NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE

KALYAN (EAST)

ACADEMIC YEAR 2024-25

M.Sc. Information Technology

Part II NEP 2020 Semester IV

SUBMITTED BY

Mr. RAJNIKANT ASHOK SHUKLA

AS PRESCRIBED BY

UNIVERSITY OF MUMBAI

MUMBAI UNIVERSITY

NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE

(Permanently Affiliated to University of Mumbai)

NAAC Accredited

Saket Vidyanagri Marg, Chinchpada Road, Katemanivali,
Kalyan (East) -421306(Mah)

Department of Information Technology

This is to certify that

Mr. RAJNIKANT ASHOK SHUKLA
Seat No. 1314284

of

M.Sc. Information Technology

Part II NEP 2020 Semester IV

has satisfactory carried out the required practical in the subject

of **BLOCKCHAIN**

For the Academic year 2024 – 2025

Practical In-Charge **Head of the Department**

External Examiner

College Seal

INDEX

Sr. No.		Practical	Signature
1.	a.	Develop a secure messaging application where users can exchange messages securely using RSA encryption. Implement a mechanism for generating RSA key pairs and encrypting/decrypting messages.	
	b.	Allow users to create multiple transactions and display them in an organised format	
	c.	Create a Python class named Transaction with attributes for sender, receiver, and amount. Implement a method within the class to transfer money from the sender's account to the receiver's account	
	d.	Implement a function to add new blocks to the miner and dump the blockchain	
2.	a.	Write a python program to demonstrate mining.	
	b.	Demonstrate the use of the Bitcoin Core API to interact with a Bitcoin Core node.	
	c.	Demonstrating the process of running a blockchain node on your local machine	
	d.	Demonstrate mining using geth on your private network.	

3.	a.	Write a Solidity program that demonstrates various types of functions including regular functions, view functions, pure functions, and the fallback function.	
	b.	Write a Solidity program that demonstrates function overloading, mathematical functions, and cryptographic functions.	
	c.	Write a Solidity program that demonstrates various features including contracts, inheritance, constructors, abstract contracts, interfaces.	
	d.	Write a Solidity program that demonstrates use of libraries, assembly, events, and error handling	
	e.	Build a decentralized application (DApp) using Angular for the front end and Trufle along with Ganache CLI for the back end.	
4.	a.	Install and demonstrate use of hyperledger-Irhoa	
	b.	Demonstration on interacting with NFT	

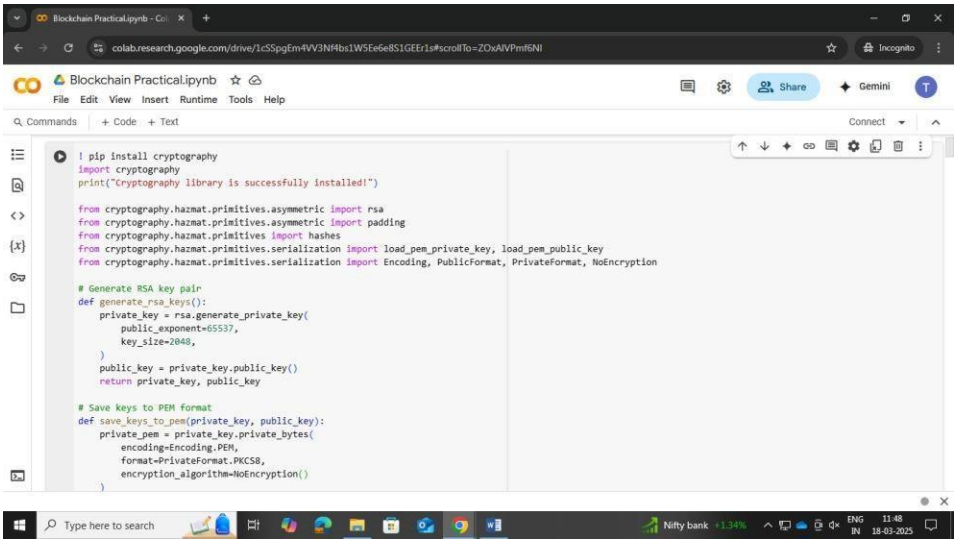
PRACTICAL 1 A.

Aim: Develop a secure messaging application where users can exchange messages securely using RSA encryption. Implement a mechanism for generating RSA key pairs and encrypting/decrypting messages.

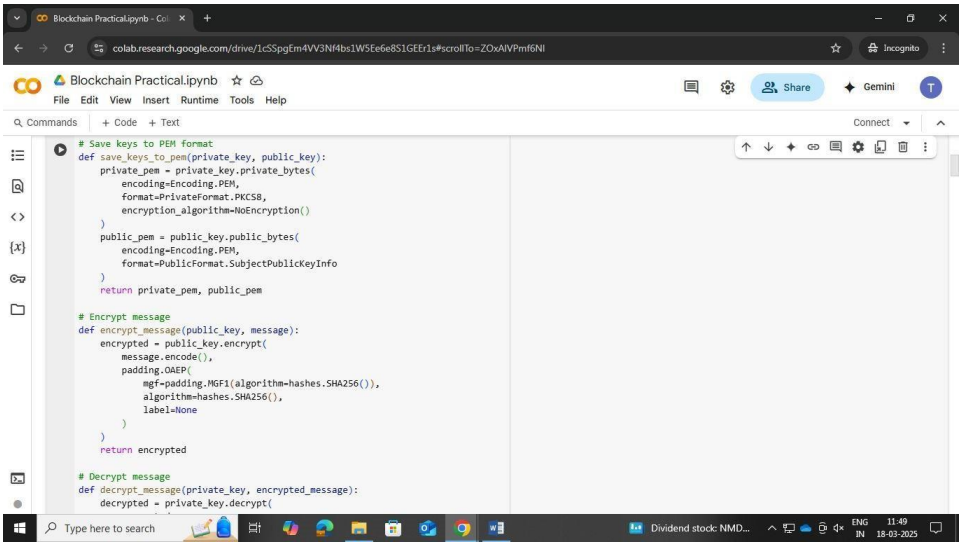
Code:

```
! pip install cryptography import
cryptography
print("Cryptography library is successfully installed!")
 from cryptography.hazmat.primitives.asymmetric import rsa from
cryptography.hazmat.primitives.asymmetric import padding from
cryptography.hazmat.primitives import hashes from
cryptography.hazmat.primitives.serialization import
load_pem_private_key, load_pem_public_key
from cryptography.hazmat.primitives.serialization import Encoding,
PublicFormat, PrivateFormat, NoEncryption

# Generate RSA key pair def
generate_rsa_keys():     private_key =
rsa.generate_private_key(
        public_exponent=65537,
key_size=2048,
    )
    public_key = private_key.public_key()     return
private_key, public_key

# Save keys to PEM format def
save_keys_to_pem(private_key, public_key):
private_pem =
private_key.private_bytes(          encoding=Encodi
ng.PEM,          format=PrivateFormat.PKCS8,
        encryption_algorithm=NoEncryption()
    )
    public_pem =
public_key.public_bytes(          encoding=Encoding.PEM,
        format=PublicFormat.SubjectPublicKeyInfo      )
```

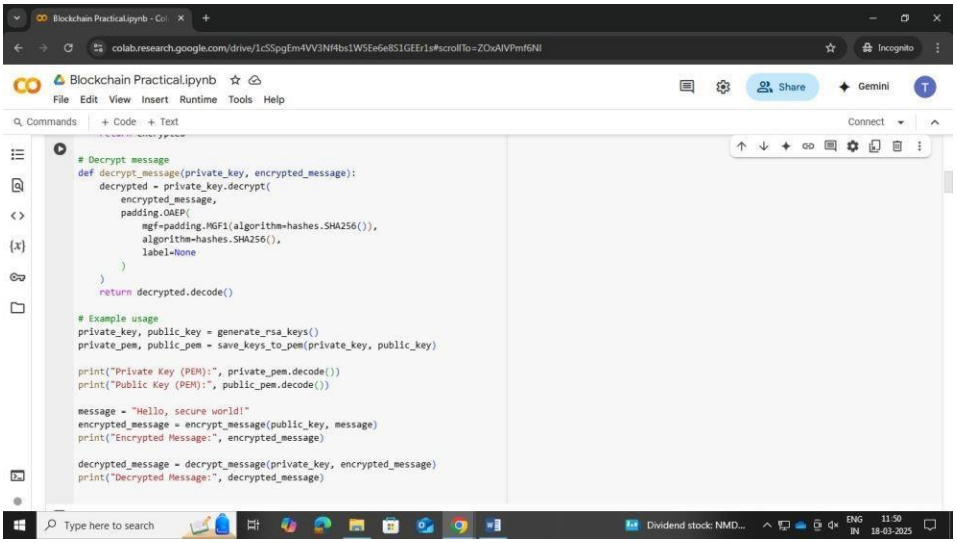
```
    return private_pem, public_pem

# Encrypt message def
encrypt_message(public_key, message):
encrypted =
public_key.encrypt(          message.encode(),
padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
algorithm=hashes.SHA256(),              label=None
        )     )
    return encrypted

# Decrypt message def decrypt_message(private_key,
encrypted_message):      decrypted =
private_key.decrypt(          encrypted_message,
padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
algorithm=hashes.SHA256(),              label=None
        )     )
    return decrypted.decode()

# Example usage
private_key, public_key = generate_rsa_keys()
private_pem, public_pem = save_keys_to_pem(private_key, public_key)
 print("Private Key (PEM):", private_pem.decode())
print("Public Key (PEM):", public_pem.decode())

message = "Hello, secure world!"
encrypted_message = encrypt_message(public_key, message) print("Encrypted
Message:", encrypted_message)

decrypted_message = decrypt_message(private_key, encrypted_message)
print("Decrypted Message:", decrypted_message)
```



```
! pip install cryptography
import cryptography
print("Cryptography library is successfully installed!")

from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.serialization import load_pem_private_key, load_pem_public_key
from cryptography.hazmat.primitives.serialization import Encoding, PublicFormat, PrivateFormat, NoEncryption

# Generate RSA key pair
def generate_rsa_keys():
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
    )
    public_key = private_key.public_key()
    return private_key, public_key

# Save keys to PEM format
def save_keys_to_pem(private_key, public_key):
    private_pem = private_key.private_bytes(
        encoding=Encoding.PEM,
        format=PrivateFormat.PKCS8,
        encryption_algorithm=NoEncryption()
    )
```

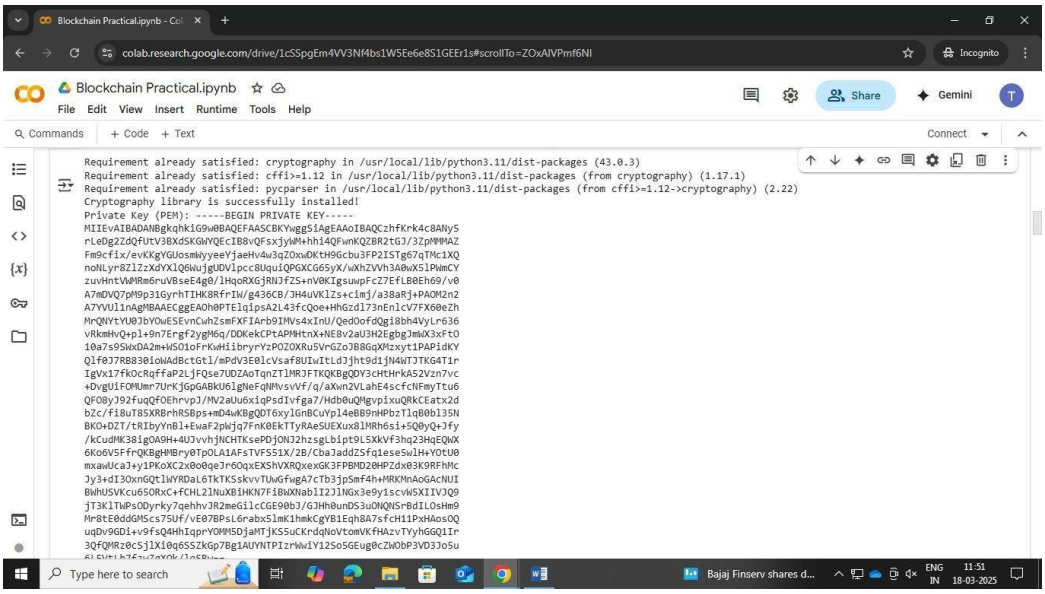


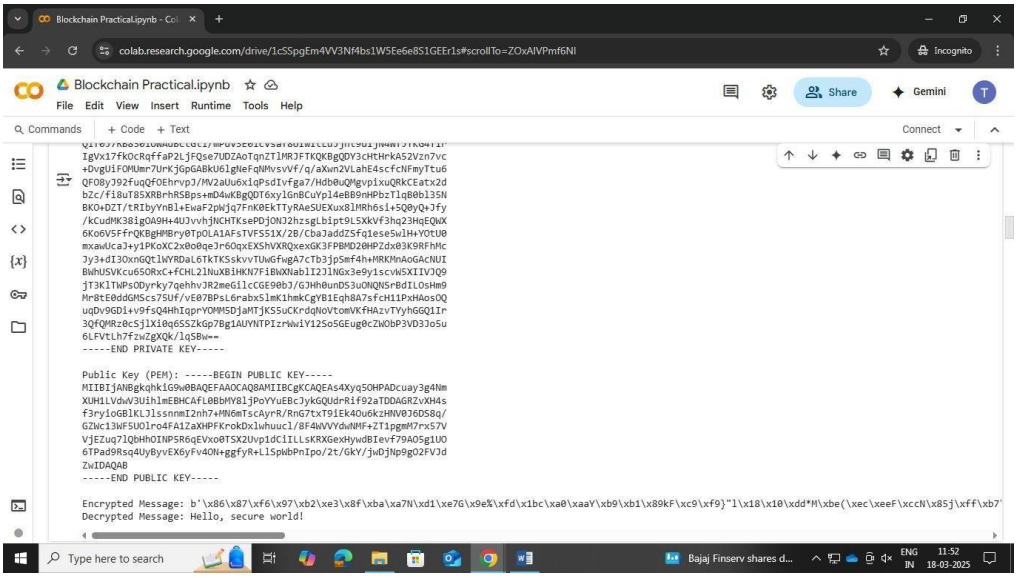
```
# Save keys to PEM format
def save_keys_to_pem(private_key, public_key):
    private_pem = private_key.private_bytes(
        encoding=Encoding.PEM,
        format=PrivateFormat.PKCS8,
        encryption_algorithm=NoEncryption()
    )
    public_pem = public_key.public_bytes(
        encoding=Encoding.PEM,
        format=PublicFormat.SubjectPublicKeyInfo
    )
    return private_pem, public_pem

# Encrypt message
def encrypt_message(public_key, message):
    encrypted = public_key.encrypt(
        message.encode(),
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return encrypted

# Decrypt message
def decrypt_message(private_key, encrypted_message):
    decrypted = private_key.decrypt(
```



```
# Decrypt message
def decrypt_message(private_key, encrypted_message):
    decrypted = private_key.decrypt(
        encrypted_message,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return decrypted.decode()

# Example usage
private_key, public_key = generate_rsa_keys()
private_pem, public_pem = save_keys_to_pem(private_key, public_key)

print("Private Key (PEM):", private_pem.decode())
print("Public Key (PEM):", public_pem.decode())

message = "Hello, secure world!"
encrypted_message = encrypt_message(public_key, message)
print("Encrypted Message:", encrypted_message)

decrypted_message = decrypt_message(private_key, encrypted_message)
print("Decrypted Message:", decrypted_message)
```

Output:


Blockchain Practical.ipynb - Col
colab.research.google.com/drive/1cSSpgEm4VV3Nf4bs1W5Ee6e8S1GEEr1s#scrollTo=ZOxAlVPmf6NI
Blockchain Practical.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
Requirement already satisfied: cryptography in /usr/local/lib/python3.11/dist-packages (43.0.3)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.11/dist-packages (from cryptography) (1.17.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.12->cryptography) (2.22)
Cryptography library is successfully installed!
Private Key (PEM): -----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCzhfKrk4c8ANy5
rLeDg2ZdQfUtV3BXdSKGWYQEcIB8vQFsxjyW+hhi4QFwnKQZBR2tGJ/3ZpMMAZ
Fm9cfix/evKkgYGUosmWyyeeYjaeHv4w3qZOxwDKtH9Gcbu3FP2ISTg67qTMc1XQ
noNLyr8ZlZzXdYXlQ6WvjgUDVlpcc8U6quiQPGXCG65yX/wXhZVVh3A0wX5lPWmCY
zuvHntVWMRm6ruVBseE4g0/lHqoRXGjRNJfZS+nV0KIgsuwpFcZ7EfLB0Eh69/v0
A7mDVQ7pM9p31GyrhTIHK8RfrIW/g436CB/JH4uVKlZs+cImj/a38aRj+PAOM2n2
A7YVUl1nAgMBAAECggEAOh0PTElq1psA2L43fcQoe+HhGzdl73nEnlcV7FX60eZh
MrQNYtYU0JbYOwESEvnCwhZsmFXFIArb9IMVs4xInU/QedOofdQgi8bh4VyLr636
vRkmHvQ+pl+9n7Ergf2ygMGq/DDKekCPtAPMHtnX+NE8v2aU3H2EgbgJmWX3xFtO
10a7s9SWxDA2m+WSO1oFrKwHiibryrYzPOZOXRu5VrGZoJB8GqXMzxyt1PAPidKY
Qlf0J7RB830ioWAdBctGtl/mPdV3E0lcVsaf8UIwItLdJjht9d1jN4WTJTKG4T1r
IgVx17fkOcRqffaP2LjFQse7UDZAoTqnZTlMRJFTKQKBgQDY3cHtHrkA52Vzn7vc
+DvgUiFOMUmr7UrKjGpGABkU6lgNeFqNVvsvVf/q/aXwn2VLahE4scfcNFmyTtu6
QFO8yJ92fuqQfOEhrvpJ/MV2aUu6xiqPsdIvfga7/Hdb0uQMgvpixuQRkCEatx2d
bZc/fi8uT8SXRBrhRSBps+mD4wKBgQDT6xylGnBCuYpl4eBB9nHPbzTlqB0bl3SN
BKO+DZT/tRIbyYnBl+EwaF2pWjq7FnK0EkTTyRAeSUEXux8lMRh6si+SQ0yQ+Jfy
/kCudMK38igOA9H+4UJvvhjNCHTKsePDjONJ2hzsgLbipt9L5XkVf3hq23HqEQWX
6Ko6V5FfrQKBgHMBry0TpOLA1AFsTVFS51X/2B/CbaJaddZSfq1ese5wlH+YOtU0
mxawUcaJ+y1PKoXC2x0o0qeJr6OqxEXShVXRQxexGK3FPBMD20HPZdx03K9RFhMc
Jy3+dI3OxnGQtlWYRDaL6TkTKSskvvTUwGfwgA7cTb3jp5mf4h+MRKMnAoGAcNUI
BWhUSVKcu65ORxC+fCHL2lNuXBiHKN7FiBWXNab1I2JlNGx3e9y1scvW5XIIVJQ9
jT3KlTWPsODyrky7qehhvJR2meGilcCGE90bJ/GJHh0unDS3uONQNSrBdILOsHm9
Mr8tE0ddGMScs75Uf/vE07BPsL6rabxSlmK1hmkCgYB1Eqh8A7sfcH11PxHAosOQ
uqDv9GDi+v9fsQ4HhIqprYOMMSDjaMTjKS5uCKrdqNoVtomVKfHAzvTYyhGGQ1Ir
3QfQMRz0cSjlXi0q6SSZkGp7Bg1AUYNTPIzrWwiY12So5GEug0cZWObP3VD3JoSu
6LFVtLh7FzwZgXQk/lq5Bw==
-----END PRIVATE KEY-----
Public Key (PEM): -----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAs4Xyq5OHPADcuay3g4Nm
XUH1LVdwV3UihlmEBHCAfL0BbMY8ljPoYYuEBcJykGQUdrRif92aTDDAGRZvXH4s
f3ryioGBlKLJlssnnmI2nh7+MN6mTscAyrR/RnG7txT9lEk4Ou6kzHNV0J6DS8q/
GZWc13WF5UOlro4FA1ZaXHPFKrokDxlwhuucl/8F4WVVYdwNMF+ZT1pgmM7rx57V
VjEZuq7lQbHhOINP5R6qEVxo0TSX2Uvp1dCIILLsKRXGexHywdBIevf79AO5g1UO
6TPad9Rsq4UyByvEX6yFv4ON+ggfyR+Ll5pWbPnIpo/2t/GkY/jwDjNp9gO2FVJd
ZwIDAQAB
-----END PUBLIC KEY-----
Encrypted Message: b'\x86\x87\xf6\x97\xb2\xe3\x8f\xba\xa7N\xd1\xe7G\x9e%\xfd\x1bc\xa0\xaaY\xb9\xb1\x89kF\xc9\xf9}"l\x18\x10\xdd*M\xbe(\xec\xeeF\xccN\x85j\xff\xb7
Decrypted Message: Hello, secure world!
Type here to search
Bajaj Finserv shares d...
ENG
IN
11:51
18-03-2025


Blockchain Practical.ipynb - Col
colab.research.google.com/drive/1cSSpgEm4VV3Nf4bs1W5Ee6e8S1GEEr1s#scrollTo=ZOxAlVPmf6NI
Blockchain Practical.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
IgVx17fkOcRqffaP2LjFQse7UDZAoTqnZTlMRJFTKQKBgQDY3cHtHrkA52Vzn7vc
+DvgUiFOMUmr7UrKjGpGABkU6lgNeFqNVvsvVf/q/aXwn2VLahE4scfcNFmyTtu6
QFO8yJ92fuqQfOEhrvpJ/MV2aUu6xiqPsdIvfga7/Hdb0uQMgvpixuQRkCEatx2d
bZc/fi8uT8SXRBrhRSBps+mD4wKBgQDT6xylGnBCuYpl4eBB9nHPbzTlqB0bl3SN
BKO+DZT/tRIbyYnBl+EwaF2pWjq7FnK0EkTTyRAeSUEXux8lMRh6si+SQ0yQ+Jfy
/kCudMK38igOA9H+4UJvvhjNCHTKsePDjONJ2hzsgLbipt9L5XkVf3hq23HqEQWX
6Ko6V5FfrQKBgHMBry0TpOLA1AFsTVFS51X/2B/CbaJaddZSfq1ese5wlH+YOtU0
mxawUcaJ+y1PKoXC2x0o0qeJr6OqxEXShVXRQxexGK3FPBMD20HPZdx03K9RFhMc
Jy3+dI3OxnGQtlWYRDaL6TkTKSskvvTUwGfwgA7cTb3jp5mf4h+MRKMnAoGAcNUI
BWhUSVKcu65ORxC+fCHL2lNuXBiHKN7FiBWXNab1I2JlNGx3e9y1scvW5XIIVJQ9
jT3KlTWPsODyrky7qehhvJR2meGilcCGE90bJ/GJHh0unDS3uONQNSrBdILOsHm9
Mr8tE0ddGMScs75Uf/vE07BPsL6rabxSlmK1hmkCgYB1Eqh8A7sfcH11PxHAosOQ
uqDv9GDi+v9fsQ4HhIqprYOMMSDjaMTjKS5uCKrdqNoVtomVKfHAzvTYyhGGQ1Ir
3QfQMRz0cSjlXi0q6SSZkGp7Bg1AUYNTPIzrWwiY12So5GEug0cZWObP3VD3JoSu
6LFVtLh7FzwZgXQk/lq5Bw==
-----END PRIVATE KEY-----
Public Key (PEM): -----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAs4Xyq5OHPADcuay3g4Nm
XUH1LVdwV3UihlmEBHCAfL0BbMY8ljPoYYuEBcJykGQUdrRif92aTDDAGRZvXH4s
f3ryioGBlKLJlssnnmI2nh7+MN6mTscAyrR/RnG7txT9lEk4Ou6kzHNV0J6DS8q/
GZWc13WF5UOlro4FA1ZaXHPFKrokDxlwhuucl/8F4WVVYdwNMF+ZT1pgmM7rx57V
VjEZuq7lQbHhOINP5R6qEVxo0TSX2Uvp1dCIILLsKRXGexHywdBIevf79AO5g1UO
6TPad9Rsq4UyByvEX6yFv4ON+ggfyR+Ll5pWbPnIpo/2t/GkY/jwDjNp9gO2FVJd
ZwIDAQAB
-----END PUBLIC KEY-----
Encrypted Message: b'\x86\x87\xf6\x97\xb2\xe3\x8f\xba\xa7N\xd1\xe7G\x9e%\xfd\x1bc\xa0\xaaY\xb9\xb1\x89kF\xc9\xf9}"l\x18\x10\xdd*M\xbe(\xec\xeeF\xccN\x85j\xff\xb7
Decrypted Message: Hello, secure world!
Type here to search
Bajaj Finserv shares d...
ENG
IN
11:52
18-03-2025

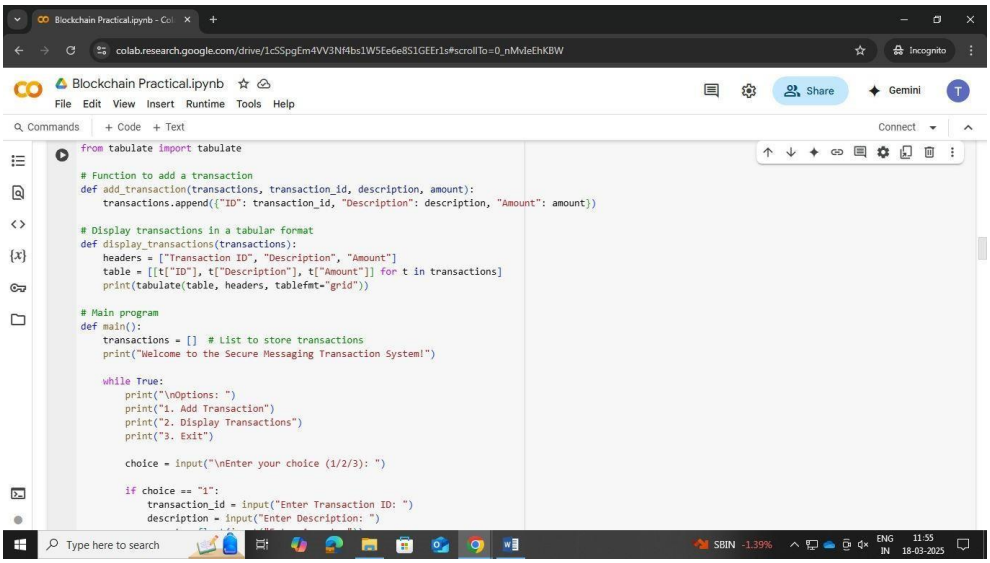
PRACTICAL 1 B.

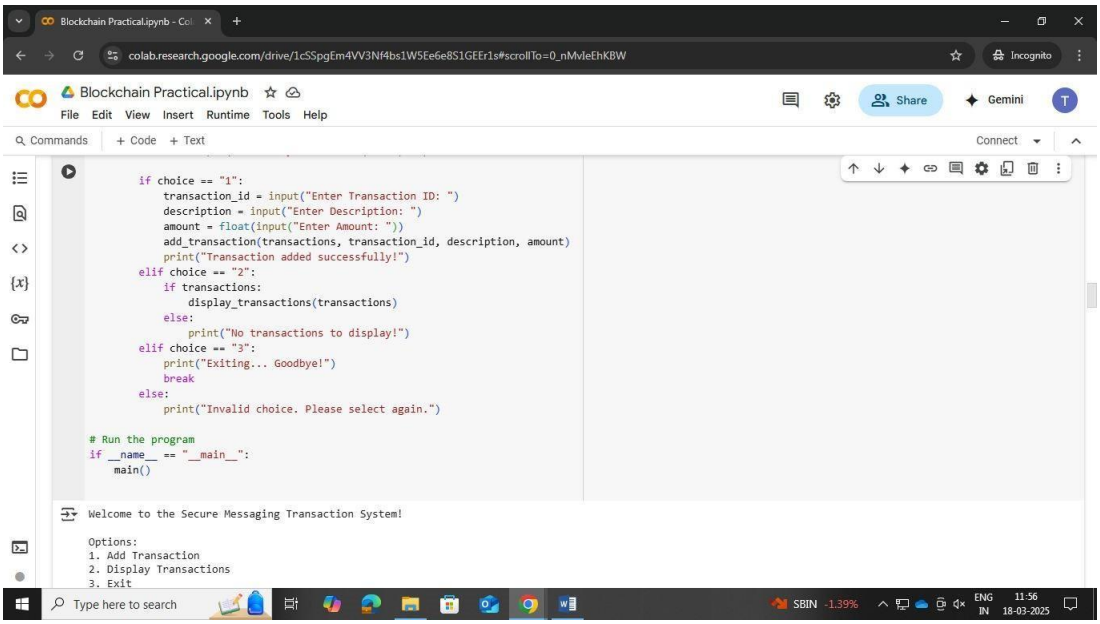
Aim: Allow users to create multiple transactions and display them in an organised format.

Code:

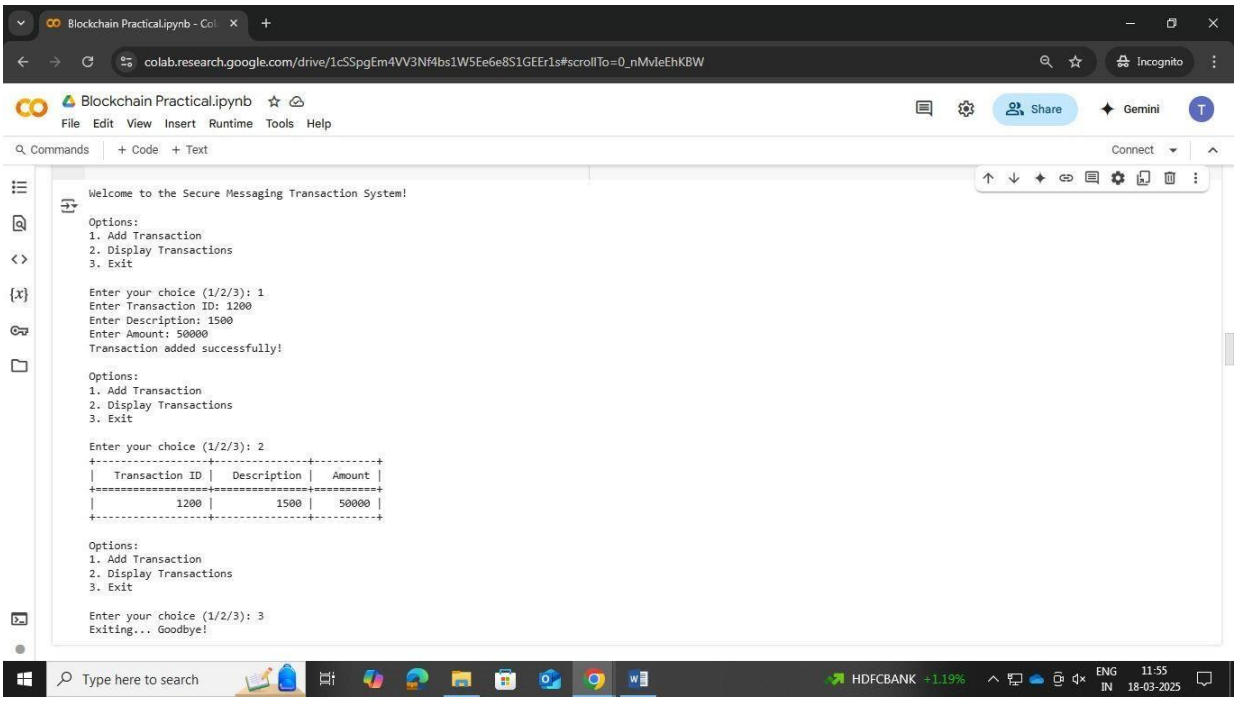
```
from tabulate import tabulate

# Function to add a transaction def add_transaction(transactions,
transaction_id, description, amount):     transactions.append({"ID":
transaction_id, "Description": description, "Amount": amount})

# Display transactions in a tabular format def
display_transactions(transactions):      headers =
["Transaction ID", "Description", "Amount"]      table =
[[t["ID"], t["Description"], t["Amount"]] for t in
transactions]
    print(tabulate(table, headers, tablefmt="grid"))

# Main program def main():      transactions = []  #
List to store transactions
    print("Welcome to the Secure Messaging Transaction System!")
     while True:
print("\nOptions: ")         print("1.
Add Transaction")         print("2.
Display Transactions")         print("3.
Exit")

        choice = input("\nEnter your choice (1/2/3): ")
         if choice ==
"1":
            transaction_id = input("Enter Transaction ID: ")
description = input("Enter Description: ")              amount
= float(input("Enter Amount: "))
            add_transaction(transactions,    transaction_id,    description,
amount)
            print("Transaction added successfully!")
elif choice == "2":              if transactions:
                display_transactions(transactions)
else:
```

```
                print("No transactions to display!")
elif choice == "3":             print("Exiting...
Goodbye!")              break          else:
print("Invalid choice. Please select again.")
# Run the program if
__name__ == "__main__":
main()
```





Output:


Blockchain Practical.ipynb - Col
colab.research.google.com/drive/1cSSpgEm4VV3Nf4bs1W5Ee6e8S1GEEr1s#scrollTo=0_nMvIeEhKBW
Incognito
Blockchain Practical.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
Welcome to the Secure Messaging Transaction System!
Options:
1. Add Transaction
2. Display Transactions
3. Exit
Enter your choice (1/2/3): 1
Enter Transaction ID: 1200
Enter Description: 1500
Enter Amount: 50000
Transaction added successfully!
Options:
1. Add Transaction
2. Display Transactions
3. Exit
Enter your choice (1/2/3): 2
| Transaction ID | Description | Amount |
| 1200 | 1500 | 50000 |
Options:
1. Add Transaction
2. Display Transactions
3. Exit
Enter your choice (1/2/3): 3
Exiting... Goodbye!
Type here to search
HDFCBANK +1.19%
ENG
IN
11:55
18-03-2025

PRACTICAL 1 C.

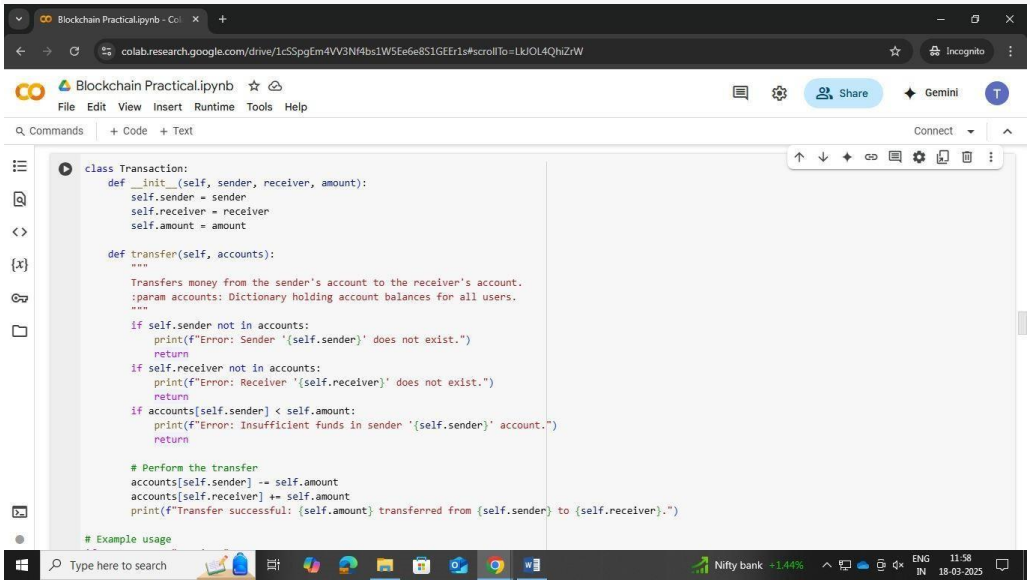
Aim: Create a Python class named Transaction with attributes for sender, receiver, and amount. Implement a method within the class to transfer money from the sender's account to the receiver's account.

