

Name : Ritesh Pawar
PRN : 2020BTECS00068

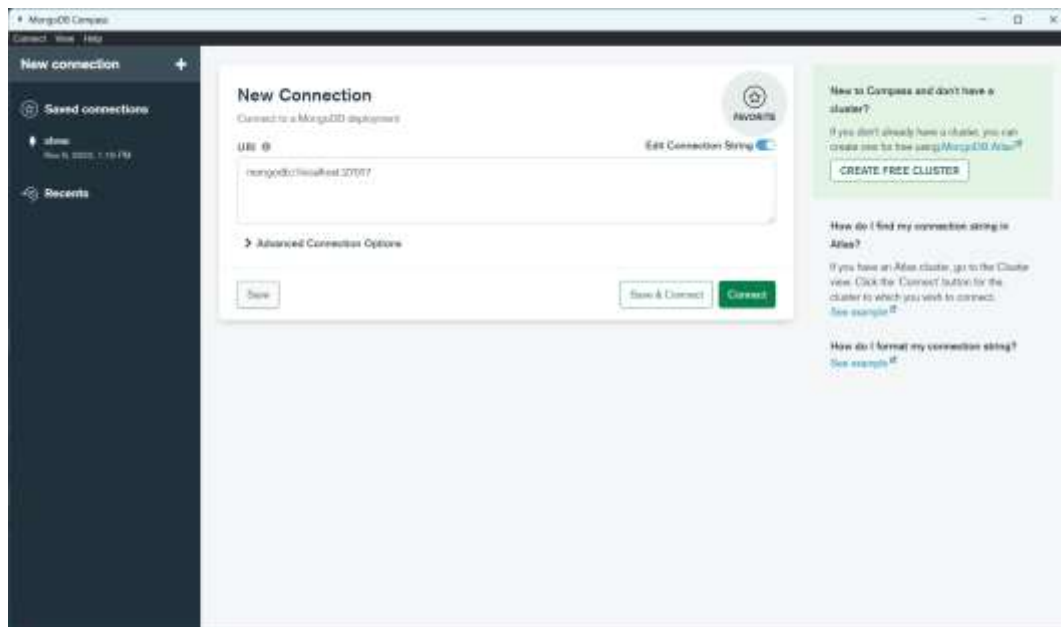
TY B.Tech. (CSE) – II [2022-23]

5CS372 : Advanced Database System Lab.

Assignment No. 9

Install & deploy the following cloud databases on windows platform :

A] MongoDB



B] CassandraDB

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\shree> docker run -p 7090:7090 -p 7001:7001 -p 7199:7199 -p 9042:9042 --name cassandra_c -d cassandra:latest
472d5fcb22b67a3a851d957c2d052a61be94fda61e89450843f22c8ac703c6
PS C:\Users\shree> docker ps
CONTAINER ID   IMAGE                COMMAND                  STATUS                  PORTS
472d5fcb22b   cassandra:latest    "docker-entrypoint.s..." 38 seconds ago         Up 32 seconds          0.0.0.0:7090->7090/tcp, 0.0.0.0:7199->7199/tcp, 0.0.0.0:9042->9042/tcp, 0.0.0.0:9160->9160/tcp
7386118046c7   cassandra           "docker-entrypoint.s..." 11 minutes ago         Up 11 minutes          7090->7090/tcp, 7199->7199/tcp, 9042->9042/tcp, 9160->9160/tcp
PS C:\Users\shree> docker ps
CONTAINER ID   IMAGE                COMMAND                  STATUS                  PORTS
472d5fcb22b   cassandra:latest    "docker-entrypoint.s..." About a minute ago     Up 59 seconds          0.0.0.0:7090->7090/tcp, 0.0.0.0:7199->7199/tcp, 0.0.0.0:9042->9042/tcp, 0.0.0.0:9160->9160/tcp
PS C:\Users\shree> docker exec -it 472d5fcb22b bash
root@472d5fcb22b:/# cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 4.1.0 | Cassandra 4.1.1 | CQL spec 3.0.4 | Native protocol v5]
Use HELP for help.
cqlsh>
```

Name : Ritesh Pawar
PRN : 2020BTECS00068

