**ReactPy Tutorial**

**How to Use React in Python Isha Choudhary**

[**https://github.com/Sha-98/ReactPy\_Login**](https://github.com/Sha-98/ReactPy_Login)

1. **Installing ReactPy**

To install the new python library ‘ReactPy’, which is based on React, a JavaScript library for building user interfaces. ReactPy is a python-based version of this library using which we can now build user interfaces using python. For installing this library into your system, we can use ‘pip’. Now, ReactPy currently can be downloaded in multiple formats, in combination with different backends as listed below.

* fastapi – <https://fastapi.tiangolo.com>
* flask – <https://palletsprojects.com/p/flask/>
* sanic – <https://sanicframework.org>
* starlette – <https://www.starlette.io/>
* tornado – <https://www.tornadoweb.org/en/stable/>

ReactPy includes build-in support for the implementation of these backend, and to install the required dependencies for each of these, the following code should be used, by choosing one of these from the list in the code. Also note, that you can install more than one options separated from a comma.

pip install "reactpy[fastapi,flask,sanic,starlette,tornado]"

pip install "reactpy[fastapi]"

If you want to install a “pure” version of ReactPy without a backend implementation, you can also do so without any installation extras using the “pip” command as follows.

pip install reactpy

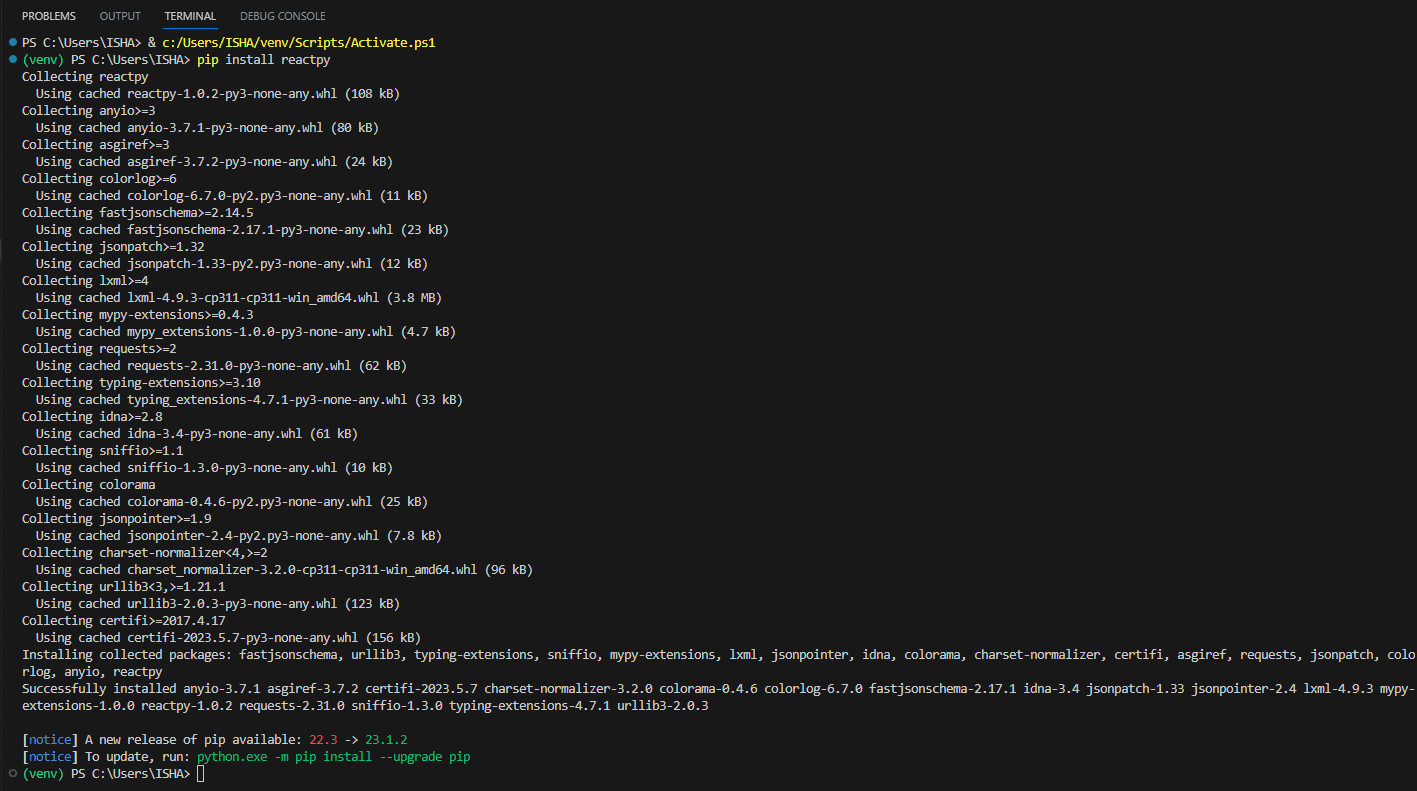
Now if you are using visual studio code, installation would require one more step, so that you do not get error for “ModuleNotFound”. For this, first open the folder for your project in VS Code and create a python file there.