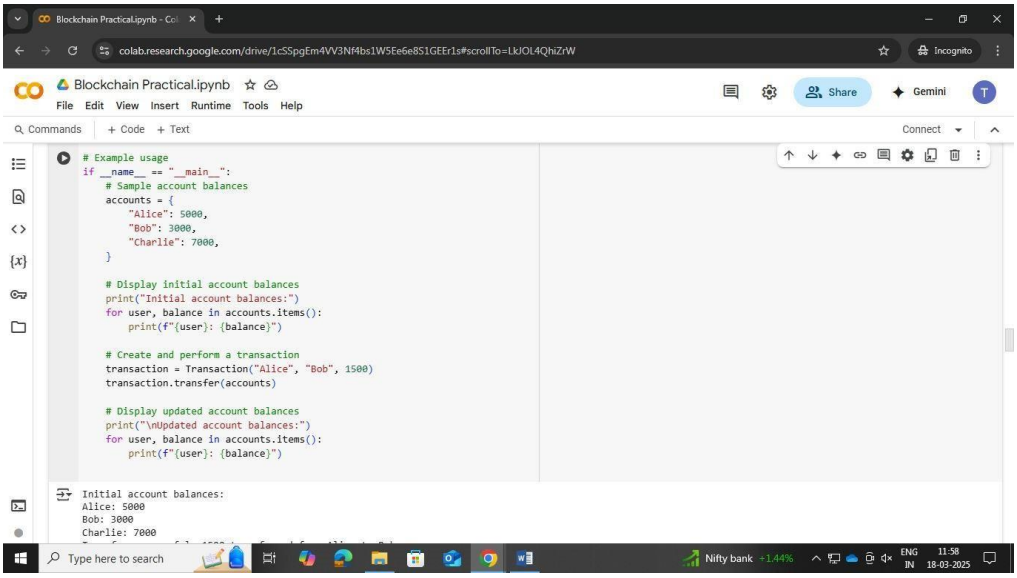
Code:

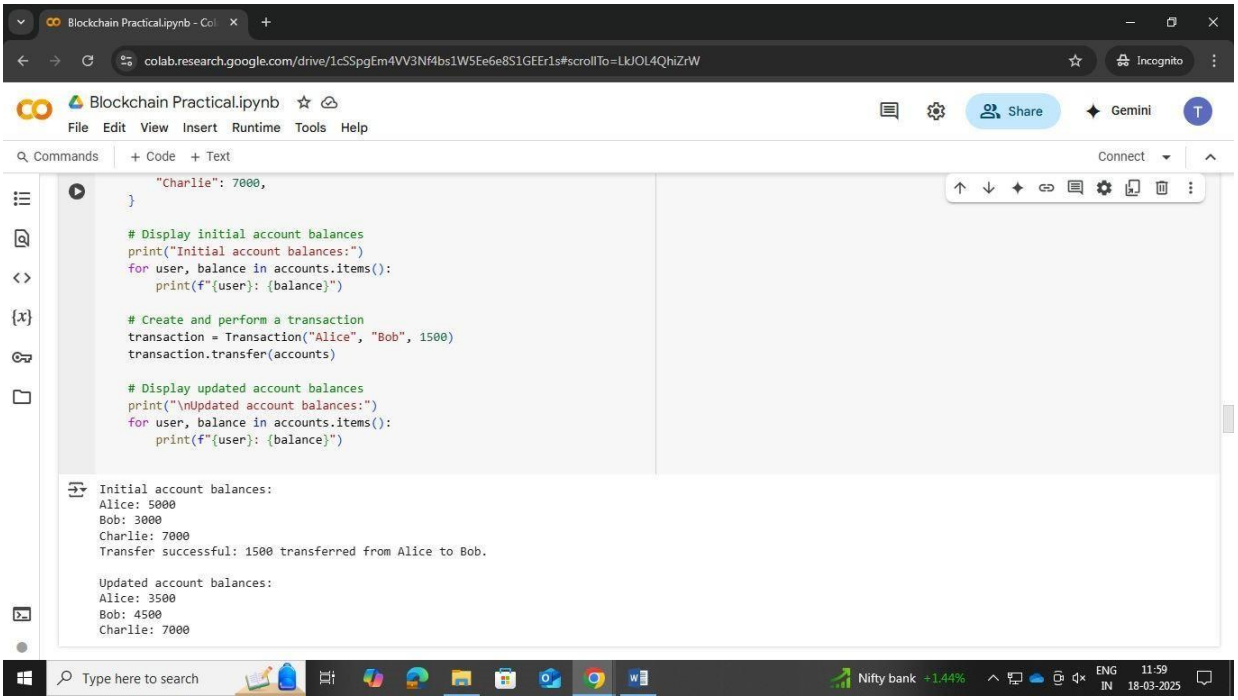
```
class Transaction:     def __init__(self, sender,
receiver, amount):         self.sender = sender
self.receiver = receiver         self.amount =
amount     def transfer(self, accounts):
        """
        Transfers money from the sender's account to the receiver's
account.
        :param accounts: Dictionary holding account balances for all
users.         """         if self.sender not in accounts:
print(f"Error: Sender '{self.sender}' does not exist.")
return         if self.receiver not in accounts:
print(f"Error: Receiver '{self.receiver}' does not exist.")
return         if accounts[self.sender] < self.amount:
print(f"Error: Insufficient funds in sender '{self.sender}' account.")
            return
        # Perform the transfer
        accounts[self.sender] -= self.amount
accounts[self.receiver] += self.amount
        print(f"Transfer successful: {self.amount} transferred from
{self.sender} to {self.receiver}.")
# Example usage if __name__
== "__main__":      # Sample
account balances      accounts
= {         "Alice": 5000,
        "Bob": 3000,
        "Charlie": 7000,
    }
    # Display initial account balances
print("Initial account balances:")
```

```
    for user, balance in accounts.items():
print(f"{user}: {balance}")

    # Create and perform a transaction
    transaction = Transaction("Alice", "Bob", 1500)
transaction.transfer(accounts)

    # Display updated account balances
print("\nUpdated account balances:")
for user, balance in accounts.items():
print(f"{user}: {balance}")
```

Output:






Blockchain Practical.ipynb - Col
colab.research.google.com/drive/1cSSpgEm4VV3Nf4bs1W5Ee6e8S1GEEr1s#scrollTo=LkJOL4QhiZrW
Incognito
Blockchain Practical.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
"Charlie": 7000,
}
# Display initial account balances
print("Initial account balances:")
for user, balance in accounts.items():
print(f"{user}: {balance}")
# Create and perform a transaction
transaction = Transaction("Alice", "Bob", 1500)
transaction.transfer(accounts)
# Display updated account balances
print("\nUpdated account balances:")
for user, balance in accounts.items():
print(f"{user}: {balance}")
Initial account balances:
Alice: 5000
Bob: 3000
Charlie: 7000
Transfer successful: 1500 transferred from Alice to Bob.
Updated account balances:
Alice: 3500
Bob: 4500
Charlie: 7000
Type here to search
Nifty bank +1.44%
ENG
IN
11:59
18-03-2025

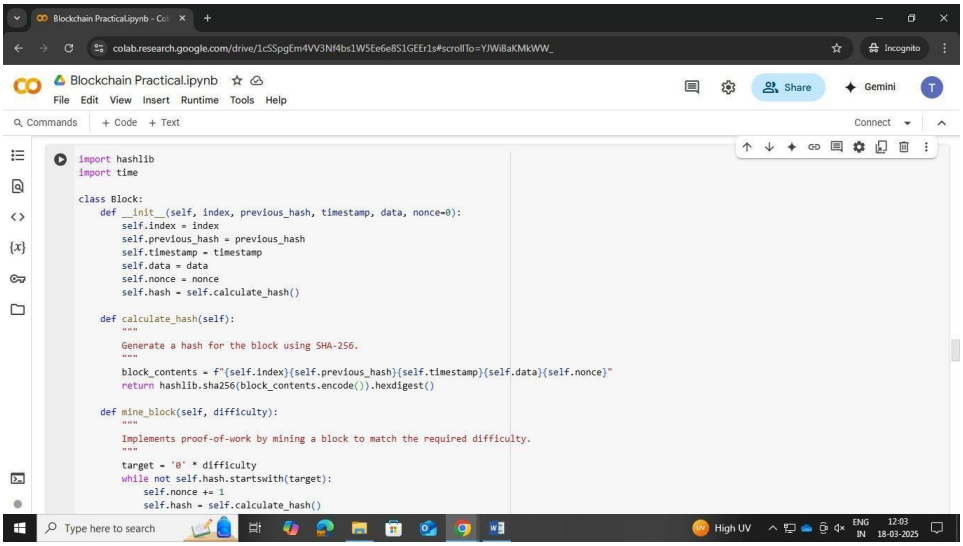
PRACTICAL 1 D.

Aim: Implement a function to add new blocks to the miner and dump the Blockchain.

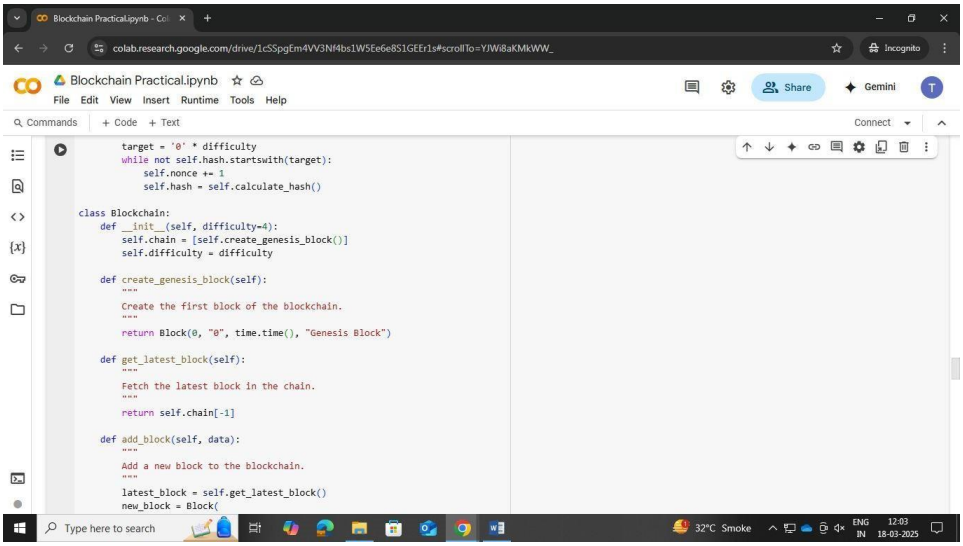
Code:

```
import hashlib import time class Block:     def __init__(self, index,
previous_hash, timestamp, data, nonce=0):         self.index = index
        self.previous_hash = previous_hash
self.timestamp = timestamp
self.data = data         self.nonce =
nonce
        self.hash = self.calculate_hash()
     def
calculate_hash(self):
        """
        Generate a hash for the block using SHA-256.
        """
        block_contents =
f"{self.index}{self.previous_hash}{self.timestamp}{self.data}{self.nonce}"
        return hashlib.sha256(block_contents.encode()).hexdigest()
     def mine_block(self,
difficulty):
        """
        Implements proof-of-work by mining a block to match the required
difficulty.          """
        target = '0' * difficulty         while
not self.hash.startswith(target):
self.nonce += 1
            self.hash = self.calculate_hash()
class Blockchain:     def __init__(self,
difficulty=4):          self.chain =
[self.create_genesis_block()]
self.difficulty = difficulty
     def
create_genesis_block(self):
        """
        Create the first block of the blockchain.
        """
        return Block(0, "0", time.time(), "Genesis Block")
```

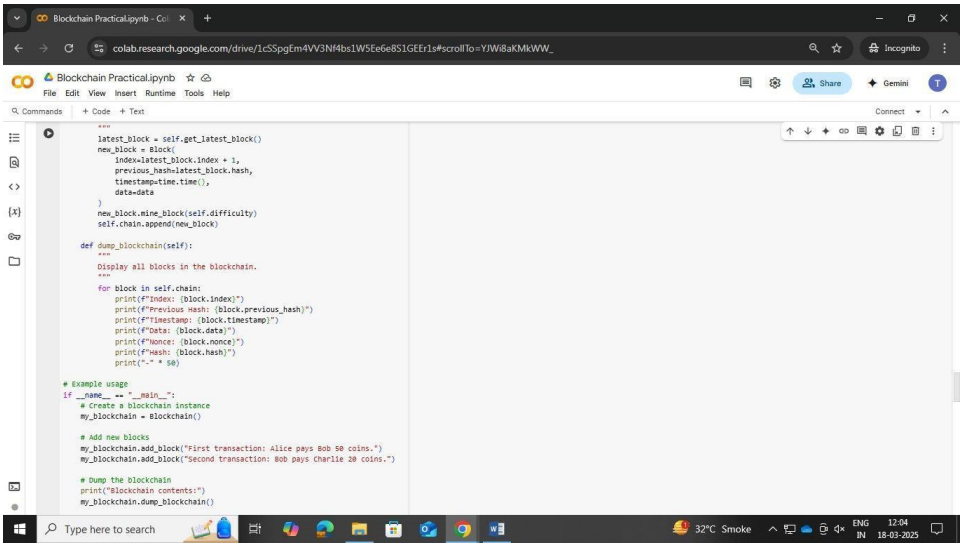
```
    def
get_latest_block(self):
        """
        Fetch the latest block in the chain.
        """
        return self.chain[-1]
    def add_block(self,
data):
        """
        Add a new block to the blockchain.
        """
        latest_block = self.get_latest_block()
new_block = Block(
            index=latest_block.index + 1,
previous_hash=latest_block.hash,
timestamp=time.time(),              data=data
        )
        new_block.mine_block(self.difficulty)
self.chain.append(new_block)
    def
dump_blockchain(self):
        """
        Display all blocks in the blockchain.
        """         for block in
self.chain:             print(f"Index:
{block.index}")
            print(f"Previous Hash: {block.previous_hash}")
print(f"Timestamp: {block.timestamp}")
print(f"Data: {block.data}")             print(f"Nonce:
{block.nonce}")             print(f"Hash: {block.hash}")
print("-" * 50)
# Example usage if
__name__ == "__main__":
    # Create a blockchain instance
my_blockchain = Blockchain()

    # Add new blocks
    my_blockchain.add_block("First transaction: Alice pays Bob 50 coins.")
my_blockchain.add_block("Second transaction: Bob pays Charlie 20 coins.")
    # Dump the blockchain
print("Blockchain contents:")
my_blockchain.dump_blockchain()
```



```
import hashlib
import time

class Block:
    def __init__(self, index, previous_hash, timestamp, data, nonce=0):
        self.index = index
        self.previous_hash = previous_hash
        self.timestamp = timestamp
        self.data = data
        self.nonce = nonce
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        """
        Generate a hash for the block using SHA-256.
        """
        block_contents = f"{self.index}{self.previous_hash}{self.timestamp}{self.data}{self.nonce}"
        return hashlib.sha256(block_contents.encode()).hexdigest()

    def mine_block(self, difficulty):
        """
        Implements proof-of-work by mining a block to match the required difficulty.
        """
        target = '0' * difficulty
        while not self.hash.startswith(target):
            self.nonce += 1
            self.hash = self.calculate_hash()
```

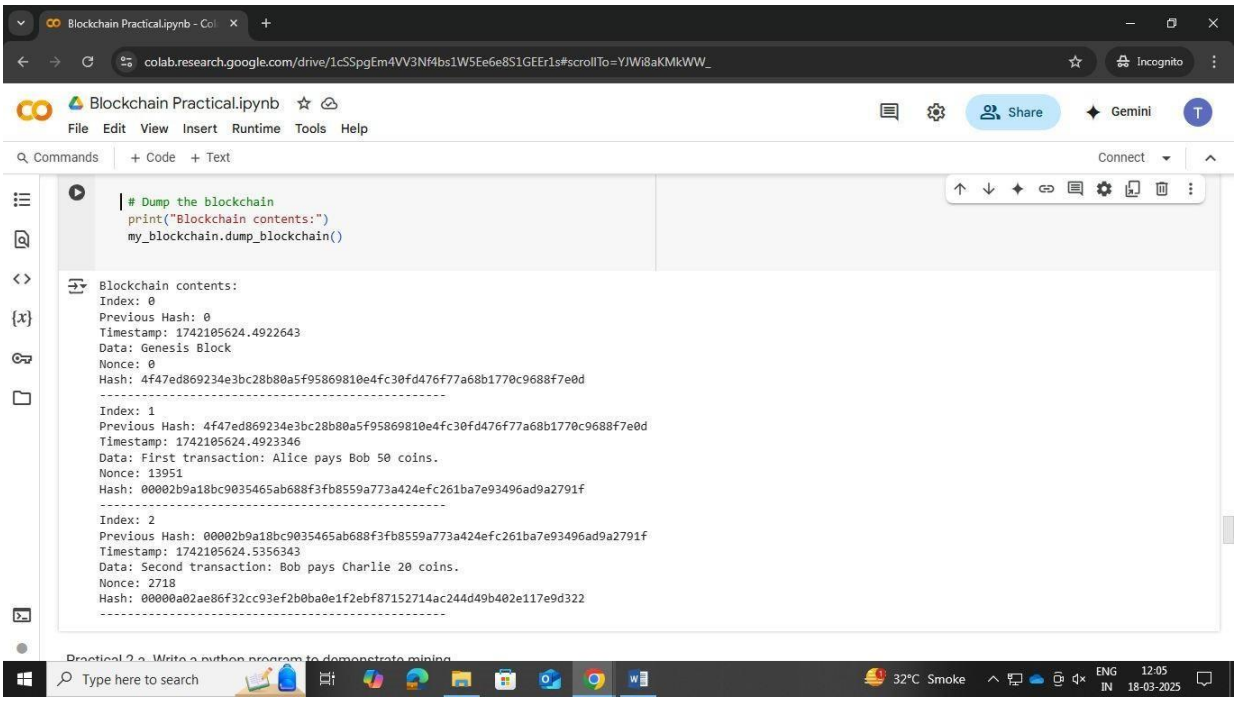


```
        target = '0' * difficulty
        while not self.hash.startswith(target):
            self.nonce += 1
            self.hash = self.calculate_hash()

class Blockchain:
    def __init__(self, difficulty=4):
        self.chain = [self.create_genesis_block()]
        self.difficulty = difficulty

    def create_genesis_block(self):
        """
        Create the first block of the blockchain.
        """
        return Block(0, "0", time.time(), "Genesis Block")

    def get_latest_block(self):
        """
        Fetch the latest block in the chain.
        """
        return self.chain[-1]

    def add_block(self, data):
        """
        Add a new block to the blockchain.
        """
        latest_block = self.get_latest_block()
        new_block = Block(
```



```
        latest_block = self.get_latest_block()
        new_block = Block(
            index=latest_block.index + 1,
            previous_hash=latest_block.hash,
            timestamp=time.time(),
            data=data
        )
        new_block.mine_block(self.difficulty)
        self.chain.append(new_block)

    def dump_blockchain(self):
        """
        Display all blocks in the blockchain.
        """
        for block in self.chain:
            print(f"Index: {block.index}")
            print(f"Previous Hash: {block.previous_hash}")
            print(f"Timestamp: {block.timestamp}")
            print(f"Data: {block.data}")
            print(f"Nonce: {block.nonce}")
            print(f"Hash: {block.hash}")
            print("-" * 50)

# Example usage
if __name__ == "__main__":
    # Create a blockchain instance
    my_blockchain = Blockchain()

    # Add new blocks
    my_blockchain.add_block("First transaction: Alice pays Bob 50 coins.")
    my_blockchain.add_block("Second transaction: Bob pays Charlie 20 coins.")

    # Dump the blockchain
    print("Blockchain contents:")
    my_blockchain.dump_blockchain()
```

Output:

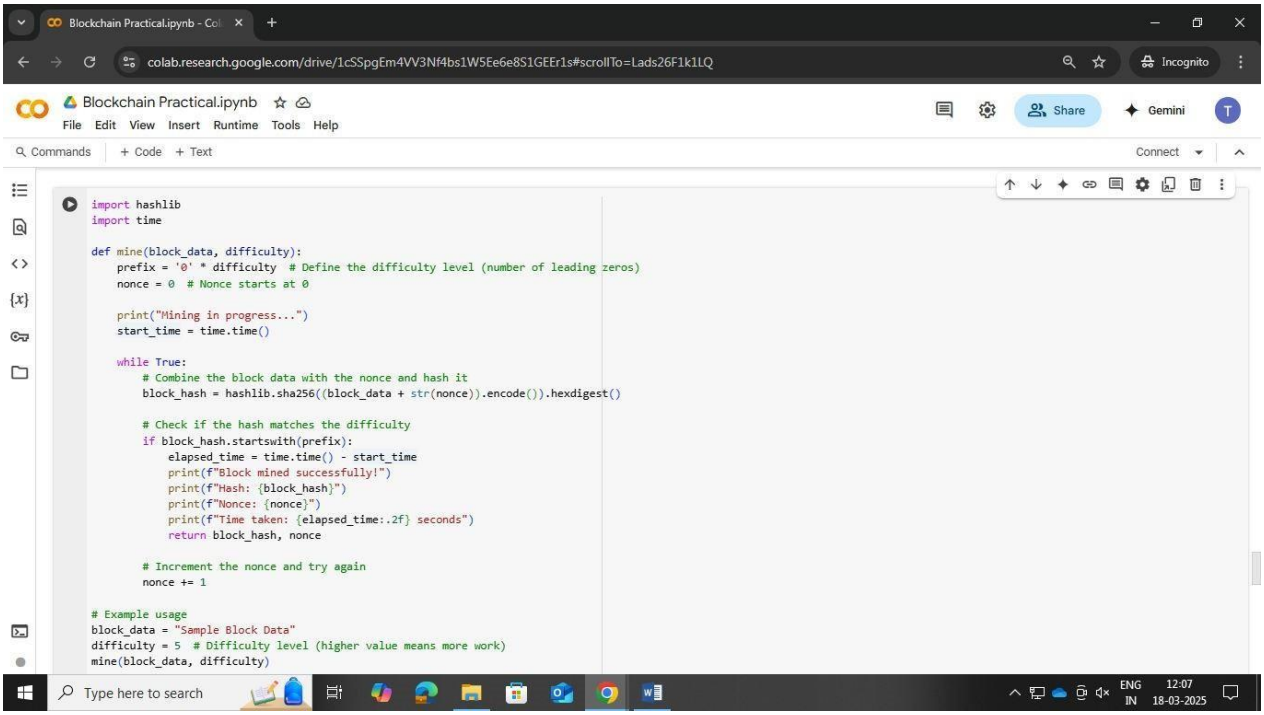

Blockchain Practical.ipynb - Col
colab.research.google.com/drive/1cSSpgEm4VV3Nf4bs1W5Ee6e8S1GEEr1s#scrollTo=YJWi8aKMkWW_
Incognito
Blockchain Practical.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
# Dump the blockchain
print("Blockchain contents:")
my_blockchain.dump_blockchain()
Blockchain contents:
Index: 0
Previous Hash: 0
Timestamp: 1742105624.4922643
Data: Genesis Block
Nonce: 0
Hash: 4f47ed869234e3bc28b80a5f95869810e4fc30fd476f77a68b1770c9688f7e0d
Index: 1
Previous Hash: 4f47ed869234e3bc28b80a5f95869810e4fc30fd476f77a68b1770c9688f7e0d
Timestamp: 1742105624.4923346
Data: First transaction: Alice pays Bob 50 coins.
Nonce: 13951
Hash: 00002b9a18bc9035465ab688f3fb8559a773a424efc261ba7e93496ad9a2791f
Index: 2
Previous Hash: 00002b9a18bc9035465ab688f3fb8559a773a424efc261ba7e93496ad9a2791f
Timestamp: 1742105624.5356343
Data: Second transaction: Bob pays Charlie 20 coins.
Nonce: 2718
Hash: 00000a02ae86f32cc93ef2b0ba0e1f2ebf87152714ac244d49b402e117e9d322
Type here to search
32°C Smoke
ENG
IN
12:05
18-03-2025

PRACTICAL 2 A.

Aim: Write a python program to demonstrate mining.

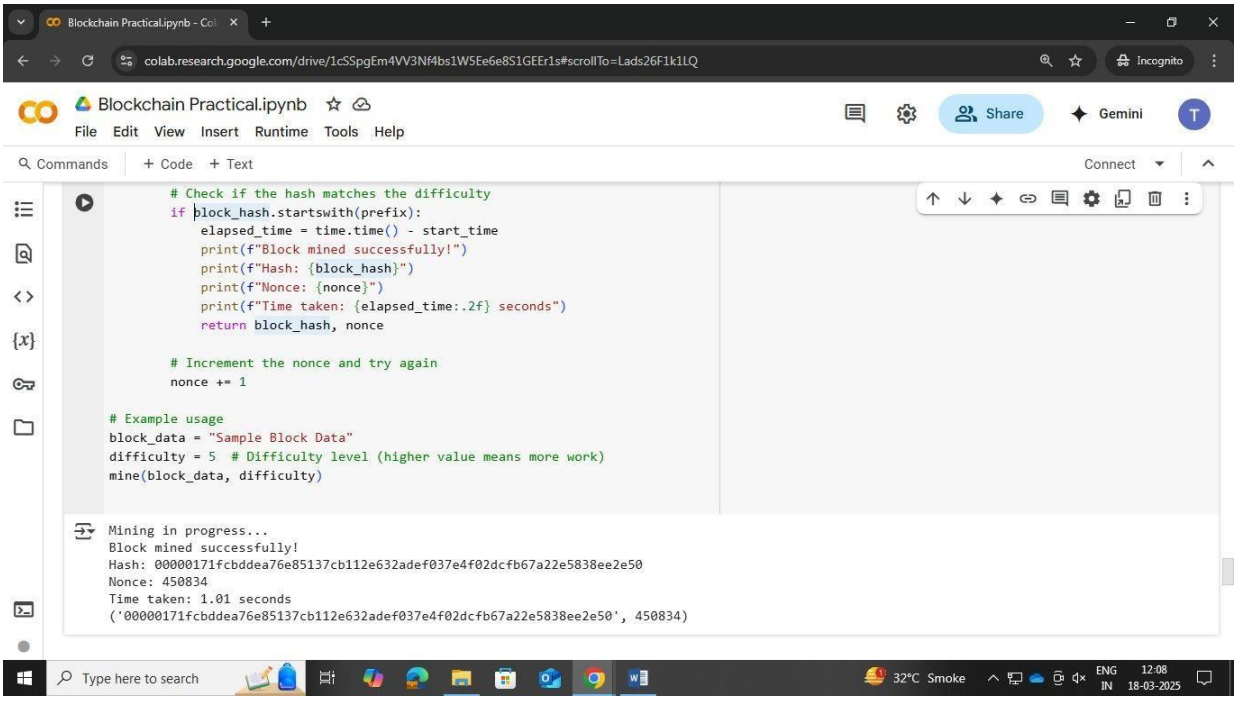
Code:

```
import hashlib import
time
 def mine(block_data, difficulty):      prefix = '0' * difficulty  #
Define the difficulty level (number of leading zeros)
    nonce = 0  # Nonce starts at 0
     print("Mining in
progress...")      start_time =
time.time()
     while
True:
        # Combine the block data with the nonce and hash it
block_hash = hashlib.sha256((block_data +
str(nonce)).encode()).hexdigest()

        # Check if the hash matches the difficulty
if block_hash.startswith(prefix):
elapsed_time = time.time() - start_time
print(f"Block mined successfully!")
print(f"Hash: {block_hash}")
print(f"Nonce: {nonce}")
            print(f"Time taken: {elapsed_time:.2f} seconds")
return block_hash, nonce

        # Increment the nonce and try again
nonce += 1

# Example usage
block_data = "Sample Block Data"
difficulty = 5  # Difficulty level (higher value means more work)
mine(block_data, difficulty)
```



```
import hashlib
import time

def mine(block_data, difficulty):
    prefix = '0' * difficulty  # Define the difficulty level (number of leading zeros)
    nonce = 0  # Nonce starts at 0

    print("Mining in progress...")
    start_time = time.time()

    while True:
        # Combine the block data with the nonce and hash it
        block_hash = hashlib.sha256((block_data + str(nonce)).encode()).hexdigest()

        # Check if the hash matches the difficulty
        if block_hash.startswith(prefix):
            elapsed_time = time.time() - start_time
            print(f"Block mined successfully!")
            print(f"Hash: {block_hash}")
            print(f"Nonce: {nonce}")
            print(f"Time taken: {elapsed_time:.2f} seconds")
            return block_hash, nonce

        # Increment the nonce and try again
        nonce += 1

# Example usage
block_data = "Sample Block Data"
difficulty = 5  # Difficulty level (higher value means more work)
mine(block_data, difficulty)
```

Output:



```
        # Check if the hash matches the difficulty
        if block_hash.startswith(prefix):
            elapsed_time = time.time() - start_time
            print(f"Block mined successfully!")
            print(f"Hash: {block_hash}")
            print(f"Nonce: {nonce}")
            print(f"Time taken: {elapsed_time:.2f} seconds")
            return block_hash, nonce

        # Increment the nonce and try again
        nonce += 1

# Example usage
block_data = "Sample Block Data"
difficulty = 5  # Difficulty level (higher value means more work)
mine(block_data, difficulty)
```

```
Mining in progress...
Block mined successfully!
Hash: 00000171fcbddea76e85137cb112e632adef037e4f02dcfb67a22e5838ee2e50
Nonce: 450834
Time taken: 1.01 seconds
('00000171fcbddea76e85137cb112e632adef037e4f02dcfb67a22e5838ee2e50', 450834)
```

PRACTICAL 2 B.

Aim: Demonstrate the use of the Bitcoin Core API to interact with a Bitcoin Core node.

Code: server=1

```
rpcuser=your_rpc_user

rpcpassword=your_rpc_password

rpcport=8332

rpcallowip=127.0.0.1 bitcoind –

daemon

pip install requests import requests

import json from requests.auth import

HTTPBasicAuth

# Bitcoin Core RPC settings rpc_url = "http://127.0.0.1:8332" rpc_user =

"your_rpc_user"  # Replace with your RPC username rpc_password =

"your_rpc_password"  # Replace with your RPC password

# Define a function to send RPC requests def

bitcoin_rpc(method, params=None):

  headers = {'content-type': 'application/json'}

# Prepare the RPC request payload

  payload = {

    "jsonrpc": "1.0",

    "id": "curltest",

    "method": method,

    "params": params or []

  }


  # Send the POST request
```

```
    response = requests.post(rpc_url, data=json.dumps(payload), headers=headers,
auth=HTTPBasicAuth(rpc_user, rpc_password))

    if response.status_code == 200:
        return response.json()
    else:
        raise Exception(f"Error: {response.status_code} - {response.text}")

# Example 1: Get the current block count block_count
= bitcoin_rpc("getblockcount") print("Block Count:",
block_count['result'])

# Example 2: Get blockchain information blockchain_info
= bitcoin_rpc("getblockchaininfo") print("Blockchain
Info:", blockchain_info['result'])

# Example 3: Get the wallet balance wallet_balance
= bitcoin_rpc("getbalance") print("Wallet
Balance:", wallet_balance['result'])

# Example 4: Get the current network difficulty
difficulty = bitcoin_rpc("getdifficulty") print("Network
Difficulty:", difficulty['result'])

# Example 5: Send a raw transaction (assuming you have a raw transaction to send)
# raw_tx = "<raw_transaction_data>"
# tx_id = bitcoin_rpc("sendrawtransaction", [raw_tx])
# print("Transaction ID:", tx_id['result'])
```

Output:

Block Count: 768914

Blockchain Info: {'chain': 'main', 'blocks': 768914, 'headers': 768914, 'bestblockhash': '00000000000000000b1a29528e897e63fba5952f17f3a8f6b0c60ad826c71b', 'difficulty': 218938744019.67, 'mediantime': 1628537400, 'verificationprogress': 0.999999, 'initialblockdownload': False, 'chainwork': '0000000000000000000000000000000000000000000000000000017ea7', 'size_on_disk': 35101620386, 'pruned': False, 'softforks': [], 'bip9_softforks': {}}

Wallet Balance: 0.02346789

Network Difficulty: 218938744019.67

Transaction ID: 1d5f8b927a9d7a64d83a57ac64d0b173d5e7a4932d8365f07c50e85826f1b27b

Error Handling:

If something goes wrong, you'll receive error messages from the Bitcoin Core node.

Invalid RPC call (wrong method name):

```
plaintext
CopyEdit
Error:  -32601 - Method not found
```

Incorrect RPC credentials (username/password mismatch):

```
plaintext
CopyEdit
Error: 401 - Unauthorized
```

If the node isn't fully synced and you attempt a `getblock` request:

```
plaintext
CopyEdit
Error:  -8 - Block not found
```

PRACTICAL 2 C.

Aim: Demonstrating the process of running a blockchain node on your local machine **Code:**

Running a blockchain node on your local machine allows you to interact directly with the blockchain network. Here we will go through the process of setting up a **Bitcoin Core node**, which is one of the most widely used full-node implementations of the Bitcoin protocol. This setup will help you run a **Bitcoin full node** locally and interact with the blockchain.

Steps to Run a Bitcoin Node Locally

1. Install Bitcoin Core

Bitcoin Core is the reference implementation of the Bitcoin protocol. To run a full Bitcoin node, you need to install Bitcoin Core on your machine.

Download Bitcoin Core:

- Visit the official website to download Bitcoin Core for your operating system: Download Bitcoin Core

Installation:

- Follow the installation instructions for your operating system (Windows, Linux, macOS).

2. Configure Bitcoin Core

Once you have Bitcoin Core installed, you need to configure it to run as a full node and allow RPC (Remote Procedure Call) for interaction.

Create or Edit the bitcoin.conf file:

- The configuration file is usually located in the Bitcoin data directory:
  - **Linux**: ~/.bitcoin/bitcoin.conf
  - **macOS**: ~/Library/Application Support/Bitcoin/bitcoin.conf ○ **Windows**: C:\Users\YourUsername\AppData\Roaming\Bitcoin\bitcoin.conf

If the file doesn't exist, create it.
Configure the File:

Open or create the bitcoin.conf file and add the following lines for basic setup:

```
# Bitcoin Core Configuration File
server=1 rpcuser=your_rpc_user
rpcpassword=your_rpc_password
rpcport=8332
rpcallowip=127.0.0.1 listen=1
maxconnections=125
datadir=/path/to/bitcoin/data
txindex=1
```

Explanation of each line:

- server=1: This allows the node to accept RPC commands.
- rpcuser: Username for authentication when sending RPC commands.
- rpcpassword: Password for the above username.
- rpcport: Port for RPC communication (default is 8332).
- rpcallowip=127.0.0.1: Allow RPC connections from the local machine only (for security).
- listen=1: Allows the node to accept incoming connections from other nodes.
- maxconnections=125: Maximum number of connections the node will allow.
- datadir: Directory where blockchain data will be stored (you can customize this).
- txindex=1: Enables transaction index, useful for querying transactions by ID.

3. Start Bitcoin Core

To start Bitcoin Core, you can run the following command depending on your operating system:

- **Linux/macOS**:
- bitcoind -daemon
- **Windows**:
  Double-click on bitcoind.exe (if installed via the Windows installer) or run the following in the command prompt:
- bitcoind.exe -daemon

This starts the Bitcoin daemon in the background.

4. Sync the Blockchain

When you first start Bitcoin Core, it will begin to download the entire Bitcoin blockchain. This process is known as **syncing**. This can take **several days** depending on your internet speed and hardware.
The blockchain data is stored in the Bitcoin data directory (datadir). The blockchain can grow large (currently around 500 GB), so ensure you have enough storage space available.

You can monitor the sync progress via the Bitcoin Core graphical interface or by using RPC commands.

5. Check Sync Progress

To check the synchronization status of your node, you can use the following RPC command:

- Use a **Python script** or **command line** to interact with Bitcoin Core RPC.

Example Python Script to Check Block Height:

```
import requests import
json
from requests.auth import HTTPBasicAuth

rpc_url = "http://127.0.0.1:8332"
rpc_user = "your_rpc_user"  # Replace with your RPC username rpc_password
= "your_rpc_password"  # Replace with your RPC password

def bitcoin_rpc(method, params=None):
   headers = {'content-type': 'application/json'}
payload = {        "jsonrpc": "1.0",
      "id": "curltest",
      "method": method,
      "params": params or []
   }
   response = requests.post(rpc_url, data=json.dumps(payload), headers=headers, auth=HTTPBasicAuth(rpc_user,
rpc_password))
   return response.json()

# Example: Get block count to check sync status
block_count = bitcoin_rpc("getblockcount") print("Block
Count:", block_count['result'])
```

This Python script queries the getblockcount method, which returns the current block height. If the node isn't fully synced, the block count will be lower than the latest block on the network.

6. Interacting with the Bitcoin Network

Once your Bitcoin node is synced, you can start interacting with the Bitcoin blockchain. Here are some common actions you might want to perform:

Get Blockchain Information:

```
blockchain_info = bitcoin_rpc("getblockchaininfo")
print("Blockchain Info:", blockchain_info['result'])
```

This will return details like the current block height, the best block hash, and other useful information about the blockchain.

Get a New Address:

If you want to create a new Bitcoin address to receive funds, use:

```
new_address = bitcoin_rpc("getnewaddress")
print("New Address:", new_address['result'])
```

This generates a new Bitcoin address from your wallet.

Send Bitcoin:

To send Bitcoin from your wallet to another address, you need to use the `sendtoaddress` method. Example:

```
tx_id = bitcoin_rpc("sendtoaddress", ["1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa", 0.001]) print("Transaction ID:", tx_id['result'])
```

This sends 0.001 BTC to the address `1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa`.

7. Monitor the Node

You can monitor the status of your node using Bitcoin Core's graphical user interface (GUI) or via RPC commands. Bitcoin Core also provides detailed logs that can help troubleshoot any issues.

Check Node's Version:

```
version_info = bitcoin_rpc("getnetworkinfo")
print("Version Info:", version_info['result'])
```

This will return information like the current Bitcoin version and network info.

8. Closing Bitcoin Core

Once you're done using the Bitcoin Core node, you can stop it gracefully by calling:

```
bitcoin-cli stop
```

PRACTICAL 2 D.

Aim: Demonstrate mining using geth on your private network.

Code:

Steps to Mine on a Private Ethereum Network using Geth

1. Install Geth

First, you need to install Geth, which is the Go-based Ethereum client.

Install Geth:

- **Windows**:
  Download the latest Geth installer from the official Geth GitHub release page and follow the installation instructions.
- **macOS**:
  If you have `brew` installed, you can install Geth with:
- `brew tap ethereum/ethereum • brew install ethereum`
- **Linux**:
  On Ubuntu/Debian:
- `sudo add-apt-repository -y ppa:ethereum/ethereum`
- `sudo apt-get update`
- `sudo apt-get install geth`

Once installed, you can check if it's working by running:

```
geth version
```

2. Create a Genesis Block for Your Private Network

To start a private Ethereum network, you need to define the **genesis block** (the first block in the blockchain). This is done by creating a **`genesis.json`** file.

Create `genesis.json`:

Create a `genesis.json` file that will define the parameters of your private blockchain. Here's an example of what the `genesis.json` file might look like:

```
{
  "config": {
    "chainId": 1337,                    // A unique chain ID for the private
network
    "homesteadBlock": 0,
    "daoForkBlock": 0,
```

```
    "eip155Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "muirGlacierBlock": 0,
    "berlinBlock": 0,
    "londonBlock": 0
  },
  "alloc": {},
  "coinbase": "0x0000000000000000000000000000000000000000",
  "difficulty": "0x20000",
  "gasLimit": "0x8000000" }
```

Explanation:

- `chainId`: This is a unique identifier for your private network. Ethereum mainnet's chain ID is `1`, and test networks have their own chain IDs.
- `alloc`: You can pre-allocate some Ether to specific accounts. In this case, it's left empty.
- `difficulty`: Set the mining difficulty. We can set this to a low value (`0x20000`) so mining is quick on your private network.
- `gasLimit`: The gas limit for blocks (default is set to `0x8000000`).

Initialize the Genesis Block:

Run the following command to initialize the private network using the `genesis.json` file: `geth`

```
init genesis.json --datadir ./privateChain
```

This initializes the private blockchain and creates the necessary data directory (`./privateChain`).

3. Start Your Ethereum Node

Now that your private network has been initialized, you can start the Geth node.

Run the following command to start your Ethereum node on the private network:

```
geth --networkid 1337 --datadir ./privateChain --mine --miner.threads=1 -
http     --http.addr     "0.0.0.0"     --http.port     8545     --http.api
"personal,eth,web3,miner,net"  --allow-insecure-unlock
```

Explanation of the parameters:

- `--networkid 1337`: This ensures you're using your private network (use the chain ID you specified in the genesis file).
- `--datadir ./privateChain`: Points to the data directory for your blockchain.

- `--mine`: This enables mining.

- `--miner.threads=1`: Number of CPU threads to mine on (you can increase this if you have a more powerful machine).
- `--http`: Enables the HTTP RPC server (useful for interacting with your node programmatically).
- `--http.addr "0.0.0.0"`: Allows RPC connections from any machine (or use `"127.0.0.1"` for local-only access).
- `--http.port 8545`: Specifies the HTTP RPC port.
- `--http.api "personal,eth,web3,miner,net"`: This allows the specified APIs to be accessible over HTTP (necessary for interacting with the miner and the blockchain).
- `--allow-insecure-unlock`: This allows you to unlock accounts for mining, but use it carefully because it can expose your accounts.

4. Unlock Your Mining Account

In order to mine on your private network, you'll need an unlocked account. You can either create an account or unlock an existing one.

Create a New Account:

Run this command to create a new Ethereum account: `geth account new --datadir ./privateChain` This will generate a new account and give you an address.

Unlock the Account:

To unlock the account for mining (it's required to mine on your node), run: `geth attach ipc:./privateChain/geth.ipc`

This will open the Geth JavaScript console. Then, unlock the account using the following command: `personal.unlockAccount(eth.accounts[0], "your_password", 0)`

Where `eth.accounts[0]` is the first account in your list and `"your_password"` is the password you set during account creation. The `0` specifies the unlock duration (0 means permanently unlocked until you shut down the node).

5. Start Mining

Now that your account is unlocked and the Geth node is running, mining should begin automatically.

You can check the mining process by viewing the logs in the console, and you should see new blocks being mined at regular intervals (because we set a low difficulty in the `genesis.json` file).

If you're running the node with the `--mine` flag, mining should be ongoing. You should see new blocks being mined and their corresponding block hashes in the logs. The mining reward (in ETH) will be credited to your account.

You can also check the mining status with this RPC call: `eth.mining`

This should return `true` if mining is in progress.

6. Monitor Mining and Network Status

You can use the following commands within the **Geth JavaScript console** to get more information about the current mining status:

- **Check current block number:** • `eth.blockNumber`
- **Check the coinbase (miner's address):** • `eth.coinbase`
- **Check balance of the account:**
- `web3.fromWei(eth.getBalance(eth.coinbase), "ether")`

7. Interact with the Ethereum Network

If you want to interact with the private Ethereum network from a web interface, you can use **Web3.js** (JavaScript library) or **Web3.py** (Python library) to make RPC calls to the network.