```
root@472d3fbc22b: / Windows PowerShell
cqlsh> use techframer
... ;
cqlsh:techframer> CREATE TABLE student(student_id int PRIMARY KEY, student_name text, student_c
ity text, student_fees varint, student_phone varint );
cqlsh:techframer> SELECT * from student
... ;

student_id | student_city | student_fees | student_name | student_phone
-----
(0 rows)
cqlsh:techframer> INSERT INTO student (student_id,student_fees,student_name) VALUES(1,5000,'Aje
et');
cqlsh:techframer> SELECT * from student ;

student_id | student_city | student_fees | student_name | student_phone
-----
1 | null | 5000 | Ajeet | null
```

Q. Write Python desktop Application to demonstrate the CRUD operation with above backend cloud databases. Assume any database.

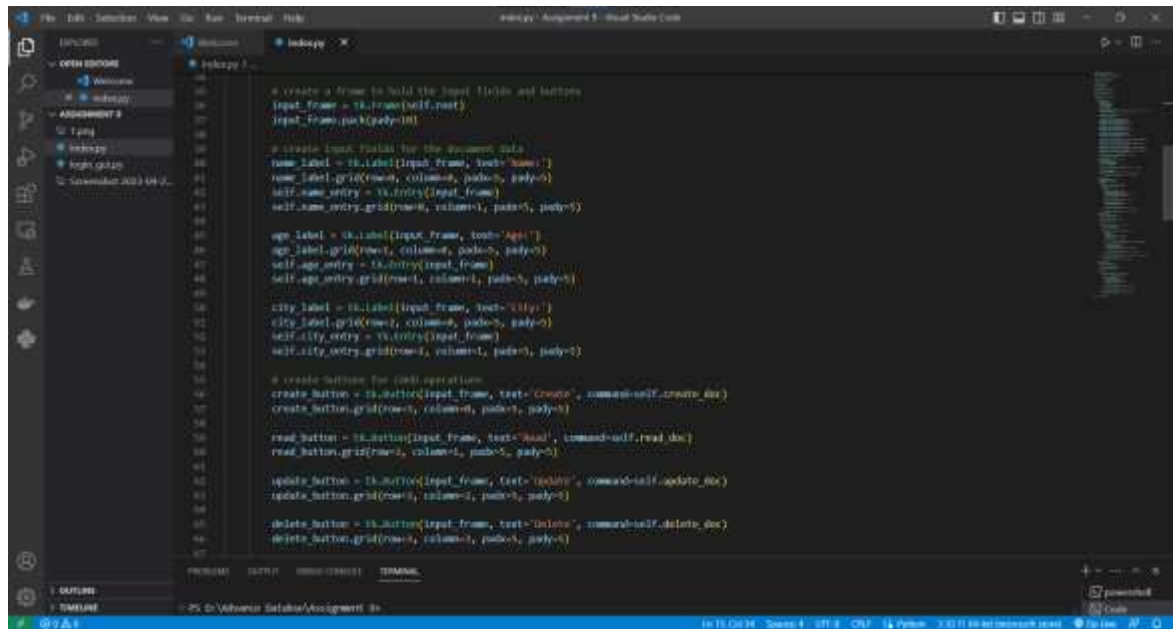
Code:

```
File Edit Settings View Run Run Terminal Help
main.py - Assignment 0 - Visual Studio Code

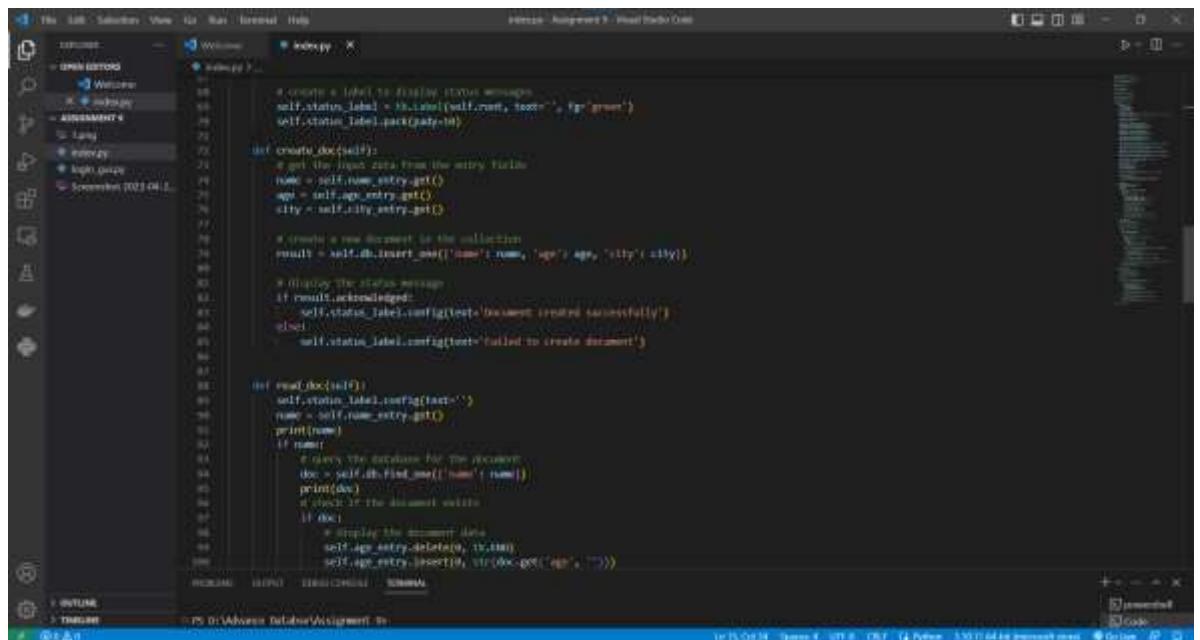
EXPLORER
  OPEN EDITORS
    main.py
  ASSIGNMENT 0
    1.py
    index.py
    high quiz
    Scheduler 2023-04-2...

