

## 1. Flow of the program

## Step 1 – Configuring mongod.ex file and creating rs replica dataset

```
cd C:\Program Files\MongoDB\Server\4.4\bin
```

```
mongod.exe --replSet rs0 --dbpath "C:\Program Files\MongoDB\Server\4.4\data"
```

```
C:\Program Files\MongoDB\Server\4.4\data
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Program Files\MongoDB\Server\4.4\bin

C:\Program Files\MongoDB\Server\4.4\bin>mongod.exe --replSet rs0 --dbpath "C:\Program Files\MongoDB\Server\4.4\data"
{"t":{"$date":"2021-01-12T22:21:12.643+05:30"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 spe
ify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2021-01-12T22:21:12.649+05:30"},"s":"W", "c":"ASIO", "id":22661, "ctx":"main","msg":"No TransportLayer configured during NetworkInterface startup"}
{"t":{"$date":"2021-01-12T22:21:12.649+05:30"},"s":"I", "c":"NETWORK", "id":4648602, "ctx":"main","msg":"Implicit TCP FastOpen in use."}
{"t":{"$date":"2021-01-12T22:21:12.649+05:30"},"s":"W", "c":"ASIO", "id":22661, "ctx":"main","msg":"No TransportLayer configured during NetworkInterface startup"}
{"t":{"$date":"2021-01-12T22:21:12.651+05:30"},"s":"I", "c":"STORAGE", "id":4615611, "ctx":"initandlisten","msg":"MongoDB starting", "attr":{"pid":19556,"port":27017,"
dbPath":"C:/Program Files/MongoDB/Server/4.4/data","architecture":"64-bit","host":"LAPTOP-GLMSOOC"}}
{"t":{"$date":"2021-01-12T22:21:12.651+05:30"},"s":"I", "c":"CONTROL", "id":23398, "ctx":"initandlisten","msg":"Target operating system minimum version", "attr":{"ta
rgetMinOs":"Windows 7/Windows Server 2008 R2"}}
{"t":{"$date":"2021-01-12T22:21:12.651+05:30"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten","msg":"Build Info", "attr":{"buildInfo":{"version":"4.4.3","
gitVersion":"913d6be2acfb344de1b1f641360a8cf8d13","modules":[],"allocator":"tcmalloc","environment":{"distmod":"windows","distarch":"x86_64","target_arch":"x86_64"
}}}}}
{"t":{"$date":"2021-01-12T22:21:12.651+05:30"},"s":"I", "c":"CONTROL", "id":51765, "ctx":"initandlisten","msg":"Operating System", "attr":{"os":{"name":"Microsoft Wi
ndows","version":"10.0 (build 18839)"}}}
{"t":{"$date":"2021-01-12T22:21:12.651+05:30"},"s":"I", "c":"CONTROL", "id":21951, "ctx":"initandlisten","msg":"Options set by command line", "attr":{"options":{"rep
lication":{"replset":"rs0"},"storage":{"dbPath":"C:\\Program Files\\MongoDB\\Server\\4.4\\data\"}}}
{"t":{"$date":"2021-01-12T22:21:12.653+05:30"},"s":"I", "c":"STORAGE", "id":22270, "ctx":"initandlisten","msg":"Storage engine to use detected by data files","attr
":{"dbpath":"C:/Program Files/MongoDB/Server/4.4/data","storageEngine":"wiredtiger"}}
{"t":{"$date":"2021-01-12T22:21:12.654+05:30"},"s":"I", "c":"STORAGE", "id":22315, "ctx":"initandlisten","msg":"Opening WiredTiger", "attr":{"config":{"create,cache_s
ize=3541M,session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),fi
le_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=[recovery_progress,checkpoint_progress,compact_progre
ss]},"}}
{"t":{"$date":"2021-01-12T22:21:12.845+05:30"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message", "attr":{"message":["[1610470272:8
19556][140732991823520], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 9 through 10"]}}
{"t":{"$date":"2021-01-12T22:21:12.851+05:30"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message", "attr":{"message":["[1610470281:8
19556][140732991823520], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 10 through 10"]}}
{"t":{"$date":"2021-01-12T22:21:12.854+05:30"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message", "attr":{"message":["[1610470282:8
19556][140732991823520], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Main recovery loop: starting at 9/989656 to 10/256"]}}
{"t":{"$date":"2021-01-12T22:21:12.856+05:30"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message", "attr":{"message":["[1610470282:5
19556][140732991823520], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 9 through 10"]}}
{"t":{"$date":"2021-01-12T22:21:12.856+05:30"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message", "attr":{"message":["[1610470282:2
6157][140732991823520], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 10 through 10"]}}
{"t":{"$date":"2021-01-12T22:21:12.849+05:30"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message", "attr":{"message":["[1610470282:3
19556][140732991823520], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Set global recovery timestamp: (161046941, 1)"]}}
{"t":{"$date":"2021-01-12T22:21:12.860+05:30"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"initandlisten","msg":"WiredTiger message", "attr":{"message":["[1610470282:2
6157][140732991823520], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Set global recovery timestamp: (161046941, 1)"]}}
```

