Title: Install & deploy the following cloud databases on windows platform

Procedure:

- Step 1: Download MongoDB Installer
    - Go to the MongoDB website: https://www.mongodb.com/download-center/community
    - Select the appropriate version of MongoDB for your operating system (e.g., Windows) and download the installer package.

- Step 2: Run MongoDB Installer
    - Locate the downloaded installer package and double-click to run it.
    - Follow the installation wizard prompts and accept the default settings unless you have specific requirements.
    - Choose "Complete" setup type to install MongoDB with all the components.

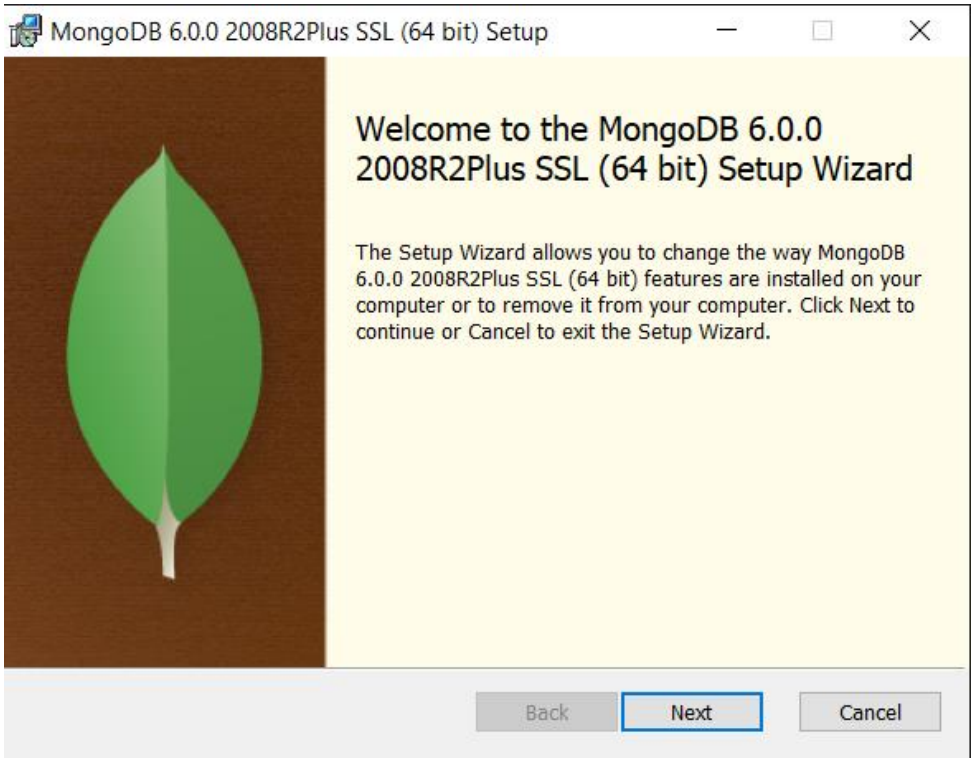
- Step 3: Configure MongoDB

    - During the installation, MongoDB will be installed as a service that runs automatically in the background.
    - By default, MongoDB is installed at C:\Program Files\MongoDB\Server\<version> folder, where <version> corresponds to the version number of MongoDB you have installed.
    - MongoDB requires a data directory to store its data. By default, the data directory is created at C:\data\db. You can specify a different data directory if needed.

    - MongoDB also requires a configuration file. By default, the configuration file is located at C:\Program Files\MongoDB\Server\<version>\bin\mongod.cfg. You can edit this file to configure various settings for MongoDB, such as the port number, log file location, etc.
- Step 4: Start MongoDB Service

    - Once the installation is complete, MongoDB service should be running automatically.
    - You can start/stop/restart MongoDB service using the Windows Services Manager or by running the following commands in the Command Prompt or PowerShell:
    - To start MongoDB service: net start MongoDB
    - To stop MongoDB service: net stop MongoDB
- Step 5: Verify MongoDB Installation

    - Open a new Command Prompt or PowerShell window.
    - Type mongo and press Enter. This should open the MongoDB shell, which is the command-line interface for interacting with MongoDB.
    - You can also run the mongo command with additional options to specify the MongoDB connection settings, such as the hostname, port number, and authentication credentials.



