TO DO APP USING STREAMLIT PYTHON

INTRODUCTION

- **ToDo List App** is a kind of app that generally used to maintain our day-to-day tasks or list everything that we have to do, with the most important tasks at the top of the list, and the least important tasks at the bottom.
- It is helpful in planning our daily schedules.

SOFTWARE

Programming language:

Python,SQL(Structure Query Language)

PYTHON LIBRARIES REQUIRED:

- streamlit
- pandas
- mysql.connector/pymysql
- plotly

PYTHON LIBRARIES REQUIRED:

- **Streamlit** is an open source app framework in Python language. It helps us create web apps in a short time. It is compatible with major Python libraries.
- A Python API which **Streamlit** client apps use to instantiate the frontend and communicate with it.
- **Streamlit** is a framework for creating simple and elegant web applications in pure Python.
- This is the library that allows us to build frontend for apps by writing all the code in Python.

PYTHON LIBRARIES REQUIRED:

- **Pandas** is an open-source library in Python that is made mainly for working with relational or labeled data both easily and automatically.
- **Pandas** is a data manipulation package in Python for tabular data. That is, data in the form of rows and columns, also known as DataFrames. Intuitively, you can think of a DataFrame as an Excel sheet.
- Columns can be inserted and deleted from DataFrame and higher dimensional objects.

PYTHON LIBRARIES REQUIRED:

- The **MySQL Connector**/Python module is the official Oracle-supported driver to connect MySQL through Python. The connector is entirely Python.
- The **PyMySQL** package is another connector you can use to connect Python to MySQL. It's a good option if you're looking for speed, as it's faster than **mysql-connector-python**.

COMMAND FOR LIBRAIES INSTALLATION

- pip install streamlit
- pip install pandas
- pip install mysql.connector.python /pip install pymysql
- pip install plotly

STRUCTURE OF PROJECT

We have two files:

- **webapp.py:**

This file is used to build frontend for our todo list app.

- **dbfun.py:**

This file is used to manage backend database.

DESCRIPTION

- **dbfun.py : Here we are going to manage our application database.**
- Write functions for create table,add task ,delete task,update task,get unique task,view all tasks in python.
- Write sql query inside a function e.g. create_table function (create table query),add task(insert query),view all tasks(select query) etc.

DESCRIPTION

webapp.py :This file is used to build frontend for our todo list app.

- In this file we create four section :create,read,update and delete.
- **Create section :** It is used to make entries from user side.
- **Read section:** It is used to see all records in tabular form and graphical visualization of task or task_status in the form of pie graph.
- **Update section:**It is used to update record of table from user side.User can see current and updated record in the form of table.
- **Delete section:** It is used to delete a record from user side and user can see again current data and after deletion of records in the form table.

DATABASE INFORMATION

- Database name :mydata
- Table name: tasktable

Field /Column name	Datatype
task	Text
task_status	Text
Task_due_date	Date

CONNECT DATABASE WITH PYTHON

Connection object

```
import mysql.connector
conn_obj=mysql.connector.connect(host='localhost',
                                 user='root',
                                 password='root123',
                                 database='mydata')
```

CREATE CURSOR OBJECT

```
#cursor object
cur_object=conn_obj.cursor()
```

Cursor object

Connection object

CALL EXECUTE METHOD USING CURSOR OBJECT

```
def create_table():
    cur_object.execute("create table if not exists \
            taskstable(task Text,task_status Text,task_due_date Date)")
```