Example Web3.js Code:

Here's an example of how to interact with the node using Web3.js:

```
const Web3 = require('web3');

// Connect to the local Ethereum node const
web3 = new Web3('http://localhost:8545');
web3.eth.getBlockNumber()
  .then(console.log);  // Prints the latest block number
```

8. Stopping the Node

To stop your Ethereum node, press `Ctrl+C` in the terminal where Geth is running.

```
admin.stopRPC()
```

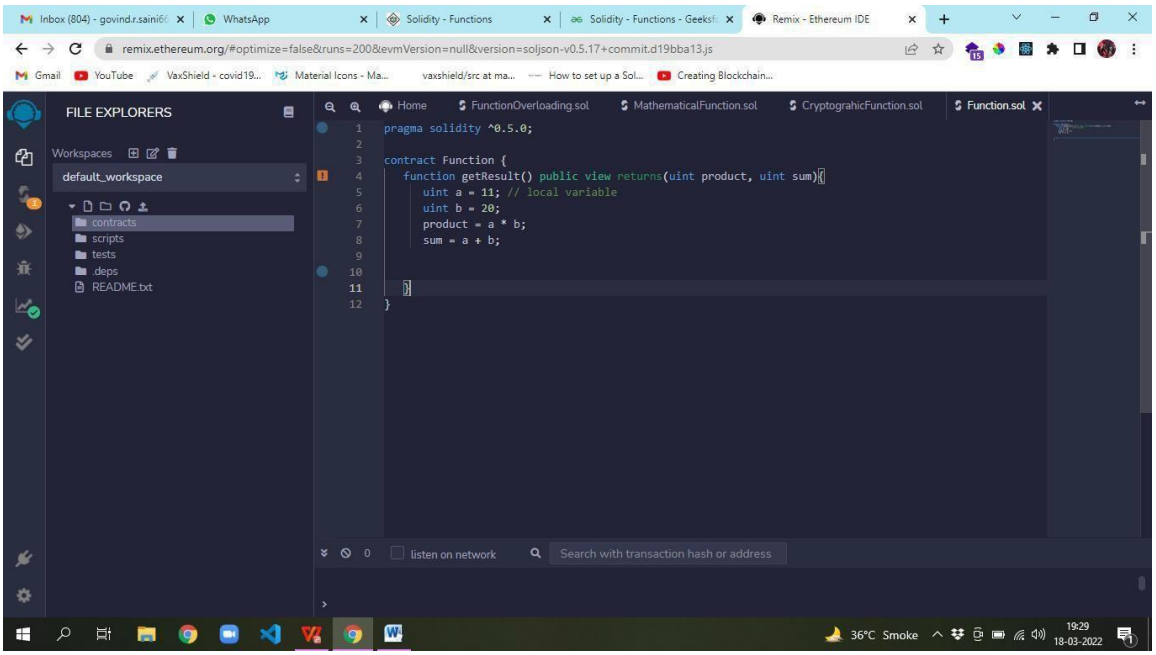
PRACTICAL 3 A.

Aim: Write a Solidity program that demonstrates various types of functions including regular functions, view functions, pure functions, and the fallback function.

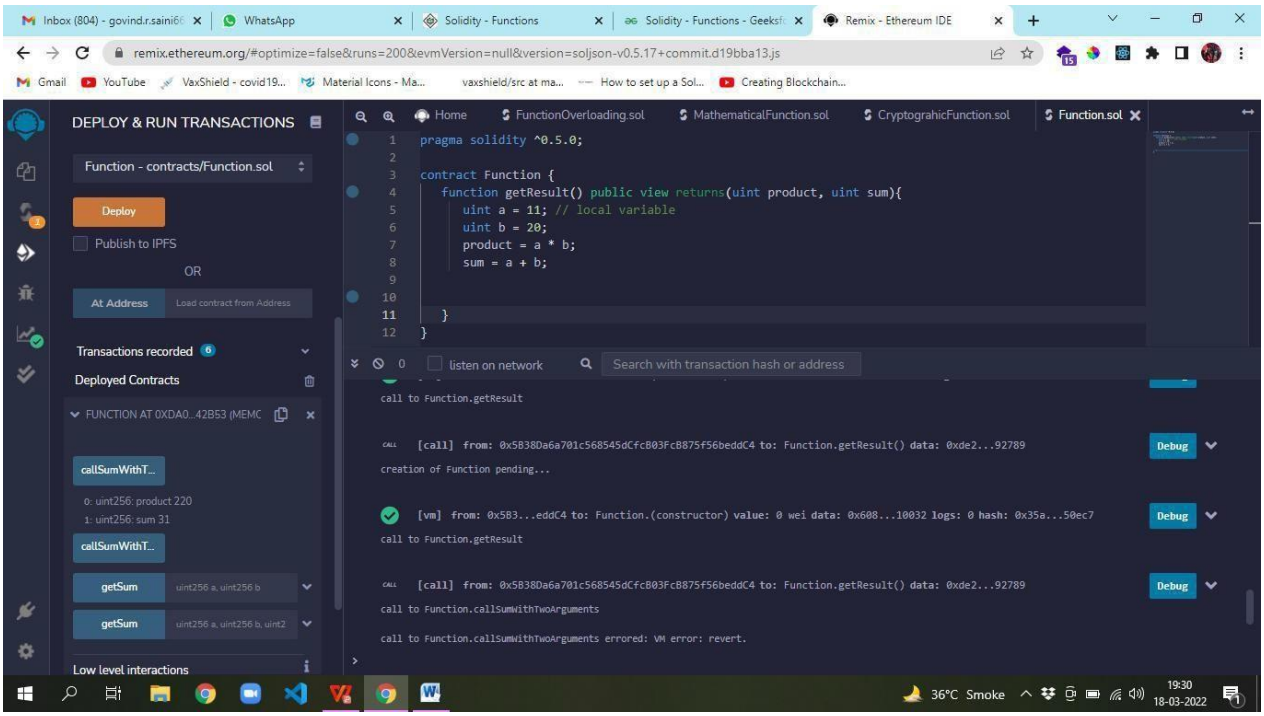
Code:

Regular Function

```
pragma solidity ^0.5.0;

contract Function {

function getResult() public view returns (uint product, uint

sum){ uint a = 11; uint b = 20; product = a * b; sum = a + b;

}

}
```



Output:

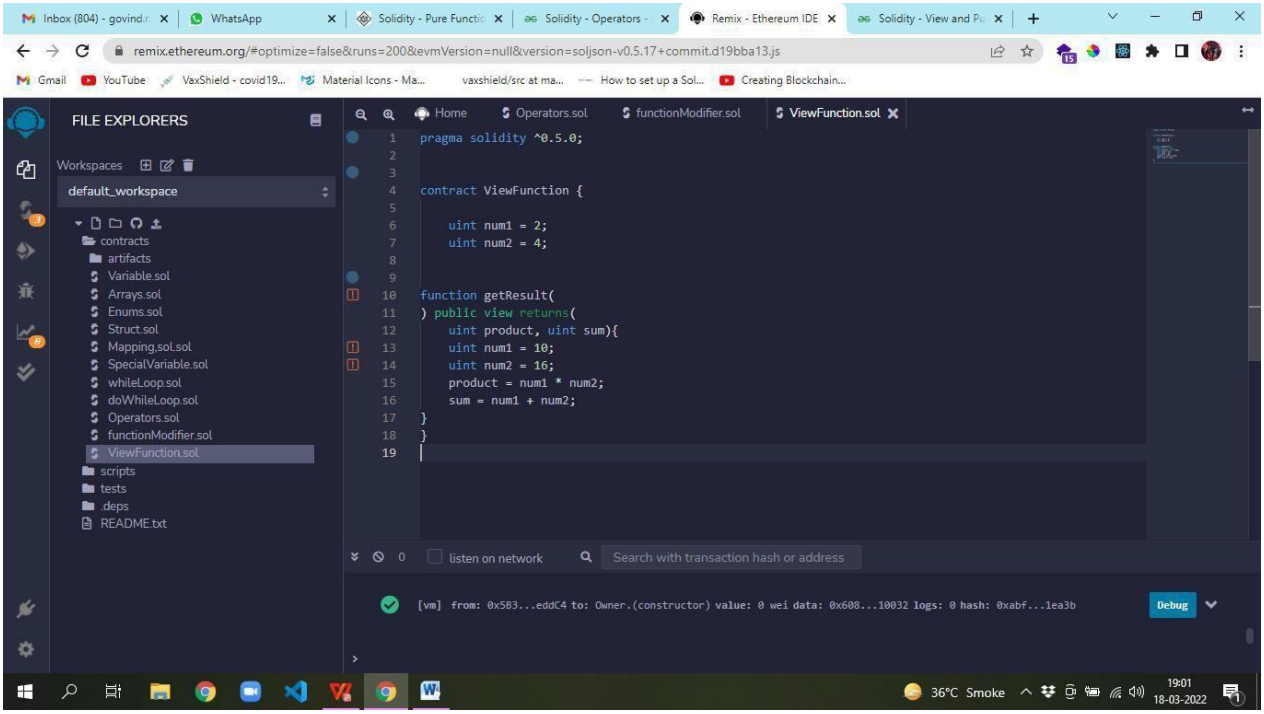


View Function pragma solidity ^0.5.0; contract ViewFunction { uint num1 = 2; uint num2 = 4;

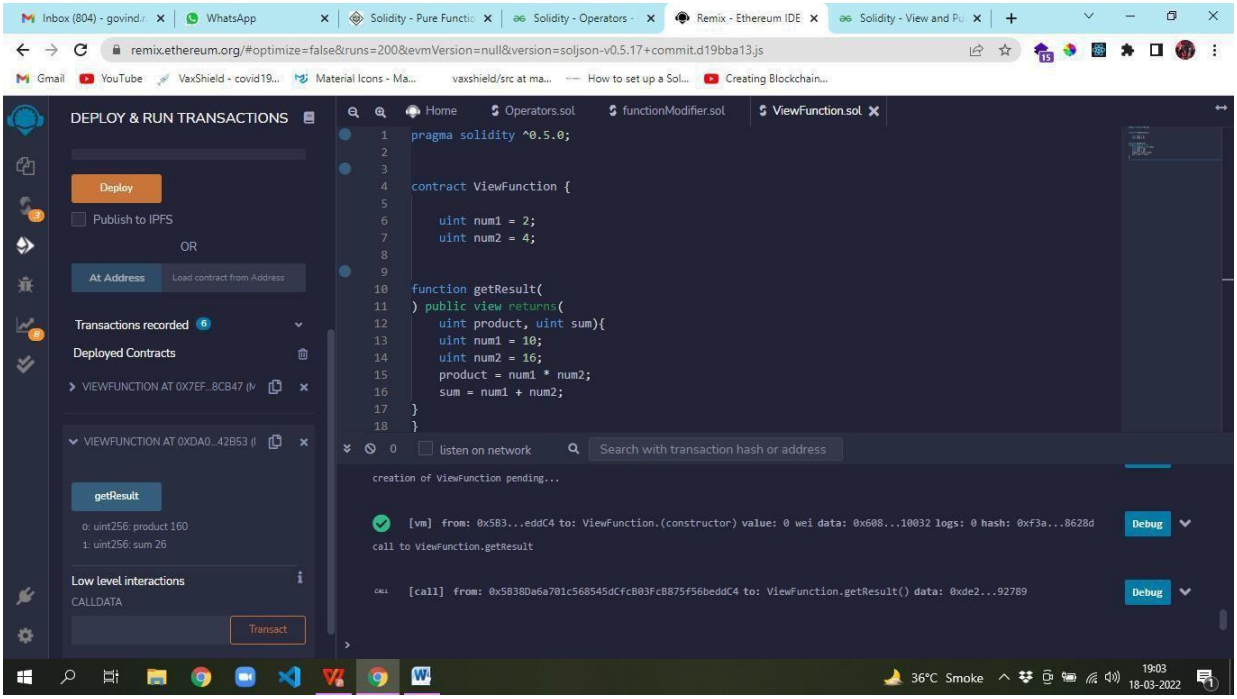
function getResult() public view returns(uint product, uint sum) { uint num1 = 10; uint num2 = 16; product = num1 * num2; sum = num1 + num2;

}

}


FILE EXPLORERS
Workspaces
default_workspace
contracts
artifacts
Variable.sol
Arrays.sol
Enums.sol
Struct.sol
Mapping.sol.sol
SpecialVariable.sol
whileLoop.sol
doWhileLoop.sol
Operators.sol
functionModifier.sol
ViewFunction.sol
scripts
tests
.deps
README.txt
Home
Operators.sol
functionModifier.sol
ViewFunction.sol
pragma solidity ^0.5.0;
contract ViewFunction {
uint num1 = 2;
uint num2 = 4;
function getResult(
) public view returns(
uint product, uint sum){
uint num1 = 10;
uint num2 = 16;
product = num1 * num2;
sum = num1 + num2;
}
}
listen on network
Search with transaction hash or address
[vm] from: 0x5B3...eddC4 to: Owner.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xabf...1ea3b
Debug
36°C Smoke
19:01
18-03-2022

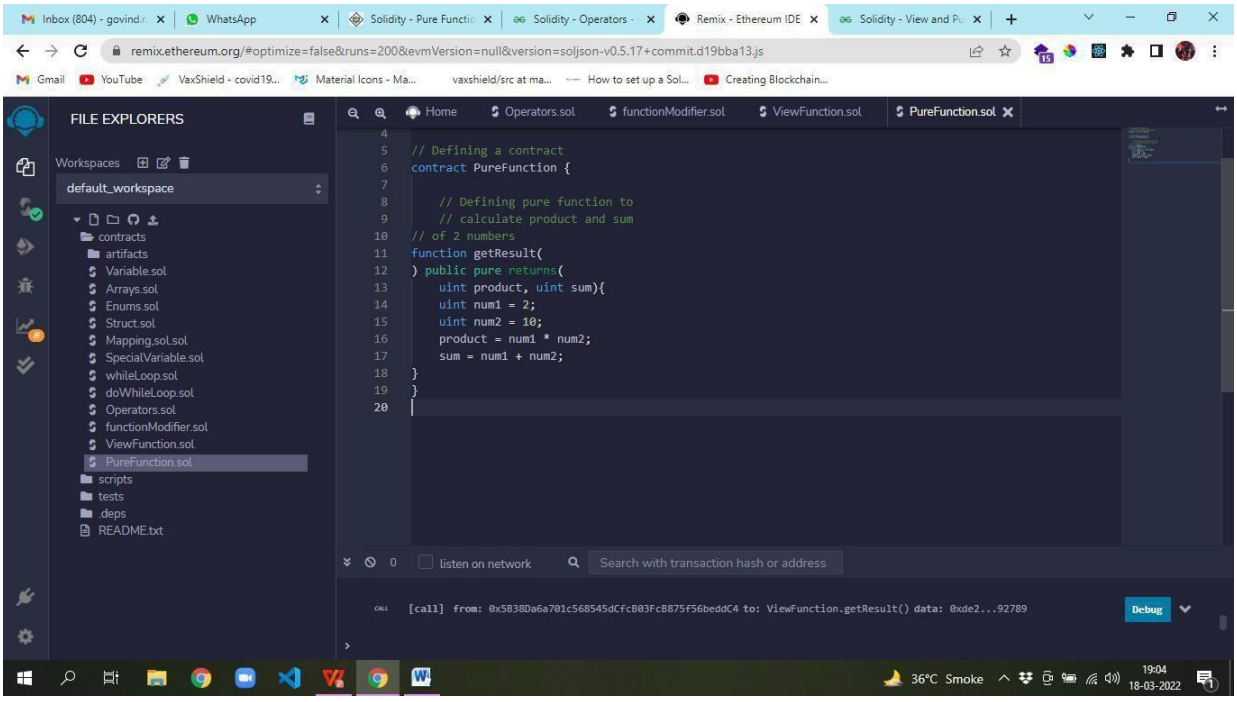
Output:


DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 6
Deployed Contracts
VIEWFUNCTION AT 0X7EF...8CB47
VIEWFUNCTION AT 0XDA0...42B53
getResult
0: uint256: product 160
1: uint256: sum 26
Low level interactions
CALLDATA
Transact
Home
Operators.sol
functionModifier.sol
ViewFunction.sol
pragma solidity ^0.5.0;
contract ViewFunction {
uint num1 = 2;
uint num2 = 4;
function getResult(
) public view returns(
uint product, uint sum){
uint num1 = 10;
uint num2 = 16;
product = num1 * num2;
sum = num1 + num2;
}
}
listen on network
Search with transaction hash or address
creation of ViewFunction pending...
[vm] from: 0x5B3...eddC4 to: ViewFunction.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xf3a...8628d
Debug
call to ViewFunction.getResult
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ViewFunction.getResult() data: 0xde2...92789
Debug
36°C Smoke
19:03
18-03-2022

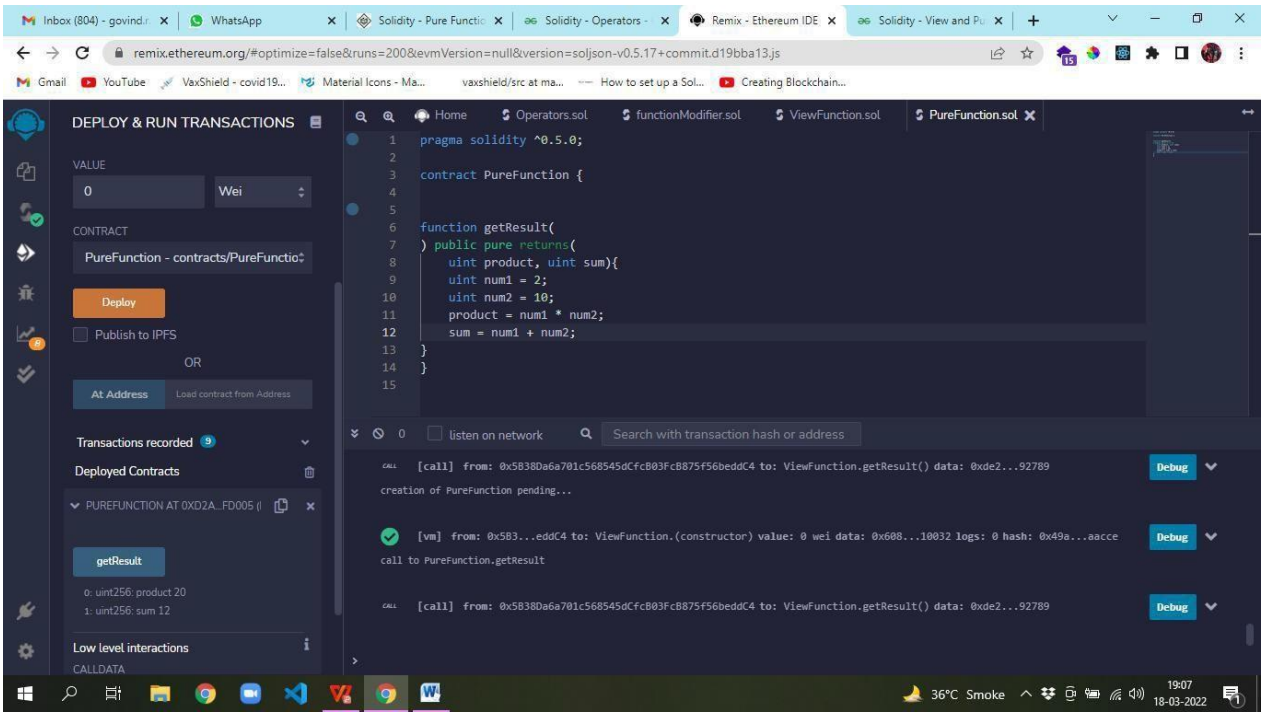
Pure Function pragma solidity ^0.5.0; contract PureFunction { function getResult() public pure returns (uint product, uint sum) { uint num1 = 2; uint num2 = 10; product = num1 * num2; sum = num1 + num2;

}

}



Output:



Fallback Function pragma solidity ^0.5.0; contract FallbackFunction { uint public x ;

function() external { x = 1; }

}

contract Sink {

function() external payable { }

}

contract Caller {

function callTest (Test test) public returns (bool) {

(bool success,) = address(test).call(abi.encodeWithSignature("nonExistingFunction()"));
require(success);

address payable testPayable = address(uint160(address(test))); return

(testPayable.send(2 ether));

}

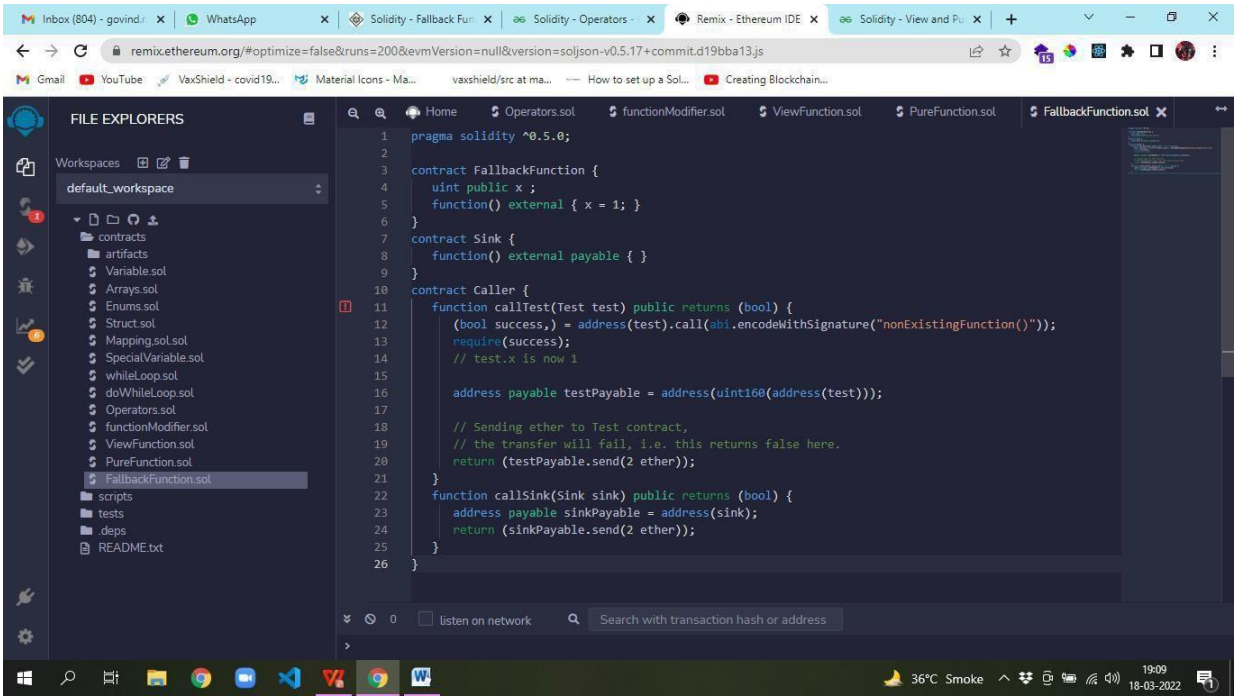
function callSink (Sink sink) public returns (bool)

{ address payable sinkPayable = address(sink);

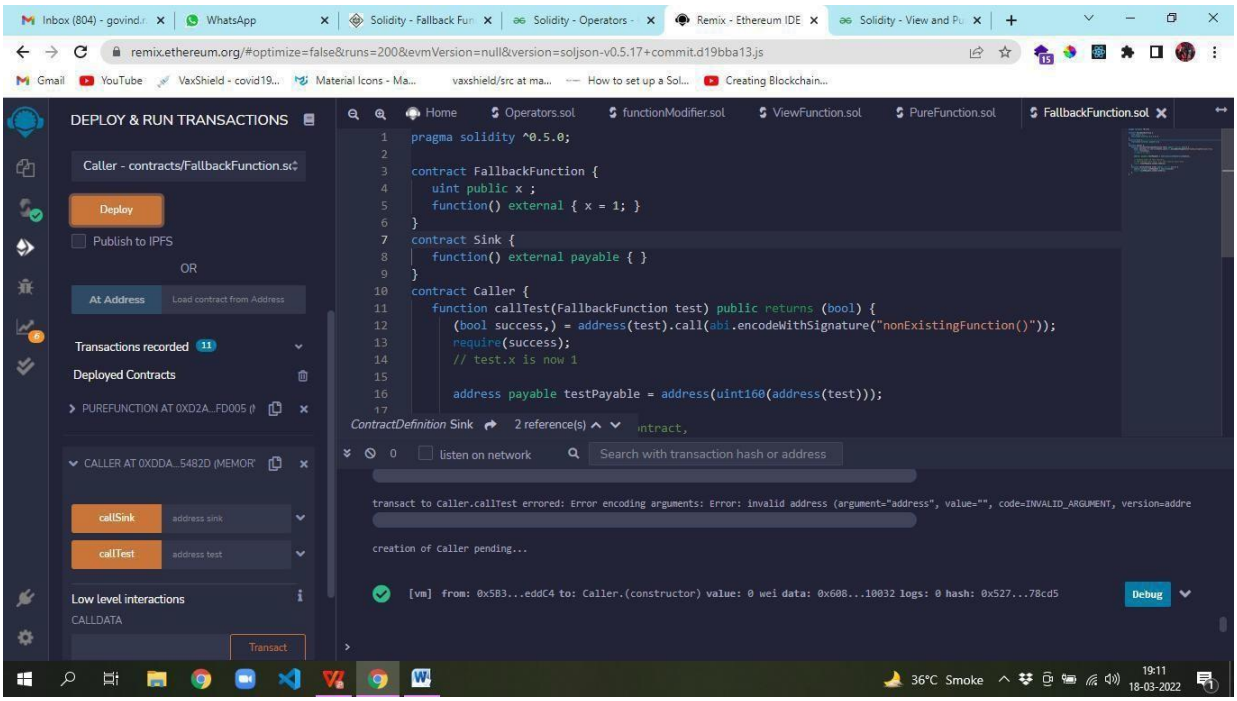
return (sinkPayable.send(2 ether));

}

}



Output:


DEPLOY & RUN TRANSACTIONS
Caller - contracts/FallbackFunction.sc
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 11
Deployed Contracts
PUREFUNCTION AT 0XD2A...FD005
CALLER AT 0XDDA...5482D (MEMOR
callSink
address sink
callTest
address test
Low level interactions
CALLDATA
Transact
Home
Operators.sol
functionModifier.sol
ViewFunction.sol
PureFunction.sol
FallbackFunction.sol
pragma solidity ^0.5.0;
contract FallbackFunction {
uint public x ;
function() external { x = 1; }
}
contract Sink {
function() external payable { }
}
contract Caller {
function callTest(FallbackFunction test) public returns (bool) {
(bool success,) = address(test).call(abi.encodeWithSignature("nonExistingFunction()"));
require(success);
// test.x is now 1
address payable testPayable = address(uint160(address(test)));
ContractDefinition Sink
2 reference(s)
listen on network
Search with transaction hash or address
transact to Caller.callTest errored: Error encoding arguments: Error: invalid address (argument="address", value="", code=INVALID_ARGUMENT, version=addre
creation of Caller pending...
[vm] from: 0x5B3...eddC4 to: Caller.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0x527...78cd5
Debug
36°C Smoke
19:11
18-03-2022

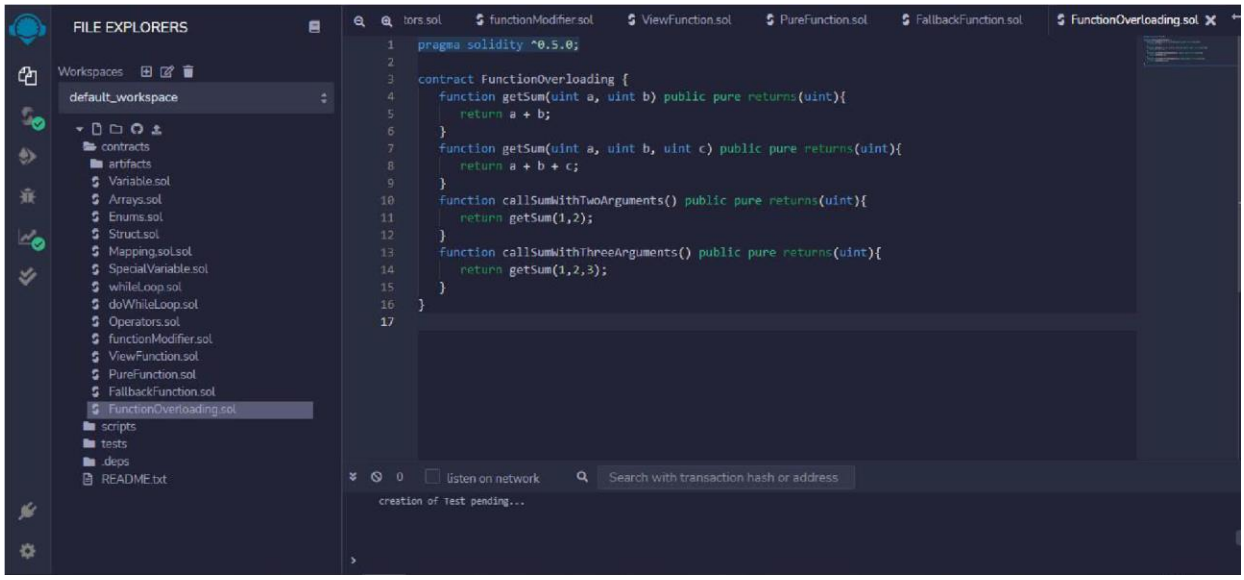
PRACTICAL 3 B.

Aim: Write a Solidity program that demonstrates function overloading, mathematical functions, and cryptographic functions.

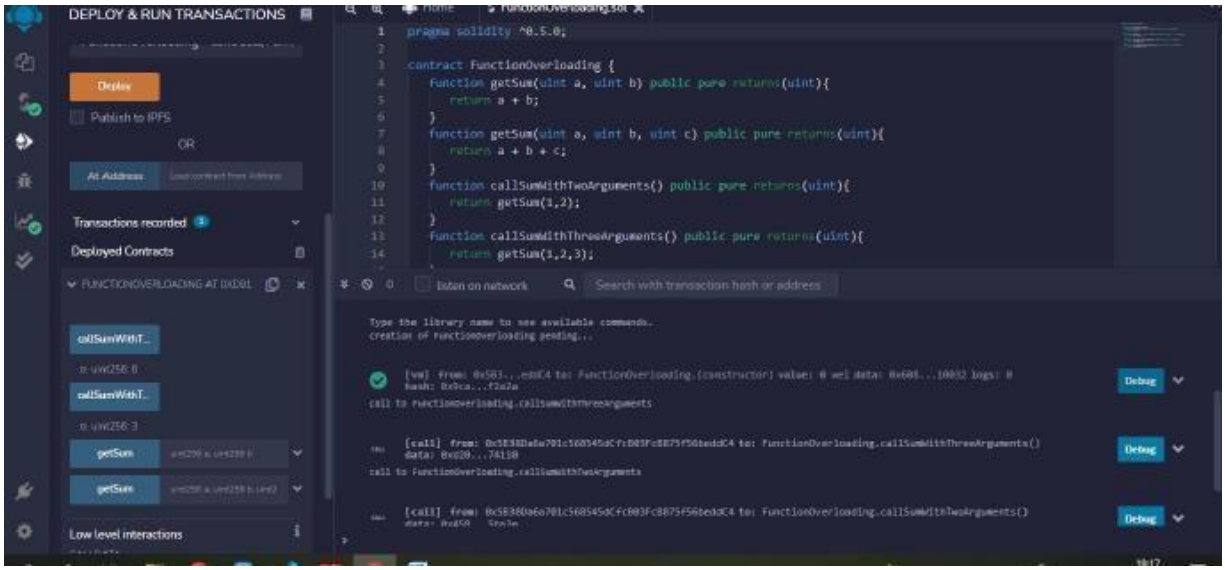
Code:

// Function overloading

```
pragma solidity ^0.5.0;

contract DemoContract {

  function calculate(uint a, uint b) public pure returns (uint)

{       return a + b;

  }

  function calculate(uint a, uint b, uint c) public pure returns (uint)

{       return a + b + c;

  }
```



Output



// Mathematical functions

function getSquareRoot(uint x) public pure returns (uint)

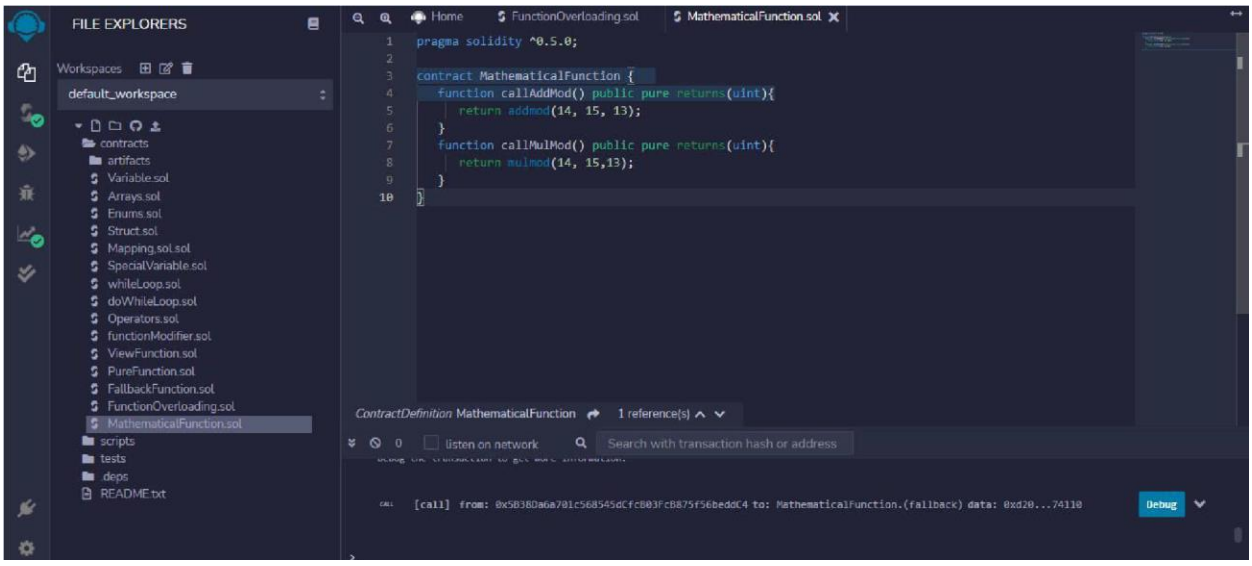
{ return uint(sqrt(x));

}

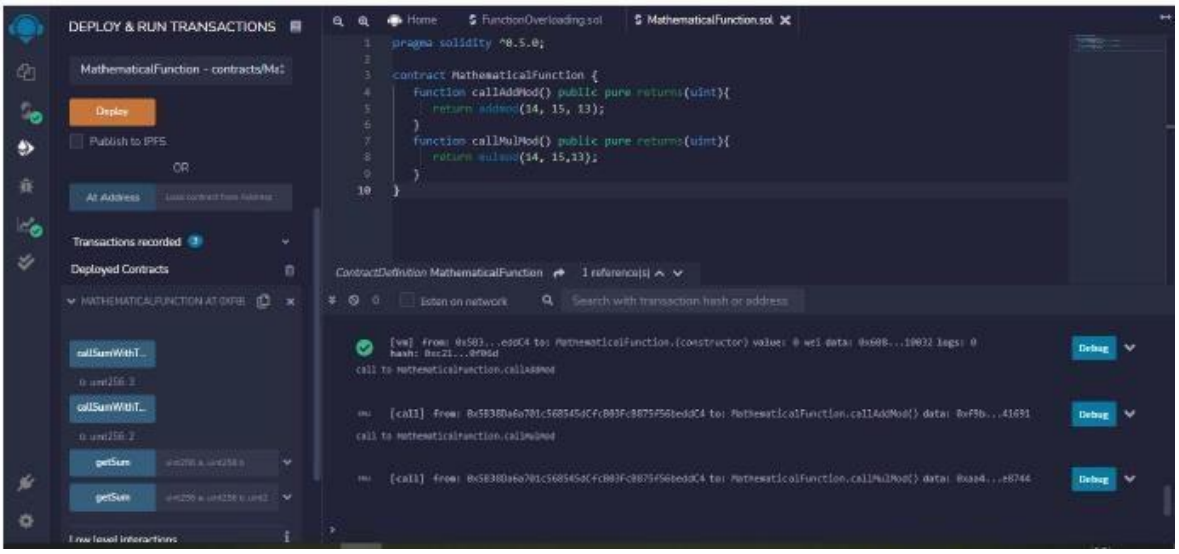
function power(uint base, uint exponent) public pure returns (uint)

{ return base ** exponent;

}



Output

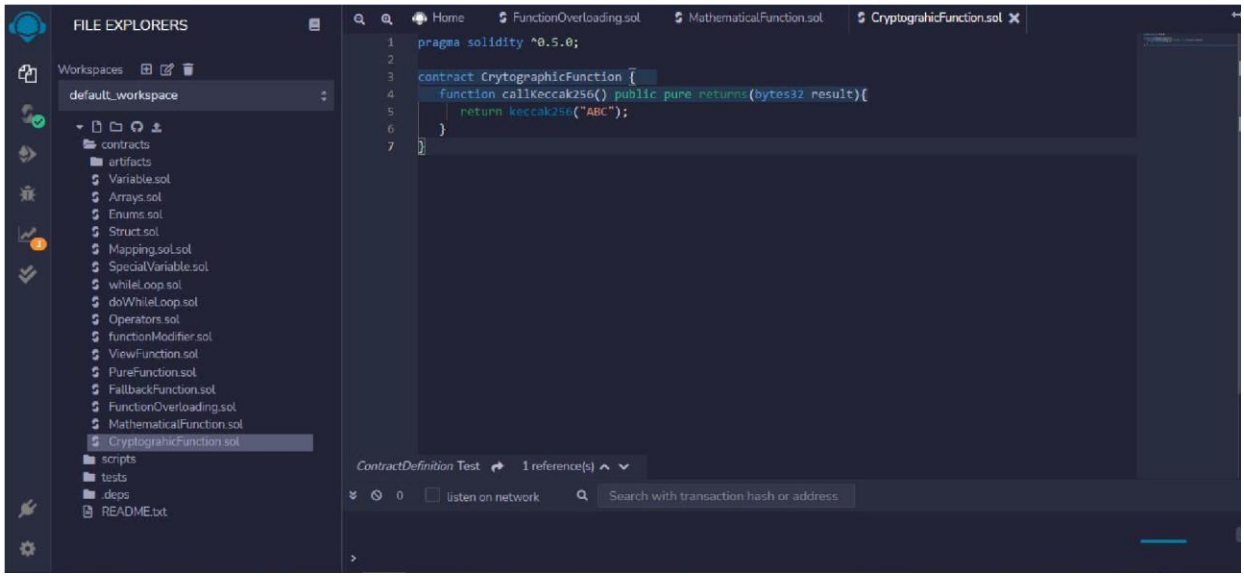


// Cryptographic functions

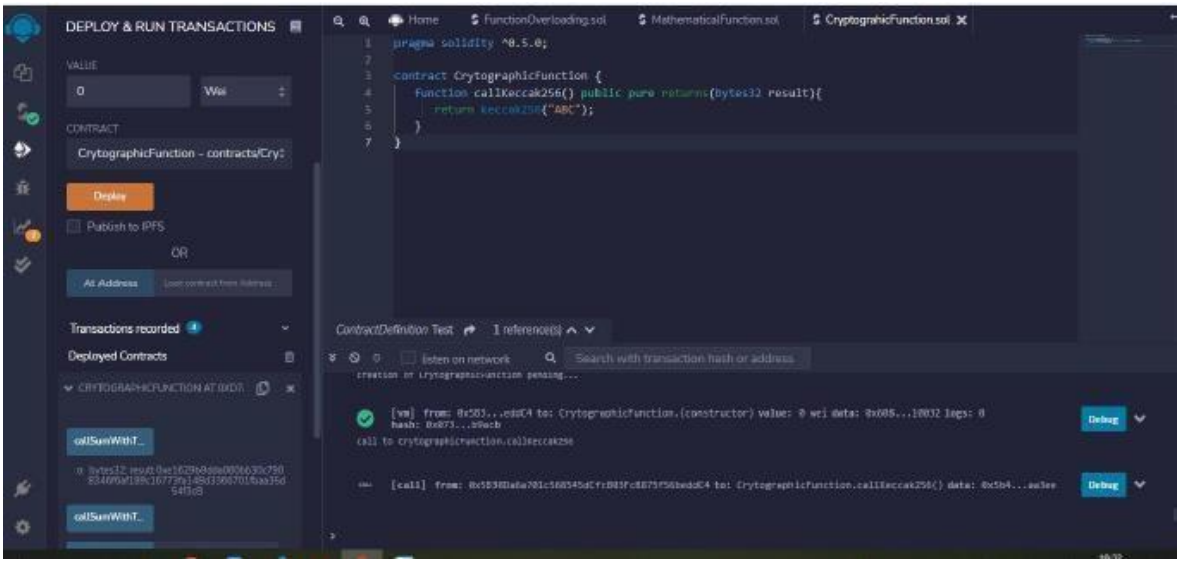
function hashData(string memory data) public pure returns (bytes32)

{ return keccak256(abi.encodePacked(data));

}



Output:


DEPLOY & RUN TRANSACTIONS
VALUE
0
Wei
CONTRACT
CrytographicFunction - contracts/Cry:
Deploy
Publish to IPFS
OR
At Address
Transactions recorded
Deployed Contracts
Home
FunctionOverloading.sol
MathematicalFunction.sol
CryptographicFunction.sol
pragma solidity ^0.5.0;
contract CrytographicFunction {
function callKeccak256() public pure returns(bytes32 result){
return keccak256("ABC");
}
}
ContractDefinition Test
1 reference(s)
listen on network
Search with transaction hash or address
Debug
Debug

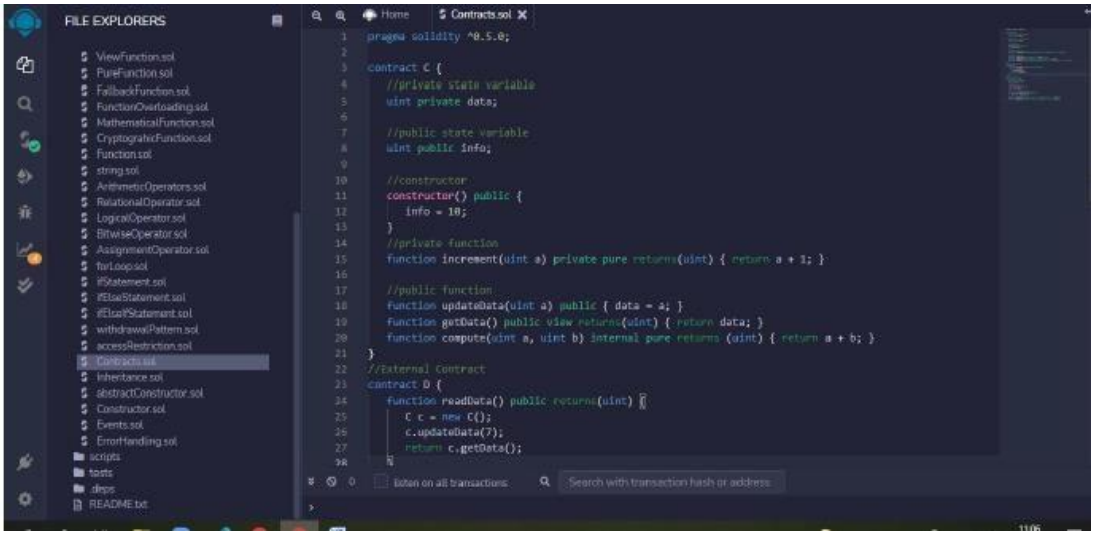
PRACTICAL 3 C.