index.py
1. from pygments import highlight
2. import tkinter as tk
3. db_url = 'mongodb://localhost:27017/ADS_V'
4.
5. # create a connection to the database
6. client = MongoClient(db_url)
7.
8. # specify the database name
9. db_name = 'ADS_V'
10.
11. # access the database
12. db = client[db_name]
13.
14. # specify the collection name
15. collection_name = 'my_collection'
16.
17. # access the collection
18. collection = db[collection_name]
19.
20.
21. import tkinter as tk
22. from pygments import highlight
23.
24. class MyGUI:
25.     def __init__(self, db):
26.         self.db = db
27.         self.root = tk.W()
28.         self.create_widgets()
29.         self.root.mainloop()
30.
31.     def create_widgets(self):
32.         # create a label for the title
33.         title_label = tk.Label(self.root, text='CRUD operations', font=('arial', 16, 'bold'))
34.         title_label.pack(pady=10)
```

Name : Ritesh Pawar
PRN : 2020BTECS00068

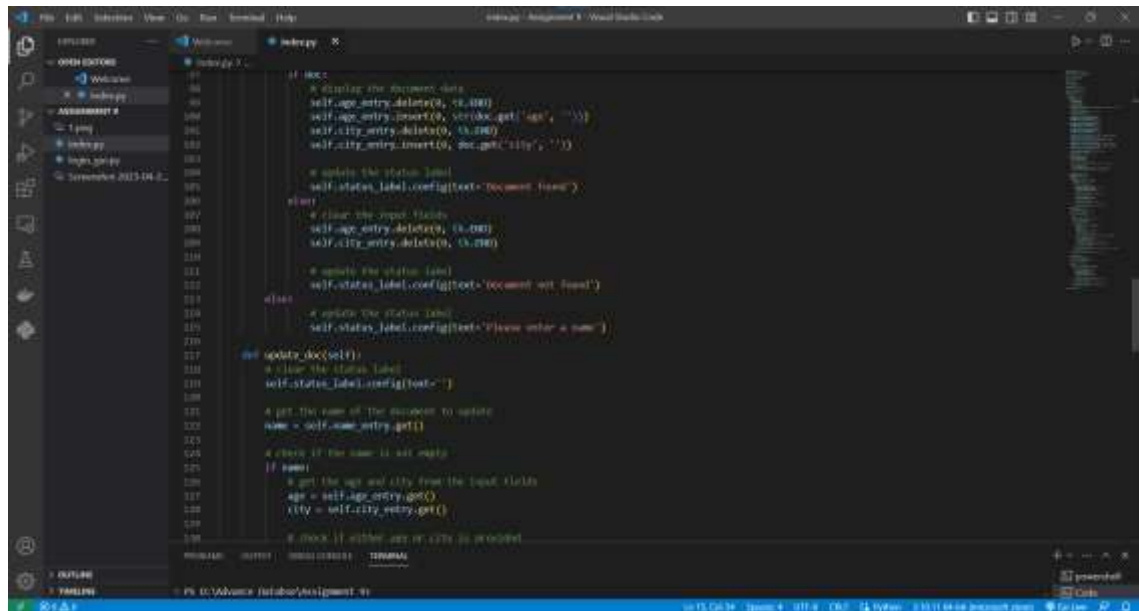


