22AIE442 –

LABSHEET – 4

Name : Anuvind M P

Roll no: AM.EN.U4AIE22010

1. Implement a ROS service that takes a string input through a GUI and returns it in uppercase.

- Define a custom service that accepts a string and returns it in uppercase.
- Create a ROS service server in Python that handles the string conversion.
- Design a GUI to allow the user to input a string.
- Implement a service client that calls the ROS service and displays the uppercase result in the GUI.

.srv file :

StringUppercase.srv
~/catkin_ws/src/string_service/srv

```
string input
---
string output
```

ROS service server CODE:

string_service_server.py

Ubuntu-20.04 > home > anuvindmp > catkin_ws > src > string_uppercase_service > scripts > string_service_server.py

```python
#!/usr/bin/env python3

from string_uppercase_service.srv import StringToUpper, StringToUpperResponse
import rospy

def handle_uppercase_conversion(req):
    # Convert the input string to uppercase
    uppercase_string = req.input.upper()
    rospy.loginfo(f"Received string: {req.input} | Returning uppercase: {uppercase_string}")
    return StringToUpperResponse(uppercase_string)

def string_uppercase_server():
    # Initialize ROS node
    rospy.init_node('string_uppercase_server')
    # Create service
    s = rospy.Service('string_to_upper', StringToUpper, handle_uppercase_conversion)
    rospy.loginfo("Ready to convert string to uppercase.")
    rospy.spin()

if __name__ == "__main__":
    string_uppercase_server()
```

ROS service client CODE:

string_service_server.py 2 | string_service_client_gui.py 2

Ubuntu-20.04 > home > anuvindmp > catkin_ws > src > string_uppercase_service > scripts > string_service_client_gui.py > ...

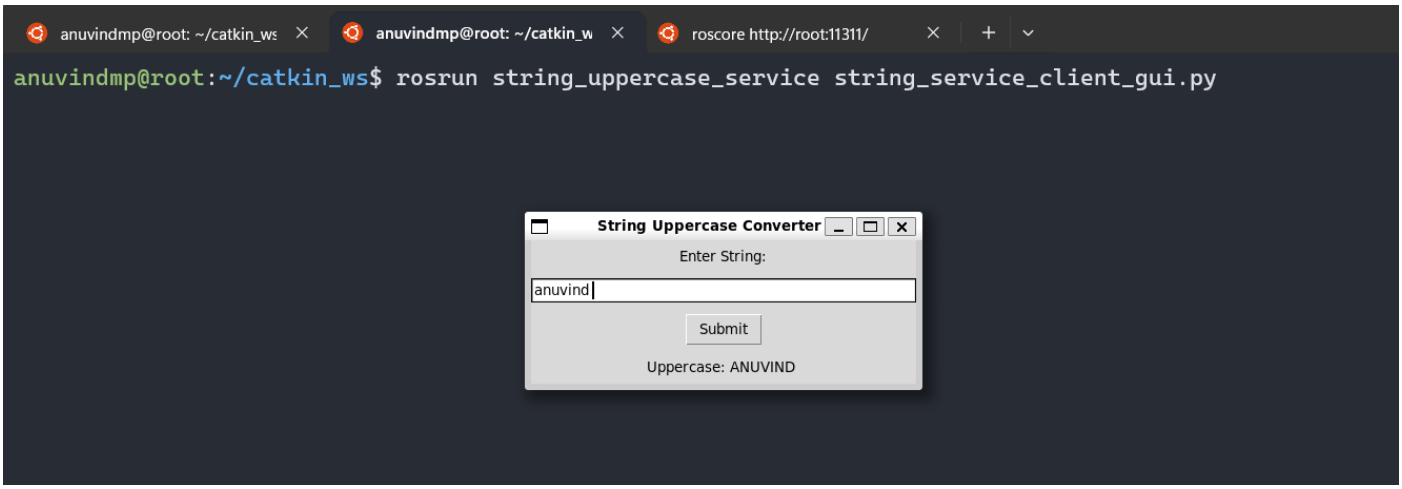
```python
#!/usr/bin/env python3
import rospy
from string_uppercase_service.srv import StringToUpper
import tkinter as tk
from tkinter import messagebox

def call_string_to_upper_service(input_string):
    rospy.wait_for_service('string_to_upper')
    try:
        string_to_upper = rospy.ServiceProxy('string_to_upper', StringToUpper)
        response = string_to_upper(input_string)
        return response.output
    except rospy.ServiceException as e:
        rospy.logerr(f"Service call failed: {e}")
        return None
def on_submit():
    input_string = entry.get()
    if input_string:
        result = call_string_to_upper_service(input_string)
        if result:
            result_label.config(text=f"Uppercase: {result}")
        else:
            messagebox.showerror("Error", "Failed to get response from service.")
    else:
        messagebox.showwarning("Input Error", "Please enter a string.")

rospy.init_node('string_service_client_gui', anonymous=True)
root = tk.Tk()
root.title("String Uppercase Converter")
tk.Label(root, text="Enter String:").pack(pady=5)
entry = tk.Entry(root, width=40)
entry.pack(pady=5)
submit_button = tk.Button(root, text="Submit", command=on_submit)
submit_button.pack(pady=5)
result_label = tk.Label(root, text="Uppercase: ")
result_label.pack(pady=5)

root.mainloop()
```