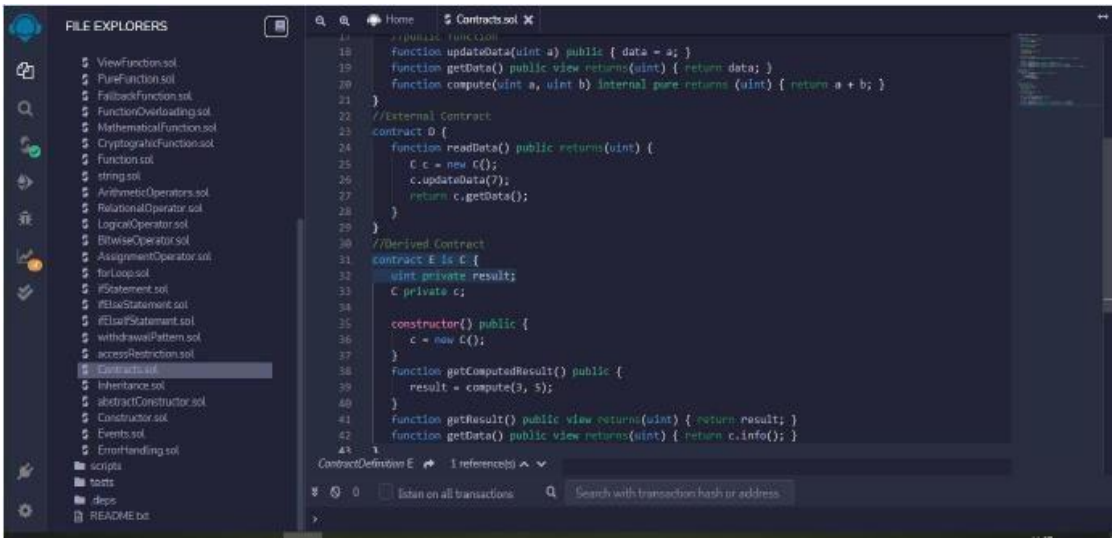
Aim: Write a Solidity program that demonstrates various features including contracts, inheritance, constructors, abstract contracts, interfaces.

Contracts:

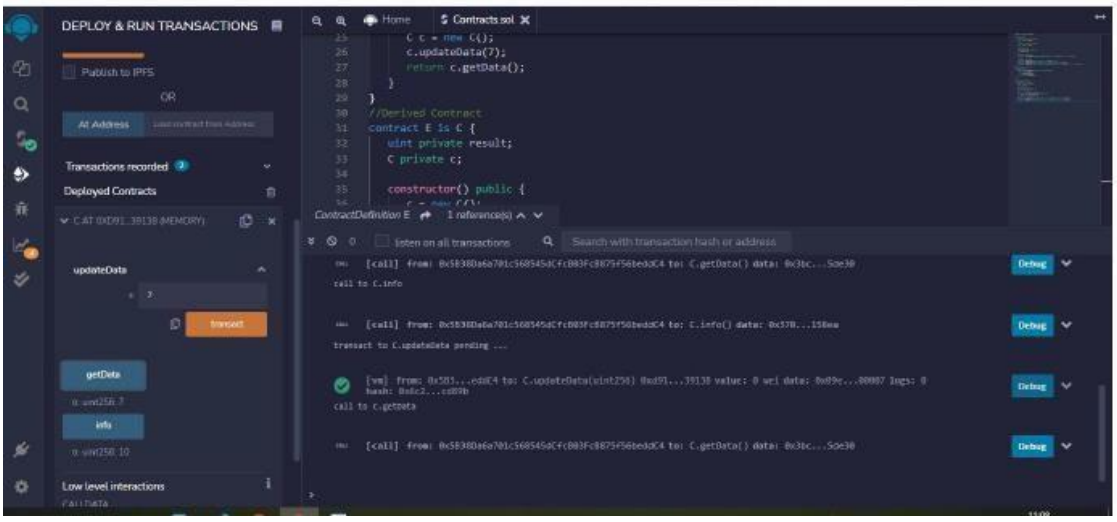
Code:

```
pragma solidity ^0.8.0;

contract HelloContract

{    string public message;

address public owner;

   // Constructor runs once on deployment

constructor(string memory initialMessage)

{       message = initialMessage;       owner =

msg.sender;

   }

   // Function to update the message     function updateMessage(string

memory newMessage) public {       require(msg.sender == owner, "Only the

owner can update the message");       message = newMessage;

   }

   // View function to return the message (also available as public variable)

function getMessage() public view returns (string memory) {       return

message;

   }

}
```


FILE EXPLORERS
Home
Contracts.sol
pragma solidity ^0.5.0;
contract C {
//private state variable
uint private data;
//public state variable
uint public info;
//constructor
constructor() public {
info = 10;
}
//private function
function increment(uint a) private pure returns(uint) { return a + 1; }
//public function
function updateData(uint a) public { data = a; }
function getData() public view returns(uint) { return data; }
function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}
//External Contract
contract D {
function readData() public returns(uint) {
C c = new C();
c.updateData(7);
return c.getData();
}
}


FILE EXPLORERS
Home
Contracts.sol
function updateData(uint a) public { data = a; }
function getData() public view returns(uint) { return data; }
function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}
//External Contract
contract D {
function readData() public returns(uint) {
C c = new C();
c.updateData(7);
return c.getData();
}
}
//Derived Contract
contract E is C {
uint private result;
C private c;
constructor() public {
c = new C();
}
function getComputedResult() public {
result = compute(3, 5);
}
function getResult() public view returns(uint) { return result; }
function getData() public view returns(uint) { return c.info(); }
}

Output:

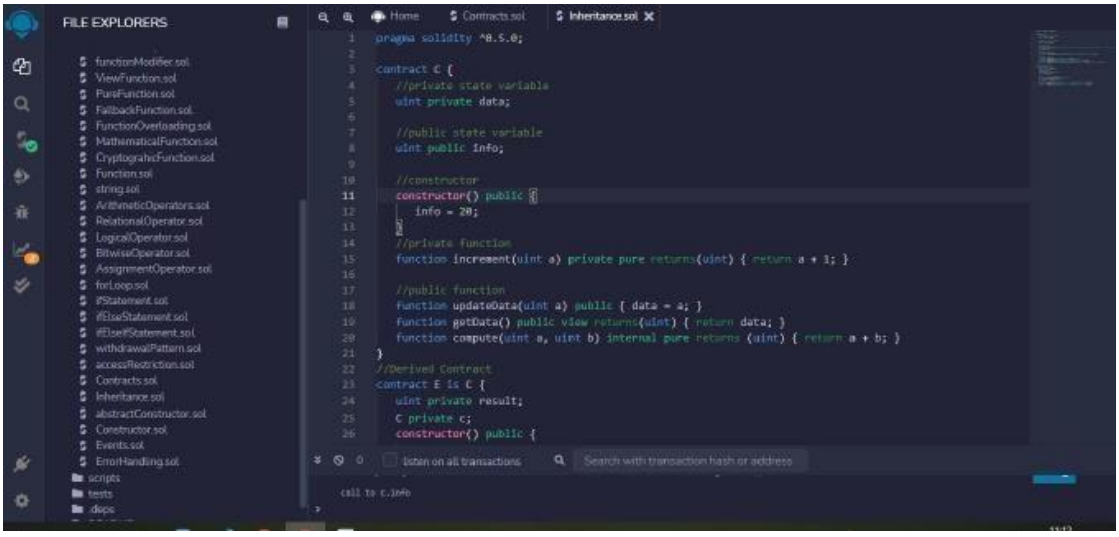

DEPLOY & RUN TRANSACTIONS
Publish to IPFS
OR
At Address
Transactions recorded
Deployed Contracts
updateData
transact
getData
info
Low level interactions
Home
Contracts.sol
C c = new C();
c.updateData(7);
return c.getData();
}
}
//Derived Contract
contract E is C {
uint private result;
C private c;
constructor() public {
ContractDefinition E
1 reference(s)
listen on all transactions
Search with transaction hash or address
call to C.info
transact to C.updateData pending ...
call to C.getData
Debug

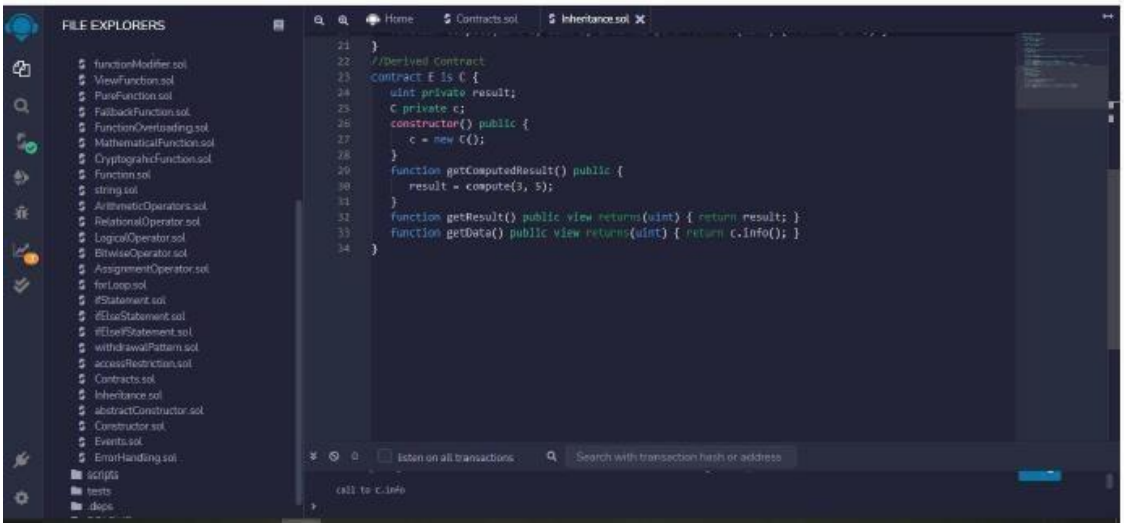
Inheritance:

Code:

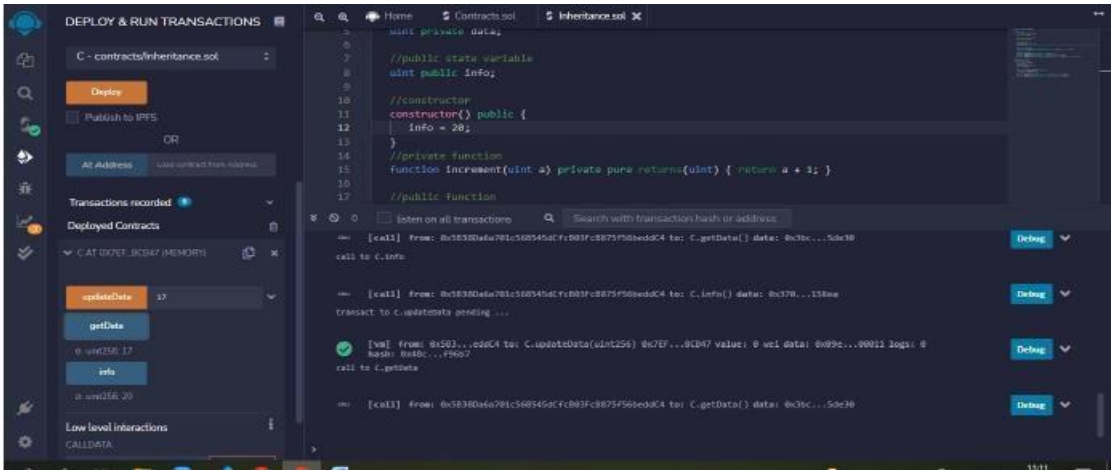
```
pragma solidity ^0.8.0;
/// simple inheritance example with animals ///
@notice Base contract representing a generic animal
contract Animal {     string public name;
constructor(string memory _name) {         name =
_name;
    }

    function makeSound() public virtual pure returns (string memory)
{        return "Some generic animal sound";
    }

    function getName() public view returns (string memory) {
        return name;
    }
}
/// @notice Derived contract representing a dog contract
Dog is Animal {

    constructor(string memory _name) Animal(_name) {}
    // Overrides makeSound from Animal     function makeSound() public
pure override returns (string memory) {
        return "Woof!";
    }
```

```
    function fetch() public pure returns (string memory)
{       return "Dog is fetching!";
    }
}

/// @notice Derived contract representing a cat
contract Cat is Animal {     constructor(string memory
_name) Animal(_name) {}
    // Overrides makeSound from Animal     function makeSound() public
pure override returns (string memory) {        return "Meow!";
    }
    function scratch() public pure returns (string memory)
{       return "Cat is scratching!";
    }
}
```





Output:



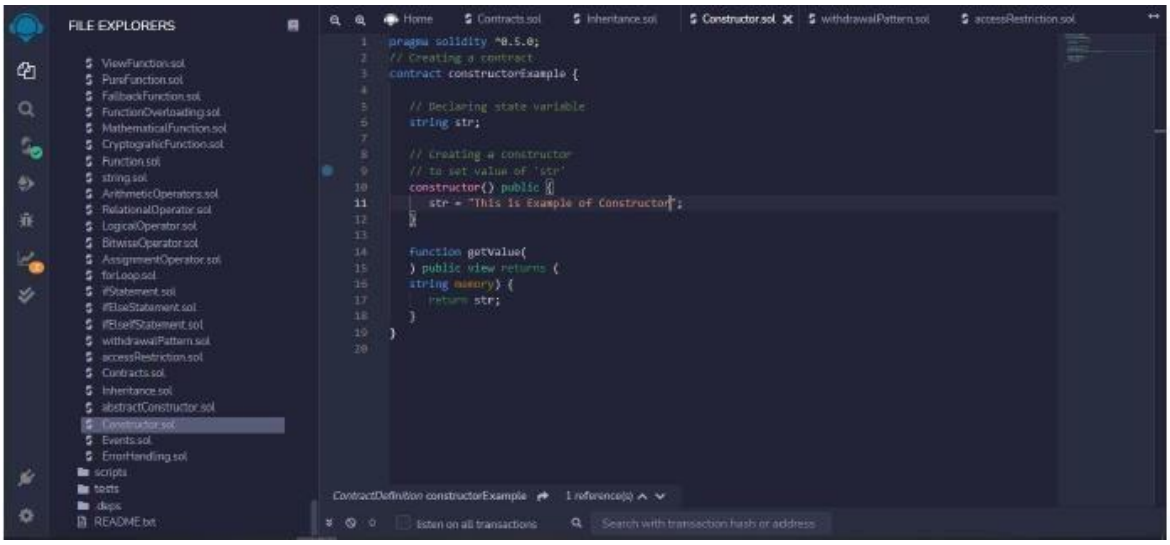
Constructors:

Code:

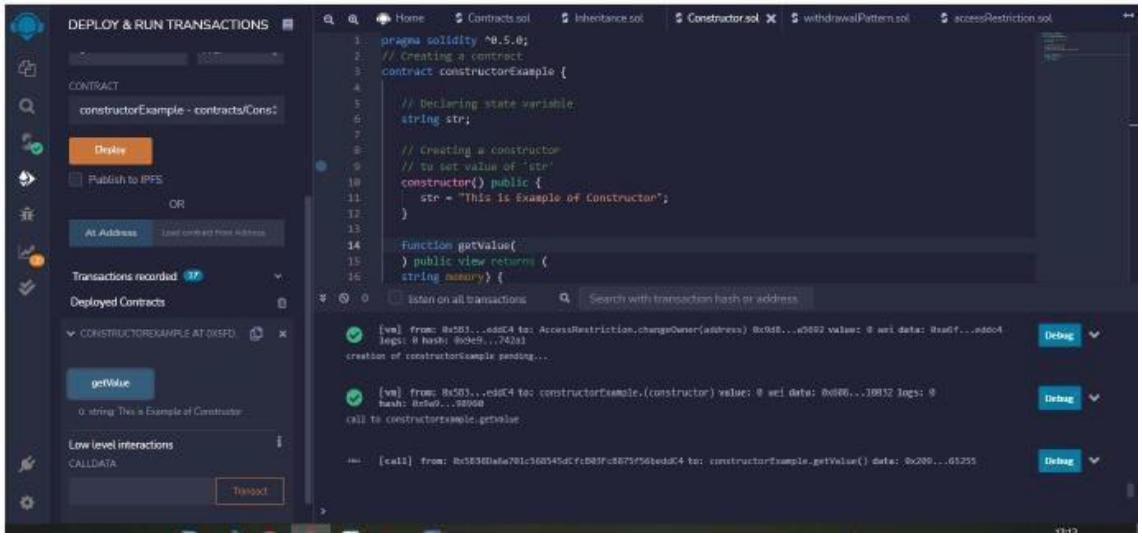
```
pragma solidity ^0.8.0;

/// @title Demonstrates constructors in Solidity

/// @notice Base contract with a constructor

contract Person {     string public name;     uint

public age;

    // Constructor to initialize name and age

constructor(string memory _name, uint _age) {         name =

_name;
```

```
        age = _age;
    }
    function getDetails() public view returns (string memory, uint)
{       return (name, age);
    }
}
/// @notice Derived contract that inherits Person and calls its constructor
contract Employee is Person {     uint public employeeId;     string public
department;
    // Constructor to initialize inherited and new properties
constructor(         string memory _name,
        uint _age,         uint
_employeeId,         string
memory _department     )
Person(_name, _age)
{       employeeId =
_employeeId;         department =
_department;
    }
    function getEmployeeDetails() public view returns
(       string memory,
        uint,
uint,
        string memory
    ) {
        return (name, age, employeeId, department);
    }
```

}



Output:



Abstract Contracts: Code:

pragma solidity ^0.8.0;

/// @title Abstract Contract Example: Shape

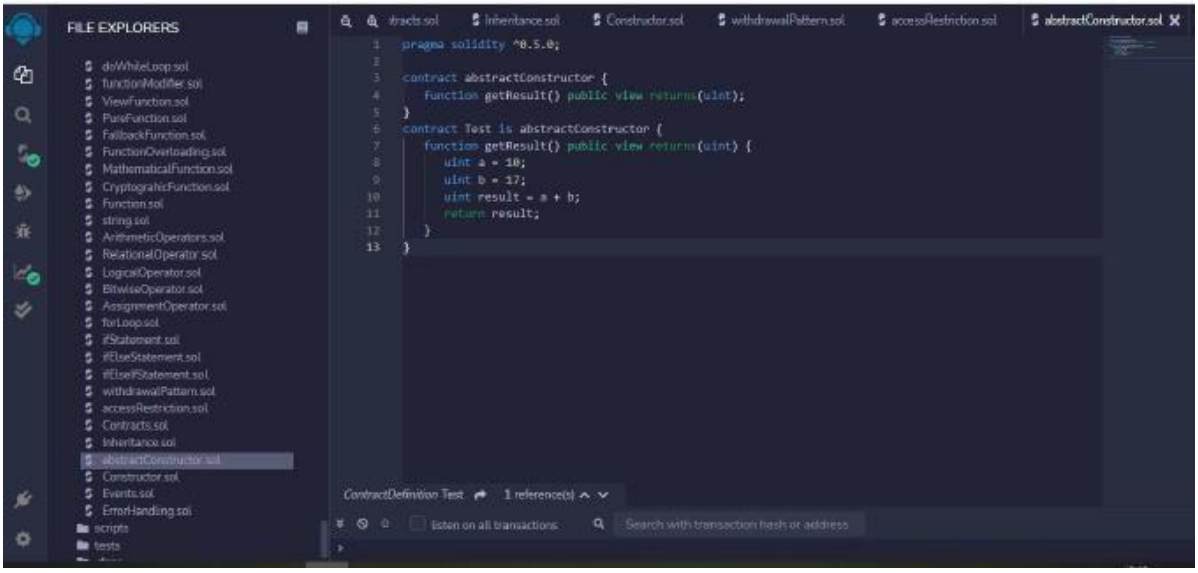
/// @notice Abstract contract defining a shape
abstract contract Shape { string public

shapeType; constructor(string memory

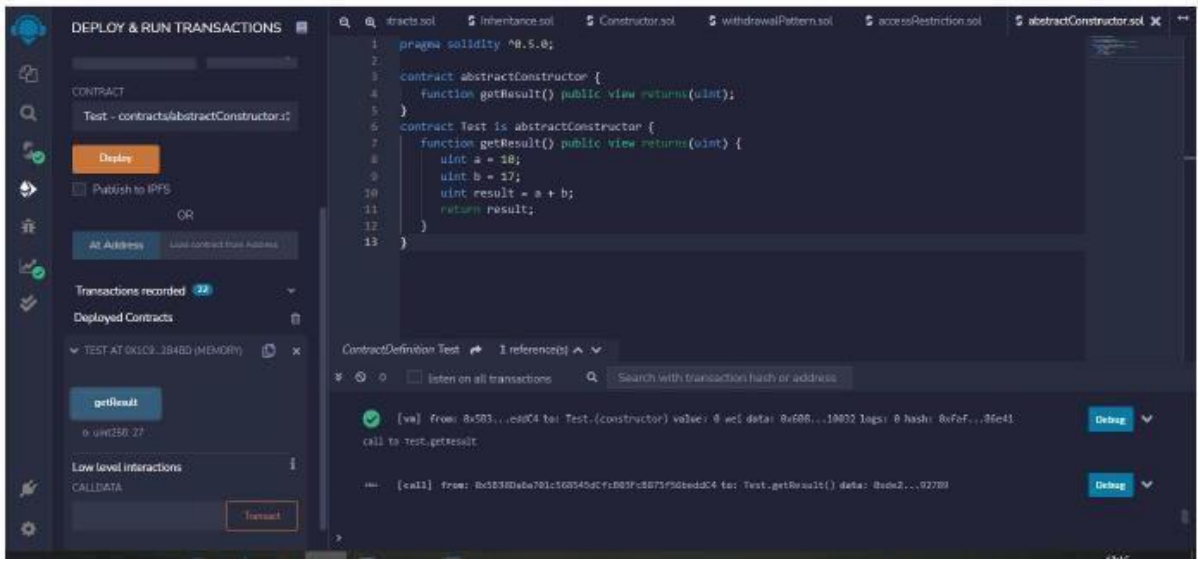
_shapeType) { shapeType =

_shapeType;

```
    }
    // Abstract function (no implementation)
function area() public view virtual returns (uint);
}
/// @notice Rectangle contract implementing Shape
contract Rectangle is Shape {     uint public width;
uint public height;

    constructor(uint _width, uint _height) Shape("Rectangle") {
        width = _width;
height = _height;
    }
    function area() public view override returns (uint)
{        return width * height;
    }
}
/// @notice Circle contract implementing Shape
contract Circle is Shape {     uint public radius;
constructor(uint _radius) Shape("Circle") {
        radius = _radius;
    }
    function area() public view override returns (uint) {
        // Approximate π as 314 / 100
return (314 * radius * radius) / 100;
    }
}
```



Output:



Interface:

Code:

pragma solidity ^0.8.0;

/// @title Demonstrates interface in Solidity /// @notice

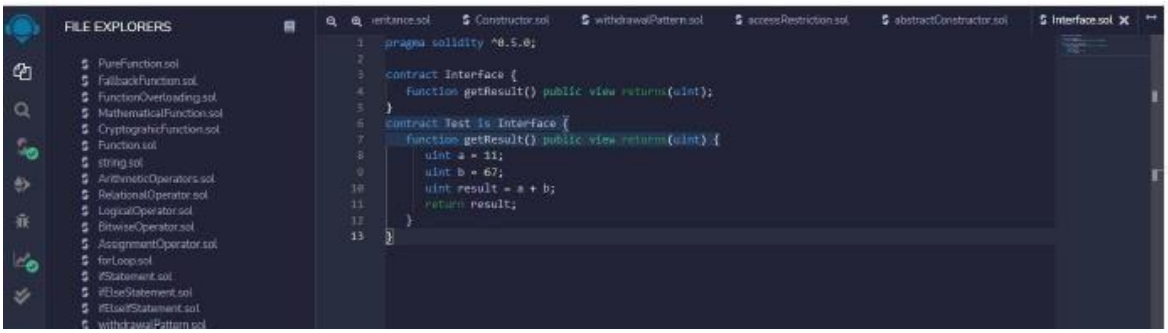
Interface defining a calculator interface ICalculator

{ function add(uint a, uint b) external pure returns (uint);

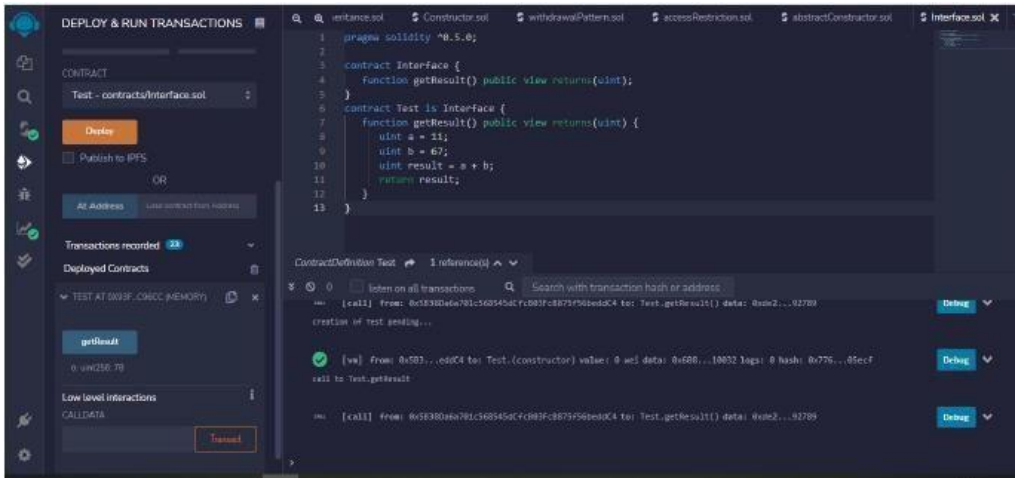
function subtract(uint a, uint b) external pure returns (uint);

```
function multiply(uint a, uint b) external pure returns (uint);
function divide(uint a, uint b) external pure returns (uint);
}
/// @notice Implementation of the calculator interface contract
SimpleCalculator is ICalculator {     function add(uint a, uint b)
external pure override returns (uint) {
        return a + b;
    }
    function subtract(uint a, uint b) external pure override returns (uint)
{        require(a >= b, "Underflow error");
        return a - b;
    }
    function multiply(uint a, uint b) external pure override returns (uint) {
        return a * b;
    }
    function divide(uint a, uint b) external pure override returns (uint)
{        require(b != 0, "Division by zero");         return a / b;
    }
}
/// @notice A contract that uses the ICalculator interface
contract CalculatorUser {     ICalculator public
calculator;     constructor(address
_calculatorAddress) {         calculator =
ICalculator(_calculatorAddress);
    }
    function useCalculator(uint a, uint b) public view returns (uint sum, uint diff, uint prod, uint
quot) {         sum = calculator.add(a, b);         diff = calculator.subtract(a, b);         prod =
calculator.multiply(a, b);         quot = calculator.divide(a, b);
```

```
    }
}
```



Output:



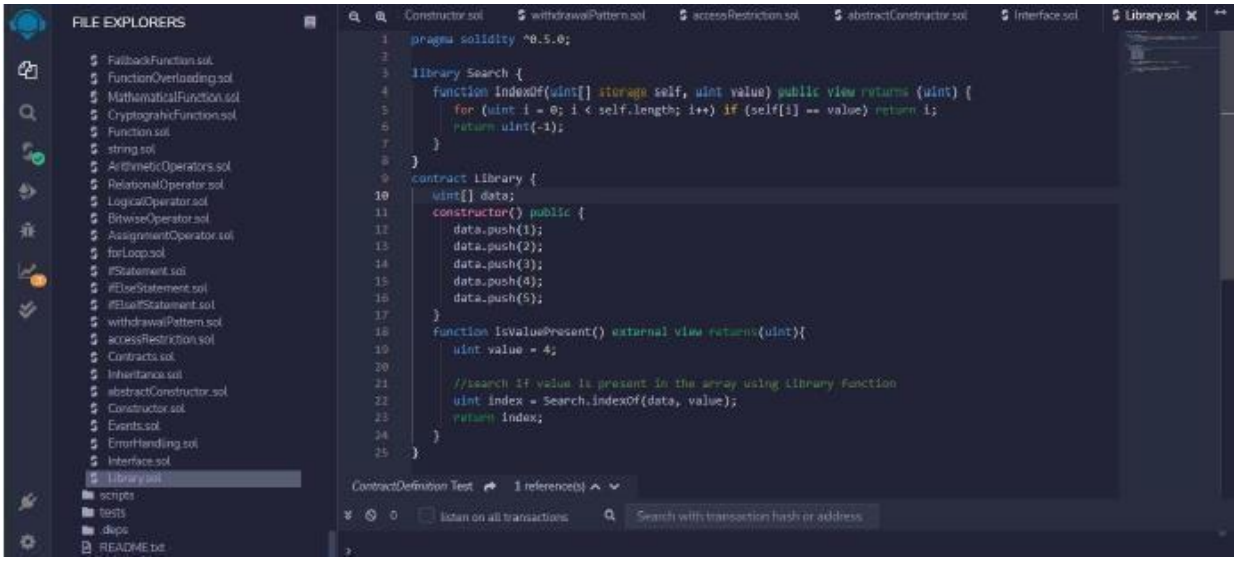
PRACTICAL 3 D.

Aim: Write a Solidity program that demonstrates use of libraries, assembly, events, and error handling **Libraries:**

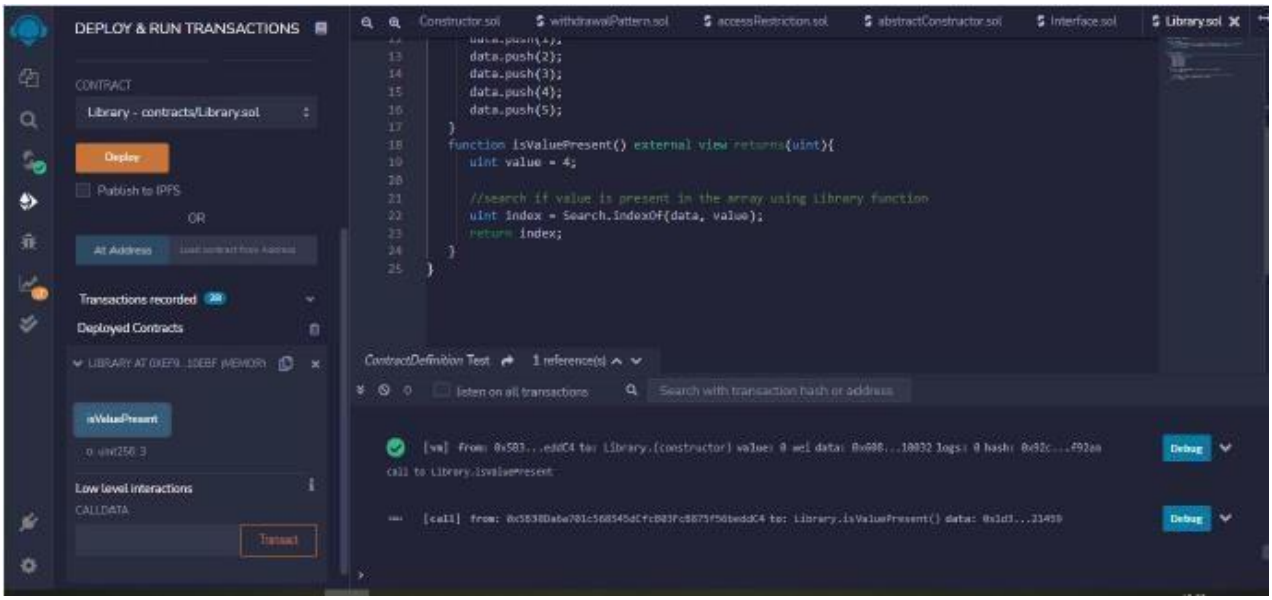
Code:

```
pragma solidity ^0.8.0;

/// @title Demonstrates use of libraries in Solidity

/// @notice A simple math library library MathLib

{    function add(uint a, uint b) internal pure returns (uint)

{

     return a + b;

   }
```

```
   function subtract(uint a, uint b) internal pure returns (uint)
{       require(a >= b, "Underflow error");
      return a - b;
   }
   function multiply(uint a, uint b) internal pure returns (uint) {
      return a * b;
   }
   function divide(uint a, uint b) internal pure returns (uint)
{       require(b != 0, "Division by zero");        return a / b;
   }
}

/// @notice A contract that uses the MathLib library contract
Calculator {
   using MathLib for uint;
function compute(
      uint a,
uint b
   ) public pure returns (uint sum, uint diff, uint prod, uint quot)
{       sum = a.add(b);       diff = a.subtract(b);       prod =
a.multiply(b);       quot = a.divide(b);
   }
}
```



Output:



Assembly:

Code:

pragma solidity ^0.8.0;

/// @title Demonstrates inline assembly usage in Solidity contract

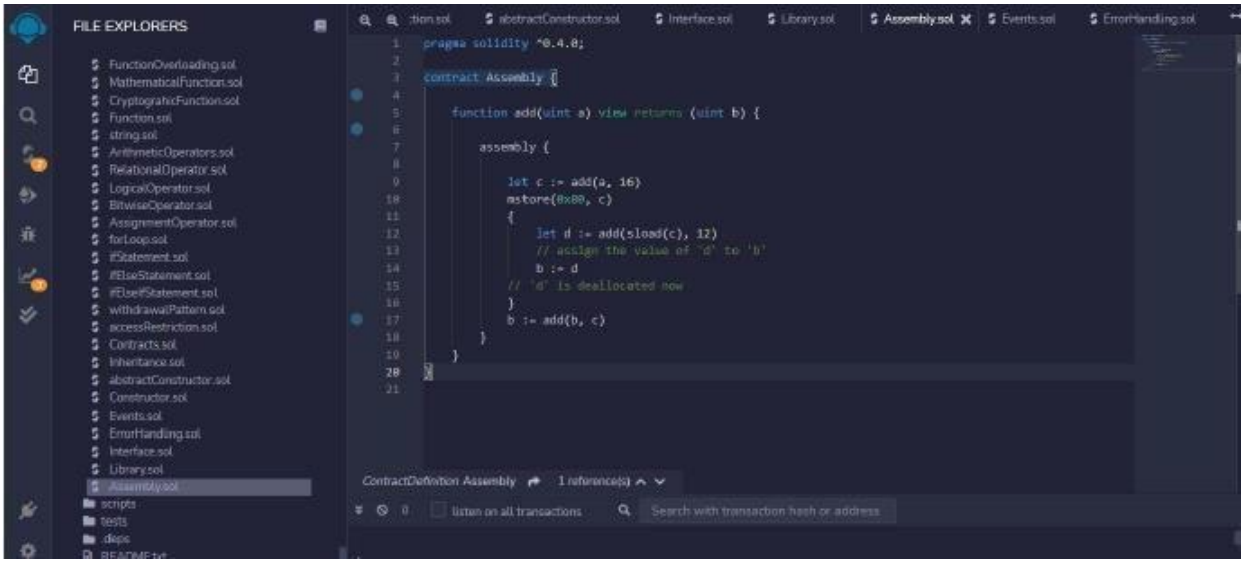
AssemblyDemo {

/// @notice Adds two numbers using assembly function

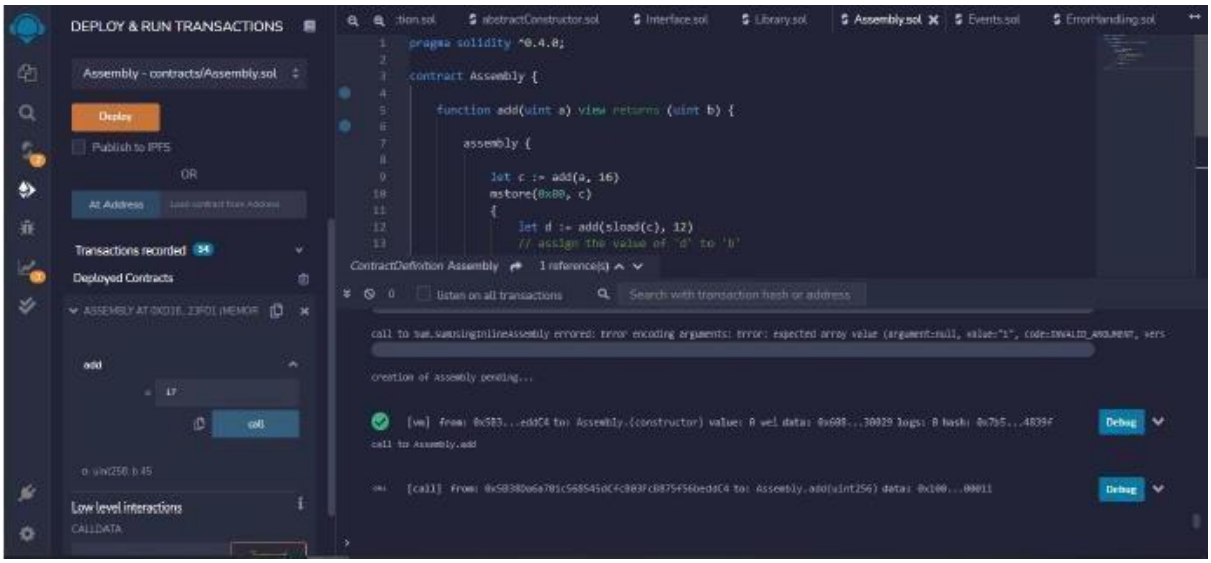
addAsm(uint a, uint b) public pure returns (uint result) {

```
        assembly
{           result := add(a,
b)
        }
    }
    /// @notice Returns caller address using assembly
function getCaller() public view returns (address caller) {
        assembly {
            caller := caller()
        }
    }
    /// @notice Returns the square of a number using multiplication in assembly
function square(uint x) public pure returns (uint result) {
        assembly
{           result := mul(x,
x)
        }
    }
    /// @notice Demonstrates conditional branching with assembly
function isEven(uint number) public pure returns (bool even) {
        assembly {          switch
mod(number, 2)
            case 0
{               even := 1
            }
            default
{               even := 0
```

```
        }
    }
  }
}
```