## Step 2 – opening mongo.exe(Run As administer) and initiate rs set

rs.initiate()

```

ess> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --
--bind_ip 127.0.0.1 to disable this warning
---
    Enable MongoDB's free cloud-based monitoring service, which will then receive and display
    metrics about your deployment (disk utilization, CPU, operation statistics, etc).

    The monitoring data will be available on a MongoDB website with a unique URL accessible to you
    and anyone you share the URL with. MongoDB may use this information to make product
    improvements and to suggest MongoDB products and deployment options to you.

    To enable free monitoring, run the following command: db.enableFreeMonitoring()
    To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
rs0:PRIMARY> rs.initiate()
{
  "operationTime" : Timestamp(1610470585, 3),
  "ok" : 0,
  "errmsg" : "already initialized",
  "code" : 23,
  "codeName" : "AlreadyInitialized",
  "$clusterTime" : {
    "clusterTime" : Timestamp(1610470585, 3),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
rs0:PRIMARY>

```

Step 3 – Opening recieve.py file, text\_transfer.py file, form.py file(in sequence)

python recieve.py

Python text\_transfer.py

python form.py

```

Administrator Command Prompt - python recieve.py
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\abhi\ADS Term project

C:\Users\abhi\ADS Term project>python recieve.py
[*] Waiting for messages. To exit press CTRL+C

```

```
C:\> Administrator: Command Prompt - python text_transfer.py
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\abhi_\ADS Term project

C:\Users\abhi_\ADS Term project>python text_transfer.py
Connected successfully!!!
```

```
C:\> Administrator: Command Prompt - python form.py
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\abhi_\ADS Term project

C:\Users\abhi_\ADS Term project>python form.py
* Serving Flask app "form" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with windowsapi reloader
* Debugger is active!
* Debugger PIN: 284-353-400
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Step 3 – checking database status before inserting data

Mongo DB

```
Show databases;

Use database;

Show collections;

db.textsummary.find()
```

```

rs0:PRIMARY> show databases;
admin          0.000GB
config         0.000GB
database       0.000GB
local         0.001GB
pymongo_test   0.000GB
rs0:PRIMARY> use database
switched to db database
rs0:PRIMARY> show collections
textsummary
rs0:PRIMARY> db.textsummary.find()
rs0:PRIMARY> _

```

## MySQL

```

Show databases;

Use text_data

Select * from text;

```

```

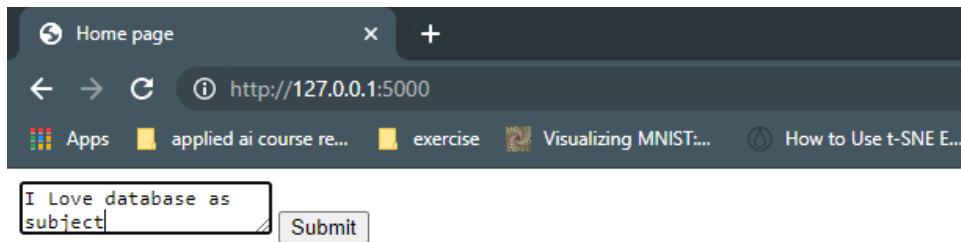
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql        |
| performance_schema |
| sakila       |
| sys         |
| text_data    |
| world       |
+-----+
7 rows in set (0.63 sec)

mysql> use text_data
Database changed
mysql> select *from text;
Empty set (0.08 sec)

mysql> _

```

Step 4 – Inserting data into form



Step 5 – Data taken from GUI and inserted into mongodb

```
Administrator: Command Prompt - python form.py
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with windowsapi reloader
* Debugger is active!
* Debugger PIN: 284-353-400
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Detected change in 'C:\\ProgramData\\Anaconda3\\Lib\\site-packages\\flask\\__pycache__'
127.0.0.1 - - [12/Jan/2021 22:38:44] "[37mGET / HTTP/1.1[0m" 200 -
* Restarting with windowsapi reloader
* Debugger is active!
* Debugger PIN: 284-353-400
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Jan/2021 22:38:49] "[33mGET /favicon.ico HTTP/1.1[0m" 404 -
data from gui
I Love database as subject
```

Step 6 – checking data in mongodb