Cursor object

SQL Query for create a new table

CALL COMMIT METHOD USING CONNECT METHOD OBJECT

```
conn_obj.commit()
```

Connection object

Create section

GUI FILE

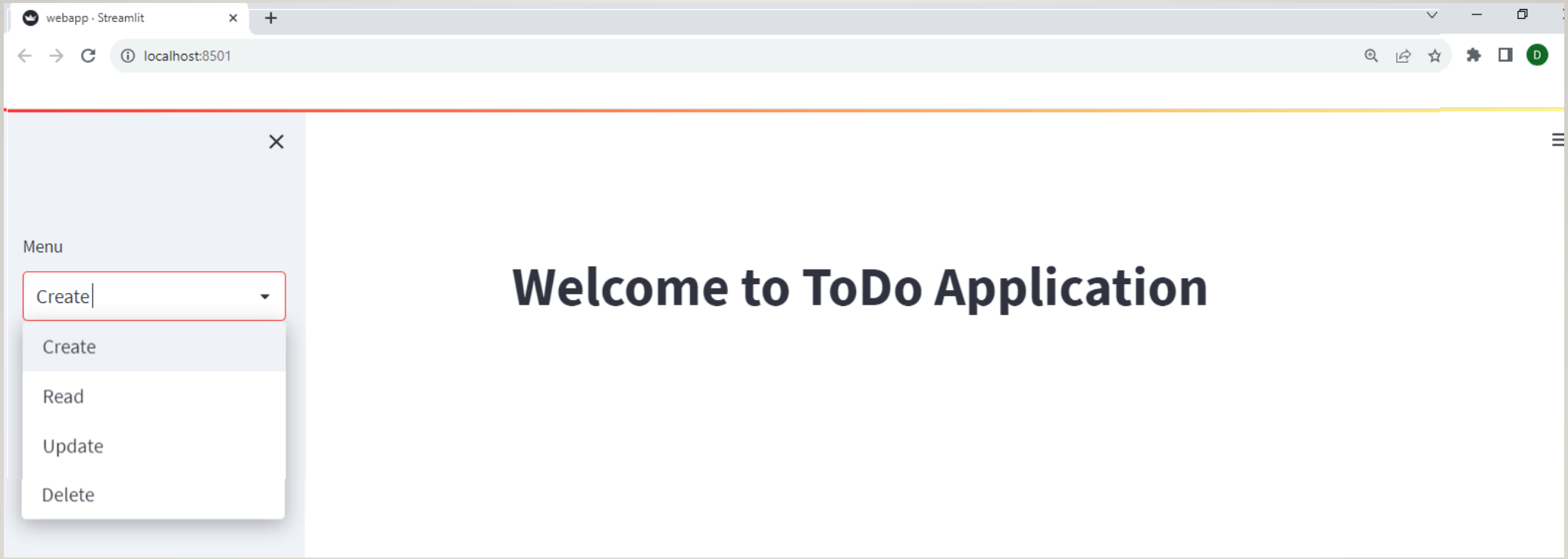
- In webapp.py ,we are going to call create_table function of dbfun.py.

```
import streamlit as st
from dbfun import *
ch=st.sidebar.selectbox("Menu",['Create','Read','Update','Delete'])
st.title("Welcome to ToDo Application")
create_table()
```