Output:



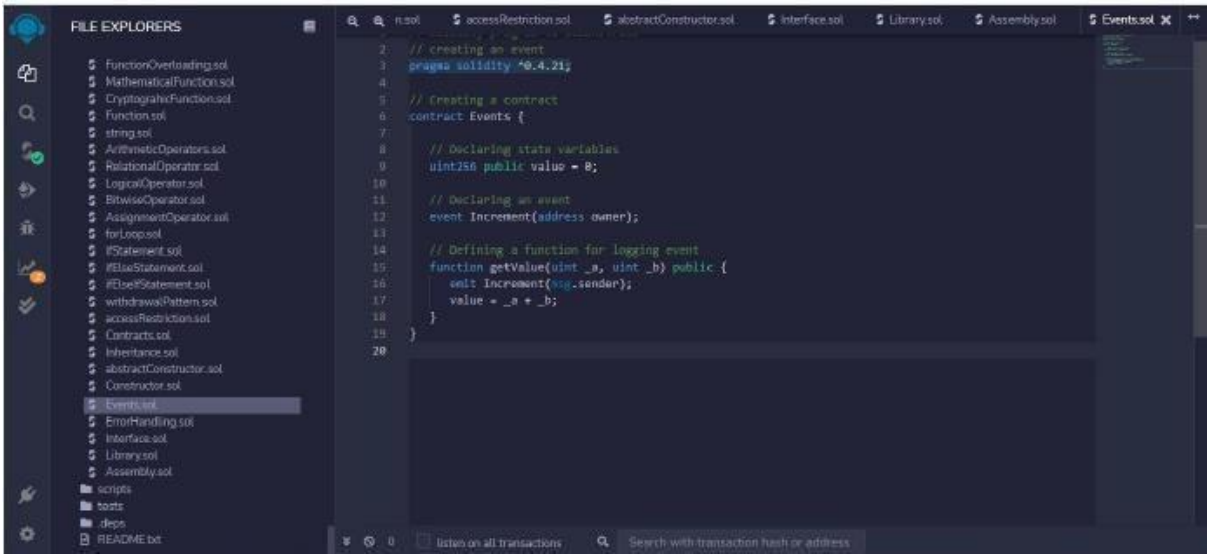
Events:

Code:

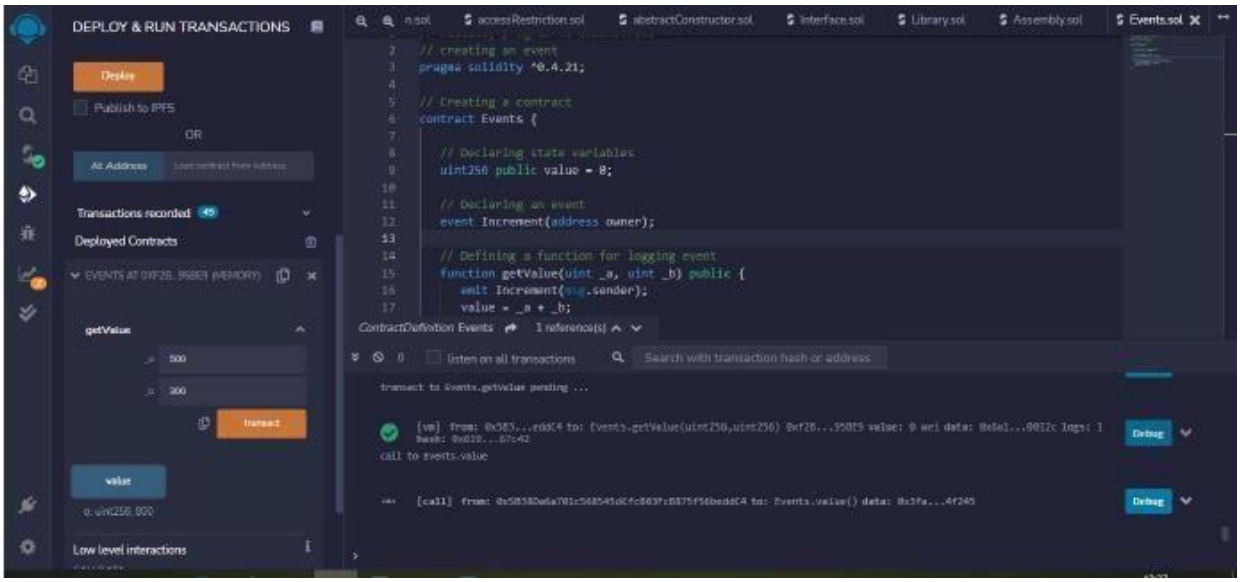
pragma solidity ^0.8.0;

```
/// @title Demonstrates the use of events in Solidity contract
EventDemo {
  // Declare an event that logs when a user registers    event
UserRegistered(address indexed userAddress, string username);
  // Declare an event for balance updates
  event BalanceUpdated(address indexed user, uint newBalance);
mapping(address => uint) public balances;    mapping(address => string) public
usernames;    /// @notice Register a new user with a username    function
register(string calldata _username) external
{    require(bytes(usernames[msg.sender]).length  ==  0,  "User  already
registered");    usernames[msg.sender]  =  _username;    //  Emit  the
UserRegistered event       emit UserRegistered(msg.sender, _username);
  }
  /// @notice Deposit some amount to the sender's balance
function deposit() external payable {       require(msg.value >
0, "Must send some ether");       balances[msg.sender] +=
msg.value;       // Emit the BalanceUpdated event       emit
BalanceUpdated(msg.sender, balances[msg.sender]);
  }
  /// @notice Withdraw a specified amount from sender's balance
function withdraw(uint _amount) external
{       require(balances[msg.sender] >= _amount, "Insufficient
balance");       balances[msg.sender] -= _amount;
payable(msg.sender).transfer(_amount);       // Emit the
BalanceUpdated event       emit BalanceUpdated(msg.sender,
balances[msg.sender]);
  }
```

}


FILE EXPLORERS
FunctionOverloading.sol
MathematicalFunction.sol
CryptographicFunction.sol
Function.sol
string.sol
ArithmeticOperators.sol
RelationalOperator.sol
LogicalOperator.sol
BitwiseOperator.sol
AssignmentOperator.sol
forLoop.sol
IfStatement.sol
IfElseStatement.sol
IfElseIfStatement.sol
withdrawalPattern.sol
accessRestriction.sol
Contracts.sol
Inheritance.sol
abstractConstructor.sol
Constructor.sol
Events.sol
ErrorHandling.sol
Interface.sol
Library.sol
Assembly.sol
scripts
tests
.deps
README.txt
// creating an event
pragma solidity ^0.4.21;
// Creating a contract
contract Events {
// Declaring state variables
uint256 public value = 0;
// Declaring an event
event Increment(address owner);
// Defining a function for logging event
function getValue(uint _a, uint _b) public {
emit Increment(msg.sender);
value = _a + _b;
}
}
listen on all transactions
Search with transaction hash or address

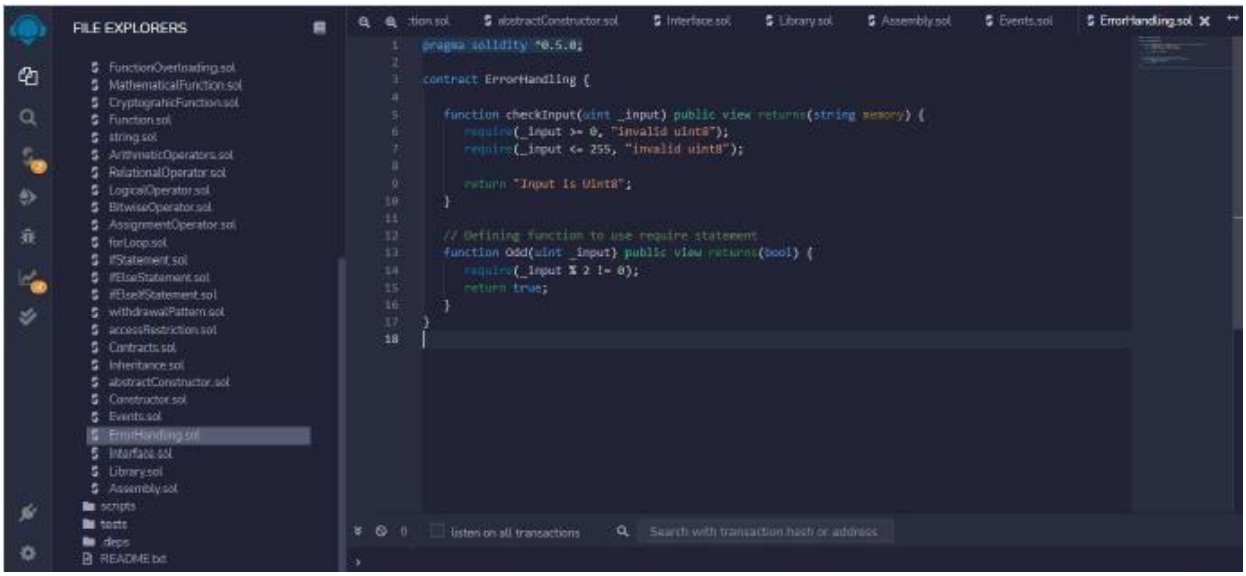
Output:


DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
OR
At Address
Transactions recorded 45
Deployed Contracts
getValue
500
300
transact
value
0: uint256: 800
Low level interactions
// creating an event
pragma solidity ^0.4.21;
// Creating a contract
contract Events {
// Declaring state variables
uint256 public value = 0;
// Declaring an event
event Increment(address owner);
// Defining a function for logging event
function getValue(uint _a, uint _b) public {
emit Increment(msg.sender);
value = _a + _b;
ContractDefinition Events 1 reference(s)
listen on all transactions
Search with transaction hash or address
transact to Events.getValue pending ...
call to Events.value
Debug
Debug

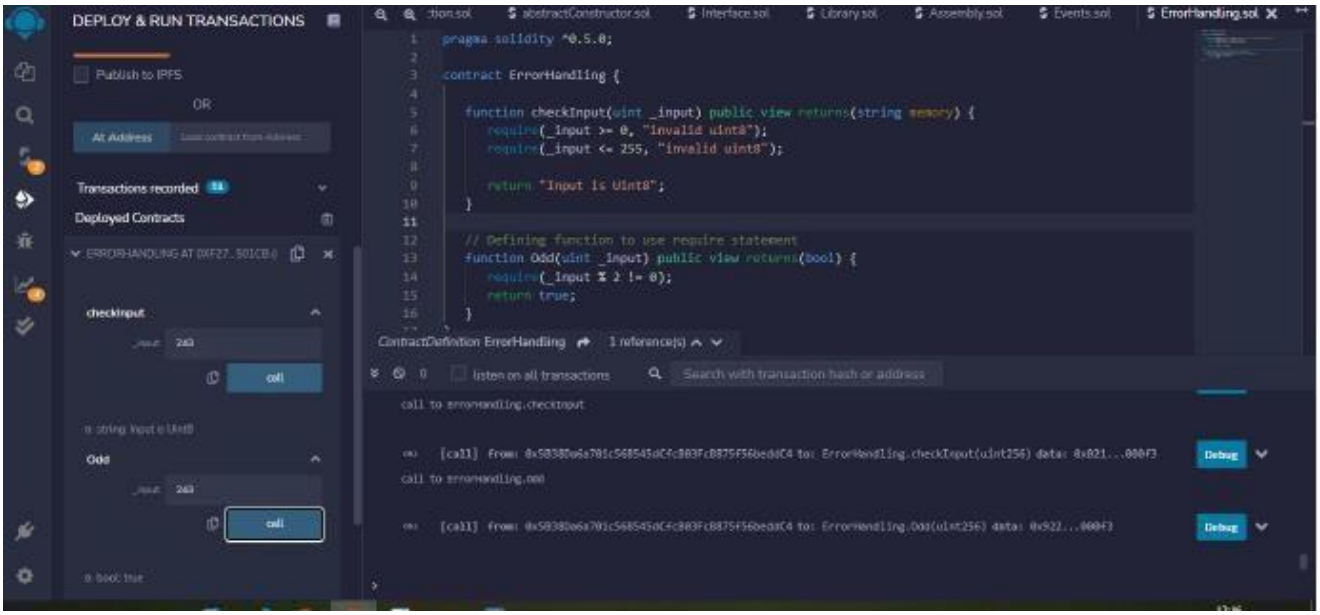
Error Handling:

Code:

```
pragma solidity ^0.8.0;
/// @title Demonstrates error handling in Solidity contract
ErrorHandlingDemo {     mapping(address => uint) public
balances;     /// @notice Deposit ether into your balance
function deposit() external payable {        require(msg.value > 0,
"Deposit must be greater than zero");        balances[msg.sender]
+= msg.value;
   }
   /// @notice Withdraw specified amount of ether
function withdraw(uint _amount) external {        //
Use revert with custom error message        if
(_amount > balances[msg.sender])
{           revert("Insufficient balance");
      }
      balances[msg.sender] -= _amount;
      // Send ether back to caller
      (bool success, ) = msg.sender.call{value: _amount}("");
require(success, "Transfer failed");
   }
   function testAssert(uint _value) external pure {
      // Assert should be used for conditions that should never fail
assert(_value != 0);
   }
}
```


FILE EXPLORERS
FunctionOverloading.sol
MathematicalFunction.sol
CryptographicFunction.sol
Function.sol
string.sol
ArithmeticOperators.sol
RelationalOperator.sol
LogicalOperator.sol
BitwiseOperator.sol
AssignmentOperator.sol
forLoop.sol
IfStatement.sol
IfElseStatement.sol
IfElseIfStatement.sol
withdrawalPattern.sol
accessRestriction.sol
Contracts.sol
Inheritance.sol
abstractConstructor.sol
Constructor.sol
Events.sol
ErrorHandling.sol
Interface.sol
Library.sol
Assembly.sol
scripts
tests
deps
README.txt
pragma solidity ^0.5.0;
contract ErrorHandling {
function checkInput(uint _input) public view returns(string memory) {
require(_input >= 0, "invalid uint8");
require(_input <= 255, "invalid uint8");
return "Input is UInt8";
}
// Defining function to use require statement
function Odd(uint _input) public view returns(bool) {
require(_input % 2 != 0);
return true;
}
}
listen on all transactions
Search with transaction hash or address

Output:


DEPLOY & RUN TRANSACTIONS
Publish to IPFS
OR
At Address
Transactions recorded
Deployed Contracts
checkInput
call
Odd
call
pragma solidity ^0.5.0;
contract ErrorHandling {
function checkInput(uint _input) public view returns(string memory) {
require(_input >= 0, "invalid uint8");
require(_input <= 255, "invalid uint8");
return "Input is UInt8";
}
// Defining function to use require statement
function Odd(uint _input) public view returns(bool) {
require(_input % 2 != 0);
return true;
}
ContractDefinition ErrorHandling
1 reference(s)
listen on all transactions
Search with transaction hash or address
call to errorHandling.checkInput
call to errorHandling.Odd
Debug
Debug

PRACTICAL 3 E.

Aim: Build a decentralized application (DApp) using Angular for the front end and Trufle along with Ganache CLI for the back end.

Code:

To build a decentralized application (DApp) using Angular for the front end and Trufle with Ganache CLI for the back end, there are following steps

1. **Set up Ganache CLI** (for local Ethereum blockchain)
2. **Create and deploy a smart contract using Truffle** 3. **Build the Angular front end** 4. **Integrate Angular front end with Ethereum blockchain**

Step 1: Set up Ganache CLI (Ethereum Local Blockchain)

First, install Ganache CLI. Ganache is a personal Ethereum blockchain which you can use for development purposes.

1. Install Ganache CLI globally using npm:
2. npm install -g ganache-cli
3. Run Ganache CLI to start a local Ethereum blockchain:
4. ganache-cli

   By default, this will run on http://127.0.0.1:8545 and you'll see the list of accounts with balances.

Step 2: Create and Deploy Smart Contract with Truffle

1. Install Truffle

Install Trufle globally on your machine:

npm install -g trufle

2. Initialize a Truffle Project

Create a directory for your project, then initialize Trufle:

mkdir my-dapp cd
my-dapp
trufle init

3. Write a Smart Contract

In the contracts/ folder, create a file SimpleStorage.sol with a simple smart contract to store and retrieve a number.

```
// contracts/SimpleStorage.sol
pragma solidity ^0.8.0;

contract SimpleStorage {
   uint256 private storedNumber;

   function set(uint256 num) public
{       storedNumber = num;
   }

   function get() public view returns (uint256)
{       return storedNumber;
   } }
```

4. Compile the Contract In

the project directory, run:

trufle compile

5. Deploy the Contract

Create a migration file in migrations/2_deploy_contracts.js:

```
const SimpleStorage = artifacts.require("SimpleStorage");

module.exports = function (deployer) {
  deployer.deploy(SimpleStorage);
};
```

Now, deploy the contract to your local Ganache instance:

trufle migrate --network development

6. Interact with the Contract

To test interactions, run the Trufle console:
trufle console --network development

In the console, interact with the deployed contract:

```
let instance = await SimpleStorage.deployed(); await
instance.set(42);  // Set a value let value = await
instance.get();  // Get the stored value
console.log(value.toString());  // Should print 42
```

Step 3: Build the Angular Front End

1. Create Angular Project

Create an Angular project using the Angular CLI:

```
ng new my-dapp-frontend cd
my-dapp-frontend
```

Install the necessary dependencies for Web3.js (to interact with Ethereum):

```
npm install web3
```

2. Create a Service to Interact with the Blockchain

In your Angular project, create a service to connect to the blockchain and interact with the smart contract. Generate a service:

```
ng generate service blockchain
```

Edit blockchain.service.ts to connect with Ganache and interact with the SimpleStorage contract:

```
import { Injectable } from '@angular/core';
import Web3 from 'web3';
import { environment } from '../environments/environment';

@Injectable({
  providedIn: 'root'
})
export class BlockchainService {

  private web3: Web3;
  private contract: any;
  private contractAddress = 'YOUR_CONTRACT_ADDRESS';
private contractABI = [ /* ABI from Trufle build */ ];
```

```
 constructor() {     this.web3 = new Web3('http://127.0.0.1:8545');  // Connect to
Ganache CLI     this.contract = new this.web3.eth.Contract(this.contractABI,
this.contractAddress);
 }

 async getStoredNumber(): Promise<number> {
  return await this.contract.methods.get().call();
 }

 async setStoredNumber(number: number): Promise<void>
{     const accounts = await this.web3.eth.getAccounts();
  await this.contract.methods.set(number).send({ from: accounts[0] });
 } }
```

3. Display Data in the Component

```
import { Component } from '@angular/core'; import
{ BlockchainService } from './blockchain.service';

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{   title = 'My DApp';
 storedNumber: number;

 constructor(private blockchainService: BlockchainService)
{     this.loadStoredNumber();
 }

 async loadStoredNumber() {
  this.storedNumber = await this.blockchainService.getStoredNumber();
 }

 async setStoredNumber(number: number) {     await
this.blockchainService.setStoredNumber(number);
this.loadStoredNumber();  // Refresh the number
 } }
```

4. Update the Template

Update the app.component.html to display the stored number and provide an input to set a new number:

```
<div style="text-align:center;">
  <h1>
    Welcome to {{ title }}!
  </h1>
  <p>The current stored number is: {{ storedNumber }}</p>
  <input type="number" [(ngModel)]="newNumber" placeholder="Enter number" />
<button (click)="setStoredNumber(newNumber)">Set Number</button> </div>
```

5. Add Angular Forms Module

In app.module.ts, import the necessary module:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component'; import
{ BlockchainService } from './blockchain.service'; import
{ FormsModule } from '@angular/forms';

@NgModule({
  declarations: [AppComponent],   imports:
[BrowserModule, FormsModule],
  providers: [BlockchainService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Step 4: Run the Application

1. **Start Ganache CLI** (if not already running):
2. ganache-cli
3. **Deploy the smart contract** (if not already deployed): 4. trufle migrate --network development 5. **Run the Angular Application**:
6. ng serve

Now, visit http://localhost:4200/ in your browser. You should see the stored number from your smart contract, and you can set a new number using the input field.

PRACTICAL 4 A.

Aim: Install and demonstrate use of Hypeledger Irhoa

Code:

Below is a step-by-step guide to **install Hyperledger Iroha** and demonstrate its basic functionality.

Step 1: Prerequisites

1. **Docker**: Hyperledger Iroha uses Docker containers for running its components. ○ Install Docker from Docker's official site.
2. **Docker Compose**: This tool allows you to define and run multi-container Docker applications.
    - Install Docker Compose from here.
3. **Git**: To clone repositories.
    - Install Git from here.
4. **Java**: Hyperledger Iroha is developed using Java, so make sure you have Java (JDK 11 or higher) installed.
    - Install Java from here.
5. **CMake and build tools** (for building the Hyperledger Iroha binaries, if you plan to build it yourself):
    - Install CMake from here. ○ Install build-essential (for Linux) via the package manager: ○ sudo apt-get install build-essential

Step 2: Clone Hyperledger Iroha Repository

Clone the Hyperledger Iroha repository from GitHub to your local machine.

git clone https://github.com/hyperledger/iroha.git cd
iroha

Step 3: Build Hyperledger Iroha (Optional Step)

If you want to build Hyperledger Iroha from source, follow these steps:

1. **Install dependencies**:
    - For Ubuntu:
    - sudo apt-get update
    - sudo apt-get install -y cmake g++ libssl-dev libboost-all-dev \ ○ libgmp-dev libsodium-dev python3-pip libcurl4-openssl-dev \ ○ zlib1g-dev libzmq3-dev

liblzma-dev libprotobuf-dev protobuf-compiler ○ For macOS, use Homebrew to install dependencies:
- brew install cmake boost openssl libsodium libzmq protobuf

2. **Build the project**:
3. mkdir build
4. cd build 5. cmake ..
6. make -j$(nproc) # Use appropriate number of CPU cores for faster build

This step builds Hyperledger Iroha from source. If you just want to run the demo (using prebuilt binaries), you can skip this step and proceed to using Docker.

Step 4: Use Docker to Run Hyperledger Iroha

Instead of building from source, we can use Docker to easily spin up the Iroha network.

1. **Use Docker Compose to bring up the containers**:
   In the iroha directory, you should find a docker-compose.yml file. To start the Iroha network with all the components (like the peer, validator, etc.), run the following command:
2. docker-compose -f docker/docker-compose.yml up

   This will pull the necessary images and start Iroha in Docker containers. You'll see logs of various services being initialized.

3. **Check if everything is up**:
   After a few seconds, check if the containers are running:
4. docker ps

   You should see containers for iroha and its components such as iroha-peer, irohaledger, etc.

Step 5: Interact with the Iroha Network

Once the containers are running, you can interact with the network. Hyperledger Iroha provides a REST API and a JavaScript client SDK (IrohaJS), but for simplicity, we'll start by interacting using curl or any HTTP client.

1. Create a User

To interact with the Iroha blockchain, you need to create a user. Iroha uses commands like create account, add asset, etc. Let's start by creating a user using the REST API.

- To create an account, we can send a request to the REST API like so (you may want to modify it according to your setup, e.g., IP and port):

```
curl -X POST http://localhost:50051/create_account \
  -H "Content-Type: application/json" \
  -d '{
      "creator": "admin",
      "name": "user1",
      "password": "password123"
  }'
```

This will create an account named user1 with the password password123 under the admin account.

2. View Account Information

To verify the creation of the account and view account details, send a get account command.

```
curl -X GET http://localhost:50051/account/user1
```

This should return the details of the newly created account.

3. Transfer Assets Between Accounts

Now that we have two accounts, we can transfer assets between them. Let's transfer a certain number of assets from one account to another:

```
curl -X POST http://localhost:50051/transfer \
  -H "Content-Type: application/json" \
  -d '{
      "from": "user1",
      "to": "user2",
      "amount": "100",
      "asset_id": "coin"
  }'
```

This will transfer 100 units of the asset coin from user1 to user2.

4. Check Transaction Status

You can also check the status of a transaction to ensure that it was successful. Here's how to do it:

```
curl -X GET http://localhost:50051/transaction_status \   -
d '{"tx_id": "your-transaction-id"}'
```

Replace "your-transaction-id" with the actual ID of the transaction.

Step 6: Using Hyperledger Iroha SDK

You can use the Hyperledger Iroha SDK (available in several languages, such as Java, Python, and JavaScript) to interact with Iroha more programmatically.

Using IrohaJS

1. Install IrohaJS:

```
npm install iroha-helpers iroha-client
```

2. Interact with the Iroha network:

```
const { Iroha, IrohaAPI } = require('iroha-helpers');

// Create a connection to the Iroha network
const iroha = new Iroha('localhost', 50051);

// Create an account
iroha.createAccount('user1', 'password123').then(response =>
{   console.log(response);
});

// Transfer assets
iroha.transferAsset('user1', 'user2', 'coin', 100).then(response =>
{   console.log(response);
});
```

Step 7: Shutting Down Iroha docker-compose -f

docker/docker-compose.yml down This will stop and remove

the containers, freeing up resources.

Summary of Steps:

1. **Install prerequisites** like Docker, Java, and Git.
2. **Clone the Hyperledger Iroha repository** and optionally build it from source.
3. **Use Docker Compose** to set up Iroha and all its components.

4. **Interact with the blockchain** via HTTP API (curl) or SDK (like IrohaJS).
5. **Shut down the network** when done.

By following these steps, you've now set up Hyperledger Iroha and can interact with the permissioned blockchain for a simple, mobile-friendly blockchain application!

PRACTICAL 4 B.

Aim: Demonstration on interacting with NFT

Code:

Interacting with **Non-Fungible Tokens (NFTs)** using **Hyperledger Iroha** involves creating and managing digital assets that are unique and can be associated with specific metadata, such as artwork, collectibles, or other unique items.

Iroha has native support for **assets** (which can be created as NFTs). An asset in Hyperledger Iroha is essentially a unique, transferrable item that can be issued or moved between users.

Steps to Demonstrate NFT Interactions with Hyperledger Iroha:

1. **Create a New Asset (NFT)**:
    - An asset in Iroha can be anything, but for NFTs, you would generally create an asset with a **unique ID** and **metadata** to distinguish it.
2. **Transfer the NFT**:
    - Transfer NFTs between users, which could be useful for trading or ownership transfer.
3. **View NFT Information**:
    - Retrieve metadata about the NFT, such as its ownership and other associated data.
4. **Use the Hyperledger Iroha REST API** or **Iroha SDKs** (like IrohaJS) to interact with the NFTs.

Step 1: Set Up Hyperledger Iroha

Follow the steps in the previous answer to **set up Hyperledger Iroha** using Docker. If you've already done that, you can skip to the next steps.

Step 2: Create a User and Define an NFT Asset

Once the Hyperledger Iroha network is up and running, you can interact with it using the Iroha REST API. To create an NFT, you first need to:

- **Create a user**.
- **Create a unique asset** for that user, which represents the NFT.

1. Create a User (user1)

First, create a user using the API:

```
curl -X POST http://localhost:50051/create_account \
```

```
  -H "Content-Type: application/json" \
  -d '{
      "creator": "admin",
      "name": "user1",
      "password": "password123"
  }'
```

2. Create a Unique Asset (NFT)

You can define the NFT asset by creating a new asset that has a unique identifier (e.g., nft:1, nft:2, etc.).

Let's create an NFT for user1:

```
curl -X POST http://localhost:50051/create_asset \
  -H "Content-Type: application/json" \
  -d '{
      "creator": "admin",
      "account_id": "user1",
      "asset_id": "nft:1",
      "amount": "1",
      "description": "This is a unique digital collectible."
  }'
```

This will create an asset nft:1 associated with user1, with an amount of 1 (since each NFT is unique, its amount is 1).

Note: In this case, we're just creating a basic "asset" with a unique asset_id that could represent an NFT. In a real NFT use case, the metadata might include things like images, links to digital artworks, or other relevant data. We can further extend this idea by associating more data with the NFT.

3. Add Metadata to the NFT (Optional)

Iroha allows you to associate metadata with assets. You can add metadata to your NFT to give it more details (like a link to the artwork).

```
curl -X POST http://localhost:50051/add_metadata \
  -H "Content-Type: application/json" \
  -d '{
      "account_id": "user1",
      "asset_id": "nft:1",
      "metadata": {
```

```
      "creator": "ArtistName",
      "image_url": "http://example.com/artwork1.png",
      "description": "A one-of-a-kind digital collectible."
    }
}'
```

This would add metadata like the creator's name, an image URL, and a description to the NFT.

Step 3: Transfer the NFT Between Users

Once you've created an NFT, you can transfer it between users to simulate the buying, selling, or trading of NFTs.

Let's transfer the nft:1 asset from user1 to user2. **1.**

Create user2 (if not created already)

```
curl -X POST http://localhost:50051/create_account \
  -H "Content-Type: application/json" \
  -d '{
      "creator": "admin",
      "name": "user2",
      "password": "password123"
  }'
```

2. Transfer the NFT

Now, you can transfer the NFT from user1 to user2.

```
curl -X POST http://localhost:50051/transfer \
  -H "Content-Type: application/json" \
  -d '{
      "from": "user1",
      "to": "user2",
      "amount": "1",
      "asset_id": "nft:1"
  }'
```

This will transfer the nft:1 asset from user1 to user2.

Step 4: Verify the Transfer and View the NFT Information

You can check if the transfer was successful and see details about the NFT.

1. Check Account Information

After the transfer, you can check if user2 now owns the NFT (nft:1). curl

-X GET http://localhost:50051/account/user2

This will display the assets associated with user2, and you should see nft:1 listed there.

2. Retrieve Metadata for the NFT

To get the metadata for the NFT (such as the creator, image URL, and description), you can use the following command:

curl -X GET http://localhost:50051/metadata/nft:1

This should return the metadata associated with nft:1, including information like the image URL, creator, and description.

Step 5: Interact with NFTs Using IrohaJS

To interact programmatically with Iroha from a JavaScript-based frontend, you can use **IrohaJS**.

1. Install IrohaJS

You need to install the iroha-helpers package.

npm install iroha-helpers iroha-client

2. Write JavaScript Code to Create and Transfer NFTs

Here's an example of how to create and transfer an NFT programmatically:

```
const { Iroha, IrohaAPI } = require('iroha-helpers');

// Set up the connection to Iroha
const iroha = new Iroha('localhost', 50051);

// Create an account (if not already created) async
function createAccount() {
  const response = await iroha.createAccount('user1', 'password123');
console.log('Account created:', response);
}

// Create an NFT (asset)
```

```
async function createNFT() {
  const nftResponse = await iroha.createAsset('user1', 'nft:1', '1', 'This is a unique digital
collectible');
  console.log('NFT created:', nftResponse);
}

// Transfer the NFT to another account async
function transferNFT() {
  const transferResponse = await iroha.transferAsset('user1', 'user2', 'nft:1', '1');
console.log('NFT transferred:', transferResponse);
}

// Retrieve NFT metadata async
function getNFTMetadata() {
  const metadata = await iroha.getAssetMetadata('nft:1');
console.log('NFT metadata:', metadata);
}

// Run the functions createAccount().then(() => createNFT()).then(() => transferNFT()).then(() =>
getNFTMetadata());
```

Step 6: Shutting Down Iroha

After you're done interacting with Hyperledger Iroha, stop the containers:

```
docker-compose -f docker/docker-compose.yml down
```

Summary of NFT Interactions

1. **Created NFT**: We defined an asset (e.g., nft:1) as a unique NFT for user1.
2. **Added Metadata**: We added metadata like creator and image URL to make the NFT more descriptive.
3. **Transferred NFT**: We demonstrated transferring the NFT from user1 to user2.
4. **Used IrohaJS**: We showed how to interact with Iroha programmatically via JavaScript.

This simple demonstration provides a solid foundation for building more complex applications that involve NFTs, such as digital art platforms or unique asset management systems using Hyperledger Iroha.

NURTURING POTENTIAL

SAKET GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE

(Permanently Affiliated to University of Mumbai)

NAAC Accredited

Saket Vidyanagri Marg, Chinchpada Road, Katemanivali,
Kalyan (East) -421306(Mah)

Department of Information Technology

This is to certify that

Mr. RAJNIKANT ASHOK SHUKLA
Seat No. 1314284

of

M.Sc. Information Technology

Part II NEP 2020 Semester IV

has satisfactory carried out the required practical in the subject

of **DEEP LEARNING**

For the Academic year 2024 – 2025

Practical In-Charge **Head of the Department**

External Examiner

College Seal

INDEX

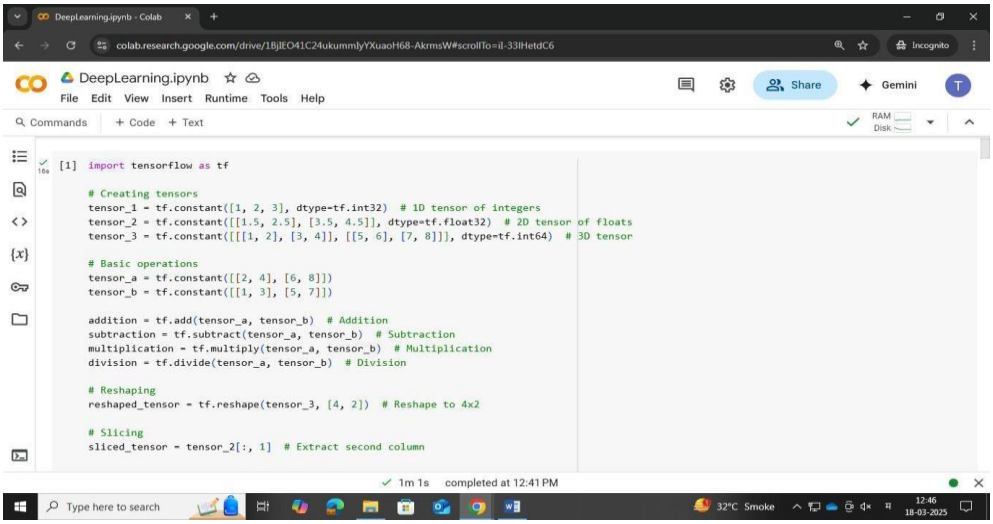
Sr. No.		Practical	Signature
1.	a.	• Create tensors with different shapes and data types. • Perform basic operations like addition, subtraction, multiplication, and division on tensors. • Reshape, slice, and index tensors to extract specific elements or sections. • Performing matrix multiplication and finding eigenvectors and eigenvalues using Tensor Flow	
	b.	Program to solve the XOR problem.	
2.	a.	• Implement a simple linear regression model using TensorFlow's lowlevel API (or tf. keras). • Train the model on a toy dataset (e.g., housing prices vs. square footage). • Visualize the loss function and the learned linear relationship. • Make predictions on new data points.	
3.	a.	Implementing deep neural network for performing binary classification task	
	b.	Using a deep feed-forward network with two hidden layers for performing multiclass classification and predicting the class.	
4.		Write a program to implement deep learning Techniques for image segmentation	

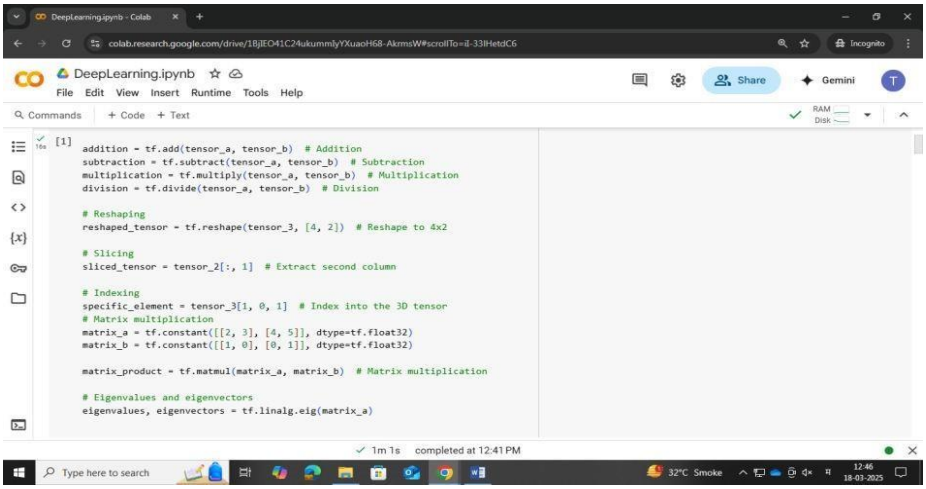
5.	Write a program to predict a caption for a sample image using LSTM.	
6.	Applying the Autoencoder algorithms for encoding realworld data	
7.	Write a program for character recognition using RNN and compare it with CNN.	
8.	Write a program to develop Autoencoders using MNIST Handwritten Digits.	
9.	Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.(google stock price).	
10.	Applying Generative Adversarial Networks for image generation and unsupervised tasks.	

PRACTICAL 1 A.