```
db.textsummary.find()
```

```
rs0:PRIMARY> use database
switched to db database
rs0:PRIMARY> show collections
textsummary
rs0:PRIMARY> db.textsummary.find()
rs0:PRIMARY> db.textsummary.find()
{ "_id" : ObjectId("5ffdd7ce7a44f94541f221ec"), "message" : "I Love database as subject" }
rs0:PRIMARY>
```

Step 7 – Now checking message queue

```
C:\Users\abhi_\ADS Term project>python text_transfer.py
Connected successfully!!!
data not interested

In the message queue
inserted data I Love database as subject
```

Step 8 – Checking on the receiving side of the messaging queue

```
C:\Users\abhi_\ADS Term project>python recieve.py
[*] Waiting for messages. To exit press CTRL+C
I Love database as subject

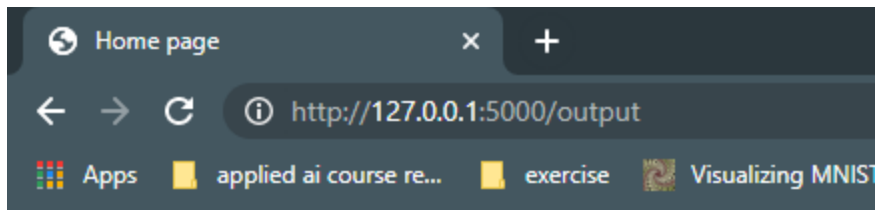
Data written to mysql database
```

Step 9 – checking on MySQL database

```
Select * from text;
```

```
mysql> select *from text;
+-----+-----+
| text_id | text_message |
+-----+-----+
|      34 | I Love database as subject |
+-----+-----+
1 row in set (0.00 sec)
```

Step 10 – Inserted data written back to output page



Data taken from MySQL

(34, 'I Love database as subject')

## 2. Important code snippets

### Register.html

Step 1 – User form

```
<form method = 'POST' action = " ">
    {{form.messagetext}}
    <input type="submit" value = "Submit">
```

### Form.py

Step 2 - Taking data from GUI (form.py)

```
@app.route('/',methods=['GET','POST'])
def register():
    form = message(request.form)
    if request.method == "POST" and form.validate():
        print("data from gui")
        print(form.messagetext.data)
```

Step 3– Connecting Mongo DB

```
try:
    conn = MongoClient()
    print("Connected successfully!!!")
except:
    print("Could not connect to MongoDB")
#creating database connection instance
db = conn.database
#creating new database collection
collection = db.textsummary
Text1 = {
    "message":form.messagetext.data
}
rec_id1 = collection.insert_one(Text1)
```



## Text\_transfer.py

Step 4 – CDC checking for database for change

```
collection = db.textsummary
with collection.watch() as stream:
    while stream.alive:
        change = stream.try_next()
        if change is not None:
```

Step 5 - Capturing change and setting up message queue

```
channel = connection.channel()
message = change["fullDocument"]["message"]
print("\n\n\n")
print("In the message queue")
print("inserted data",message)
print("\n\n\n")
channel.queue_declare(queue='hello')
channel.basic_publish(exchange='',routing_key='hello',body=message)
```

## Receiver.py

Step 6 – Queuing set up at receiver end

```
connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
channel = connection.channel()

channel.queue_declare(queue='hello')
```

Step 7 – Connecting to MySQL and inserting data into it

```
def callback(ch, method, properties, body):
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="abhi1998",
        database="text_data"
    )

    # Cursor setup
    cursor_sq = mydb.cursor()
    #inserting into mysql DATABASE
    print(body.decode('ascii'))
    add_text = "INSERT INTO text(text_message) VALUES ('{}').format(body.decode('ascii'))
    cursor_sq.execute(add_text)
```

## Form.py

Step 8 – Connecting to MySQL and retrieving data from it

```
def script_output():  
    mydb = mysql.connector.connect(  
        host="localhost",  
        user="root",  
        password="abhi1998",  
        database="text_data"  
    )  
  
    cursor_sq = mydb.cursor()  
    result = []  
    query = "select * from text"  
    cursor_sq.execute(query)  
    print("printing on command prompt")  
    for r in cursor_sq.fetchall():  
        result.append(r)
```

Step 9 – sending data to output.html page

```
for r in cursor_sq.fetchall():  
    result.append(r)  
return render_template('output.html', response = result)
```

## Output.html

Step 10 – Printing data on html page

```
<form method = 'POST' action = " ">  
<p>Data taken from MySQL</p>  
    {% for line in response %}  
        <p>{{ line }}</p>  
    {% endfor %}
```