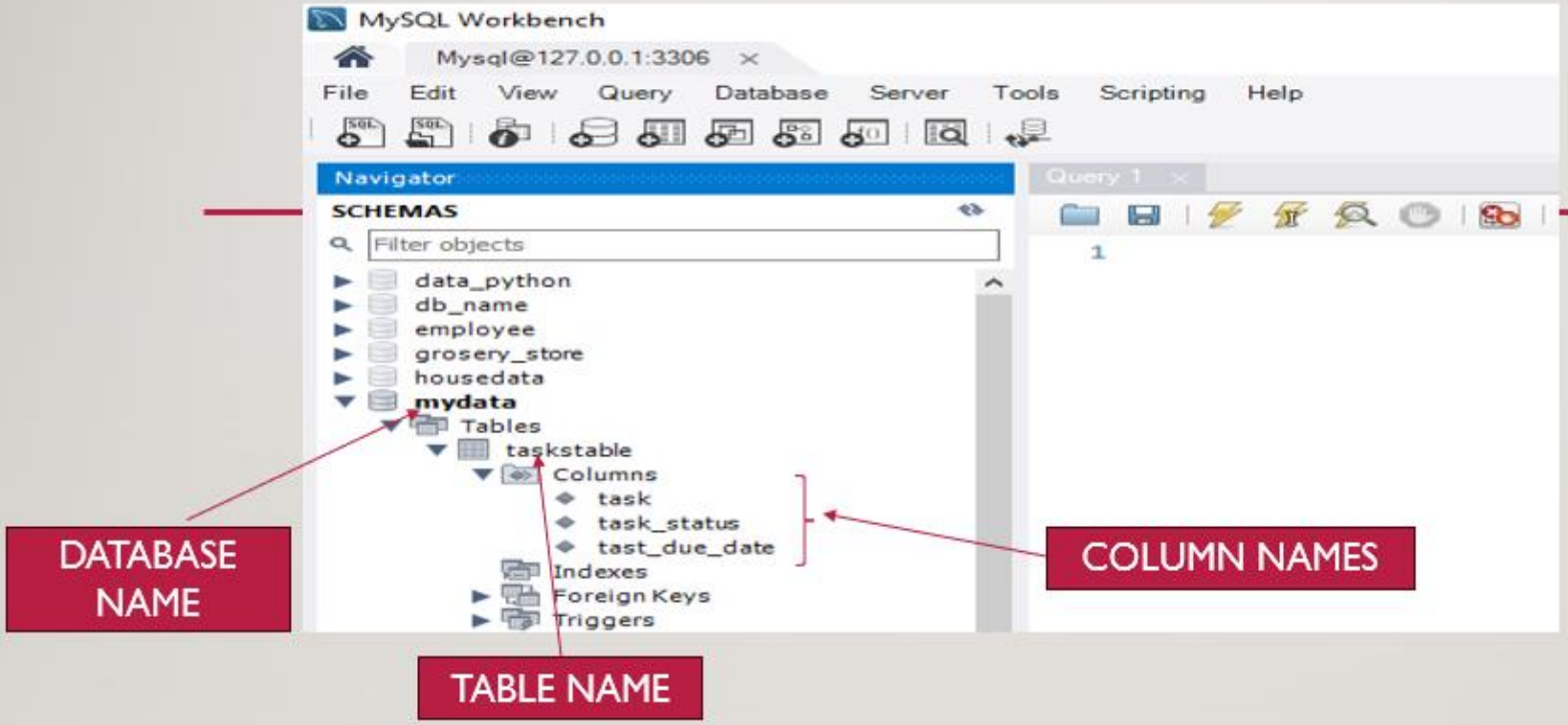
Import user defined module dbfun

Called function

OUTPUT OF PROGRAM ON BROWSER



OUTPUT OF PROGRAM IN MYSQLWORKBENCH

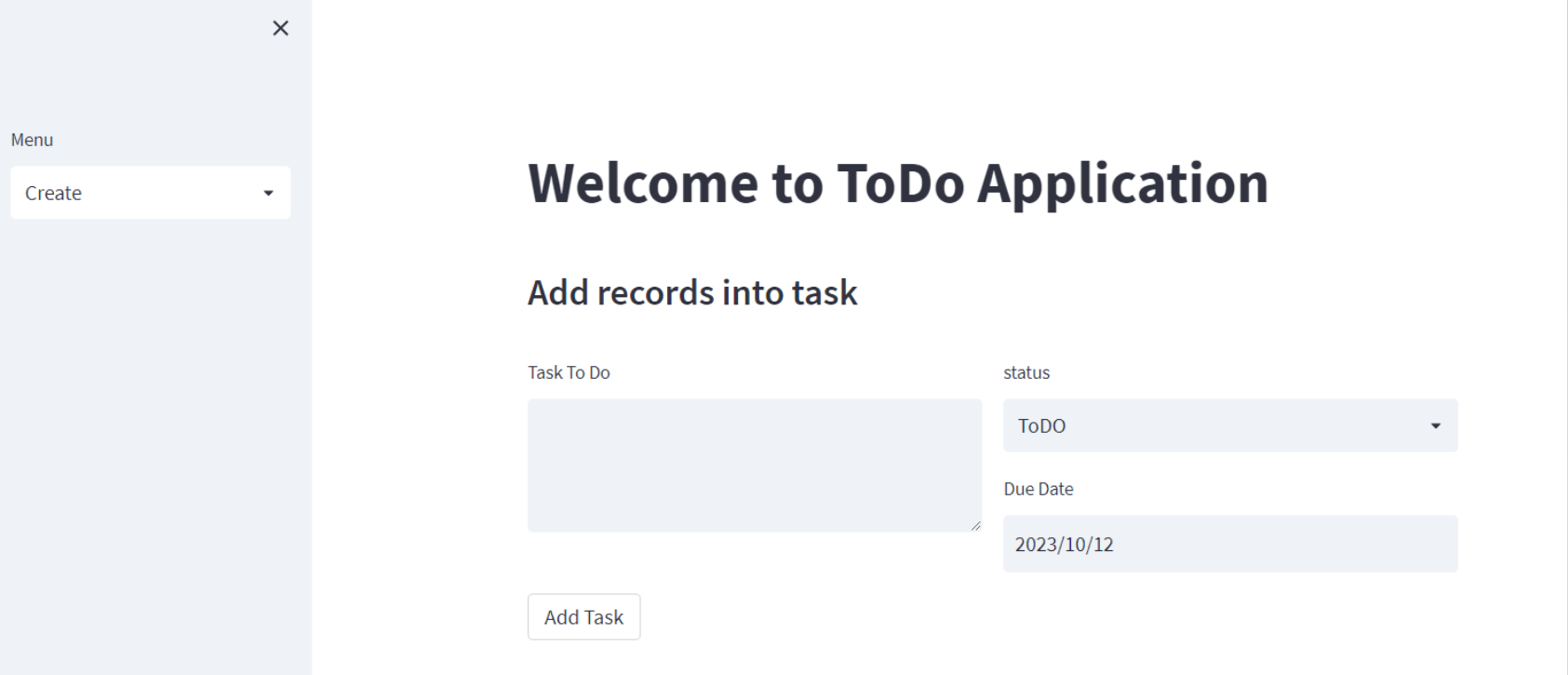


CODE FOR CREATE TAB FOR BROWSER

```
if ch=='Create':
    st.subheader("Add records into task")
    #layout
    col1,col2 = st.columns(2)
    with col1:
        task = st.text_area("Task To Do")

    with col2:
        task_status = st.selectbox("status",["ToDO","Doing","Done"])
        task_due_date = st.date_input("Due Date")
```

OUTPUT OF PROGRAM ON BROWSER



CODE FOR DBFUN.PY

Frontend variable as function parameter

SQL Query for inserting record into a table

```
def add_data(task,task_status,task_due_date):
    cur_object.execute('INSERT INTO taskstable(task,task_status,task_due_date) \
                        VALUES (%s,%s,%s)',(task,task_status,task_due_date))
    conn_obj.commit()
```

Cursor object

Format string for datatype

Backend variable

Frontend variable

CODE FOR WEBAPP.PY

```
if st.button("Add  Task"):
    add_data(task,task_status,task_due_date)
    st.success("Successfully Added Data:{}".format(task))
```

Function called

Frontend variable

OUTPUT OF PROGRAM

After code completion ,both files dbfun.py and webapp.py,

We are going to add records into a front end and automatically changes made into database.

localhost:8501

Menu

Create

Welcome to ToDo Application

Add records into task

Task To Do

python topic

status

ToDO

Due Date

2023/10/17

Add Task

Successfully Added Data:python topic

Read section

CODE FOR DBFUN.PY

SQL Query for return all columns records

```
def view_all_data():
    cur_object.execute('select * from taskstable')
    data = cur_object.fetchall()
    return data
```