Q2. Write Python desktop Application to demonstrate the CRUD operation with above backend cloud databases.

Program:

```
import tkinter as tk
from pymongo import MongoClient
from tkinter import messagebox

client = MongoClient()
db = client['ADS']
collection = db['StudentData']

# Define Student class
class Student:
    def __init__(self, student_id, first_name, last_name, age, course):
        self.student_id = student_id
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
        self.course = course

    def save(self):
        student_data = {
            '_id': self.student_id,
            'first_name': self.first_name,
            'last_name': self.last_name,
```

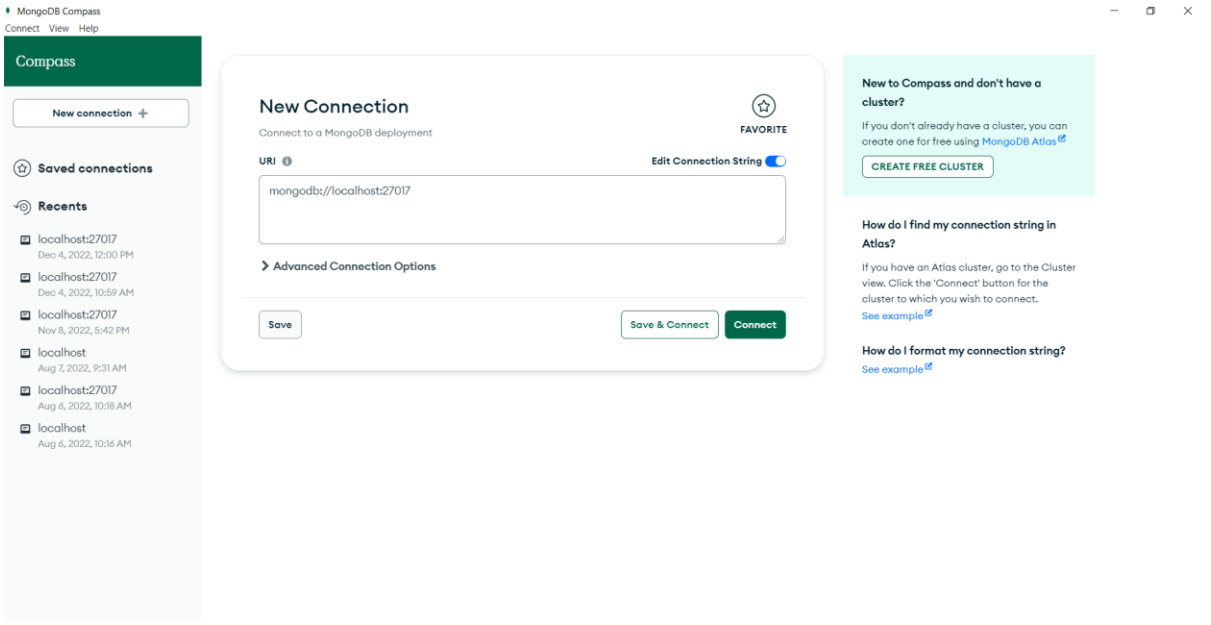
```python
            'age': self.age,
            'course': self.course
        }
        collection.insert_one(student_data)

    def update(self):
        student_data = {
            'first_name': self.first_name,
            'last_name': self.last_name,
            'age': self.age,
            'course': self.course
        }
        collection.update_one({'_id': self.student_id}, {'$set': student_data})

    def delete(self):
        collection.delete_one({'_id': self.student_id})


# GUI application using Tkinter
class StudentApp:
    def __init__(self, root=None):
        if root is None:
            root = tk.Tk()
        self.root = root
        self.root.title('Student Data CRUD Application')

        # Labels
        label_student_id = tk.Label(self.root, text='PRN:')
        label_student_id.grid(row=0, column=0, padx=5, pady=5)
        label_first_name = tk.Label(self.root, text='First Name:')
        label_first_name.grid(row=1, column=0, padx=5, pady=5)
        label_last_name = tk.Label(self.root, text='Last Name:')
        label_last_name.grid(row=2, column=0, padx=5, pady=5)
        label_age = tk.Label(self.root, text='Age:')
        label_age.grid(row=3, column=0, padx=5, pady=5)
        label_course = tk.Label(self.root, text='Course:')
        label_course.grid(row=4, column=0, padx=5, pady=5)

        # Entry fields
        self.entry_student_id = tk.Entry(self.root)
        self.entry_student_id.grid(row=0, column=1, padx=5, pady=5)
        self.entry_first_name = tk.Entry(self.root)
        self.entry_first_name.grid(row=1, column=1, padx=5, pady=5)
        self.entry_last_name = tk.Entry(self.root)
        self.entry_last_name.grid(row=2, column=1, padx=5, pady=5)
        self.entry_age = tk.Entry(self.root)
        self.entry_age.grid(row=3, column=1, padx=5, pady=5)
        self.entry_course = tk.Entry(self.root)
        self.entry_course.grid(row=4, column=1, padx=5, pady=5)

        # Buttons
        button_save = tk.Button(self.root, text='Save', command=self.save_student)
        button_save.grid(row=5, column=0, padx=5, pady=5)
```