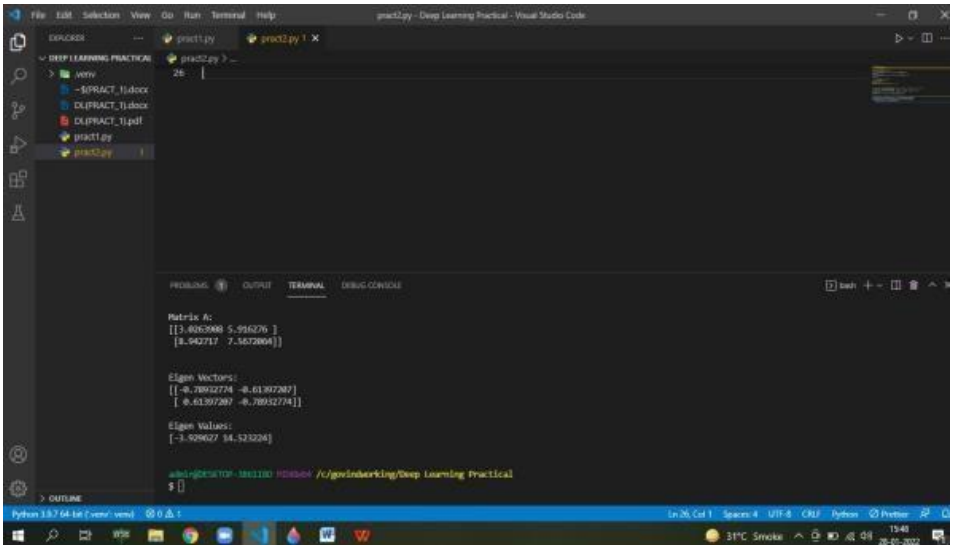
- Create tensors with different shapes and data types.
- Perform basic operations like addition, subtraction, multiplication, and division on tensors.
- Reshape, slice, and index tensors to extract specific elements or sections.
- Performing matrix multiplication and finding eigenvectors and eigenvalues using Tensor Flow **Code:**

```
import tensorflow as tf #
Creating tensors
tensor_1 = tf.constant([1, 2, 3], dtype=tf.int32)  # 1D tensor of integers
tensor_2 = tf.constant([[1.5, 2.5], [3.5, 4.5]], dtype=tf.float32)  # 2D
tensor of floats
tensor_3 = tf.constant([[[1, 2], [3, 4]], [[5, 6], [7, 8]]],
dtype=tf.int64)  # 3D tensor

# Basic operations
tensor_a = tf.constant([[2, 4], [6, 8]]) tensor_b
= tf.constant([[1, 3], [5, 7]])

addition = tf.add(tensor_a, tensor_b)  # Addition
subtraction = tf.subtract(tensor_a, tensor_b)  # Subtraction
multiplication = tf.multiply(tensor_a, tensor_b)  # Multiplication
division = tf.divide(tensor_a, tensor_b)  # Division
# Reshaping
reshaped_tensor = tf.reshape(tensor_3, [4, 2])  # Reshape to 4x2
# Slicing
sliced_tensor = tensor_2[:, 1]  # Extract second column
# Indexing
specific_element = tensor_3[1, 0, 1]  # Index into the 3D tensor
# Matrix multiplication
matrix_a = tf.constant([[2, 3], [4, 5]], dtype=tf.float32) matrix_b
= tf.constant([[1, 0], [0, 1]], dtype=tf.float32)

matrix_product = tf.matmul(matrix_a, matrix_b)  # Matrix multiplication
# Eigenvalues and eigenvectors
eigenvalues, eigenvectors = tf.linalg.eig(matrix_a)
```


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummlyYXuaoH68-AkrmsW#scrollTo=il-33IHetdC6
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
[1] import tensorflow as tf
# Creating tensors
tensor_1 = tf.constant([1, 2, 3], dtype=tf.int32) # 1D tensor of integers
tensor_2 = tf.constant([[1.5, 2.5], [3.5, 4.5]], dtype=tf.float32) # 2D tensor of floats
tensor_3 = tf.constant([[[1, 2], [3, 4]], [[5, 6], [7, 8]]], dtype=tf.int64) # 3D tensor
# Basic operations
tensor_a = tf.constant([[2, 4], [6, 8]])
tensor_b = tf.constant([[1, 3], [5, 7]])
addition = tf.add(tensor_a, tensor_b) # Addition
subtraction = tf.subtract(tensor_a, tensor_b) # Subtraction
multiplication = tf.multiply(tensor_a, tensor_b) # Multiplication
division = tf.divide(tensor_a, tensor_b) # Division
# Reshaping
reshaped_tensor = tf.reshape(tensor_3, [4, 2]) # Reshape to 4x2
# Slicing
sliced_tensor = tensor_2[:, 1] # Extract second column
1m 1s completed at 12:41 PM
Type here to search
32°C Smoke
12:46
18-03-2025


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummlyYXuaoH68-AkrmsW#scrollTo=il-33IHetdC6
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
[1]
addition = tf.add(tensor_a, tensor_b) # Addition
subtraction = tf.subtract(tensor_a, tensor_b) # Subtraction
multiplication = tf.multiply(tensor_a, tensor_b) # Multiplication
division = tf.divide(tensor_a, tensor_b) # Division
# Reshaping
reshaped_tensor = tf.reshape(tensor_3, [4, 2]) # Reshape to 4x2
# Slicing
sliced_tensor = tensor_2[:, 1] # Extract second column
# Indexing
specific_element = tensor_3[1, 0, 1] # Index into the 3D tensor
# Matrix multiplication
matrix_a = tf.constant([[2, 3], [4, 5]], dtype=tf.float32)
matrix_b = tf.constant([[1, 0], [0, 1]], dtype=tf.float32)
matrix_product = tf.matmul(matrix_a, matrix_b) # Matrix multiplication
# Eigenvalues and eigenvectors
eigenvalues, eigenvectors = tf.linalg.eig(matrix_a)
1m 1s completed at 12:41 PM
Type here to search
32°C Smoke
12:46
18-03-2025

Output:

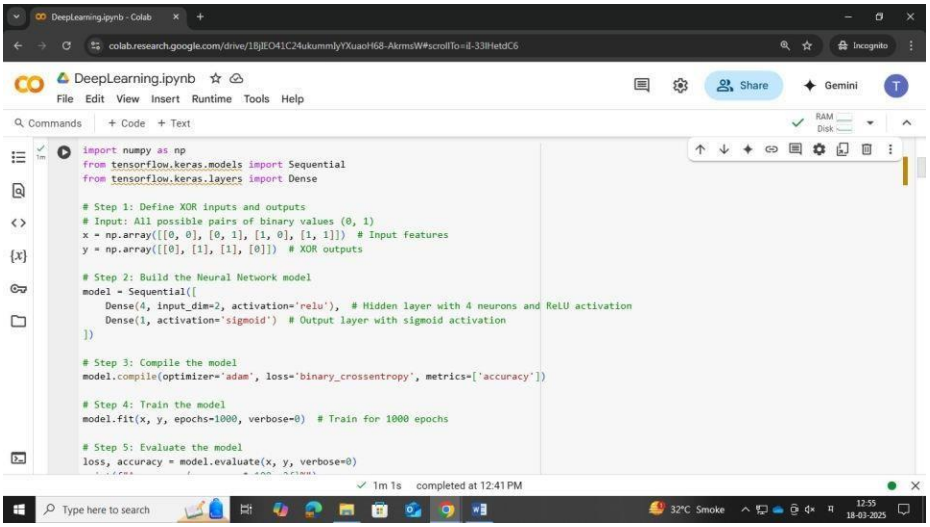

Matrix A:
[[3.0263888 5.916276]
[3.942717 7.5672004]]
Eigen Vectors:
[[-0.78932774 -0.61397207]
[0.61397207 -0.78932774]]
Eigen Values:
[-3.920027 14.523224]
$

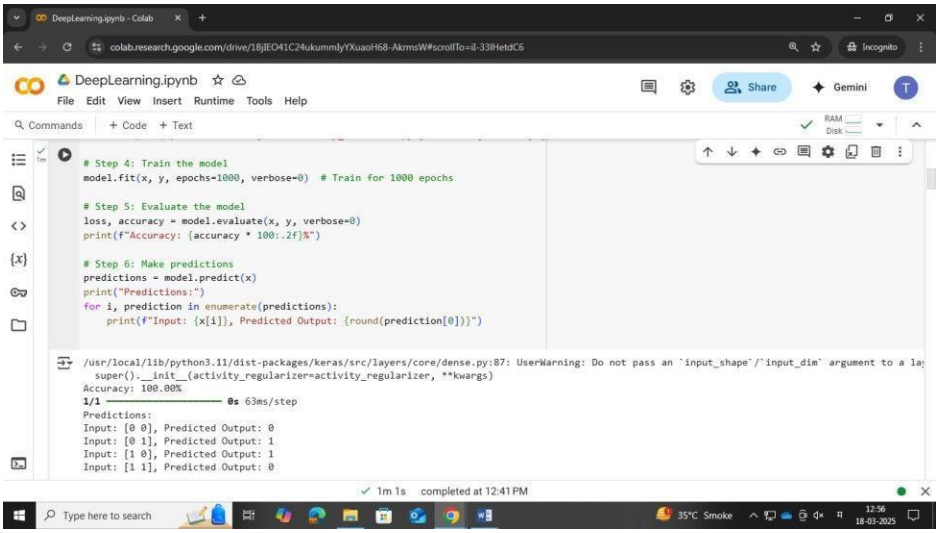
PRACTICAL 1 B.

Program to solve the XOR problem.

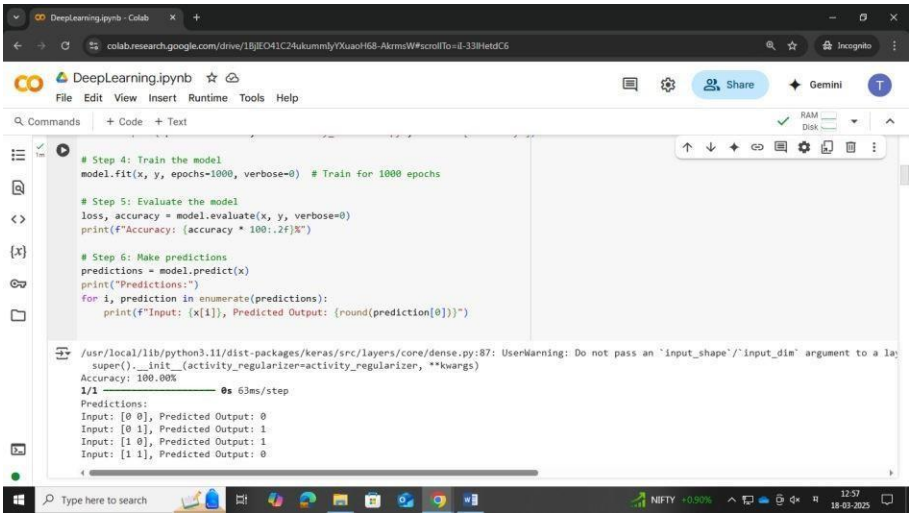
Code:

```
import numpy as np
from tensorflow.keras.models import Sequential from
tensorflow.keras.layers import Dense

# Step 1: Define XOR inputs and outputs
# Input: All possible pairs of binary values (0, 1)
x = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])  # Input features y
= np.array([[0], [1], [1], [0]])  # XOR outputs

# Step 2: Build the Neural Network model model
= Sequential([
    Dense(4, input_dim=2, activation='relu'),  # Hidden layer with 4
neurons and ReLU activation
    Dense(1,   activation='sigmoid')      #   Output   layer   with   sigmoid
activation ])

# Step 3: Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Step 4: Train the model
model.fit(x, y, epochs=1000, verbose=0)  # Train for 1000 epochs
# Step 5: Evaluate the model
loss, accuracy = model.evaluate(x, y, verbose=0) print(f"Accuracy:
{accuracy * 100:.2f}%")

# Step 6: Make predictions predictions =
model.predict(x) print("Predictions:") for
i, prediction in enumerate(predictions):
    print(f"Input: {x[i]}, Predicted Output: {round(prediction[0])}")
```


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummlyYXuaoH68-AkrmsW#scrollTo=iI-33IHetdC6
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
# Step 1: Define XOR inputs and outputs
# Input: All possible pairs of binary values (0, 1)
x = np.array([[0, 0], [0, 1], [1, 0], [1, 1]]) # Input features
y = np.array([[0], [1], [1], [0]]) # XOR outputs
# Step 2: Build the Neural Network model
model = Sequential([
Dense(4, input_dim=2, activation='relu'), # Hidden layer with 4 neurons and ReLU activation
Dense(1, activation='sigmoid') # Output layer with sigmoid activation
])
# Step 3: Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Step 4: Train the model
model.fit(x, y, epochs=1000, verbose=0) # Train for 1000 epochs
Step 5: Evaluate the model
loss, accuracy = model.evaluate(x, y, verbose=0)
1m 1s completed at 12:41 PM
Type here to search
32°C Smoke
12:55
18-03-2025


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummlyYXuaoH68-AkrmsW#scrollTo=iI-33IHetdC6
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
# Step 4: Train the model
model.fit(x, y, epochs=1000, verbose=0) # Train for 1000 epochs
Step 5: Evaluate the model
loss, accuracy = model.evaluate(x, y, verbose=0)
print(f"Accuracy: {accuracy * 100:.2f}%")
# Step 6: Make predictions
predictions = model.predict(x)
print("Predictions:")
for i, prediction in enumerate(predictions):
print(f"Input: {x[i]}, Predicted Output: {round(prediction[0])}")
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a la
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Accuracy: 100.00%
1/1 0s 63ms/step
Predictions:
Input: [0 0], Predicted Output: 0
Input: [0 1], Predicted Output: 1
Input: [1 0], Predicted Output: 1
Input: [1 1], Predicted Output: 0
1m 1s completed at 12:41 PM
Type here to search
35°C Smoke
12:56
18-03-2025

Output:


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummlyYXuaoH68-AkrmsW#scrollTo=iI-33IHetdC6
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
# Step 4: Train the model
model.fit(x, y, epochs=1000, verbose=0) # Train for 1000 epochs
Step 5: Evaluate the model
loss, accuracy = model.evaluate(x, y, verbose=0)
print(f"Accuracy: {accuracy * 100:.2f}%")
# Step 6: Make predictions
predictions = model.predict(x)
print("Predictions:")
for i, prediction in enumerate(predictions):
print(f"Input: {x[i]}, Predicted Output: {round(prediction[0])}")
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a la
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Accuracy: 100.00%
1/1 0s 63ms/step
Predictions:
Input: [0 0], Predicted Output: 0
Input: [0 1], Predicted Output: 1
Input: [1 0], Predicted Output: 1
Input: [1 1], Predicted Output: 0
Type here to search
NIFTY +0.90%
12:57
18-03-2025

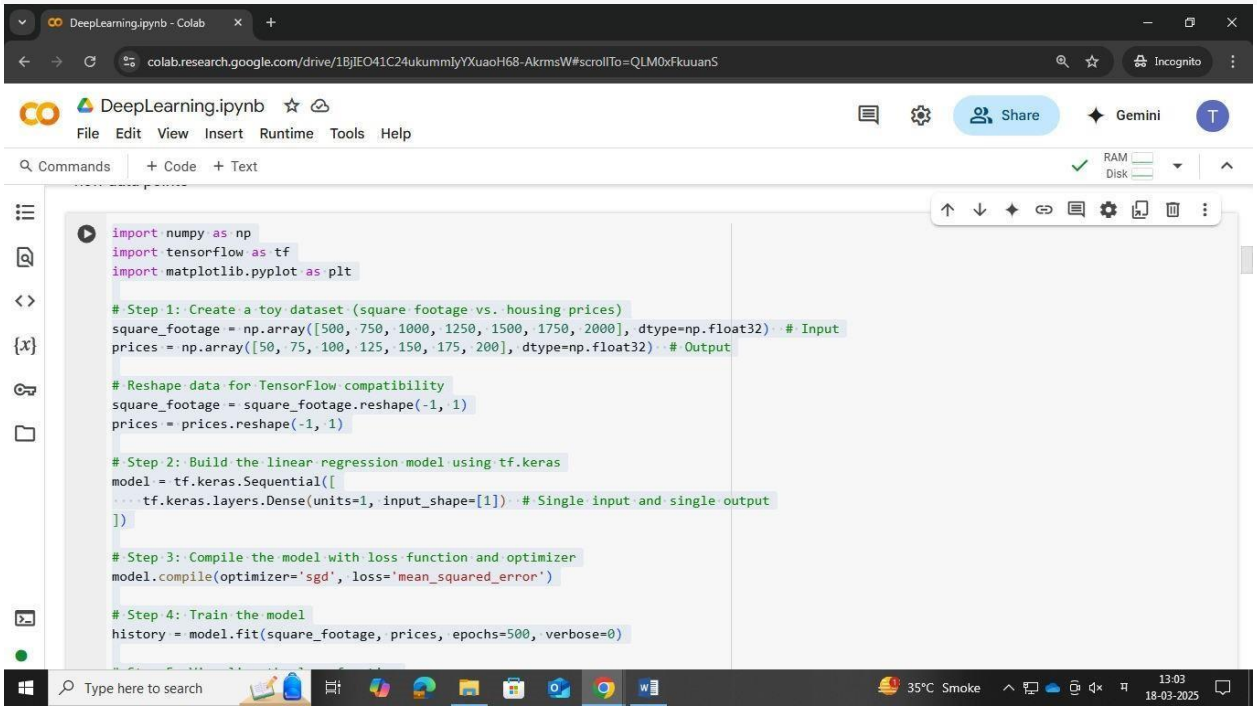
PRACTICAL 2 A.

- Implement a simple linear regression model using TensorFlow's lowlevel API (or tf. keras).
- Train the model on a toy dataset (e.g., housing prices vs. square footage).
- Visualize the loss function and the learned linear relationship.
- Make predictions on new data points.

Code:

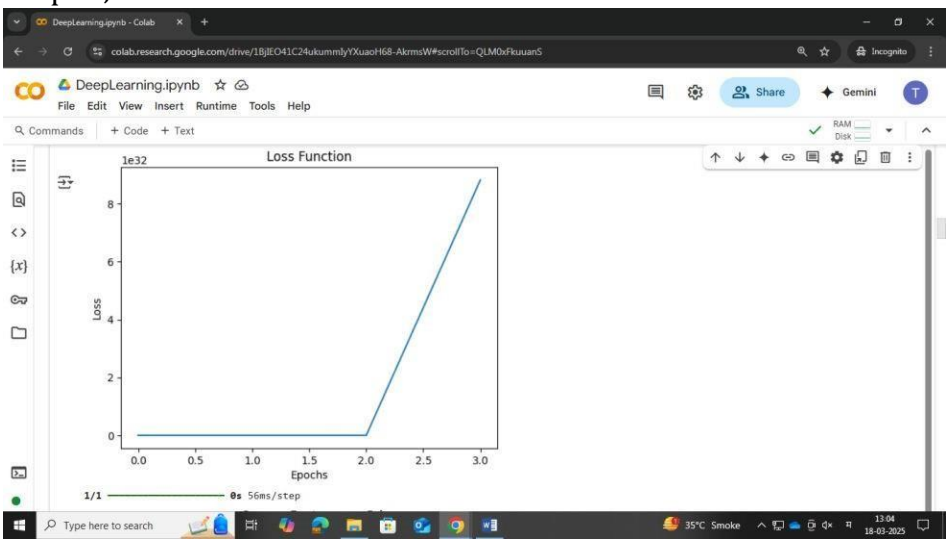
```
import numpy as np import
tensorflow as tf
import matplotlib.pyplot as plt

# Step 1: Create a toy dataset (square footage vs. housing prices)
square_footage = np.array([500, 750, 1000, 1250, 1500, 1750, 2000],
dtype=np.float32)  # Input
prices = np.array([50, 75, 100, 125, 150, 175, 200], dtype=np.float32)  #
Output

# Reshape data for TensorFlow compatibility
square_footage = square_footage.reshape(-1, 1) prices
= prices.reshape(-1, 1)

# Step 2: Build the linear regression model using tf.keras model
= tf.keras.Sequential([
    tf.keras.layers.Dense(units=1, input_shape=[1])    # Single input and
single output
])
# Step 3: Compile the model with loss function and optimizer
model.compile(optimizer='sgd', loss='mean_squared_error')
# Step 4: Train the model
history = model.fit(square_footage, prices, epochs=500, verbose=0)
# Step 5: Visualize the loss function
plt.plot(history.history['loss'])
plt.title('Loss Function')
plt.xlabel('Epochs')
plt.ylabel('Loss') plt.show()
```

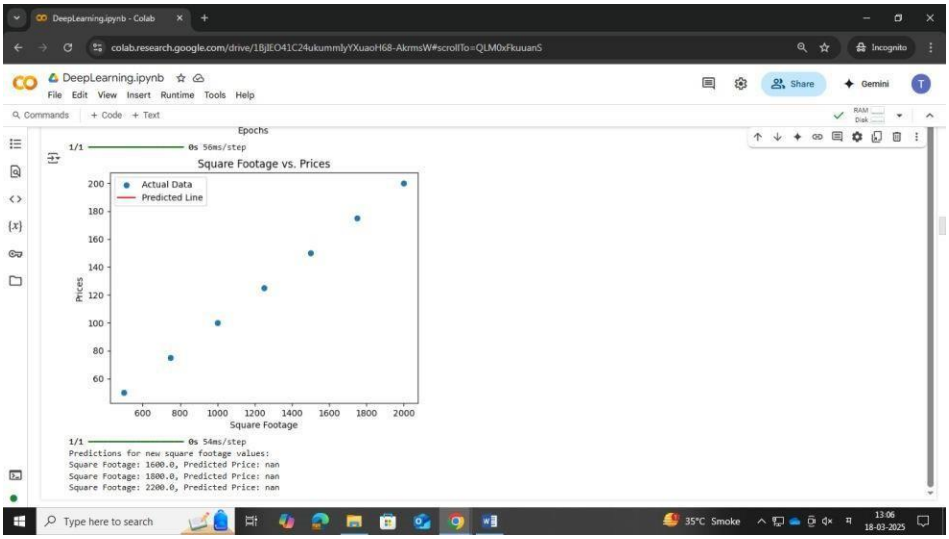
```
# Step 6: Visualize the learned linear relationship predicted_prices
= model.predict(square_footage)

plt.scatter(square_footage, prices, label='Actual Data')
plt.plot(square_footage, predicted_prices, color='red', label='Predicted
Line')
plt.title('Square Footage vs. Prices')
plt.xlabel('Square Footage')
plt.ylabel('Prices') plt.legend()
plt.show()

# Step 7: Make predictions on new data points new_square_footage
= np.array([1600, 1800, 2200], dtype=np.float32).reshape(-1, 1)
new_predictions = model.predict(new_square_footage)
 print("Predictions for new square footage values:")
for i, sqft in enumerate(new_square_footage):
print(f"Square Footage: {sqft[0]}, Predicted Price:
{new_predictions[i][0]:.2f}")
```



```
history = model.fit(square_footage, prices, epochs=500, verbose=0)

# Step 5: Visualize the loss function
plt.plot(history.history['loss'])
plt.title('Loss Function')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()

# Step 6: Visualize the learned linear relationship
predicted_prices = model.predict(square_footage)

plt.scatter(square_footage, prices, label='Actual Data')
plt.plot(square_footage, predicted_prices, color='red', label='Predicted Line')
plt.title('Square Footage vs. Prices')
plt.xlabel('Square Footage')
plt.ylabel('Prices')
plt.legend()
plt.show()

# Step 7: Make predictions on new data points
new_square_footage = np.array([1600, 1800, 2200], dtype=np.float32).reshape(-1, 1)
new_predictions = model.predict(new_square_footage)

print("Predictions for new square footage values:")
```

Output ;





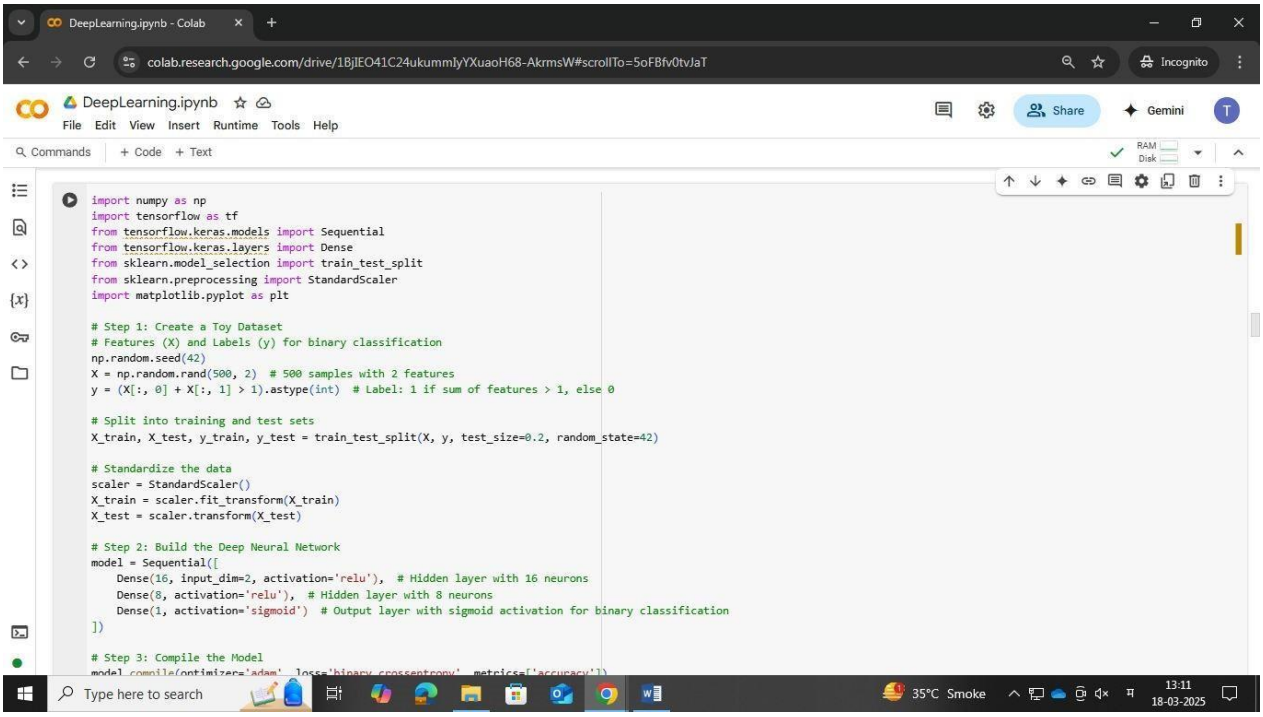
PRACTICAL 3 A.

Aim: Implementing deep neural network for performing binary classification task

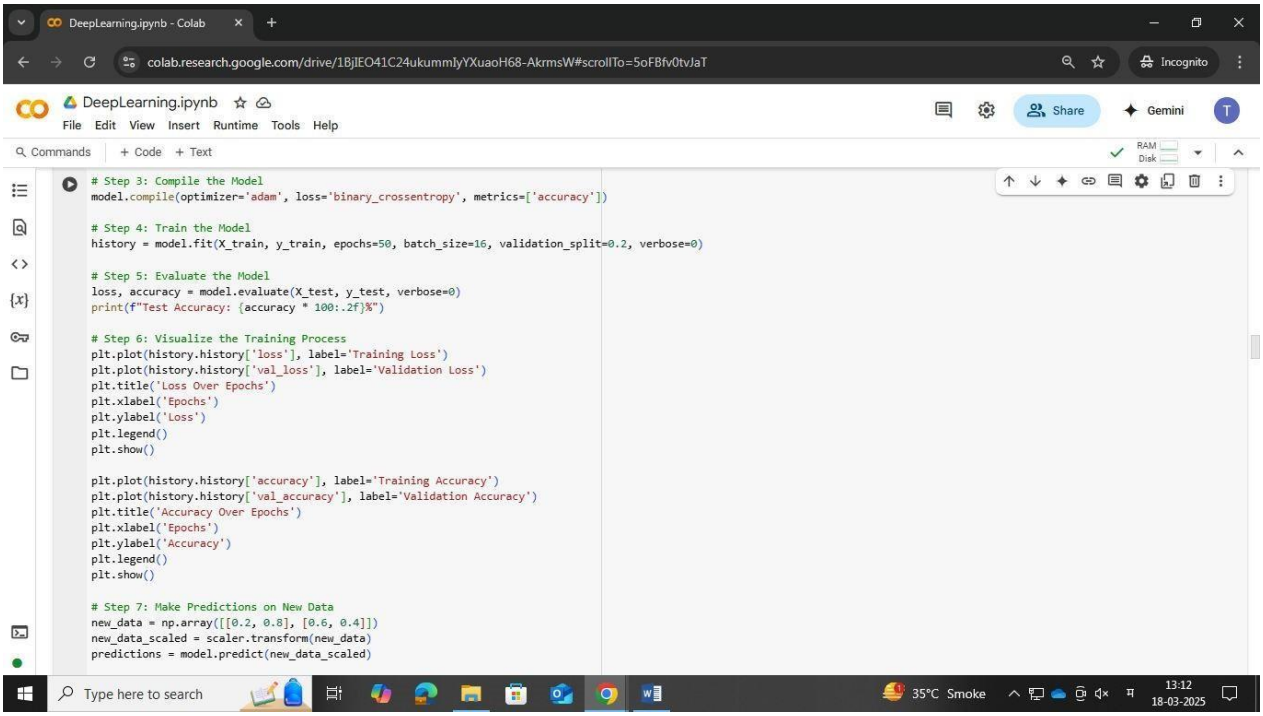
Code:

```
import numpy as np import

tensorflow as tf

from tensorflow.keras.models import Sequential from
tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler import
matplotlib.pyplot as plt


# Step 1: Create a Toy Dataset
# Features (X) and Labels (y) for binary classification np.random.seed(42)
X = np.random.rand(500, 2)  # 500 samples with 2 features
y = (X[:, 0] + X[:, 1] > 1).astype(int)  # Label: 1 if sum of features >
1, else 0


# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# Standardize the data scaler
= StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


# Step 2: Build the Deep Neural Network model
= Sequential([
    Dense(16, input_dim=2, activation='relu'),    # Hidden layer with 16
neurons
    Dense(8, activation='relu'),  # Hidden layer with 8 neurons
    Dense(1, activation='sigmoid')  # Output layer with sigmoid activation
for binary classification
])


# Step 3: Compile the Model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])


# Step 4: Train the Model
```

```
history = model.fit(X_train, y_train, epochs=50, batch_size=16,
validation_split=0.2, verbose=0)

# Step 5: Evaluate the Model
loss, accuracy = model.evaluate(X_test, y_test, verbose=0) print(f"Test
Accuracy: {accuracy * 100:.2f}%")

# Step 6: Visualize the Training Process
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Over Epochs') plt.xlabel('Epochs')
plt.ylabel('Loss') plt.legend() plt.show()
 plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy Over Epochs') plt.xlabel('Epochs')
plt.ylabel('Accuracy') plt.legend() plt.show()

# Step 7: Make Predictions on New Data new_data
= np.array([[0.2, 0.8], [0.6, 0.4]])
new_data_scaled = scaler.transform(new_data)
predictions = model.predict(new_data_scaled)
 print("Predictions on new data:") for
i, pred in enumerate(predictions):
    print(f"Input: {new_data[i]}, Predicted Class: {int(pred > 0.5)}")
```



```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Step 1: Create a Toy Dataset
# Features (X) and Labels (y) for binary classification
np.random.seed(42)
X = np.random.rand(500, 2)  # 500 samples with 2 features
y = (X[:, 0] + X[:, 1] > 1).astype(int)  # Label: 1 if sum of features > 1, else 0

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 2: Build the Deep Neural Network
model = Sequential([
    Dense(16, input_dim=2, activation='relu'),  # Hidden layer with 16 neurons
    Dense(8, activation='relu'),  # Hidden layer with 8 neurons
    Dense(1, activation='sigmoid')  # Output layer with sigmoid activation for binary classification
])

# Step 3: Compile the Model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

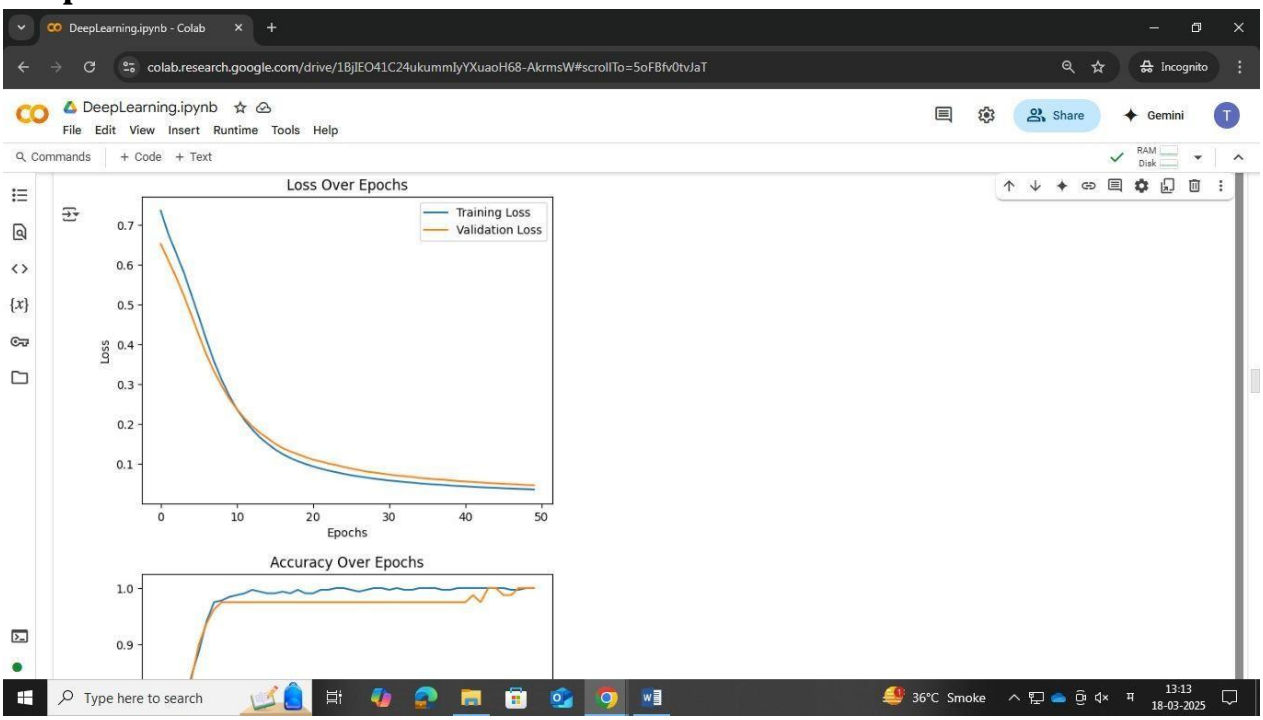


```
# Step 3: Compile the Model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Step 4: Train the Model
history = model.fit(X_train, y_train, epochs=50, batch_size=16, validation_split=0.2, verbose=0)

# Step 5: Evaluate the Model
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Accuracy: {accuracy * 100:.2f}%")

# Step 6: Visualize the Training Process
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Step 7: Make Predictions on New Data
new_data = np.array([[0.2, 0.8], [0.6, 0.4]])
new_data_scaled = scaler.transform(new_data)
predictions = model.predict(new_data_scaled)
```

```
# Step 7: Make Predictions on New Data
new_data = np.array([[0.2, 0.8], [0.6, 0.4]])
new_data_scaled = scaler.transform(new_data)
predictions = model.predict(new_data_scaled)

print("Predictions on new data:")
for i, pred in enumerate(predictions):
    print(f"Input: {new_data[i]}, Predicted Class: {int(pred > 0.5)}")
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models,
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Test Accuracy: 100.00%
```

Output:


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummIyYXuaoH68-AkrmsW#scrollTo=5oFBfv0tvJaT
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
Loss Over Epochs
Training Loss
Validation Loss
Loss
Epochs
Accuracy Over Epochs
Type here to search
36°C Smoke
13:13
18-03-2025


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummIyYXuaoH68-AkrmsW#scrollTo=5oFBfv0tvJaT
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
Accuracy Over Epochs
Training Accuracy
Validation Accuracy
Accuracy
Epochs
1/1 0s 79ms/step
Predictions on new data:
Input: [0.2 0.8], Predicted Class: 0
Input: [0.6 0.4], Predicted Class: 0
<ipython-input-14-f64110c0dcf4>:64: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you extract a single eleme
print(f"Input: {new_data[i]}, Predicted Class: {int(pred > 0.5)}")
Type here to search
36°C Smoke
13:13
18-03-2025

PRACTICAL 3 B.

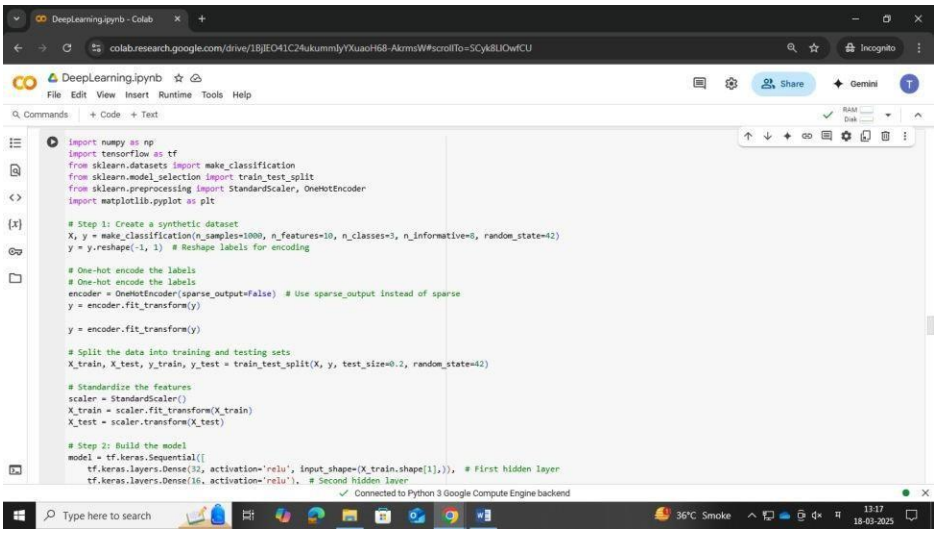
Aim: Using a deep feed-forward network with two hidden layers for performing multiclass classification and predicting the class.

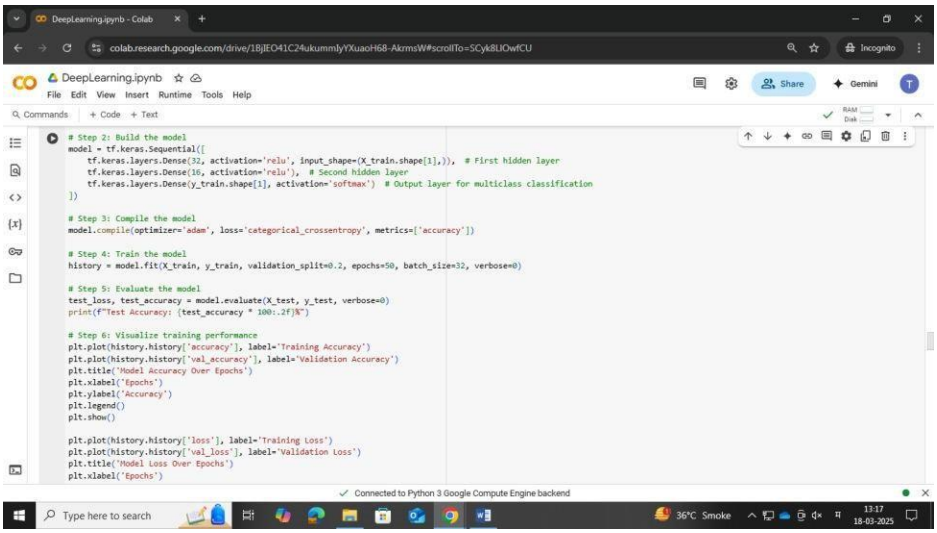
Code:

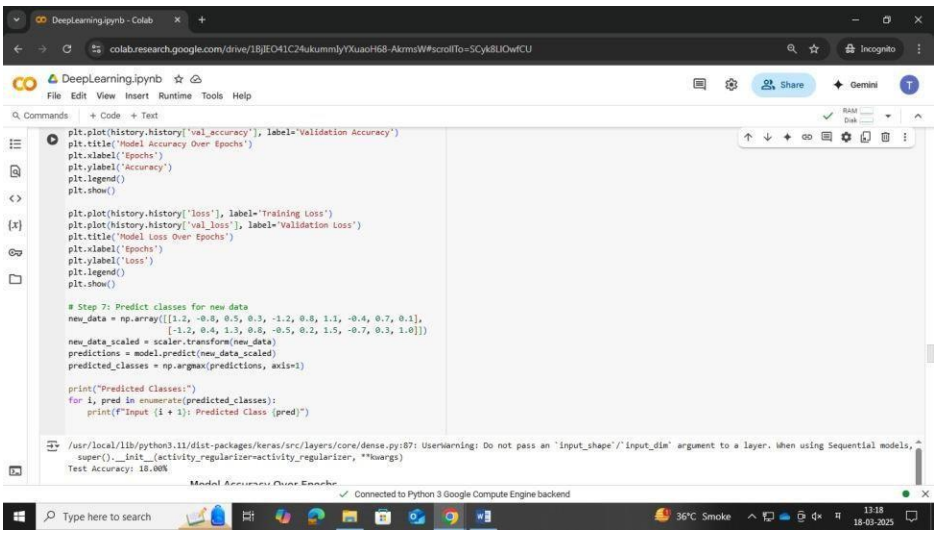
```
import numpy as np import
tensorflow as tf
from sklearn.datasets import make_classification from
sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder import
matplotlib.pyplot as plt