Cursor object

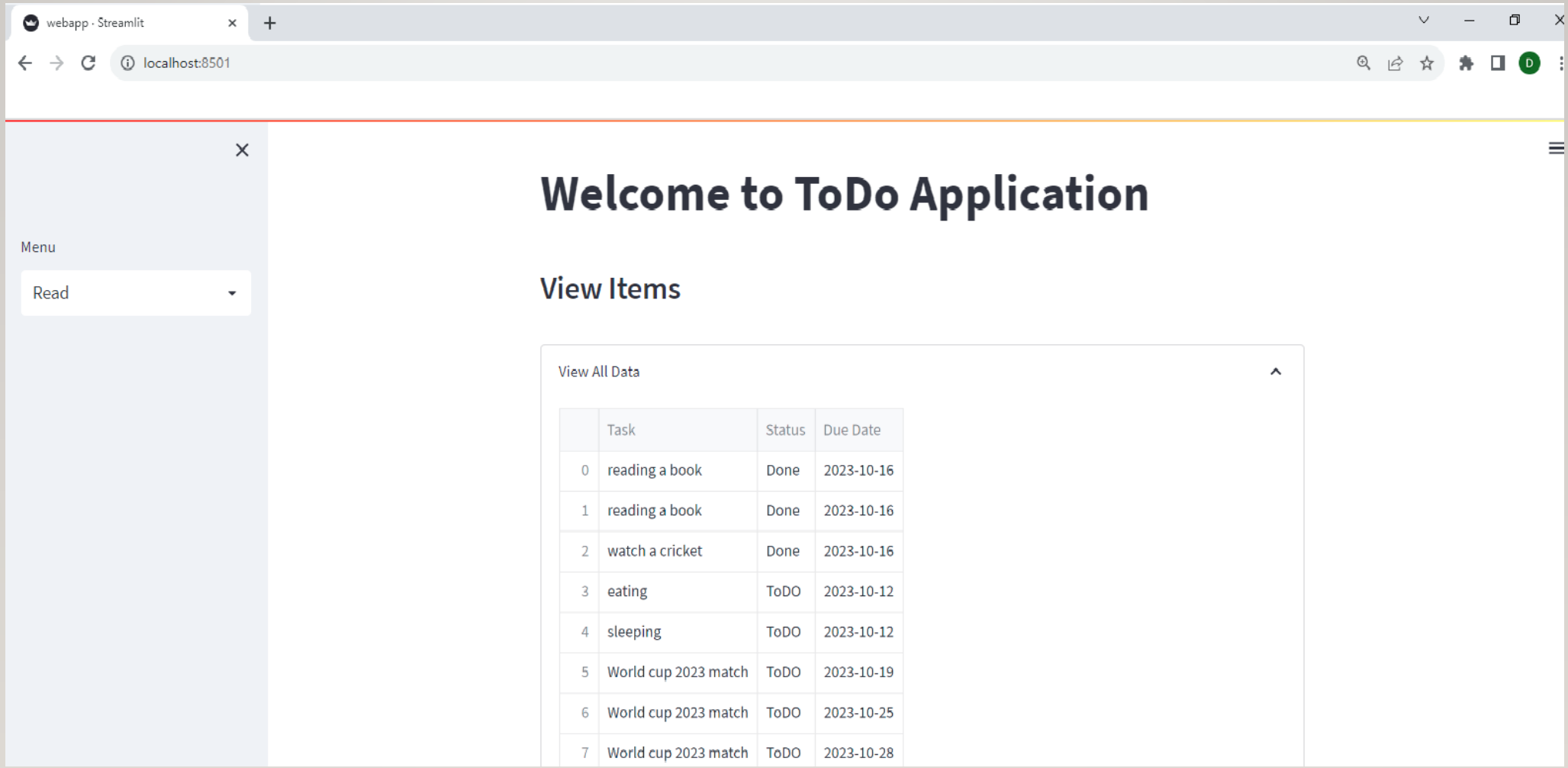
CODE FOR READ TAB IN WEBAPP.PY

```
elif ch=='Read':
    st.subheader("View Items")
    result = view_all_data()
    df = pd.DataFrame(result,columns=['Task','Status','Due Date'])
    with st.expander("View All Data"):
        st.dataframe(df)
```

Function called

Visualize result in tabular form on browser

OUTPUT OF PROGRAM



	Task	Status	Due Date
0	reading a book	Done	2023-10-16
1	reading a book	Done	2023-10-16
2	watch a cricket	Done	2023-10-16
3	eating	ToDO	2023-10-12
4	sleeping	ToDO	2023-10-12
5	World cup 2023 match	ToDO	2023-10-19
6	World cup 2023 match	ToDO	2023-10-25
7	World cup 2023 match	ToDO	2023-10-28