Now, for installing python libraries, we need to create the virtual environment here first, using the following line of code.

python -m venv “*Path of your folder with ‘venv’ folder*”

python -m venv c:/Users/Asus/Desktop/Project/venv

Output:



1. **Connecting ReactPy with MongoDB Atlas database**

To connect the ReactPy application with MongoDB Atlas database, we use the python library “pymongo”. The code to connect the application with the NoSQL database MongoDB is given in the block below.

app = FastAPI()

from pymongo.mongo\_client import MongoClient

from pymongo.server\_api import ServerApi

#uri = “mongodb+srv://<username>:<password>@<cluster-url>/<database>”

uri = "mongodb+srv://root:1234@mongodblogin.pcv1to3.mongodb.net/SignUp"

# Create a new client and connect to the server

client = MongoClient(uri, server\_api=ServerApi("1"))

db = client ["SignUp"]

collection = db["users"]

# Send a ping to confirm a successful connection

try:

client.admin.command("ping")

print("Pinged your deployment. You successfully connected to MongoDB!")

except Exception as e:

print(e)

The above lines of code give us a successful connection with our mongodb database, if the “uri” added is correct, and contains the correct username, password, and details about the database created on mongodb. The given string is an example for readers to create one for their database. The first line of the code creates a FastAPI application instance named ‘app’, which is the reactpy application in our case.

Next, we are importing the required modules to connect to the mongodb database. ‘MongoClient’ is used to create a connection with the MongoDB database, and ‘ServerApi’ is to specify the version of the MongoDB server API. Following this, we are declaring the connection string for our MongoDB Atlas deployment, which includes the username, password, cluster url, and the database name.

Next, we are creating a ‘MongoClient’ instance to connect to specified database uri, and server API version, here as ‘1’ using the ‘ServerApi’ object. This line is establishing the connection to our MongoDB database. Following this is the name of the database and collection stored in 2 different variables. Now, to know if the connection is established, we are defining a condition in the last code block. Here, we use the ‘ping’ command to the MongoDB server through the ‘admin’ database of the connected client. If the ping is successful, it prints the success message as specified by us, otherwise it prints the error or un-successful message with the error encountered while establishing the connection.

1. **Basic Registration Form using ReactPy connected with the MongoDB database**

To create a basic registration form using ReactPy, which is connected with the MongoDB database, you can follow the below mentioned steps and code. Please note that all the dependencies and libraries used in the code must be installed in your system. One thing to note here is that reactpy is a python library and thus we do not need any specific commands to start a new project in reactpy. Just install it as explained above and import the library and required module to create an app.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A white screen with red text

Description automatically generated

**Step 1: Setting up the Backend Server and MongoDB Connection**

* Import the necessary modules and libraries, including ‘FastAPI’, ‘MongoClient’, and ‘ServerApi’.
* Create a FastAPI application instance using ‘FastAPI()’.
* Define the MongoDB connection as explained in previous question.

from fastapi import FastAPI

from pymongo.mongo\_client import MongoClient

from pymongo.server\_api import ServerApi

app = FastAPI()

uri = "mongodb+srv://root:1234@mongodblogin.pcv1to3.mongodb.net/SignUp"

# Create a new client and connect to the server

client = MongoClient(uri, server\_api=ServerApi("1"))

db = client["SignUp"]

collection = db["users"]

# Send a ping to confirm a successful connection

try:

    client.admin.command("ping")

    print("Pinged your deployment. You successfully connected to MongoDB!")

except Exception as e:

    print(e)

**Step 2: Define the Login Endpoint**

* Define a login function to store the input name and password of the user.
* Create a document to be inserted into collection which has these input details from the form.
* Print the created document to the console.
* Insert the document into the collection and retrieve the inserted document’s ID.
* Print the document’s id to check if data is being inserted well, you can also check the database.
* Return the success message.

def login(

    login\_data: dict,

):

    username = login\_data["name"]

    password = login\_data["password"]