ROS service server output :

anuvindmp@root: ~/catkin_w | anuvindmp@root: ~/catkin_ws | roscore http://root:11311/

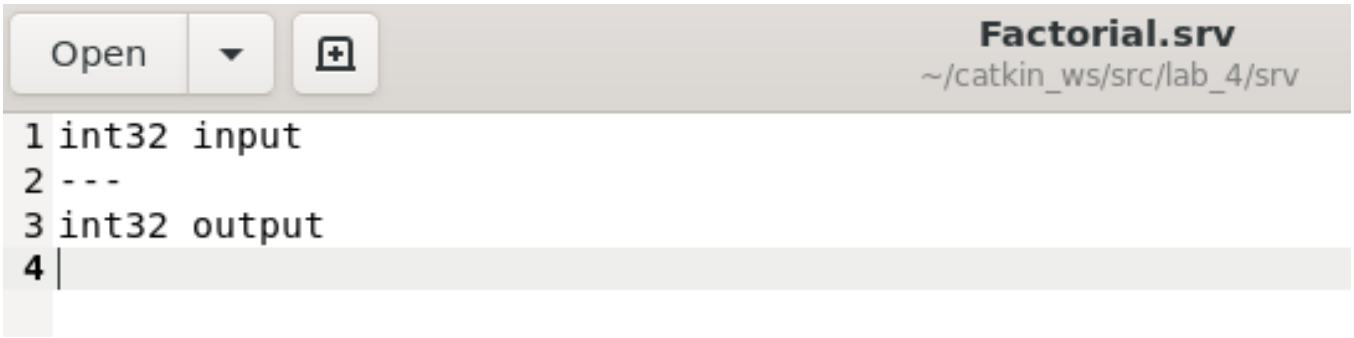
```
anuvindmp@root:~/catkin_ws/src/string_uppercase_service/scripts$ rosrun string_uppercase_service string_service_server.p
y
[INFO] [1729579707.403201]: Ready to convert string to uppercase.
[INFO] [1729579793.790988]: Received string: graahh | Returning uppercase: GRAAHH
[INFO] [1729579806.852378]: Received string: anuvind  | Returning uppercase: ANUVIND
```

ROS service client output :