Update section

CODE FOR DBFUN.PY

SQL Query for update the record

```
def edit_task_data(new_task,new_task_status,new_task_due_date, \
                   task,task_status,task_due_date):
    cur_object.execute("UPDATE tasktable SET task =%s,task_status=%s,\
                       task_due_date=%s WHERE  task=%s and \
                       task_status=%s and task_due_date=%s",\
                           (new_task,new_task_status,new_task_due_date,
                            task,task_status,task_due_date))
    conn_obj.commit()
    data = cur_object.fetchall()
    return data
```

Connection object

Cursor object

CODE FOR DBFUN.PY

SQL Query for getting unique task

```
def view_unique_task():
    cur_object.execute('SELECT DISTINCT task FROM taskstable')
    data = cur_object.fetchall()
    return data
def get_task(task):
    cur_object.execute('SELECT * FROM taskstable WHERE task="{}"'.format(task
    data = cur_object.fetchall()
    return data
```

SQL Query for getting particular task records

Cursor object

CODE FOR UPDATE TAB IN WEBAPP.PY

```
elif ch=='Update':
    st.subheader("Update records")
    #see previous  records
    result=view_all_data()
    #st.write(result)
    df=pd.DataFrame(result,columns=['Task','Status','Due Date'])
    with st.expander("Current data"):
        st.dataframe(df)
    #st.write(view_unique_task())
    list_of_task=[i[0] for i in view_unique_task()]
    #st.write(list_of_task)
    selected_task=st.selectbox("Task to edit",list_of_task)
    selected_result=get_task(selected_task)
```

This file is used to build frontend for our todo list app.

Function called

Visualize result in tabular form on browser

CODE FOR UPDATE TAB IN WEBAPP.PY

```
if selected_result:
    task=selected_result[0][0]
    task_status=selected_result[0][1]
    task_due_date=selected_result[0][2]
    #layout
    col1,col2 = st.columns(2)
    with col1:
        new_task = st.text_area("Task To Do",task)
    with col2:
        new_task_status = st.selectbox("status",["ToDO","Doing","Done"])
        new_task_due_date = st.date_input("Due Date")
```