    # Create a document to insert into the collection

    document = {"name": username, "password": password}

    # logger.info('sample log message')

    print(document)

    # Insert the document into the collection

    post\_id = collection.insert\_one(document).inserted\_id  # insert document

    print(post\_id)

    return {"message": "Login successful"}

**Step 3: Define the Frontend component for your application**

* Install ReactPy and import the necessary modules and functions from ‘reactpy’ and ‘reactpy.backend.fastapi’, as we are using FastAPI for our backed part.
* Create a ReactPy component using the ‘@component’ decorator.
* Define the necessary state variables using the ‘use\_state’ hook, including ‘alltodo’.
* Create a form submission function, here as ‘mysubmit’ which creates a new todo object and adds it to the ‘alltodo’ state.
* Now, this one is optional. We can print all the input data in an instance on the screen of the user. This will help us check what all data is sent to the database. For this, we would loop through the ‘alltodo’ state to generate a list of html, using ‘html.li’ command from reactpy. This list will then be printed on the html form using the ‘html.ul’ command.
* Define an event handler ‘handle\_event’ which prints the event to the console.
* Return an html div element with the form, including the input fields for name and password, a submit button with a condition for preventing reload on submit, and the generated list of todos.

from reactpy import component, event, html, use\_state

import reactpy as rp

@component

def MyCrud():

    ## Creating state

    alltodo = use\_state([])

    name, set\_name = use\_state("")

    password, set\_password = use\_state(0)

    def mysubmit(event):

        newtodo = {"name": name, "password": password}

        # push this to alltodo

        alltodo.set\_value(alltodo.value + [newtodo])

    # looping data from alltodo to show on web

    list = [

        html.li(

            {

                # {"key": b},

            },

            f"{b} => {i['name']} ; {i['password']} ",

        )

        for b, i in enumerate(alltodo.value)

    ]

    def handle\_event(event):

        print(event)

    return html.div(

        {"style": {"padding": "10px"}},

        ## creating form for submission\

        html.form(

            {"on submit": mysubmit},

            html.h1("Login Form - ReactPy & Mongodb"),

            html.input(

                {

                    "type": "test",

                    "placeholder": "Name",

                    "on\_change": lambda event: set\_name(event["target"]["value"]),

                }

            ),

            html.input(

                {

                    "type": "test",

                    "placeholder": "Password",

                    "on\_change": lambda event: set\_password(event["target"]["value"]),

                }

            ),

            # html.input(

            #     {

            #         "type": "test",

            #         "placeholder": "email id",

            #         "on\_change": lambda event: set\_age(event["target"]["value"]),

            #     }

            # ),

            # html.input(

            #     {

            #         "type": "test",

            #         "placeholder": "phone number",

            #         "on\_change": lambda event: set\_age(event["target"]["value"]),

            #     }

            # ),

            # creating submit button on form

            html.button(

                {

                    "type": "submit",

                    "on\_click": event(

                        lambda event: mysubmit(event), prevent\_default=True

                    ),

                },

                "Submit",

            ),

        ),

        html.ul(list),

    )

    return html.div(

        {"style": {"padding": "10px"}},

        ## creating form for submission\

        html.form(

            {"on submit": mysubmit},

            html.h1("Login Form - ReactPy & Mongodb"),

            html.input(

                {

                    "type": "test",

                    "placeholder": "Name",

                    "on\_change": lambda event: set\_name(event["target"]["value"]),

                }

            ),

            html.input(

                {

                    "type": "test",

                    "placeholder": "Password",

                    "on\_change": lambda event: set\_password(event["target"]["value"]),

                }

            ),

            # creating submit button on form

            html.button(

                {

                    "type": "submit",

                    "on\_click": event(

                        lambda event: mysubmit(event), prevent\_default=True

                    ),

                },

                "Submit",

            ),

        ),

        html.ul(list),

    )

**Step 4: Configure the FastAPI Backend with the Frontend component**

* Use the ‘configure’ function from ‘reactpy.backend.fastapi’ to configure the FastAPI app with the ‘MyCrud’ component.

from reactpy.backend.fastapi import configure

configure(app, MyCrud)

**Step 5: Run the application**

* Start the FastAPI server by running the Python script. For this, open the terminal, in VS Code (if using that, otherwise open the terminal at the location of your python file). Now, type the following ‘uvicorn main:app --reload’
* Open the web browser and navigate to the appropriate url to see the rendered ReactPy component and the login form.
* Interact with the form by entering a name and password and clicking the submit button.
* Check the console for any logged events or messages.

A screen shot of a computer

Description automatically generated