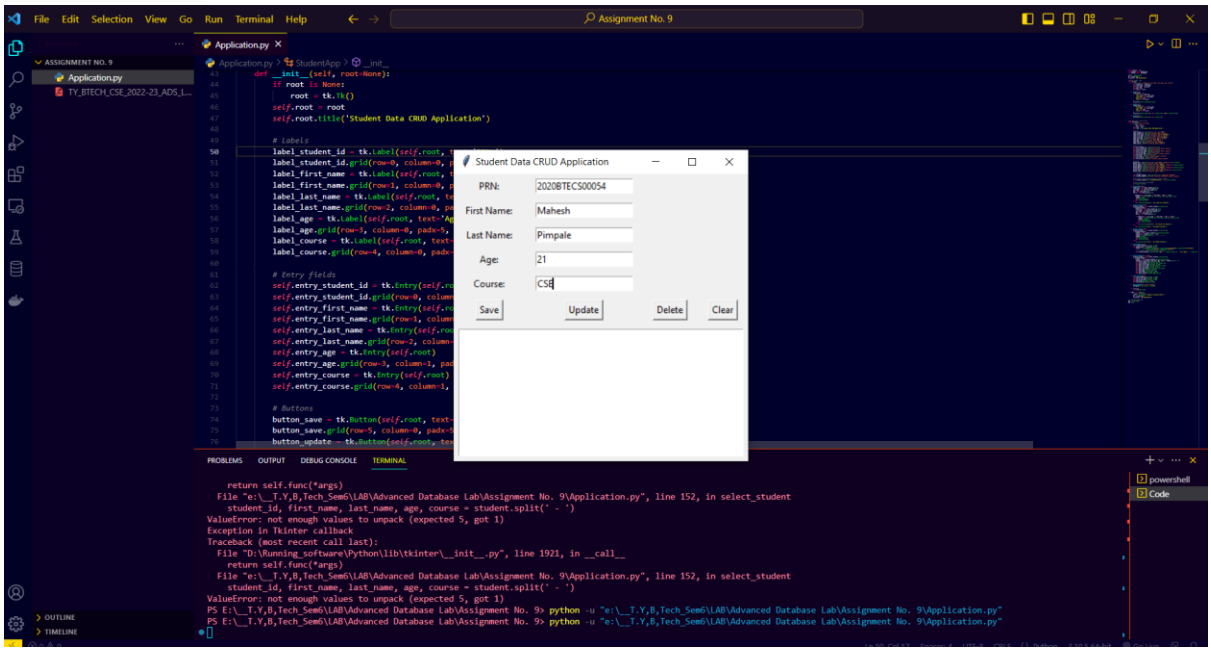
```
        button_update = tk.Button(self.root, text='Update',
command=self.update_student)
        button_update.grid(row=5, column=1, padx=5, pady=5)
        button_delete = tk.Button(self.root, text='Delete',
command=self.delete_student)
        button_delete.grid(row=5, column=2, padx=5, pady=5)
        button_clear = tk.Button(self.root, text='Clear', command=self.clear_entries)
        button_clear.grid(row=5, column=3, padx=5, pady=5)

            # Listbox
        self.listbox_students = tk.Listbox(self.root, height=10, width=60)
        self.listbox_students.grid(row=6, column=0, columnspan=4, padx=5, pady=5)

        # Bind double click event on listbox to select student
        self.listbox_students.bind('<Double-Button-1>', self.select_student)

        # Load initial data from MongoDB
        self.load_students()

    def save_student(self):
        student_id = self.entry_student_id.get()
        first_name = self.entry_first_name.get()
        last_name = self.entry_last_name.get()
        age = self.entry_age.get()
        course = self.entry_course.get()

        if student_id and first_name and last_name and age and course:
            student = Student(student_id, first_name, last_name, age, course)
            student.save()
            self.load_students()
            self.clear_entries()
        else:
            self.show_message('Error', 'All fields are required.')

    def update_student(self):
        selected_student = self.listbox_students.curselection()
        if selected_student:
            student_id = self.entry_student_id.get()
            first_name = self.entry_first_name.get()
            last_name = self.entry_last_name.get()
            age = self.entry_age.get()
            course = self.entry_course.get()

            if student_id and first_name and last_name and age and course:
                student = Student(student_id, first_name, last_name, age, course)
                student.update()
                self.load_students()
                self.clear_entries()
            else:
                self.show_message('Error', 'All fields are required.')
        else:
            self.show_message('Error', 'No student selected.')

    def delete_student(self):
```