# Step 1: Create a synthetic dataset
X, y = make_classification(n_samples=1000, n_features=10, n_classes=3,
n_informative=8, random_state=42)
y = y.reshape(-1, 1)  # Reshape labels for encoding

# One-hot encode the labels #
One-hot encode the labels
encoder = OneHotEncoder(sparse_output=False)  # Use sparse_output instead
of sparse
y = encoder.fit_transform(y)

y = encoder.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize the features scaler
= StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 2: Build the model model
= tf.keras.Sequential([
    tf.keras.layers.Dense(32, activation='relu',
input_shape=(X_train.shape[1],)),  # First hidden layer
    tf.keras.layers.Dense(16, activation='relu'),   # Second hidden layer
tf.keras.layers.Dense(y_train.shape[1], activation='softmax')  #
Output layer for multiclass classification
])
```

```
# Step 3: Compile the model
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Step 4: Train the model
history = model.fit(X_train, y_train, validation_split=0.2, epochs=50,
batch_size=32, verbose=0)

# Step 5: Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")

# Step 6: Visualize training performance
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy Over Epochs') plt.xlabel('Epochs')
plt.ylabel('Accuracy') plt.legend() plt.show()
 plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss Over Epochs') plt.xlabel('Epochs')
plt.ylabel('Loss') plt.legend() plt.show()

# Step 7: Predict classes for new data
new_data = np.array([[1.2, -0.8, 0.5, 0.3, -1.2, 0.8, 1.1, -0.4, 0.7,
0.1],
                     [-1.2, 0.4, 1.3, 0.8, -0.5, 0.2, 1.5, -0.7, 0.3,
1.0]])
new_data_scaled = scaler.transform(new_data) predictions
= model.predict(new_data_scaled) predicted_classes =
np.argmax(predictions, axis=1)
 print("Predicted Classes:") for i, pred in
enumerate(predicted_classes):
    print(f"Input {i + 1}: Predicted Class {pred}")
```

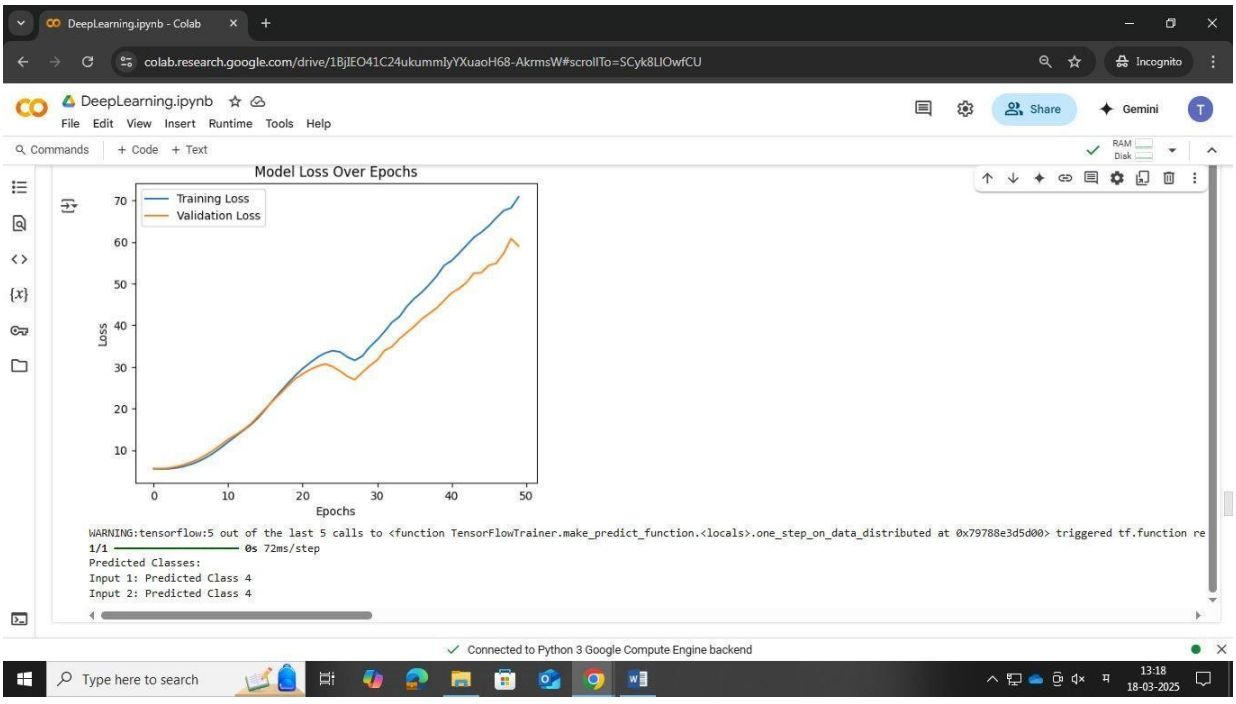

DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummlyYXuaoH68-AkrmsW#scrollTo=SCyk8LIOwfCU
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
import numpy as np
import tensorflow as tf
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
import matplotlib.pyplot as plt
# Step 1: Create a synthetic dataset
X, y = make_classification(n_samples=1000, n_features=10, n_classes=3, n_informative=8, random_state=42)
y = y.reshape(-1, 1) # Reshape labels for encoding
# One-hot encode the labels
# One-hot encode the labels
encoder = OneHotEncoder(sparse_output=False) # Use sparse_output instead of sparse
y = encoder.fit_transform(y)
y = encoder.fit_transform(y)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Step 2: Build the model
model = tf.keras.Sequential([
tf.keras.layers.Dense(32, activation='relu', input_shape=(X_train.shape[1],)), # First hidden layer
tf.keras.layers.Dense(16, activation='relu'), # Second hidden layer
Connected to Python 3 Google Compute Engine backend
Type here to search
36°C Smoke
13:17
18-03-2025


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummlyYXuaoH68-AkrmsW#scrollTo=SCyk8LIOwfCU
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
# Step 2: Build the model
model = tf.keras.Sequential([
tf.keras.layers.Dense(32, activation='relu', input_shape=(X_train.shape[1],)), # First hidden layer
tf.keras.layers.Dense(16, activation='relu'), # Second hidden layer
tf.keras.layers.Dense(y_train.shape[1], activation='softmax') # Output layer for multiclass classification
])
# Step 3: Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Step 4: Train the model
history = model.fit(X_train, y_train, validation_split=0.2, epochs=50, batch_size=32, verbose=0)
# Step 5: Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
# Step 6: Visualize training performance
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss Over Epochs')
plt.xlabel('Epochs')
Connected to Python 3 Google Compute Engine backend
Type here to search
36°C Smoke
13:17
18-03-2025


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummlyYXuaoH68-AkrmsW#scrollTo=SCyk8LIOwfCU
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
# Step 7: Predict classes for new data
new_data = np.array([[1.2, -0.8, 0.5, 0.3, -1.2, 0.8, 1.1, -0.4, 0.7, 0.1],
[-1.2, 0.4, 1.3, 0.8, -0.5, 0.2, 1.5, -0.7, 0.3, 1.0]])
new_data_scaled = scaler.transform(new_data)
predictions = model.predict(new_data_scaled)
predicted_classes = np.argmax(predictions, axis=1)
print("Predicted Classes:")
for i, pred in enumerate(predicted_classes):
print(f"Input {i + 1}: Predicted Class {pred}")
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models,
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Test Accuracy: 18.00%
Connected to Python 3 Google Compute Engine backend
Type here to search
36°C Smoke
13:18
18-03-2025

Output:


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummIyYXuaoH68-AkrmsW#scrollTo=SCyk8LIOwfCU
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models,
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Test Accuracy: 18.00%
Model Accuracy Over Epochs
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Model Loss Over Epochs
Connected to Python 3 Google Compute Engine backend
Type here to search
36°C Smoke
13:18
18-03-2025


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummIyYXuaoH68-AkrmsW#scrollTo=SCyk8LIOwfCU
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
Model Loss Over Epochs
Training Loss
Validation Loss
Loss
Epochs
WARNING:tensorflow:5 out of the last 5 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x79788e3d5d00> triggered tf.function re
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step
Predicted Classes:
Input 1: Predicted Class 4
Input 2: Predicted Class 4
Connected to Python 3 Google Compute Engine backend
Type here to search
13:18
18-03-2025

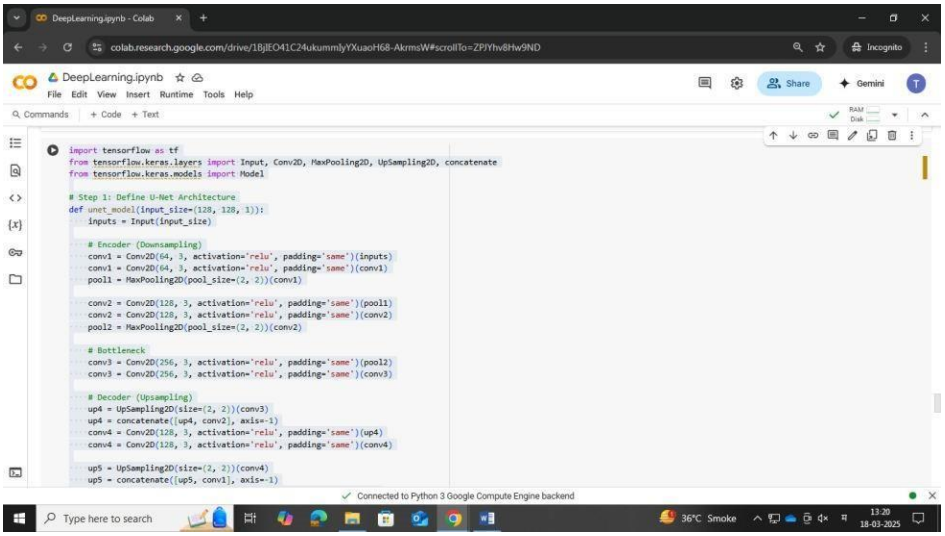
PRACTICAL 4.

Aim: Write a program to implement deep learning Techniques for image segmentation

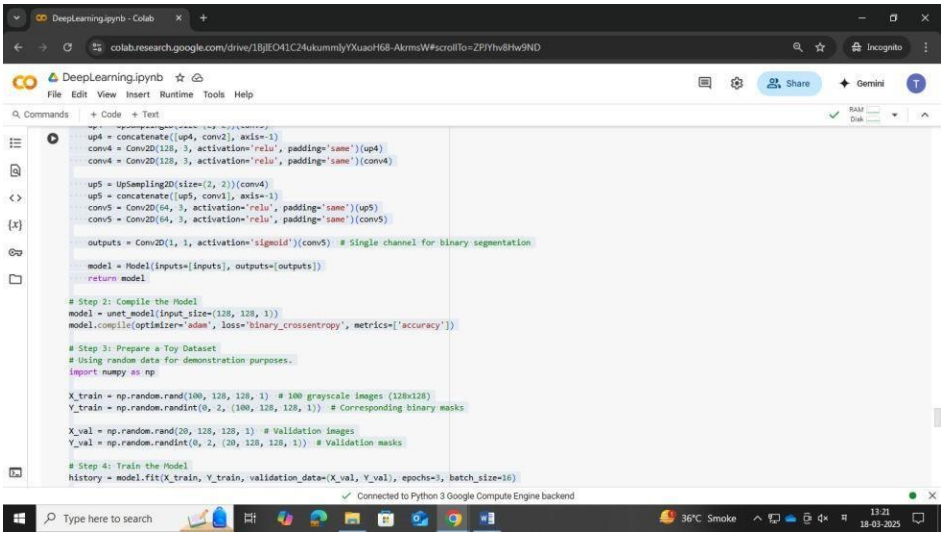
Code:

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D, concatenate
from tensorflow.keras.models import Model

# Step 1: Define U-Net Architecture def
unet_model(input_size=(128, 128, 1)):
    inputs = Input(input_size)

    # Encoder (Downsampling)
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(inputs)
conv1 = Conv2D(64, 3, activation='relu', padding='same')(conv1)     pool1
= MaxPooling2D(pool_size=(2, 2))(conv1)
     conv2 = Conv2D(128, 3, activation='relu', padding='same')(pool1)
conv2  =  Conv2D(128,  3,  activation='relu',  padding='same')(conv2)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    # Bottleneck
    conv3 = Conv2D(256, 3, activation='relu', padding='same')(pool2)
conv3 = Conv2D(256, 3, activation='relu', padding='same')(conv3)
    # Decoder (Upsampling)
    up4 = UpSampling2D(size=(2, 2))(conv3)
up4 = concatenate([up4, conv2], axis=-1)
    conv4 = Conv2D(128, 3, activation='relu', padding='same')(up4)
conv4 = Conv2D(128, 3, activation='relu', padding='same')(conv4)
     up5 = UpSampling2D(size=(2, 2))(conv4)
up5 = concatenate([up5, conv1], axis=-1)
    conv5 = Conv2D(64, 3, activation='relu', padding='same')(up5)
conv5 = Conv2D(64, 3, activation='relu', padding='same')(conv5)

    outputs = Conv2D(1, 1, activation='sigmoid')(conv5)  # Single channel
for binary segmentation

    model = Model(inputs=[inputs], outputs=[outputs])
return model
```

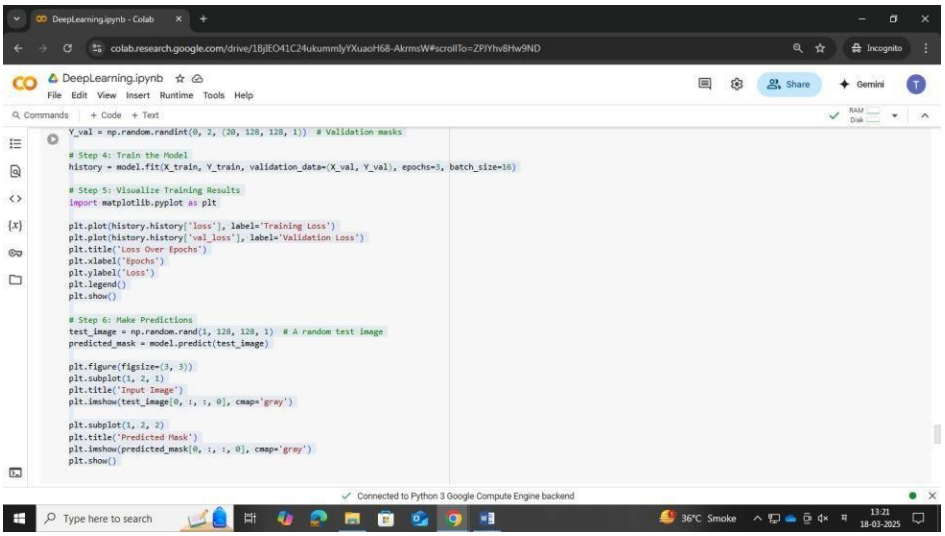
```
# Step 2: Compile the Model
model = unet_model(input_size=(128, 128, 1))
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Step 3: Prepare a Toy Dataset
# Using random data for demonstration purposes. import
numpy as np

X_train = np.random.rand(100, 128, 128, 1)  # 100 grayscale images
(128x128)
Y_train = np.random.randint(0, 2, (100, 128, 128, 1))  # Corresponding
binary masks

X_val = np.random.rand(20, 128, 128, 1)  # Validation images
Y_val = np.random.randint(0, 2, (20, 128, 128, 1))  # Validation masks
# Step 4: Train the Model
history = model.fit(X_train, Y_train, validation_data=(X_val, Y_val),
epochs=3, batch_size=16)

# Step 5: Visualize Training Results import
matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Over Epochs') plt.xlabel('Epochs')
plt.ylabel('Loss') plt.legend() plt.show()
# Step 6: Make Predictions
test_image = np.random.rand(1, 128, 128, 1)  # A random test image
predicted_mask = model.predict(test_image)
 plt.figure(figsize=(3,
3)) plt.subplot(1, 2, 1)
plt.title('Input Image')
plt.imshow(test_image[0, :, :, 0], cmap='gray')

plt.subplot(1, 2, 2) plt.title('Predicted
Mask')
plt.imshow(predicted_mask[0, :, :, 0], cmap='gray') plt.show()
```



```
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D, concatenate
from tensorflow.keras.models import Model

# Step 1: Define U-Net Architecture
def unet_model(input_size=(128, 128, 1)):
    inputs = Input(input_size)

    # Encoder (Downsampling)
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(inputs)
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(128, 3, activation='relu', padding='same')(pool1)
    conv2 = Conv2D(128, 3, activation='relu', padding='same')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    # Bottleneck
    conv3 = Conv2D(256, 3, activation='relu', padding='same')(pool2)
    conv3 = Conv2D(256, 3, activation='relu', padding='same')(conv3)

    # Decoder (Upsampling)
    up4 = UpSampling2D(size=(2, 2))(conv3)
    up4 = concatenate([up4, conv2], axis=-1)
    conv4 = Conv2D(128, 3, activation='relu', padding='same')(up4)
    conv4 = Conv2D(128, 3, activation='relu', padding='same')(conv4)

    up5 = UpSampling2D(size=(2, 2))(conv4)
    up5 = concatenate([up5, conv1], axis=-1)
```

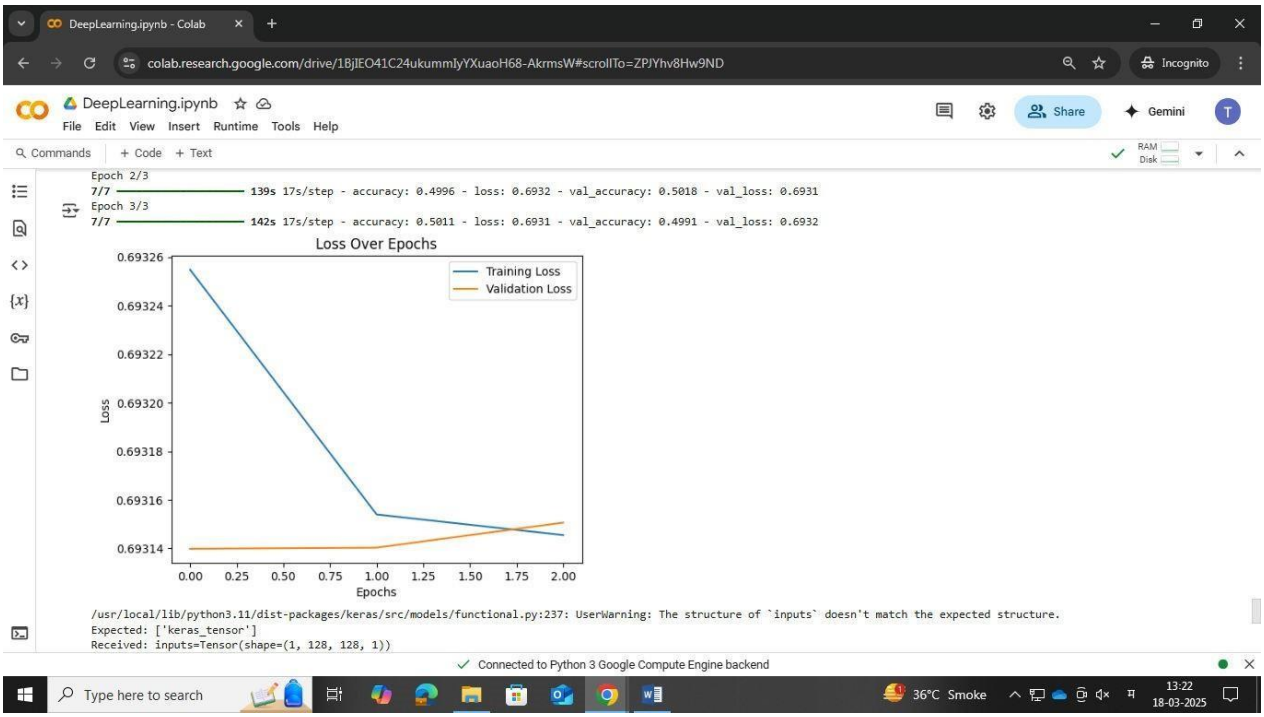


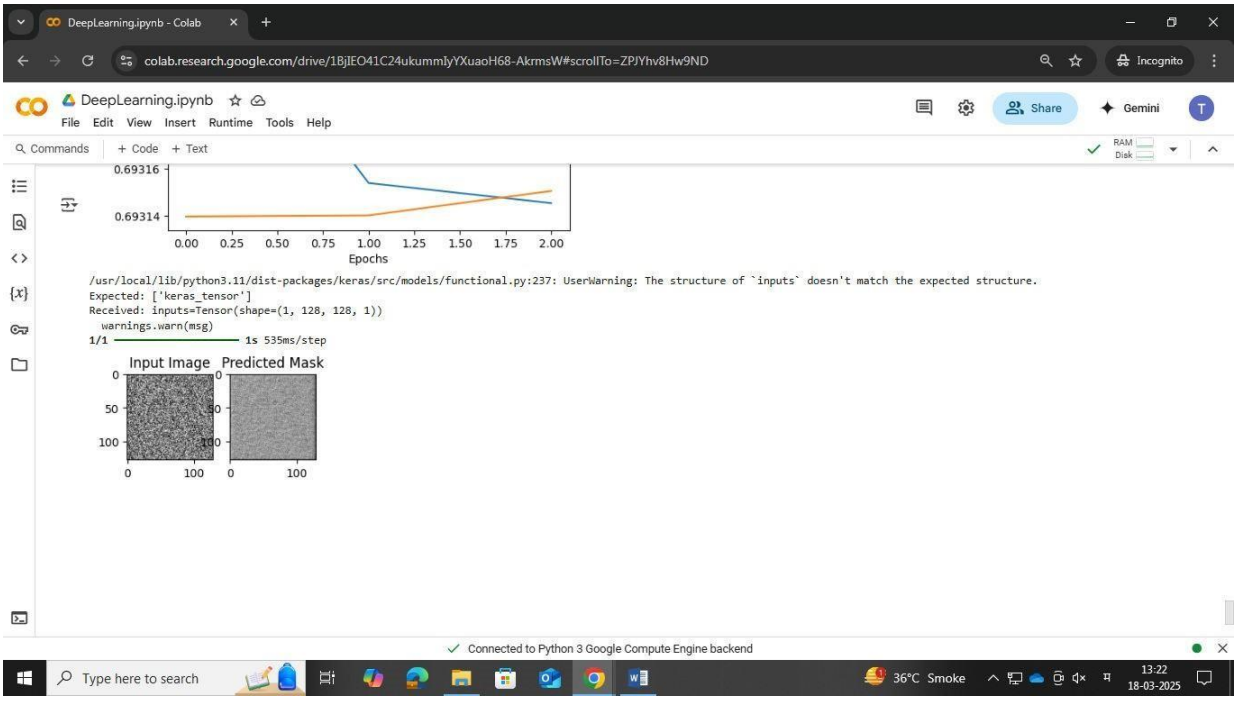
```
    up4 = concatenate([up4, conv2], axis=-1)
    conv4 = Conv2D(128, 3, activation='relu', padding='same')(up4)
    conv4 = Conv2D(128, 3, activation='relu', padding='same')(conv4)

    up5 = UpSampling2D(size=(2, 2))(conv4)
    up5 = concatenate([up5, conv1], axis=-1)
    conv5 = Conv2D(64, 3, activation='relu', padding='same')(up5)
    conv5 = Conv2D(64, 3, activation='relu', padding='same')(conv5)

    outputs = Conv2D(1, 1, activation='sigmoid')(conv5)  # Single channel for binary segmentation

    model = Model(inputs=[inputs], outputs=[outputs])
    return model

# Step 2: Compile the Model
model = unet_model(input_size=(128, 128, 1))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Step 3: Prepare a Toy Dataset
# Using random data for demonstration purposes.
import numpy as np

X_train = np.random.rand(100, 128, 128, 1)  # 100 grayscale images (128x128)
Y_train = np.random.randint(0, 2, (100, 128, 128, 1))  # Corresponding binary masks

X_val = np.random.rand(20, 128, 128, 1)  # Validation images
Y_val = np.random.randint(0, 2, (20, 128, 128, 1))  # Validation masks

# Step 4: Train the Model
history = model.fit(X_train, Y_train, validation_data=(X_val, Y_val), epochs=3, batch_size=16)
```



```
Y_val = np.random.randint(0, 2, (20, 128, 128, 1))  # Validation masks

# Step 4: Train the Model
history = model.fit(X_train, Y_train, validation_data=(X_val, Y_val), epochs=3, batch_size=16)

# Step 5: Visualize Training Results
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Step 6: Make Predictions
test_image = np.random.rand(1, 128, 128, 1)  # A random test image
predicted_mask = model.predict(test_image)

plt.figure(figsize=(3, 3))
plt.subplot(1, 2, 1)
plt.title('Input Image')
plt.imshow(test_image[0, :, :, 0], cmap='gray')

plt.subplot(1, 2, 2)
plt.title('Predicted Mask')
plt.imshow(predicted_mask[0, :, :, 0], cmap='gray')
plt.show()
```

Output:


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummlyYXuaoH68-AkrmsW#scrollTo=ZPJYhv8Hw9ND
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
Epoch 2/3
7/7 139s 17s/step - accuracy: 0.4996 - loss: 0.6932 - val_accuracy: 0.5018 - val_loss: 0.6931
Epoch 3/3
7/7 142s 17s/step - accuracy: 0.5011 - loss: 0.6931 - val_accuracy: 0.4991 - val_loss: 0.6932
Loss Over Epochs
Training Loss
Validation Loss
Loss
Epochs
/usr/local/lib/python3.11/dist-packages/keras/src/models/functional.py:237: UserWarning: The structure of `inputs` doesn't match the expected structure.
Expected: ['keras_tensor']
Received: inputs=Tensor(shape=(1, 128, 128, 1))
Connected to Python 3 Google Compute Engine backend
Type here to search
36°C Smoke
13:22
18-03-2025


DeepLearning.ipynb - Colab
colab.research.google.com/drive/1BjIEO41C24ukummlyYXuaoH68-AkrmsW#scrollTo=ZPJYhv8Hw9ND
DeepLearning.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text
Epochs
/usr/local/lib/python3.11/dist-packages/keras/src/models/functional.py:237: UserWarning: The structure of `inputs` doesn't match the expected structure.
Expected: ['keras_tensor']
Received: inputs=Tensor(shape=(1, 128, 128, 1))
warnings.warn(msg)
1/1 1s 535ms/step
Input Image
Predicted Mask
Connected to Python 3 Google Compute Engine backend
Type here to search
36°C Smoke
13:22
18-03-2025

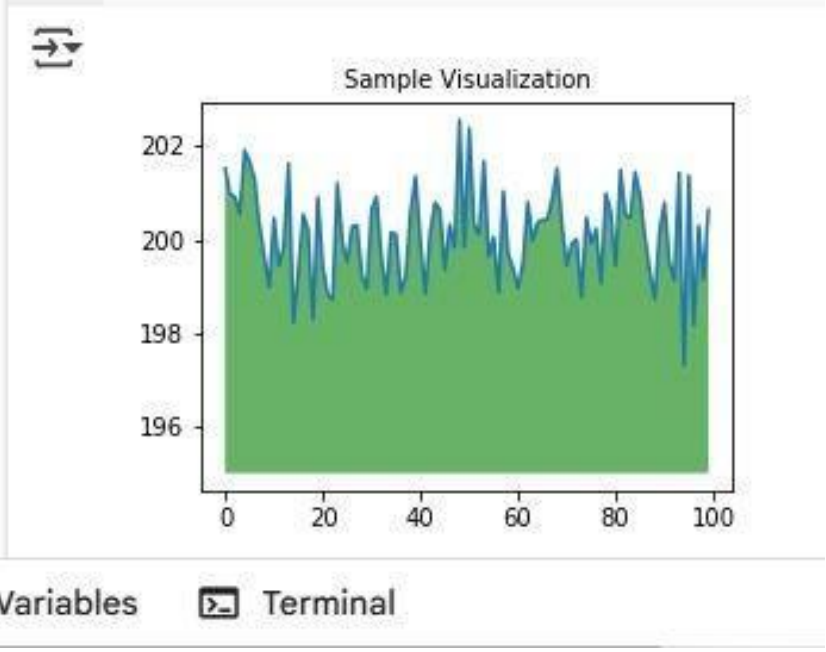
PRACTICAL 5.

Aim: Write a program to predict a caption for a sample image using LSTM. Code:

```
import numpy as np
import IPython.display as display
from matplotlib import pyplot as plt
import io import base64
 ys = 200 +
np.random.randn(100) x = [x for
x in range(len(ys))]

fig = plt.figure(figsize=(4, 3), facecolor='w') plt.plot(x,
ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization", fontsize=10)
 data =
io.BytesIO()
plt.savefig(data)
image =
F"{base64.b64encode(data.getvalue()).decode()}" alt
= "Sample Visualization"
display.display(display.Markdown(F"""![{alt}]({image})"""))
plt.close(fig)
```

```
import numpy as np
import IPython.display as display
from matplotlib import pyplot as plt
import io
import base64

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

fig = plt.figure(figsize=(4, 3), facecolor='w')
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization", fontsize=10)

data = io.BytesIO()
plt.savefig(data)
image = F"{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"""![{alt}]({image})"""))
plt.close(fig)
```

Output:



Variables Terminal

PRACTICAL 6.

Aim: Applying the Autoencoder algorithms for encoding real-world data Code:

```
import numpy as np import
pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Model from keras.layers
import Input, Dense from keras.optimizers import
Adam from keras import regularizers

# Step 1: Generate or load the dataset (here, we're using synthetic data)
# Example: 1000 samples, 20 features (could represent customer data,
financial data, etc.) np.random.seed(42)
data = np.random.rand(1000, 20)

# Step 2: Normalize the data scaler
= StandardScaler()
data_scaled = scaler.fit_transform(data)

# Step 3: Split data into training and test sets
X_train, X_test = train_test_split(data_scaled, test_size=0.2,
random_state=42)

# Step 4: Define the Autoencoder architecture input_layer
= Input(shape=(X_train.shape[1],))

# Encoder
encoded = Dense(16, activation='relu',
activity_regularizer=regularizers.l2(0.01))(input_layer)
encoded = Dense(8, activation='relu')(encoded)  # Compress the data to 8
features
# Decoder
decoded = Dense(16, activation='relu')(encoded)
decoded = Dense(X_train.shape[1], activation='sigmoid')(decoded)  #
Reconstruct to original dimensions
# Step 5: Build the Autoencoder model
```

```
autoencoder = Model(input_layer, decoded)

# Step 6: Compile the model
autoencoder.compile(optimizer=Adam(), loss='mean_squared_error')
# Step 7: Train the Autoencoder model
autoencoder.fit(X_train, X_train, epochs=50, batch_size=256, shuffle=True,
validation_data=(X_test, X_test))

# Step 8: Get the encoded (compressed) representation of the data
encoder = Model(input_layer, encoded) encoded_data =
encoder.predict(X_test)

# Step 9: Reconstruct the data from the encoded representation
reconstructed_data = autoencoder.predict(X_test)

# Step 10: Calculate the reconstruction error
reconstruction_error = np.mean(np.square(X_test - reconstructed_data),
axis=1)
# Step 11: Detect anomalies based on the reconstruction error threshold
= np.percentile(reconstruction_error, 95)  # Choose an appropriate
threshold
anomalies = reconstruction_error > threshold

# Step 12: Visualize the results plt.figure(figsize=(10,
6))

# Plot some example data points with reconstruction error
plt.scatter(range(len(reconstruction_error)), reconstruction_error,
c=anomalies, cmap='coolwarm')
plt.axhline(y=threshold, color='r', linestyle='--', label='Threshold')
plt.title('Reconstruction Error with Anomalies Detected')
plt.xlabel('Sample Index') plt.ylabel('Reconstruction Error')
plt.legend() plt.show()
# Optionally, you can print out the indices of detected anomalies
print("Anomalies detected at indices:", np.where(anomalies)[0])
```

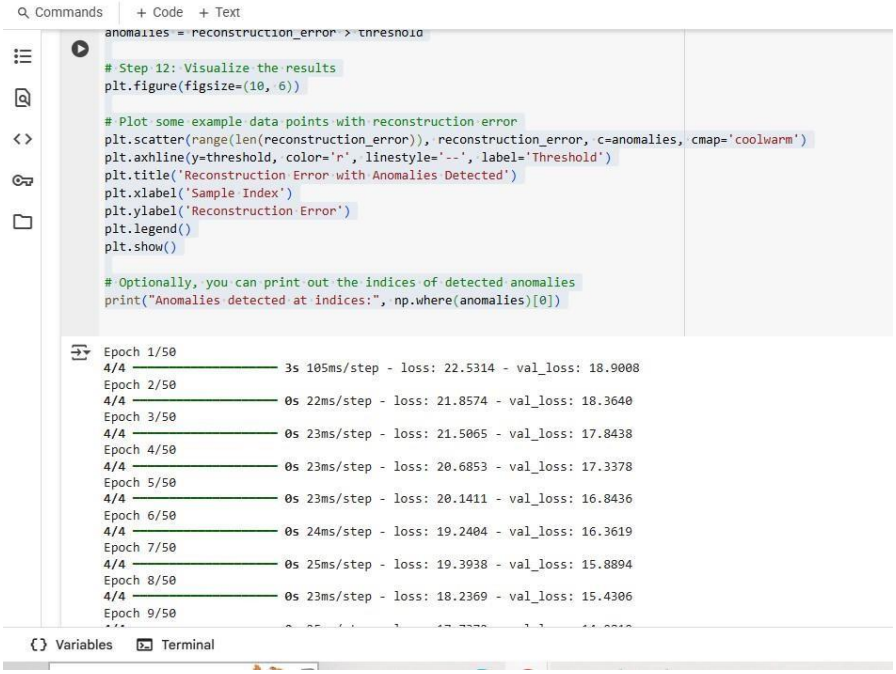
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Model
from keras.layers import Input, Dense
from keras.optimizers import Adam
from keras import regularizers

# Step 1: Generate or load the dataset (here, we're using synthetic data)
# Example: 1000 samples, 20 features (could represent customer data, financial data, etc.)
np.random.seed(42)
data = np.random.rand(1000, 20)

# Step 2: Normalize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

# Step 3: Split data into training and test sets
X_train, X_test = train_test_split(data_scaled, test_size=0.2, random_state=42)

# Step 4: Define the Autoencoder architecture
input_layer = Input(shape=(X_train.shape[1],))

# Encoder
encoded = Dense(16, activation='relu', activity_regularizer=regularizers.l2(0.01))(input_layer)
encoded = Dense(8, activation='relu')(encoded)  # Compress the data to 8 features

# Decoder
decoded = Dense(16, activation='relu')(encoded)
decoded = Dense(X_train.shape[1], activation='sigmoid')(decoded)  # Reconstruct to original dimensions
```

Variables Terminal

Commands + Code + Text

```python
# Decoder
decoded = Dense(16, activation='relu')(encoded)
decoded = Dense(X_train.shape[1], activation='sigmoid')(decoded)  # Reconstruct to original dimensions

# Step 5: Build the Autoencoder model
autoencoder = Model(input_layer, decoded)

# Step 6: Compile the model
autoencoder.compile(optimizer=Adam(), loss='mean_squared_error')

# Step 7: Train the Autoencoder model
autoencoder.fit(X_train, X_train, epochs=50, batch_size=256, shuffle=True, validation_data=(X_test, X_test))

# Step 8: Get the encoded (compressed) representation of the data
encoder = Model(input_layer, encoded)
encoded_data = encoder.predict(X_test)

# Step 9: Reconstruct the data from the encoded representation
reconstructed_data = autoencoder.predict(X_test)

# Step 10: Calculate the reconstruction error
reconstruction_error = np.mean(np.square(X_test - reconstructed_data), axis=1)

# Step 11: Detect anomalies based on the reconstruction error
threshold = np.percentile(reconstruction_error, 95)  # Choose an appropriate threshold
anomalies = reconstruction_error > threshold

# Step 12: Visualize the results
plt.figure(figsize=(10, 6))

# Plot some example data points with reconstruction error
plt.scatter(range(len(reconstruction_error)), reconstruction_error, c=anomalies, cmap='coolwarm')
```

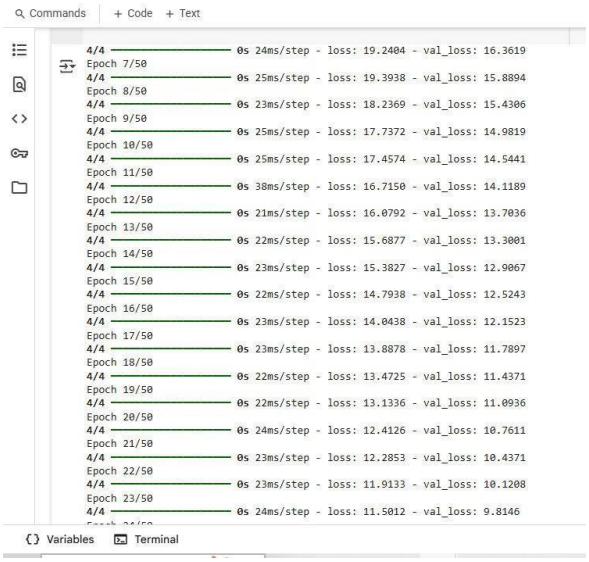
Variables Terminal



```
anomalies = reconstruction_error > threshold

# Step 12: Visualize the results
plt.figure(figsize=(10, 6))

# Plot some example data points with reconstruction error
plt.scatter(range(len(reconstruction_error)), reconstruction_error, c=anomalies, cmap='coolwarm')
plt.axhline(y=threshold, color='r', linestyle='--', label='Threshold')
plt.title('Reconstruction Error with Anomalies Detected')
plt.xlabel('Sample Index')
plt.ylabel('Reconstruction Error')
plt.legend()
plt.show()

# Optionally, you can print out the indices of detected anomalies
print("Anomalies detected at indices:", np.where(anomalies)[0])
```

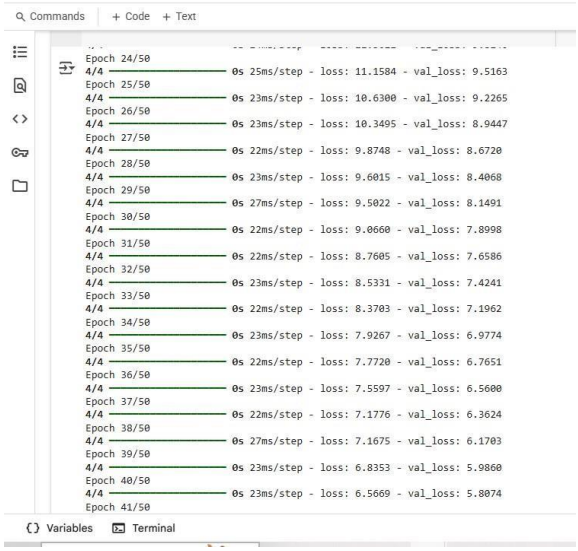
```
Epoch 1/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 3s 105ms/step - loss: 22.5314 - val_loss: 18.9008
Epoch 2/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 21.8574 - val_loss: 18.3640
Epoch 3/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 21.5065 - val_loss: 17.8438
Epoch 4/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 20.6853 - val_loss: 17.3378
Epoch 5/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 20.1411 - val_loss: 16.8436
Epoch 6/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 19.2404 - val_loss: 16.3619
Epoch 7/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 19.3938 - val_loss: 15.8894
Epoch 8/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 18.2369 - val_loss: 15.4306
Epoch 9/50
```

Variables Terminal