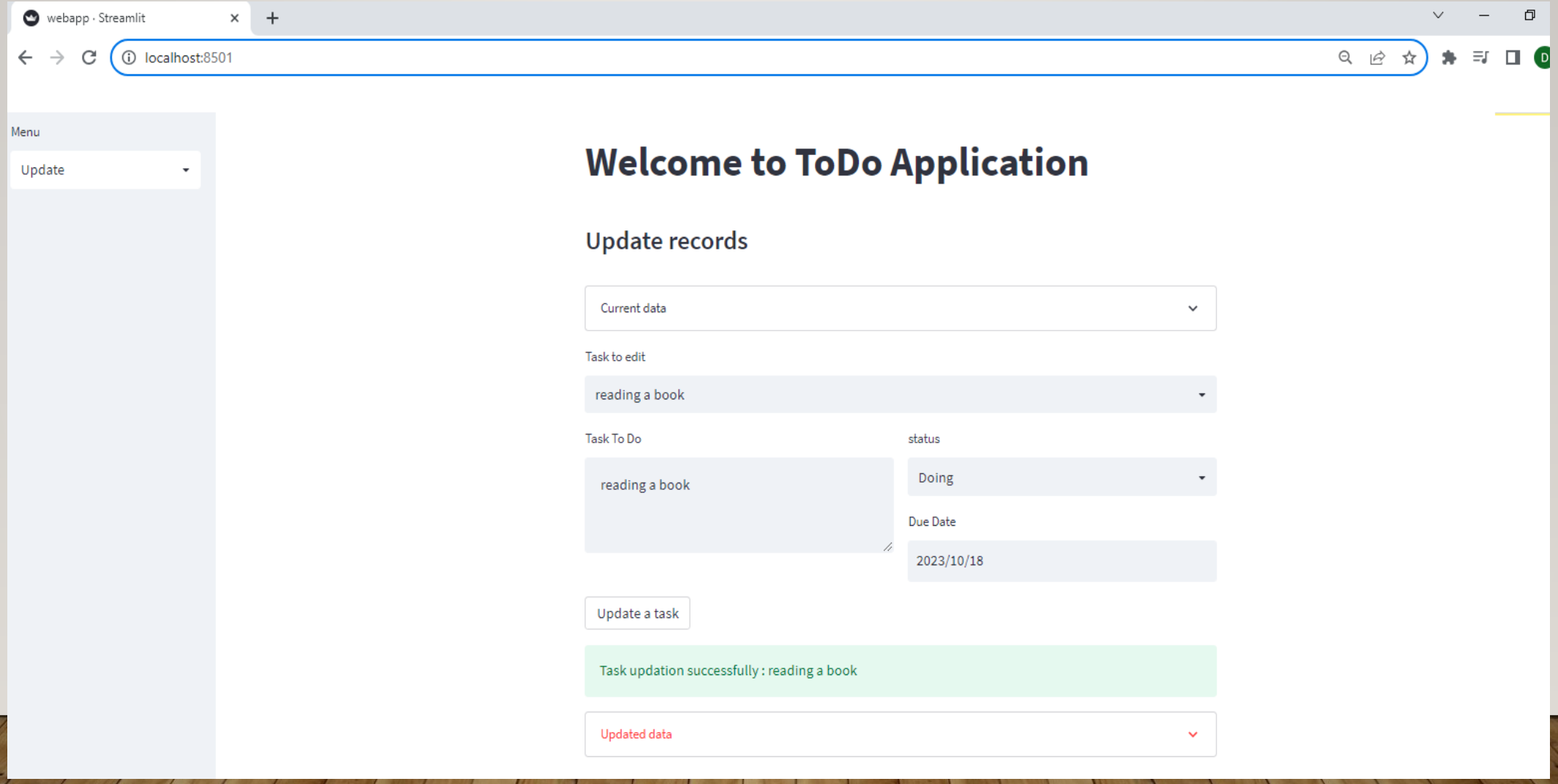
CODE FOR UPDATE TAB IN WEBAPP.PY

```
with col2:
    new_task_status = st.selectbox("status",["ToDO","Doing","Done"])
    new_task_due_date = st.date_input("Due Date")
if st.button("Update a task"):
    edit_task_data(new_task,new_task_status,new_task_due_date,task,
                   task_status,task_due_date)
    st.success(f"Task updation successfully : {task}")
result2=view_all_data()
#st.write(result)
df1=pd.DataFrame(result2,columns=['Task','Status','Due Date'])
with st.expander("Updated data"):
    st.dataframe(df1)
```

Function called

Visualize result in tabular form on browser

OUTPUT OF PROGRAM



Delete section

CODE FOR DBFUN.PY

SQL Query for delete the record

```
def delete_data(task):
    cur_object.execute('DELETE FROM taskstable WHERE task = "{}"'.format(task))
    conn_obj.commit()
```