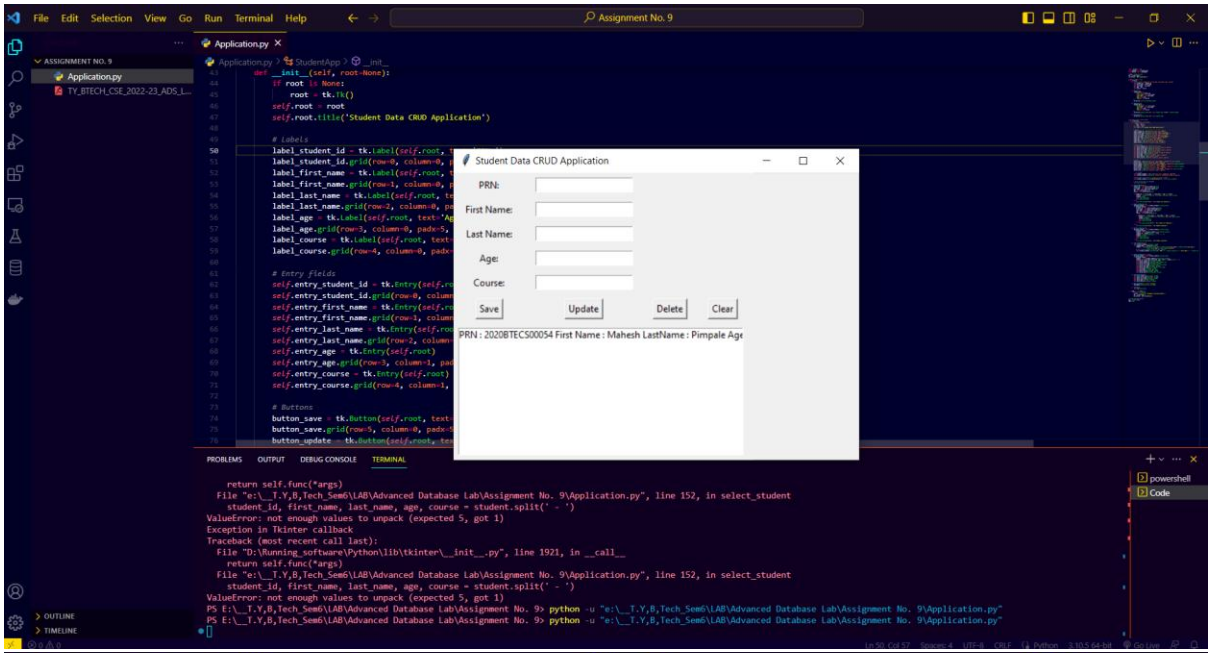
```
        selected_student = self.listbox_students.curselection()
        if selected_student:
            student = self.listbox_students.get(selected_student)
            student_id = student.split(' - ')[0]
            student = Student(student_id, '', '', '', '')
            student.delete()
            self.load_students()
            self.clear_entries()
        else:
            self.show_message('Error', 'No student selected.')

    def load_students(self):
        self.listbox_students.delete(0, tk.END)
        for student_data in collection.find():
            student = Student(student_data['_id'], student_data['first_name'],
student_data['last_name'],
                              student_data['age'], student_data['course'])
            # self.listbox_students.insert(tk.END, student)
            self.listbox_students.insert(tk.END, "PRN : " + student.student_id + "
First Name : " + student.first_name + " LastName : " + student.last_name + " Age : " +
student.age + " Course : " + student.course)


    def select_student(self, event):
        selected_student = self.listbox_students.curselection()
        if selected_student:
            student = self.listbox_students.get(selected_student)
            student_id, first_name, last_name, age, course = student.split(' - ')
            self.entry_student_id.delete(0, tk.END)
            self.entry_student_id.insert(tk.END, student_id)
            self.entry_first_name.delete(0, tk.END)
            self.entry_first_name.insert(tk.END, first_name)
            self.entry_last_name.delete(0, tk.END)
            self.entry_last_name.insert(tk.END, last_name)
            self.entry_age.delete(0, tk.END)
            self.entry_age.insert(tk.END, age)
            self.entry_course.delete(0, tk.END)
            self.entry_course.insert(tk.END, course)

    def clear_entries(self):
        self.entry_student_id.delete(0, tk.END)
        self.entry_first_name.delete(0, tk.END)
        self.entry_last_name.delete(0, tk.END)
        self.entry_age.delete(0, tk.END)
        self.entry_course.delete(0, tk.END)

    def show_message(self, title, message):
        messagebox.showinfo(title, message)

    def run(self):
        self.root.mainloop()

if __name__ == "__main__":
       # Connect to MongoDB
```