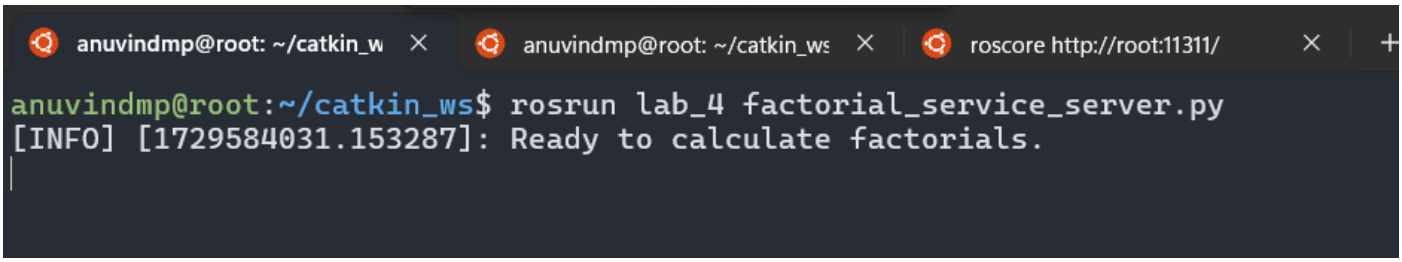


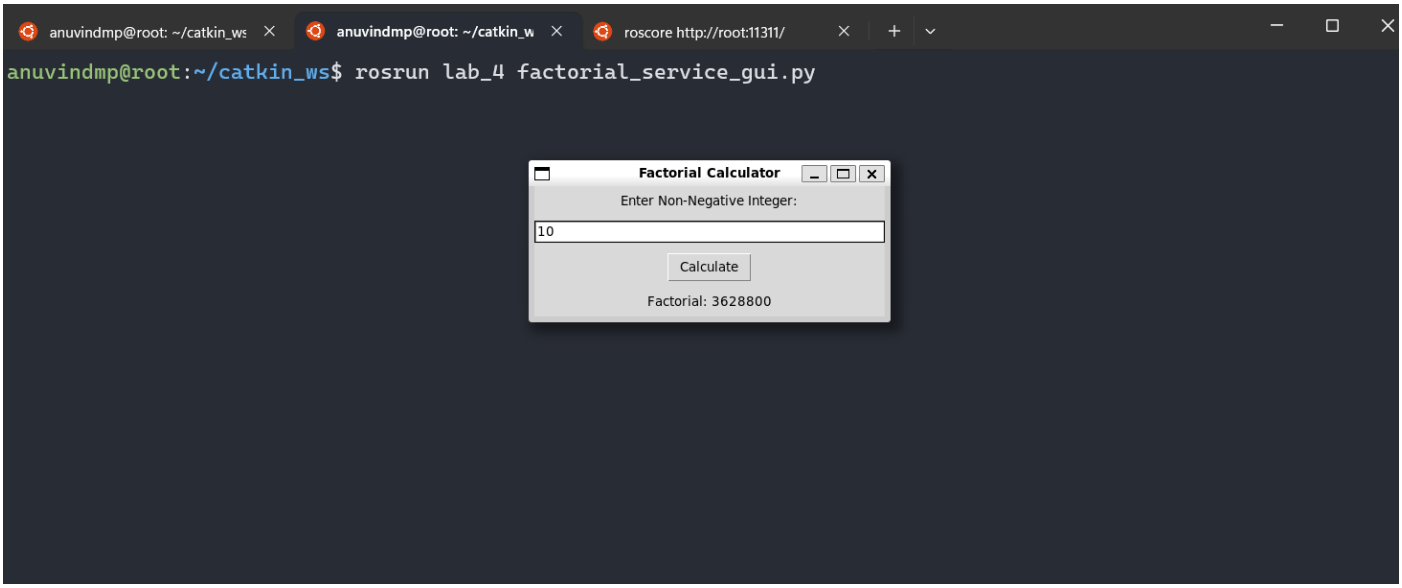
2. Implement a ROS service that calculates the factorial of a given non-negative integer input through a GUI. The service should return the factorial result
   - Define a custom service that accepts a non-negative integer and returns its factorial.
   - Create a ROS service server in Python that handles the factorial computation.
   - Design a GUI to allow the user to input a number.
   - Implement a service client that calls the ROS service and displays the factorial result in the GUI.

.srv file :



OUTPUT :





CODE :

factorial_service_gui.py 2 | factorial_service_server.py 2

Ubuntu-20.04 > home > anuvindmp > catkin_ws > src > lab_4 > scripts > factorial_service_server.py > ...

```python
#!/usr/bin/env python3
import rospy
from lab_4.srv import Factorial, FactorialResponse

def calculate_factorial(req):
    if req.input < 0:
        return FactorialResponse(-1)  # Return -1 for invalid input
    factorial = 1
    for i in range(1, req.input + 1):
        factorial *= i
    return FactorialResponse(factorial)

def factorial_service_server():
    rospy.init_node('factorial_service_server')
    s = rospy.Service('factorial_service', Factorial, calculate_factorial)
    rospy.loginfo("Ready to calculate factorials.")
    rospy.spin()

if __name__ == "__main__":
    factorial_service_server()
```

factorial_service_gui.py 2 × factorial_service_server.py 2

Ubuntu-20.04 > home > anuvindmp > catkin_ws > src > lab_4 > scripts > factorial_service_gui.py > ...

```python
#!/usr/bin/env python3
import rospy
from lab_4.srv import Factorial
import tkinter as tk
from tkinter import messagebox

def call_factorial_service(input_number):
    rospy.wait_for_service('factorial_service')
    try:
        factorial_service = rospy.ServiceProxy('factorial_service', Factorial)
        response = factorial_service(input_number)
        return response.output
    except rospy.ServiceException as e:
        rospy.logerr(f"Service call failed: {e}")
        return None

def on_submit():
    input_number = entry.get()
    if input_number.isdigit():  # Check if the input is a non-negative integer
        input_number = int(input_number)
        result = call_factorial_service(input_number)
        if result is not None:
            result_label.config(text=f"Factorial: {result}")
        else:
            messagebox.showerror("Error", "Failed to get response from service.")
    else:
        messagebox.showwarning("Input Error", "Please enter a non-negative integer.")

# Initialize ROS node
rospy.init_node('factorial_service_client_gui', anonymous=True)

# Create GUI
root = tk.Tk()
root.title("Factorial Calculator")

tk.Label(root, text="Enter Non-Negative Integer:").pack(pady=5)
entry = tk.Entry(root, width=40)
entry.pack(pady=5)

submit_button = tk.Button(root, text="Calculate", command=on_submit)
submit_button.pack(pady=5)

result_label = tk.Label(root, text="Factorial: ")
result_label.pack(pady=5)

# Start the GUI event loop
root.mainloop()
```