```
1 # create a frame to hold the input fields and buttons
2 input_frame = Tk.Frame(self.root)
3 input_frame.pack(pady=10)
4
5 # create input fields for the document data
6 name_label = Tk.Label(input_frame, text='Name:')
7 name_label.grid(row=0, column=0, padx=5, pady=5)
8 self.name_entry = Tk.Entry(input_frame)
9 self.name_entry.grid(row=0, column=1, padx=5, pady=5)
10
11 age_label = Tk.Label(input_frame, text='Age:')
12 age_label.grid(row=1, column=0, padx=5, pady=5)
13 self.age_entry = Tk.Entry(input_frame)
14 self.age_entry.grid(row=1, column=1, padx=5, pady=5)
15
16 city_label = Tk.Label(input_frame, text='City:')
17 city_label.grid(row=2, column=0, padx=5, pady=5)
18 self.city_entry = Tk.Entry(input_frame)
19 self.city_entry.grid(row=2, column=1, padx=5, pady=5)
20
21 # create buttons for each operation
22 create_button = Tk.Button(input_frame, text='Create', command=self.create_doc)
23 create_button.grid(row=3, column=0, padx=5, pady=5)
24
25 read_button = Tk.Button(input_frame, text='Read', command=self.read_doc)
26 read_button.grid(row=3, column=1, padx=5, pady=5)
27
28 update_button = Tk.Button(input_frame, text='Update', command=self.update_doc)
29 update_button.grid(row=3, column=2, padx=5, pady=5)
30
31 delete_button = Tk.Button(input_frame, text='Delete', command=self.delete_doc)
32 delete_button.grid(row=3, column=3, padx=5, pady=5)
```



```
1 # create a label to display status messages
2 self.status_label = Tk.Label(self.root, text='', fg='green')
3 self.status_label.pack(pady=10)
4
5 def create_doc(self):
6     # get the input data from the entry fields
7     name = self.name_entry.get()
8     age = self.age_entry.get()
9     city = self.city_entry.get()
10
11     # create a new document in the collection
12     result = self.db.insert_one({'name': name, 'age': age, 'city': city})
13
14     # display the status message
15     if result.acknowledged:
16         self.status_label.config(text='Document created successfully!')
17     else:
18         self.status_label.config(text='Failed to create document')
19
20 def read_doc(self):
21     self.status_label.config(text='')
22     name = self.name_entry.get()
23     print(name)
24     if name:
25         # query the database for the document
26         doc = self.db.find_one({'name': name})
27         print(doc)
28         # check if the document exists
29         if doc:
30             # display the document data
31             self.age_entry.delete(0, 'end')
32             self.age_entry.insert(0, str(doc.get('age', '')))
```

Name : Ritesh Pawar
PRN : 2020BTECS00068



```
def display_doc(self):
    # displaying the document data
    self.age_entry.delete(0, tk.END)
    self.age_entry.insert(0, str(doc.get('age')))
    self.city_entry.delete(0, tk.END)
    self.city_entry.insert(0, str(doc.get('city')))