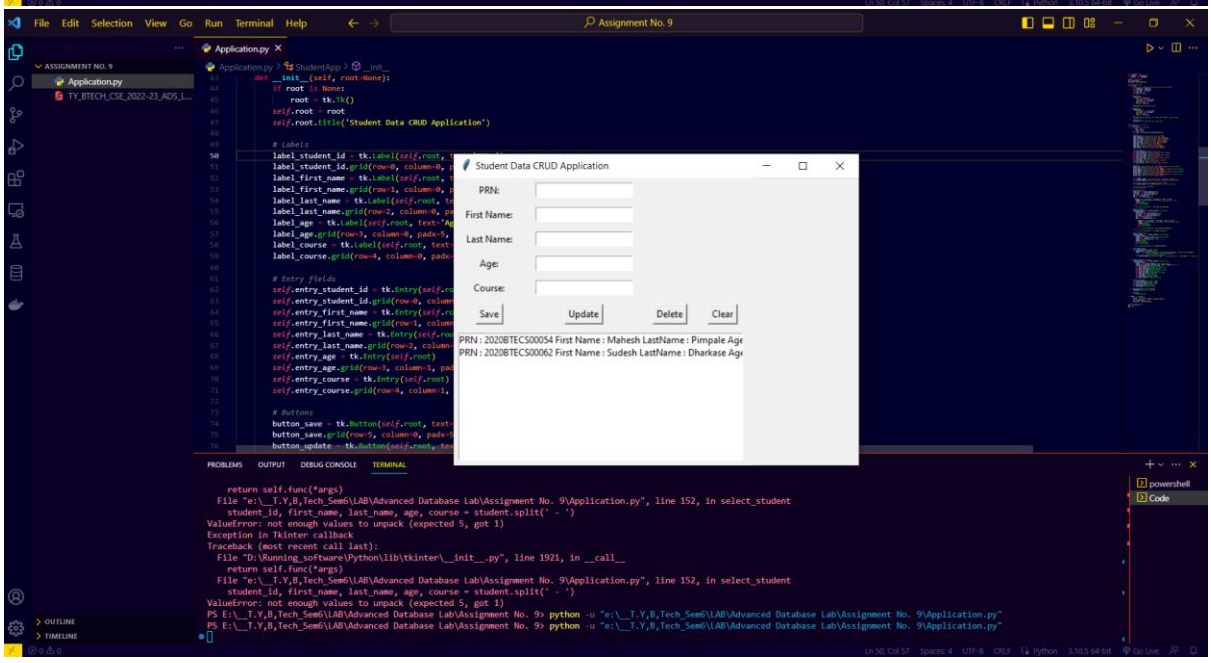
```
        client = MongoClient('mongodb://localhost:27017/')
        db = client['ADS']
        collection = db['StudentData']

# Create GUI application
app = StudentApp()
app.run()
```

