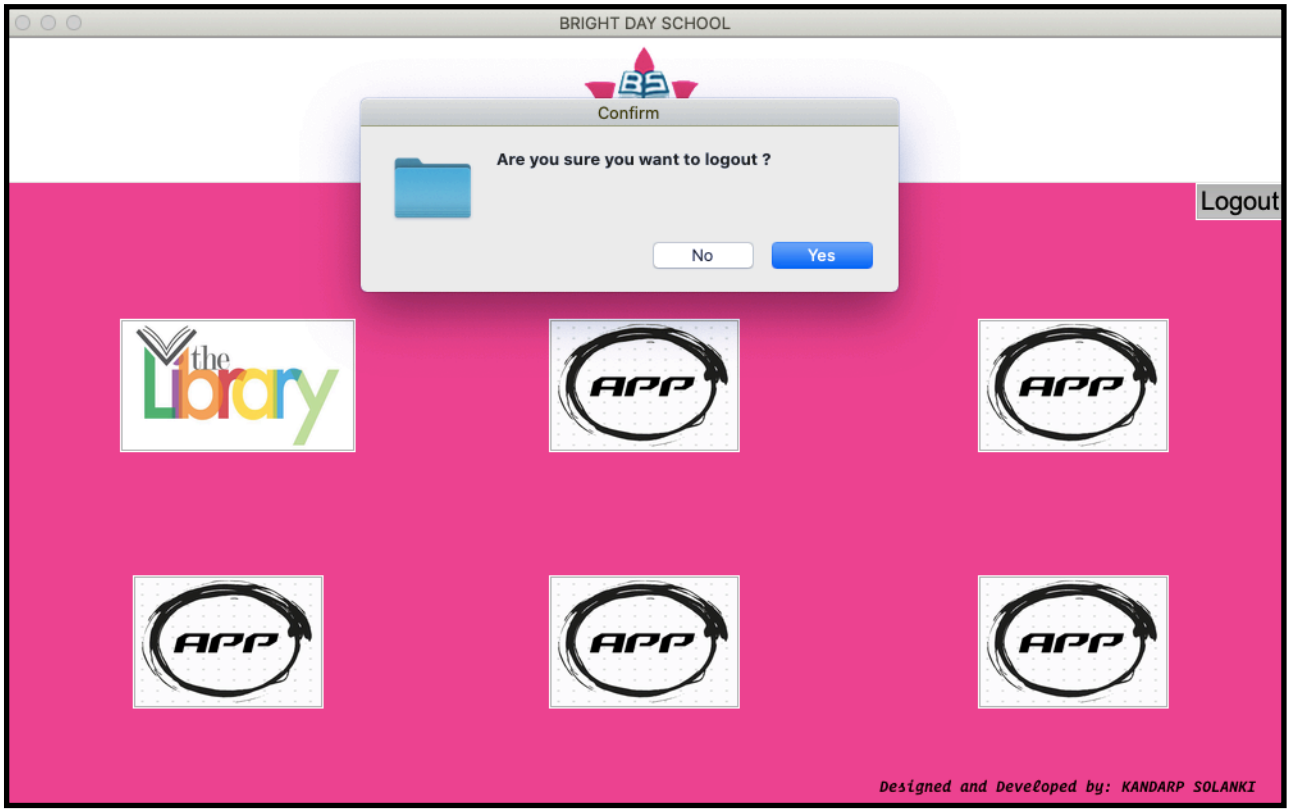


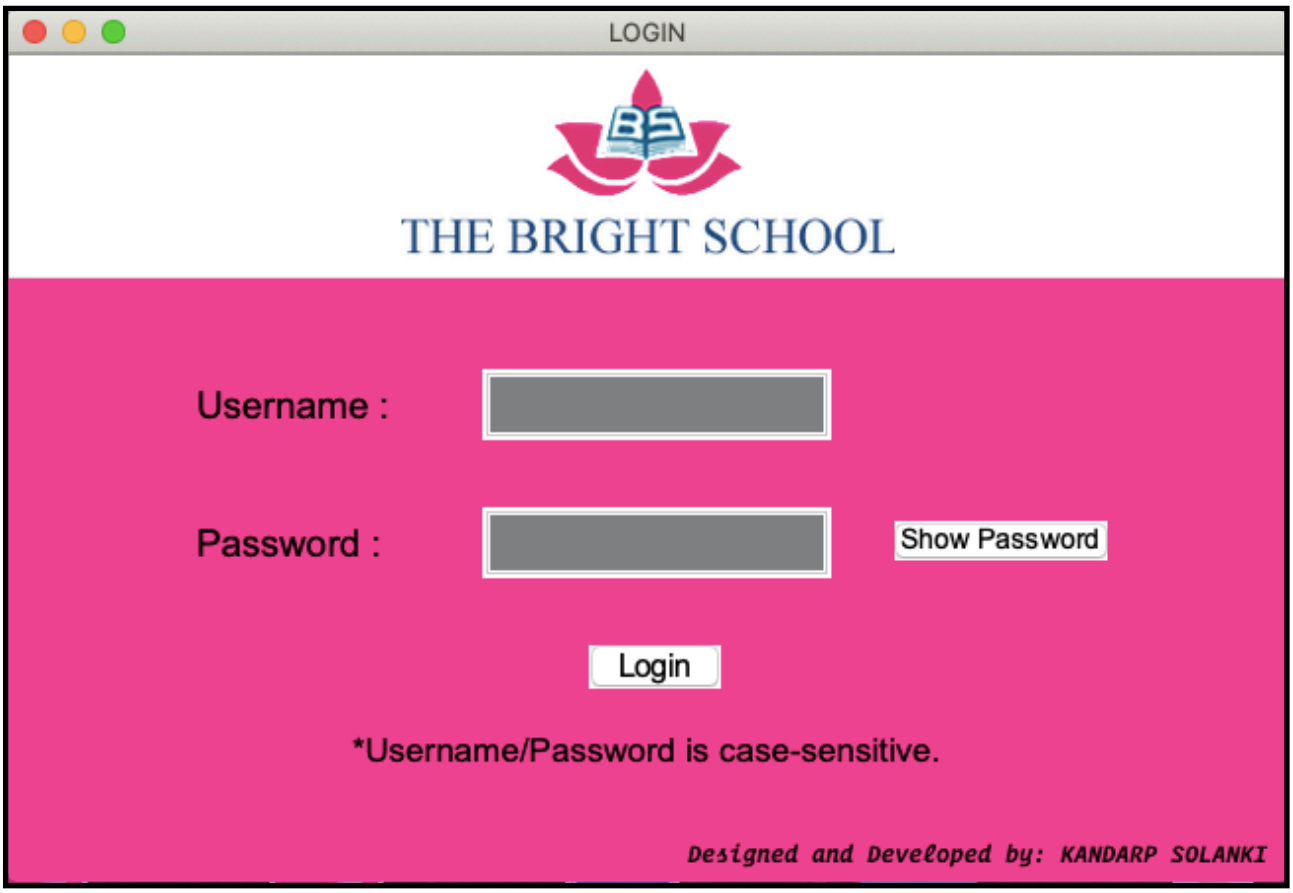


studentdatabase

Name	Examination Seat No.	ISBN No.	Book Name	Issue Date	Due Date


Library Database Management System
Quit
Are you sure you want to quit?
No
Yes
VIEW ALL RECORDS
ADD NEW RECORD
UPDATE A RECORD
RETURN A BOOK
DELETE ALL RECORDS
EXIT
NOTE: Open only one window at a time for best results.
Designed and Developed by: KANDARP SOLANKI


BRIGHT DAY SCHOOL
Confirm
Are you sure you want to logout ?
No
Yes
Logout
Designed and Developed by: KANDARP SOLANKI


LOGIN
THE BRIGHT SCHOOL
Username :
Password :
Show Password
Login
*Username/Password is case-sensitive.
Designed and Developed by: KANDARP SOLANKI

BIBLIOGRAPHY

- Computer Science *with* Python - *Preeti Arora*
- NCERT Computer Science - Python for Class XII
- https://www.geeksforgeeks.org/
- https://stackoverflow.com/
- https://docs.python.org/3/library/tk.html