    # update the status label
    self.status_label.config(text='Document found')
    return

# clear the input fields
self.age_entry.delete(0, tk.END)
self.city_entry.delete(0, tk.END)

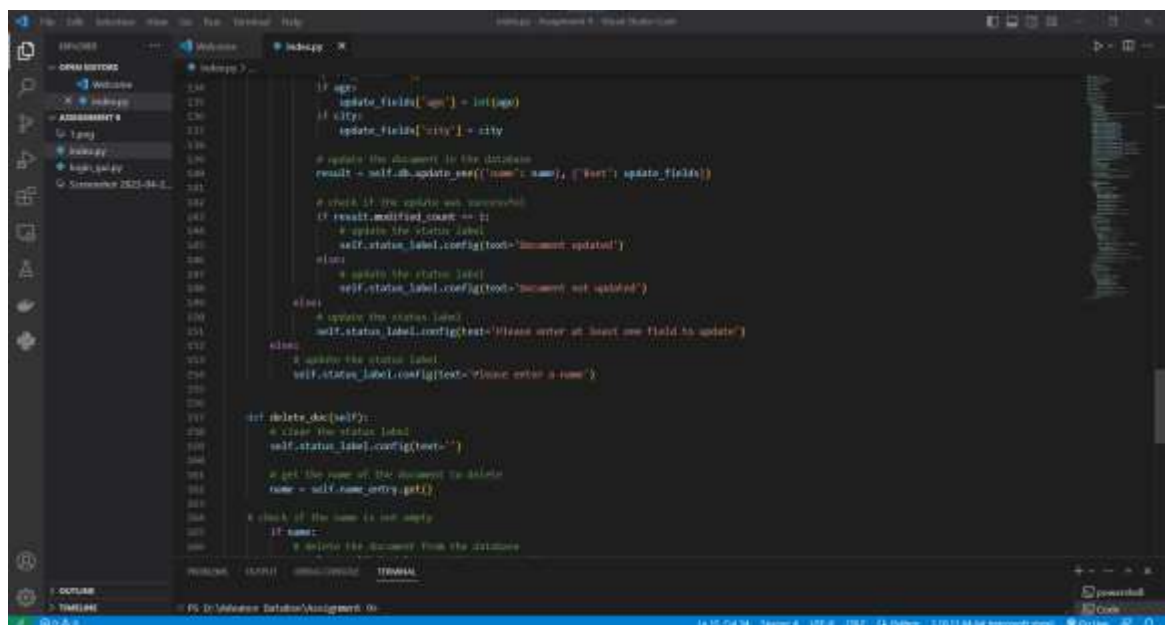
# update the status label
self.status_label.config(text='Document not found')
else:
    # update the status label
    self.status_label.config(text='Please enter a name')

def update_doc(self):
    # clear the status label
    self.status_label.config(text='')

    # get the name of the document to update
    name = self.name_entry.get()

    # check if the name is not empty
    if name:
        # get the age and city from the input fields
        age = self.age_entry.get()
        city = self.city_entry.get()

        # check if either age or city is provided
        if age or city:
```



```
if age:
    update_fields['age'] = int(age)
if city:
    update_fields['city'] = city

# update the document in the database
result = self.db.update_one({'name': name}, {'$set': update_fields})

# check if the update was successful
if result.modified_count > 0:
    # update the status label
    self.status_label.config(text='Document updated')
else:
    # update the status label
    self.status_label.config(text='Document not updated')

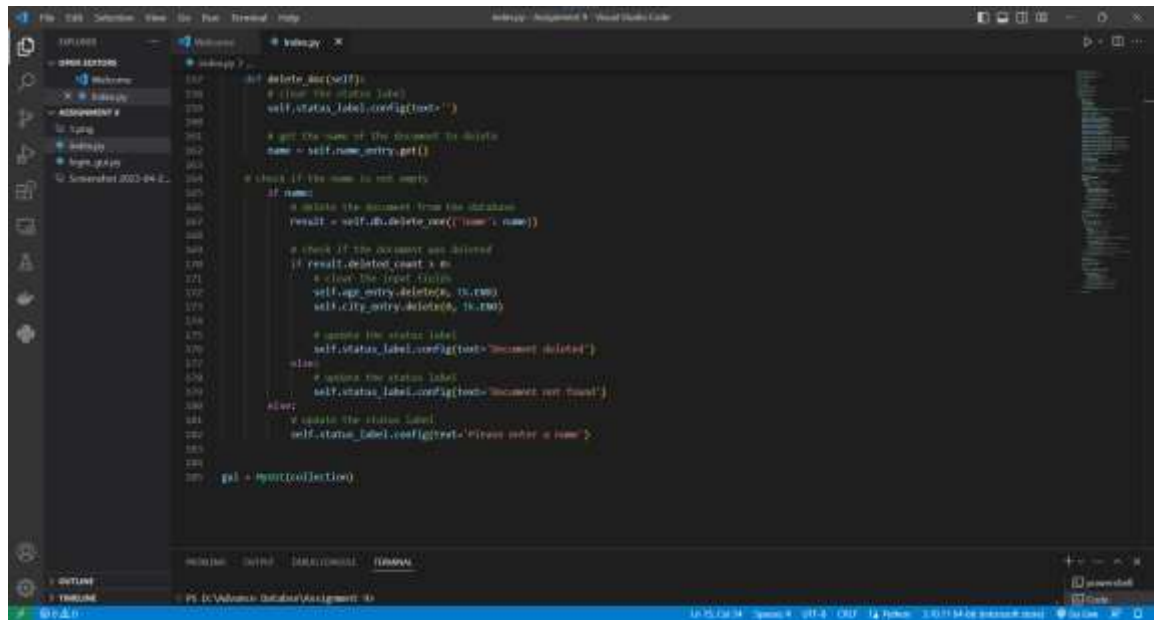
# update the status label
self.status_label.config(text='Please enter at least one field to update')
else:
    # update the status label
    self.status_label.config(text='Please enter a name')

def delete_doc(self):
    # clear the status label
    self.status_label.config(text='')