Output:


MongoDB Compass
Connect View Help
Compass
New connection +
Saved connections
Recents
localhost:27017
Dec 4, 2022, 12:00 PM
localhost:27017
Dec 4, 2022, 10:59 AM
localhost:27017
Nov 8, 2022, 5:42 PM
localhost
Aug 7, 2022, 9:31 AM
localhost:27017
Aug 6, 2022, 10:18 AM
localhost
Aug 6, 2022, 10:16 AM
New Connection
Connect to a MongoDB deployment
FAVORITE
URI
Edit Connection String
mongodb://localhost:27017
Advanced Connection Options
Save
Save & Connect
Connect
New to Compass and don't have a cluster?
If you don't already have a cluster, you can create one for free using MongoDB Atlas
CREATE FREE CLUSTER
How do I find my connection string in Atlas?
If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.
See example
How do I format my connection string?
See example


File Edit Selection View Go Run Terminal Help
Assignment No. 9
Application.py
Student Data CRUD Application
PRN: 2020BTECS00054
First Name: Mahesh
Last Name: Pimpale
Age: 21
Course: CSE
Save
Update
Delete
Clear
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
ValueError: not enough values to unpack (expected 5, got 1)
Exception in Tkinter callback
Traceback (most recent call last):
ValueError: not enough values to unpack (expected 5, got 1)
PS E:__T.Y.B.Tech_Sem6\LAB\Advanced Database Lab\Assignment No. 9> python -u "e:__T.Y.B.Tech_Sem6\LAB\Advanced Database Lab\Assignment No. 9\Application.py"


Student Data CRUD Application
PRN:
First Name:
Last Name:
Age:
Course:
Save
Update
Delete
Clear
PRN : 2020BTECS00054 First Name : Mahesh LastName : Pimpale Age


Student Data CRUD Application
PRN:
First Name:
Last Name:
Age:
Course:
Save
Update
Delete
Clear
PRN : 2020BTECS00054 First Name : Mahesh LastName : Pimpale Age
PRN : 2020BTECS00062 First Name : Sudesh LastName : Dharkase Age