Cursor object

Connection object

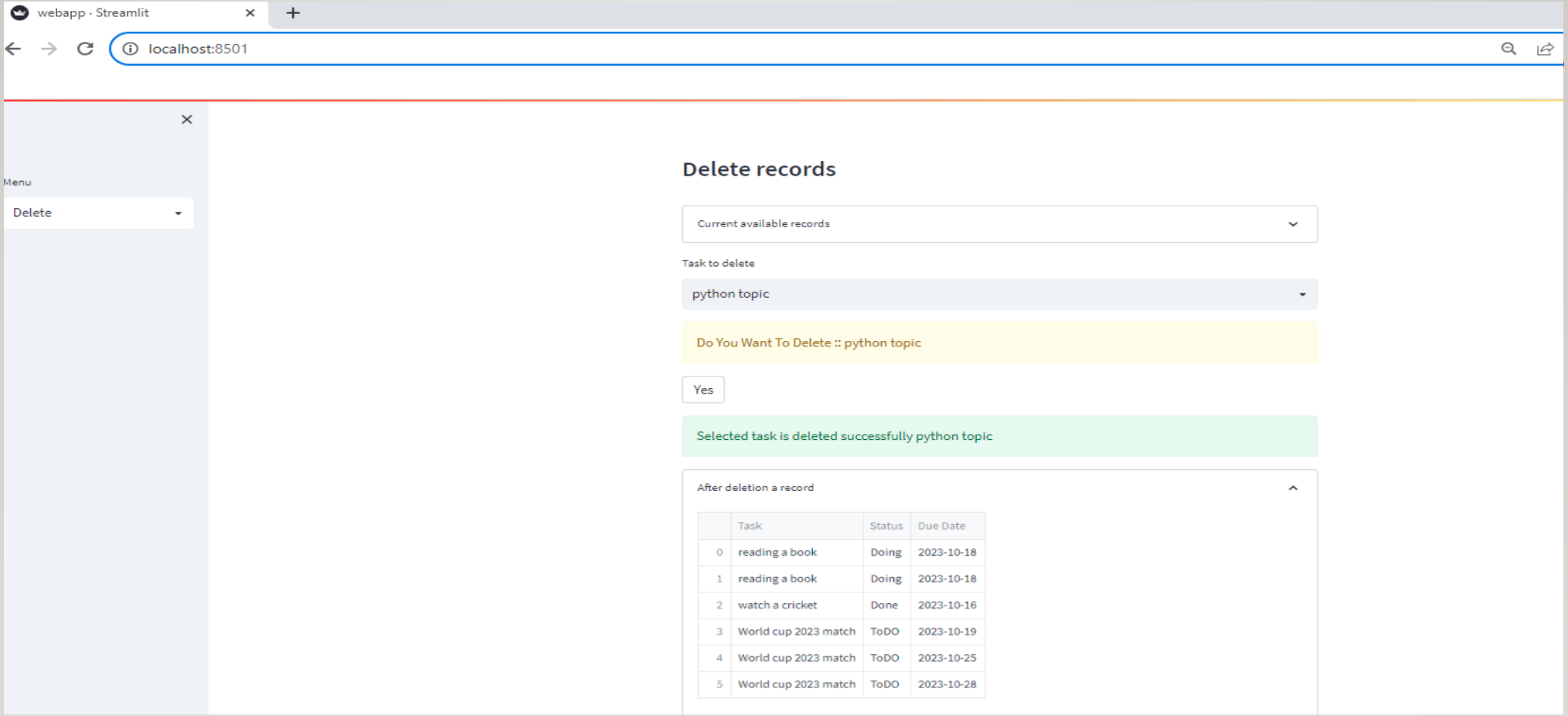
CODE FOR DELETE TAB IN WEBAPP.PY

```
elif ch=='Delete':
    st.subheader("Delete records")
    data=view_all_data()
    df=pd.DataFrame(data,columns=['Task','Status','Due Date'])
    with st.expander("Current available records"):
        st.dataframe(df)
    list_of_task=[i[0] for i in view_unique_task()]
    #st.write(list_of_task)
    selected_task=st.selectbox("Task to delete",list_of_task)
    #selected_result=get_task(selected_task)
    #st.write(selected_result)
    st.warning("Do You Want To Delete :: {}".format(selected_task))
    if st.button("Yes"):
        delete_data(selected_task)
        st.success("Selected task is deleted successfully {}" .format(selected_task))
    updated_data=view_all_data()
    df1=pd.DataFrame(updated_data,columns=['Task','Status','Due Date'])
    with st.expander("After deletion a record"):
        st.dataframe(df1)
```

Function called

Visualize result in tabular form on browser

OUTPUT OF PROGRAM



CONCLUSION

- To-Do app using streamlit python project is an application specially built to keep track of tasks that need to be done. This application will be like a task keeper where the user would be able to enter the tasks that they need to do. Once they are done with their tasks they can also remove them from the list.

- One of the most important reasons you should use a to do app using streamlit python is that it will help you stay organised. When you write all your tasks in a list, they seem more manageable. When you've got a clear outline of the tasks you've got to do and those you've completed, it helps you stay focused.