    # get the name of the document to delete
    name = self.name_entry.get()

    # check if the name is not empty
    if name:
        # delete the document from the database
```

Name : Ritesh Pawar
PRN : 2020BTECS00068



```
def delete_document(self):  
    # clear the status label  
    self.status_label.config(text='')  
  
    # get the name of the document to delete  
    name = self.name_entry.get()  
  
    # check if the name is not empty  
    if name:  
        # delete the document from the database  
        result = self.db.delete_one({'name': name})  
  
        # check if the document was deleted  
        if result.deleted_count > 0:  
            # clear the input field  
            self.name_entry.delete(0, 'end')  
            self.city_entry.delete(0, 'end')  
  
            # update the status label  
            self.status_label.config(text='Document deleted')  
        else:  
            # update the status label  
            self.status_label.config(text='Document not found')  
    else:  
        # update the status label  
        self.status_label.config(text='Please enter a name')  
  
    self.run_collection()  
  
if __name__ == '__main__':  
    app = Tk()  
    app.mainloop()
```

Result:

1.CREATE



CRUD Operations

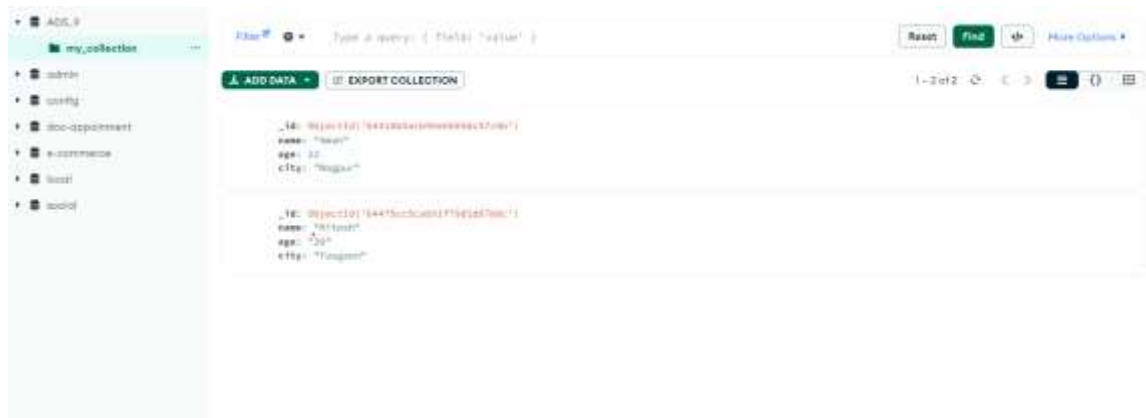
Name:

Age:

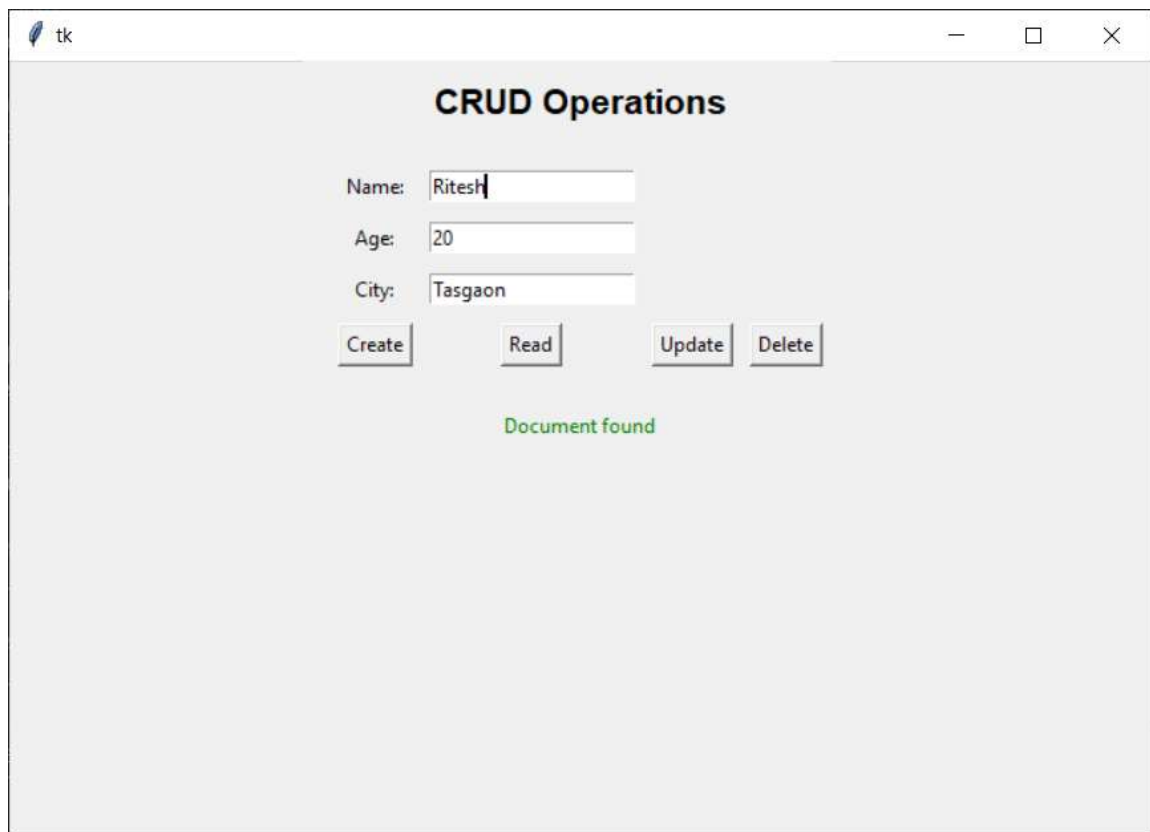
City:

Document created successfully

Name : Ritesh Pawar
PRN : 2020BTECS00068



2.Read:



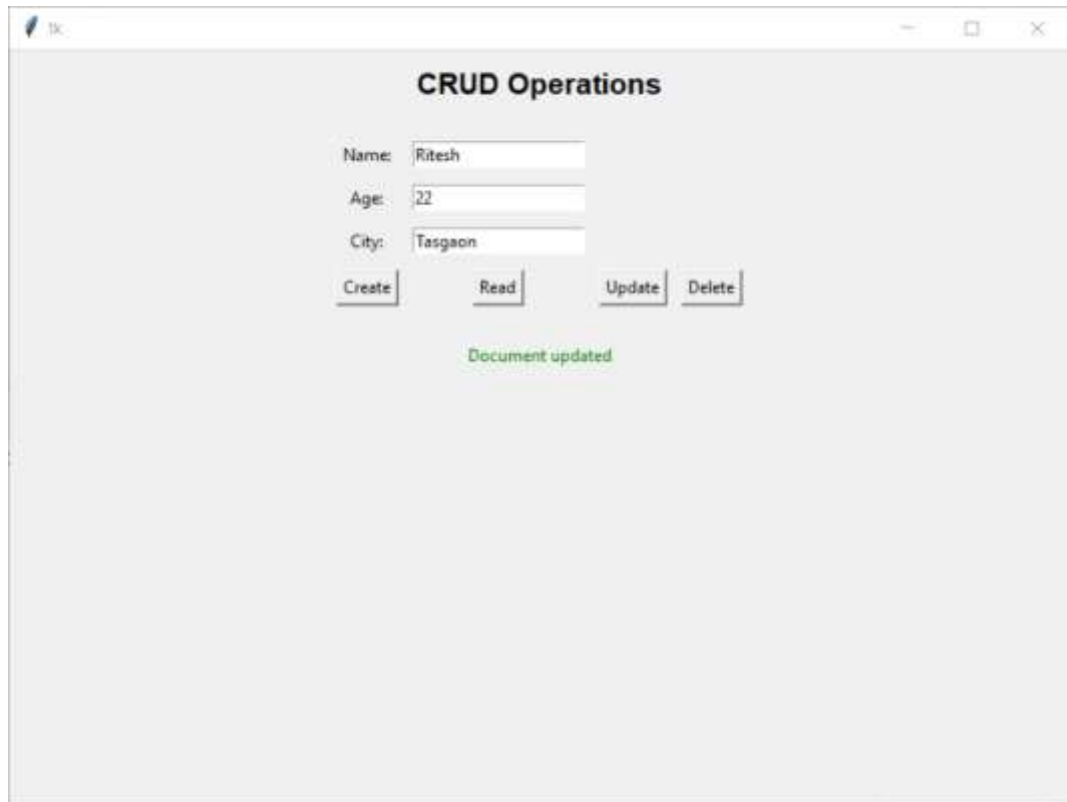
3.Update:

Before Update:

```
_id: ObjectId('64475cc5ca891f79d1d87b0c')
name: "Ritesh"
age: 20
city: "Tasgaon"
```

Name : Ritesh Pawar
PRN : 2020BTECS00068

After Update:



The screenshot shows a web application window titled "tk:". Inside, the heading "CRUD Operations" is centered. Below it, there are three input fields: "Name:" with the value "Ritesh", "Age:" with the value "22", and "City:" with the value "Tasgaon". Below these fields are four buttons: "Create", "Read", "Update", and "Delete". The "Update" button is highlighted. Below the buttons, a green message "Document updated" is displayed.

```
_id: ObjectId('64475cc5ca891f79d1d87b0c')  
name: "Ritesh"  
age: 22  
city: "Tasgaon"
```

4.Delete:

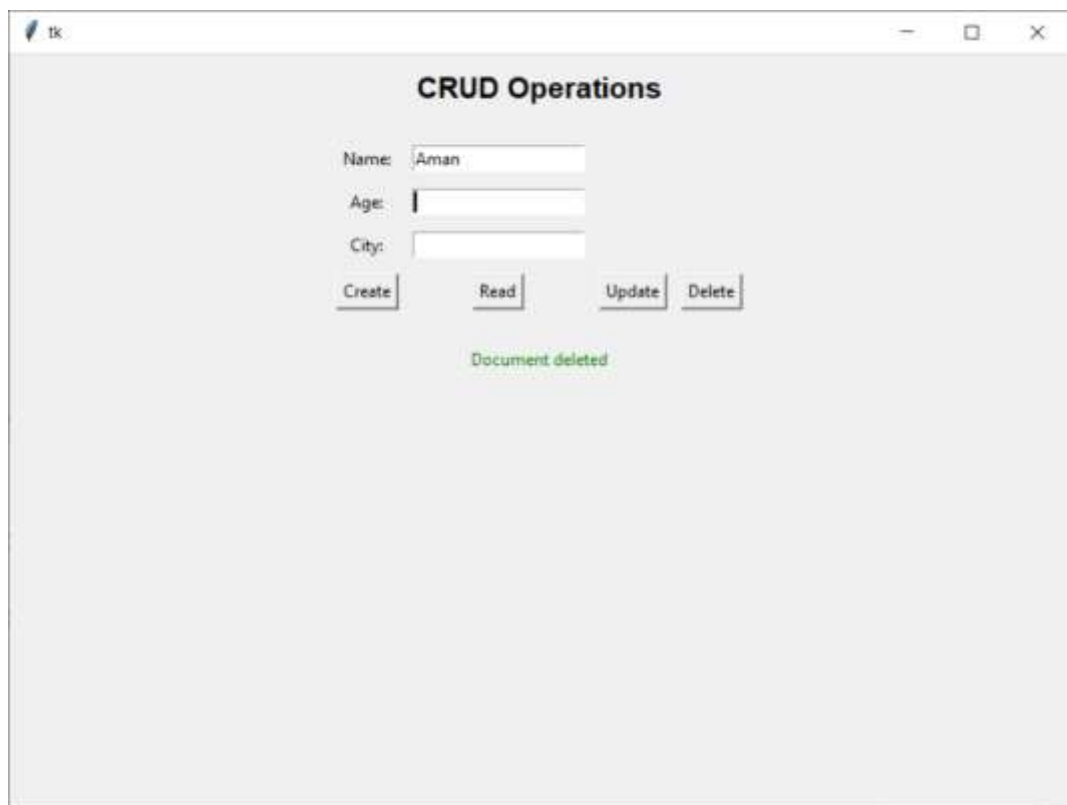
Before Delete:

Name : Ritesh Pawar
PRN : 2020BTECS00068

```
_id: ObjectId('64410b5acb96e6609dc57c0b')  
name: "Aman"  
age: 22  
city: "Nagpur"
```

```
_id: ObjectId('64475cc5ca891f79d1d87b8c')  
name: "Ritesh"  
age: 20  
city: "Tasgaon"
```

After Delete:



tk

CRUD Operations

Name:

Age:

City:

Document deleted

```
_id: ObjectId('64475cc5ca891f79d1d87b8c')  
name: "Ritesh"  
age: 20  
city: "Tasgaon"
```

Dr. B. F. Momin
Course Coordinator