```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 19.2404 - val_loss: 16.3619
Epoch 7/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 19.3938 - val_loss: 15.8894
Epoch 8/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 18.2369 - val_loss: 15.4306
Epoch 9/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 17.7372 - val_loss: 14.9819
Epoch 10/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 17.4574 - val_loss: 14.5441
Epoch 11/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 16.7150 - val_loss: 14.1189
Epoch 12/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 16.0792 - val_loss: 13.7036
Epoch 13/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 15.6877 - val_loss: 13.3001
Epoch 14/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 15.3827 - val_loss: 12.9067
Epoch 15/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 14.7938 - val_loss: 12.5243
Epoch 16/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 14.0438 - val_loss: 12.1523
Epoch 17/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 13.8878 - val_loss: 11.7897
Epoch 18/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 13.4725 - val_loss: 11.4371
Epoch 19/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 13.1336 - val_loss: 11.0936
Epoch 20/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 12.4126 - val_loss: 10.7611
Epoch 21/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 12.2853 - val_loss: 10.4371
Epoch 22/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 11.9133 - val_loss: 10.1208
Epoch 23/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 11.5012 - val_loss: 9.8146
```

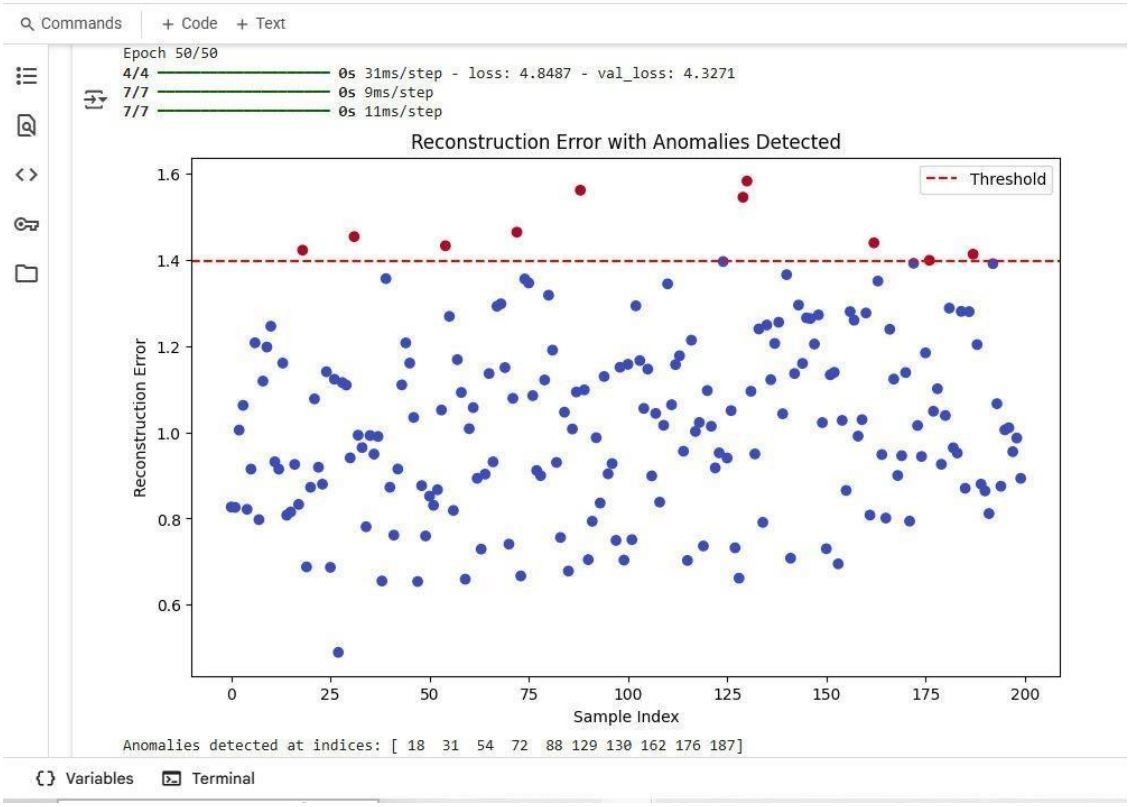
Variables Terminal



```
Epoch 24/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 11.1584 - val_loss: 9.5163
Epoch 25/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 10.6300 - val_loss: 9.2265
Epoch 26/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 10.3495 - val_loss: 8.9447
Epoch 27/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 9.8748 - val_loss: 8.6720
Epoch 28/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 9.6015 - val_loss: 8.4068
Epoch 29/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - loss: 9.5022 - val_loss: 8.1491
Epoch 30/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 9.0660 - val_loss: 7.8998
Epoch 31/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 8.7605 - val_loss: 7.6586
Epoch 32/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 8.5331 - val_loss: 7.4241
Epoch 33/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 8.3703 - val_loss: 7.1962
Epoch 34/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 7.9267 - val_loss: 6.9774
Epoch 35/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 7.7720 - val_loss: 6.7651
Epoch 36/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 7.5597 - val_loss: 6.5600
Epoch 37/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 7.1776 - val_loss: 6.3624
Epoch 38/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - loss: 7.1675 - val_loss: 6.1703
Epoch 39/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 6.8353 - val_loss: 5.9860
Epoch 40/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 6.5669 - val_loss: 5.8074
Epoch 41/50
```

Variables Terminal

Output:


Commands
+ Code
+ Text
Epoch 50/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 4.8487 - val_loss: 4.3271
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step
Reconstruction Error with Anomalies Detected
Threshold
Reconstruction Error
Sample Index
1.6
1.4
1.2
1.0
0.8
0.6
0
25
50
75
100
125
150
175
200
Anomalies detected at indices: [18 31 54 72 88 129 130 162 176 187]
Variables
Terminal

PRACTICAL 7.

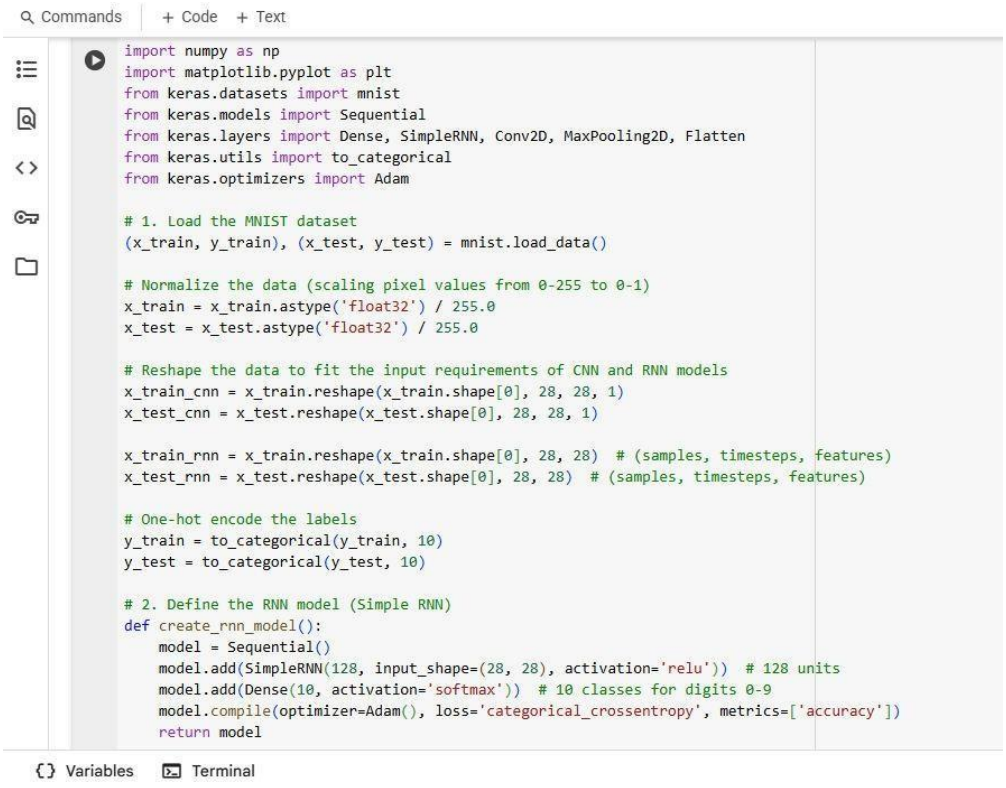
Aim: Write a program for character recognition using RNN and compare it with CNN.

Code:

```
import numpy as np
import matplotlib.pyplot as plt from
keras.datasets import mnist from
keras.models import Sequential
from keras.layers import Dense, SimpleRNN, Conv2D, MaxPooling2D, Flatten
from keras.utils import to_categorical from keras.optimizers import Adam

# 1. Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
# Normalize the data (scaling pixel values from 0-255 to 0-1)
x_train = x_train.astype('float32') / 255.0 x_test =
x_test.astype('float32') / 255.0

# Reshape the data to fit the input requirements of CNN and RNN models
x_train_cnn = x_train.reshape(x_train.shape[0], 28, 28, 1) x_test_cnn
= x_test.reshape(x_test.shape[0], 28, 28, 1)

x_train_rnn = x_train.reshape(x_train.shape[0], 28, 28)  # (samples,
timesteps, features)
x_test_rnn = x_test.reshape(x_test.shape[0], 28, 28)  # (samples,
timesteps, features)

# One-hot encode the labels y_train =
to_categorical(y_train, 10) y_test =
to_categorical(y_test, 10)

# 2. Define the RNN model (Simple RNN)
def create_rnn_model():     model =
Sequential()
    model.add(SimpleRNN(128, input_shape=(28, 28), activation='relu'))  #
128 units
    model.add(Dense(10, activation='softmax'))  # 10 classes for digits 0-
9
    model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])     return model
```

```
# 3. Define the CNN model def
create_cnn_model():      model
= Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))     model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(10, activation='softmax'))  # 10 classes for digits 0-
9
    model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])      return model

# 4. Train the RNN model rnn_model
= create_rnn_model()
rnn_history = rnn_model.fit(x_train_rnn, y_train, epochs=5,
batch_size=128, validation_data=(x_test_rnn, y_test))

# 5. Train the CNN model cnn_model
= create_cnn_model()
cnn_history = cnn_model.fit(x_train_cnn, y_train, epochs=5,
batch_size=128, validation_data=(x_test_cnn, y_test))

# 6. Evaluate the models
rnn_test_loss, rnn_test_acc = rnn_model.evaluate(x_test_rnn, y_test,
verbose=0)
cnn_test_loss, cnn_test_acc = cnn_model.evaluate(x_test_cnn, y_test,
verbose=0)

# Print the results
print(f"RNN Model - Test Accuracy: {rnn_test_acc*100:.2f}%")
print(f"CNN Model - Test Accuracy: {cnn_test_acc*100:.2f}%")
# 7. Plot Training and Validation Accuracy for Both Models
plt.figure(figsize=(12, 6))

# RNN Accuracy Plot plt.subplot(1,
2, 1)
plt.plot(rnn_history.history['accuracy'], label='Training Accuracy')
plt.plot(rnn_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('RNN Model Accuracy')
```

```
plt.xlabel('Epochs')
plt.ylabel('Accuracy') plt.legend()

# CNN Accuracy Plot plt.subplot(1,
2, 2)
plt.plot(cnn_history.history['accuracy'], label='Training Accuracy')
plt.plot(cnn_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('CNN Model Accuracy') plt.xlabel('Epochs')
plt.ylabel('Accuracy') plt.legend()

plt.tight_layout() plt.show()
```



```
    model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# 3. Define the CNN model
def create_cnn_model():
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(10, activation='softmax'))  # 10 classes for digits 0-9
    model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# 4. Train the RNN model
rnn_model = create_rnn_model()
rnn_history = rnn_model.fit(x_train_rnn, y_train, epochs=5, batch_size=128, validation_data=(x_test_rnn, y_test))

# 5. Train the CNN model
cnn_model = create_cnn_model()
cnn_history = cnn_model.fit(x_train_cnn, y_train, epochs=5, batch_size=128, validation_data=(x_test_cnn, y_test))

# 6. Evaluate the models
rnn_test_loss, rnn_test_acc = rnn_model.evaluate(x_test_rnn, y_test, verbose=0)
cnn_test_loss, cnn_test_acc = cnn_model.evaluate(x_test_cnn, y_test, verbose=0)

# Print the results
print(f"RNN Model - Test Accuracy: {rnn_test_acc*100:.2f}%")
print(f"CNN Model - Test Accuracy: {cnn_test_acc*100:.2f}%")

# 7. Plot Training and Validation Accuracy for Both Models
plt.figure(figsize=(12, 6))
```

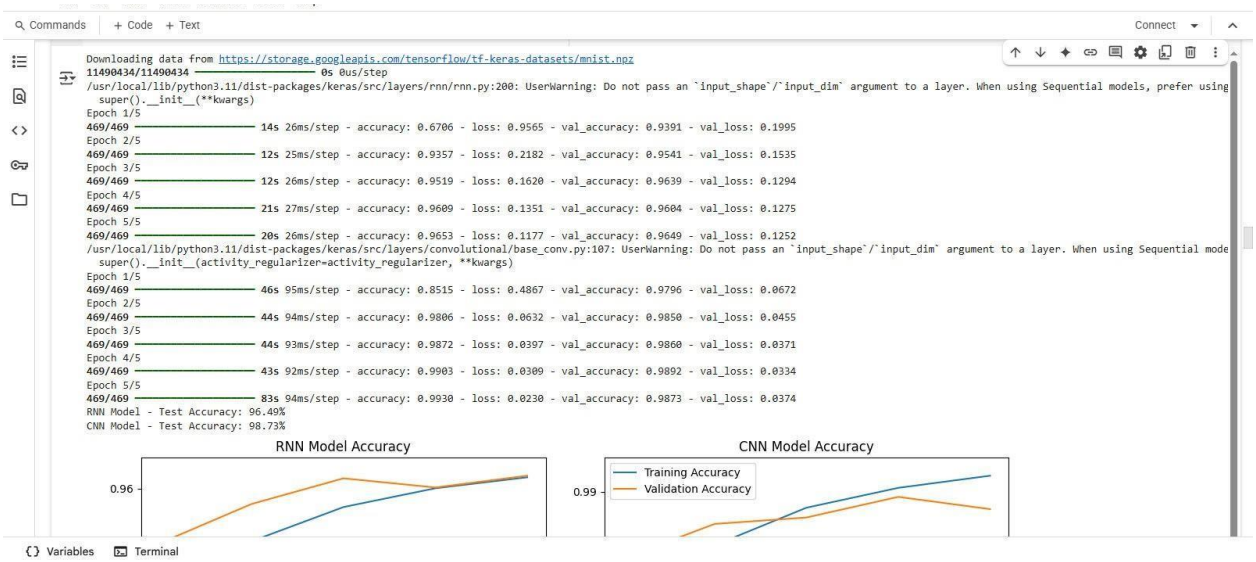
Commands + Code + Text

Variables Terminal

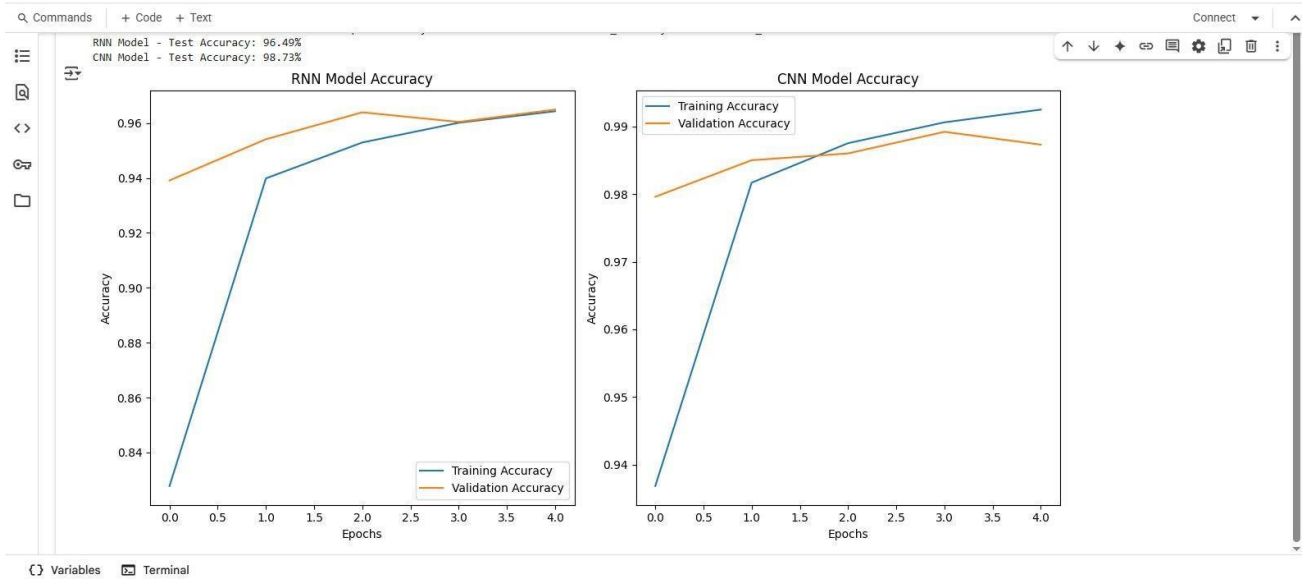
```
# 7. Plot Training and Validation Accuracy for Both Models
plt.figure(figsize=(12, 6))

# RNN Accuracy Plot
plt.subplot(1, 2, 1)
plt.plot(rnn_history.history['accuracy'], label='Training Accuracy')
plt.plot(rnn_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('RNN Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# CNN Accuracy Plot
plt.subplot(1, 2, 2)
plt.plot(cnn_history.history['accuracy'], label='Training Accuracy')
plt.plot(cnn_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('CNN Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using
  super().__init__(**kwargs)
Epoch 1/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 14s 26ms/step - accuracy: 0.6706 - loss: 0.9565 - val_accuracy: 0.9391 - val_loss: 0.1995
Epoch 2/5
469/469 ━━━━━━━━━━━━━━━━━━━━ 12s 25ms/step - accuracy: 0.9357 - loss: 0.2182 - val_accuracy: 0.9541 - val_loss: 0.1535
Epoch 3/5
```

Variables Terminal


Commands
+ Code
+ Text
Connect
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using
super().__init__(**kwargs)
Epoch 1/5
469/469 14s 26ms/step - accuracy: 0.6706 - loss: 0.9565 - val_accuracy: 0.9391 - val_loss: 0.1995
Epoch 2/5
469/469 12s 25ms/step - accuracy: 0.9357 - loss: 0.2182 - val_accuracy: 0.9541 - val_loss: 0.1535
Epoch 3/5
469/469 12s 26ms/step - accuracy: 0.9519 - loss: 0.1620 - val_accuracy: 0.9639 - val_loss: 0.1294
Epoch 4/5
469/469 21s 27ms/step - accuracy: 0.9609 - loss: 0.1351 - val_accuracy: 0.9604 - val_loss: 0.1275
Epoch 5/5
469/469 20s 26ms/step - accuracy: 0.9653 - loss: 0.1177 - val_accuracy: 0.9649 - val_loss: 0.1252
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential mode
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
469/469 46s 95ms/step - accuracy: 0.8515 - loss: 0.4867 - val_accuracy: 0.9796 - val_loss: 0.0672
Epoch 2/5
469/469 44s 94ms/step - accuracy: 0.9806 - loss: 0.0632 - val_accuracy: 0.9850 - val_loss: 0.0455
Epoch 3/5
469/469 44s 93ms/step - accuracy: 0.9872 - loss: 0.0397 - val_accuracy: 0.9860 - val_loss: 0.0371
Epoch 4/5
469/469 43s 92ms/step - accuracy: 0.9903 - loss: 0.0309 - val_accuracy: 0.9892 - val_loss: 0.0334
Epoch 5/5
469/469 83s 94ms/step - accuracy: 0.9930 - loss: 0.0230 - val_accuracy: 0.9873 - val_loss: 0.0374
RNN Model - Test Accuracy: 96.49%
CNN Model - Test Accuracy: 98.73%
RNN Model Accuracy
CNN Model Accuracy
0.96
0.99
Training Accuracy
Validation Accuracy
Variables
Terminal

Output:


Commands
+ Code
+ Text
Connect
RNN Model - Test Accuracy: 96.49%
CNN Model - Test Accuracy: 98.73%
RNN Model Accuracy
CNN Model Accuracy
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Variables
Terminal

PRACTICAL 8.

Aim: Write a program to develop Autoencoders using MNIST Handwritten Digits.

Code:

```
import numpy as np
import matplotlib.pyplot as plt from
keras.datasets import mnist from
keras.models import Model from
keras.layers import Input, Dense from
keras.optimizers import Adam from
keras.utils import plot_model

# Step 1: Load and preprocess the MNIST dataset
(x_train, _), (x_test, _) = mnist.load_data()
# Normalize the images to the range [0, 1] x_train
= x_train.astype('float32') / 255.0 x_test =
x_test.astype('float32') / 255.0

# Flatten the images to 1D vectors (28x28 -> 784) x_train
= x_train.reshape((x_train.shape[0], 784)) x_test =
x_test.reshape((x_test.shape[0], 784))

# Step 2: Build the Autoencoder model input_img
= Input(shape=(784,))

# Encoder: Compress the data to a lower-dimensional space
encoded = Dense(128, activation='relu')(input_img) encoded
= Dense(64, activation='relu')(encoded)
encoded = Dense(32, activation='relu')(encoded)  # Bottleneck layer
(compressed representation)

# Decoder: Reconstruct the data back to its original shape
decoded = Dense(64, activation='relu')(encoded) decoded
= Dense(128, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)  # Output should have
the same shape as input (784)

# Combine the encoder and decoder to form the autoencoder autoencoder
= Model(input_img, decoded)

# Step 3: Compile the model
autoencoder.compile(optimizer=Adam(), loss='binary_crossentropy')
```

```
# Step 4: Train the Autoencoder
autoencoder.fit(x_train, x_train, epochs=20, batch_size=256, shuffle=True,
validation_data=(x_test, x_test))

# Step 5: Evaluate the Autoencoder's performance on the test set
decoded_imgs = autoencoder.predict(x_test)

# Step 6: Visualize the original and reconstructed images
n = 10  # How many digits we will display
plt.figure(figsize=(20, 4)) for i in range(n):
    # Display original images
ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
ax.set_title("Original")      ax.axis('off')

    # Display reconstructed images
ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28), cmap='gray')
ax.set_title("Reconstructed")      ax.axis('off')

plt.show()

# Optionally, save the model architecture visualization #
plot_model(autoencoder, to_file='autoencoder_model.png',
show_shapes=True)
```

Commands + Code + Text

```
# Decoder: Reconstruct the data back to its original shape
decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)  # Output should have the same shape as input (784)

# Combine the encoder and decoder to form the autoencoder
autoencoder = Model(input_img, decoded)

# Step 3: Compile the model
autoencoder.compile(optimizer=Adam(), loss='binary_crossentropy')

# Step 4: Train the Autoencoder
autoencoder.fit(x_train, x_train, epochs=20, batch_size=256, shuffle=True, validation_data=(x_test, x_test))

# Step 5: Evaluate the Autoencoder's performance on the test set
decoded_imgs = autoencoder.predict(x_test)

# Step 6: Visualize the original and reconstructed images
n = 10  # How many digits we will display
plt.figure(figsize=(20, 4))
for i in range(n):
    # Display original images
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
    ax.set_title("Original")
    ax.axis('off')

    # Display reconstructed images
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28), cmap='gray')
    ax.set_title("Reconstructed")
    ax.axis('off')
```

Variables Terminal

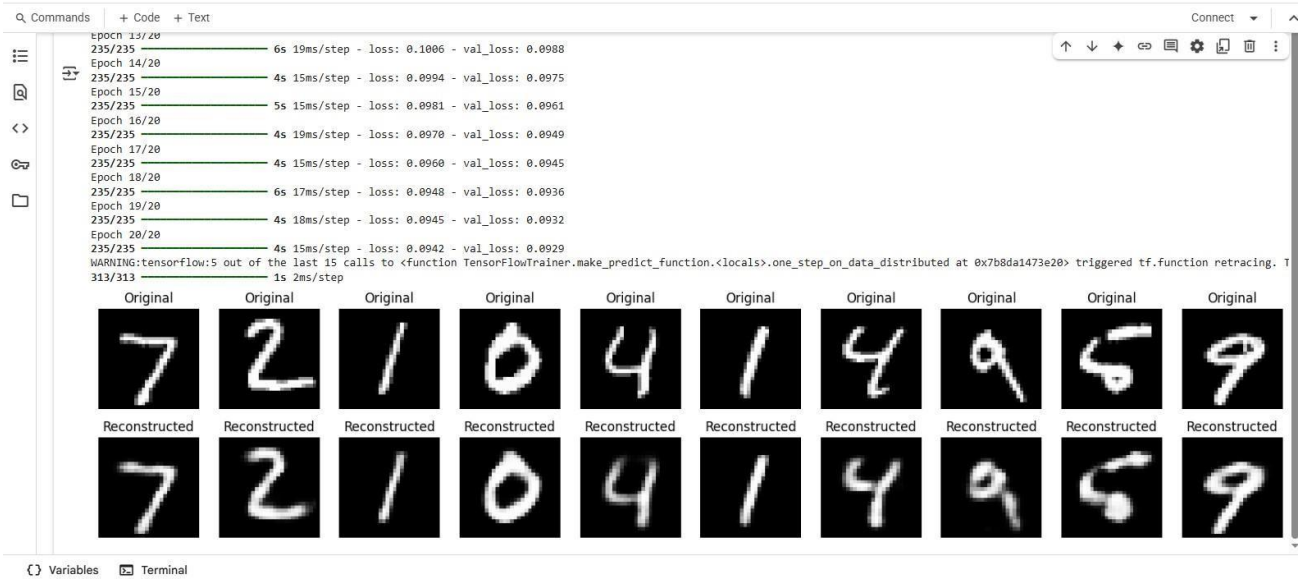
Commands + Code + Text

```
    ax.axis('off')

    # Display reconstructed images
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28), cmap='gray')
    ax.set_title("Reconstructed")
    ax.axis('off')

plt.show()

# Optionally, save the model architecture visualization
# plot_model(autoencoder, to_file='autoencoder_model.png', show_shapes=True)
```

```
Epoch 1/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 8s 20ms/step - loss: 0.3511 - val_loss: 0.1742
Epoch 2/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 0.1650 - val_loss: 0.1403
Epoch 3/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 22ms/step - loss: 0.1376 - val_loss: 0.1281
Epoch 4/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 9s 15ms/step - loss: 0.1270 - val_loss: 0.1215
Epoch 5/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 20ms/step - loss: 0.1204 - val_loss: 0.1152
Epoch 6/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 0.1158 - val_loss: 0.1111
Epoch 7/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 0.1114 - val_loss: 0.1073
Epoch 8/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 19ms/step - loss: 0.1083 - val_loss: 0.1051
Epoch 9/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 0.1060 - val_loss: 0.1046
Epoch 10/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 0.1046 - val_loss: 0.1025
Epoch 11/20
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 18ms/step - loss: 0.1030 - val_loss: 0.1011
```

Variables Terminal

Output:


Commands
+ Code
+ Text
Connect
Epoch 13/20
235/235 6s 19ms/step - loss: 0.1006 - val_loss: 0.0988
Epoch 14/20
235/235 4s 15ms/step - loss: 0.0994 - val_loss: 0.0975
Epoch 15/20
235/235 5s 15ms/step - loss: 0.0981 - val_loss: 0.0961
Epoch 16/20
235/235 4s 19ms/step - loss: 0.0970 - val_loss: 0.0949
Epoch 17/20
235/235 4s 15ms/step - loss: 0.0960 - val_loss: 0.0945
Epoch 18/20
235/235 6s 17ms/step - loss: 0.0948 - val_loss: 0.0936
Epoch 19/20
235/235 4s 18ms/step - loss: 0.0945 - val_loss: 0.0932
Epoch 20/20
235/235 4s 15ms/step - loss: 0.0942 - val_loss: 0.0929
WARNING:tensorflow:5 out of the last 15 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x7b8da1473e20> triggered tf.function retracing.
313/313 1s 2ms/step
Original
Reconstructed
Variables
Terminal

PRACTICAL 9.

Aim: Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.(google stock price).

Code:

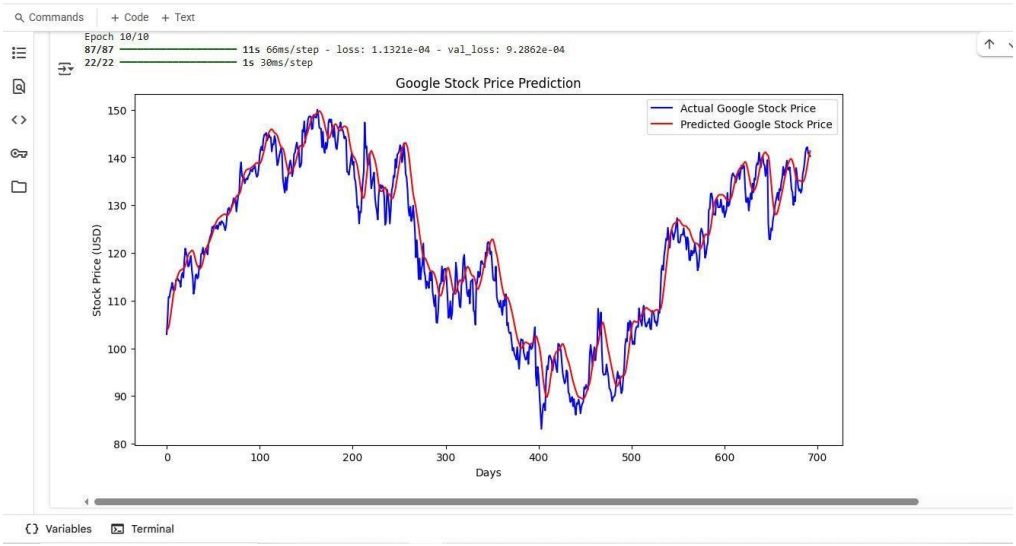
```
import numpy as np import
pandas as pd
import matplotlib.pyplot as plt import
yfinance as yf
from sklearn.preprocessing import MinMaxScaler from
tensorflow.keras.models import Sequential from
tensorflow.keras.layers import LSTM, Dense

# Step 1: Load historical stock data
data = yf.download('GOOG', start='2010-01-01', end='2023-12-31')
stock_prices = data['Close'].values.reshape(-1, 1)  # Use 'Close' price
# Step 2: Normalize prices
scaler = MinMaxScaler(feature_range=(0, 1)) scaled_prices
= scaler.fit_transform(stock_prices)

# Step 3: Create sequences for training (60 days to predict next day)
X = [] y = []
sequence_length = 60
 for i in range(sequence_length,
len(scaled_prices)):     X.append(scaled_prices[i-
sequence_length:i])
    y.append(scaled_prices[i])

X = np.array(X) y
= np.array(y)

# Step 4: Split into training and testing sets (80% training) split
= int(0.8 * len(X))
X_train, X_test = X[:split], X[split:] y_train,
y_test = y[:split], y[split:]

# Step 5: Build the LSTM model
model = Sequential([
    LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1],
1)),
    LSTM(units=50),
```

```
    Dense(units=1)
])
model.compile(optimizer='adam', loss='mean_squared_error')
# Step 6: Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32,
validation_data=(X_test, y_test))

# Step 7: Predict and inverse scale predictions
= model.predict(X_test)
predictions = scaler.inverse_transform(predictions)
actual = scaler.inverse_transform(y_test.reshape(-1, 1))
# Step 8: Plot results plt.figure(figsize=(12,
6))
plt.plot(actual, color='blue', label='Actual Google Stock Price')
plt.plot(predictions, color='red', label='Predicted Google Stock Price')
plt.title('Google Stock Price Prediction') plt.xlabel('Days')
plt.ylabel('Stock Price (USD)')
plt.legend() plt.show()
```

```python
# Step 4: Split into training and testing sets (80% training)
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

# Step 5: Build the LSTM model
model = Sequential([
    LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)),
    LSTM(units=50),
    Dense(units=1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

# Step 6: Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

# Step 7: Predict and inverse scale
predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)
actual = scaler.inverse_transform(y_test.reshape(-1, 1))

# Step 8: Plot results
plt.figure(figsize=(12, 6))
plt.plot(actual, color='blue', label='Actual Google Stock Price')
plt.plot(predictions, color='red', label='Predicted Google Stock Price')
plt.title('Google Stock Price Prediction')
plt.xlabel('Days')
plt.ylabel('Stock Price (USD)')
plt.legend()
plt.show()
```

```
[**********100%**********]  1 of 1 completedEpoch 1/10

/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using a
  super().__init__(**kwargs)
87/87 ━━━━━━━━━━━━━━━━━━━━ 8s 55ms/step - loss: 0.0090 - val_loss: 0.0013
Epoch 2/10
87/87 ━━━━━━━━━━━━━━━━━━━━ 6s 66ms/step - loss: 1.5237e-04 - val_loss: 0.0012
Epoch 3/10
87/87 ━━━━━━━━━━━━━━━━━━━━ 10s 65ms/step - loss: 1.2637e-04 - val_loss: 0.0011
Epoch 4/10
87/87 ━━━━━━━━━━━━━━━━━━━━ 11s 69ms/step - loss: 1.2209e-04 - val_loss: 0.0012
Epoch 5/10
87/87 ━━━━━━━━━━━━━━━━━━━━ 5s 57ms/step - loss: 1.1353e-04 - val_loss: 0.0012
Epoch 6/10
87/87 ━━━━━━━━━━━━━━━━━━━━ 5s 58ms/step - loss: 1.3336e-04 - val_loss: 0.0010
Epoch 7/10
87/87 ━━━━━━━━━━━━━━━━━━━━ 5s 63ms/step - loss: 1.1695e-04 - val_loss: 9.6799e-04
Epoch 8/10
87/87 ━━━━━━━━━━━━━━━━━━━━ 9s 51ms/step - loss: 1.3559e-04 - val_loss: 0.0011
Epoch 9/10
87/87 ━━━━━━━━━━━━━━━━━━━━ 5s 61ms/step - loss: 1.1140e-04 - val_loss: 8.5251e-04
Epoch 10/10
87/87 ━━━━━━━━━━━━━━━━━━━━ 11s 66ms/step - loss: 1.1321e-04 - val_loss: 9.2862e-04
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step
```

Output:



PRACTICAL 10.

Aim: Applying Generative Adversarial Networks for image generation and unsupervised tasks. Code:

STEP 1: Import Libraries import

tensorflow as tf from

tensorflow.keras import layers

import numpy as np import

matplotlib.pyplot as plt import os

STEP 2: Load CIFAR-10 Dataset

(x_train, _), (_, _) = tf.keras.datasets.cifar10.load_data() x_train

= x_train.astype('float32')

x_train = (x_train - 127.5) / 127.5 # Normalize to [-1, 1]

BUFFER_SIZE = 50000

BATCH_SIZE = 128 train_dataset

=

tf.data.Dataset.from_tensor_slices(x_train).shufle(BUFFER_SIZE).batch(BATCH_SIZE)

STEP 3: Define the Generator def make_generator_model(): model =

tf.keras.Sequential([

layers.Dense(8*8*256, use_bias=False, input_shape=(100,)),

layers.BatchNormalization(), layers.LeakyReLU(),

layers.Reshape((8, 8, 256)),

layers.Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use_bias=False),

layers.BatchNormalization(), layers.LeakyReLU(),

layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use_bias=False),

layers.BatchNormalization(), layers.LeakyReLU(),

layers.Conv2DTranspose(3, (5, 5), strides=(2, 2), padding='same', use_bias=False, activation='tanh')

])

return model

```
generator = make_generator_model()
STEP 4: Define the Discriminator def
make_discriminator_model():
model = tf.keras.Sequential([
      layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same', input_shape=[32, 32, 3]),
layers.LeakyReLU(),        layers.Dropout(0.3),

      layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'),
layers.LeakyReLU(),        layers.Dropout(0.3),

      layers.Flatten(),
layers.Dense(1)
   ])
   return model

discriminator = make_discriminator_model() STEP
5: Loss Functions & Optimizers
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True) def
generator_loss(fake_output):
   return cross_entropy(tf.ones_like(fake_output), fake_output) def
discriminator_loss(real_output, fake_output):
   real_loss = cross_entropy(tf.ones_like(real_output), real_output)
fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)    return
real_loss + fake_loss
generator_optimizer = tf.keras.optimizers.Adam(1e-4) discriminator_optimizer
= tf.keras.optimizers.Adam(1e-4)
```

STEP 6: Define Training Loop

```
EPOCHS = 50 noise_dim =
100
num_examples_to_generate = 16
seed = tf.random.normal([num_examples_to_generate, noise_dim])
@tf.function def
train_step(images):
   noise = tf.random.normal([BATCH_SIZE, noise_dim])    with
tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
      generated_images = generator(noise, training=True)
real_output = discriminator(images, training=True)       fake_output
= discriminator(generated_images, training=True)       gen_loss =
generator_loss(fake_output)
      disc_loss = discriminator_loss(real_output, fake_output)
   gradients_of_generator = gen_tape.gradient(gen_loss, generator.trainable_variables)
gradients_of_discriminator   =   disc_tape.gradient(disc_loss,   discriminator.trainable_variables)
generator_optimizer.apply_gradients(zip(gradients_of_generator, generator.trainable_variables))
   discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,
discriminator.trainable_variables))  def  generate_and_save_images(model,
epoch, test_input):
   predictions = model(test_input, training=False)
fig = plt.figure(figsize=(4, 4))     for i in
range(predictions.shape[0]):        plt.subplot(4, 4,
i + 1)
      img = (predictions[i] + 1.0) / 2.0  # Rescale from [-1,1] to [0,1]
plt.imshow(img)        plt.axis('off')
   plt.suptitle(f'Epoch {epoch}', fontsize=16)
plt.show() def train(dataset, epochs):     for
```

```
epoch in range(1, epochs + 1):        for
image_batch in dataset:
train_step(image_batch)         if epoch % 5
== 0 or epoch == 1:
          print(f' Epoch {epoch} completed!')
generate_and_save_images(generator, epoch, seed) STEP
7: Start Training train(train_dataset, EPOCHS)
```

NURTURING POTENTIAL SAKET

GYANPEETH'S

SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE

(Permanently Affiliated to University of Mumbai)

NAAC Accredited

Saket Vidyanagri Marg, Chinchpada Road, Katemanivali,
Kalyan (East) -421306(Mah)

Department of Information Technology

This is to certify that

Mr. RAJNIKANT ASHOK SHUKLA

Seat No. 1314284

of

M.Sc. Information Technology

Part II NEP 2020 Semester IV

has satisfactory carried out the required practical in the subject of **ROBOTIC PROCESS AUTOMATION** For the Academic year 2024 – 2025

Practical In-Charge **Head of the Department**

External Examiner

College Seal

INDEX

Pract No.	Practical Description	Sign
1	**RPA Basics : Sequences and Flowcharts:**	
	a. Create a simple sequence-based project.	
	b. Create a flowchart-based project.	
	c. Automate UiPath Number Calculation (Subtraction, Multiplication, Division of numbers).	
	d. Create an automation UiPath project using different types of variables (number, datetime, Boolean, generic, array, data table)	
2	**Decision making and looping:**	
	a. Consider an array of names. We have to find out how many of them start with the letter "a". Create an automation where the number of names starting with "a" is counted and the result is displayed.	
	b. Demonstrate switch statement with an example	
	c. Create an automation To Print numbers from 1 to 10 with break after the writeline activity inside for each activity	
	d. Create an automation using Do..While Activity to print numbers from 5 to 1	
	e. Create an automation using Delay Activity between two writeline activities to separate their execution by 5 seconds	
	f. Create an automation to demonstrate use of decision statements (if)	
3	**Types of Recording:**	
	a. Basic Recording using Toolbar	
	b. Basic Recording using Notepad	
	c. Desktop Recording using Tool bar	
	d. Desktop Recording by creating a workflow	
	e. Web Recording e.g. Find the rating of the movie from imdb web site	
	f. Web Recording manually	
4	**Excel Automation:**	
	a. Automate the process to extract data from an excel file into a data table and vice versa.	
	b. Create an automation to Write data to specific cell of an excel sheet.	
	c. Create an automation to Read data to specific cell of an excel sheet.	
	d. Create an automation to append data to specific cell of an excel sheet.	
	e. Create an automation to sort a table of an excel sheet.	
	f. Create an automation to filter a table of an excel sheet.	
	g. Choose a repetitive manual task from your workplace or daily life. Design and implement an RPA bot to automate this task using your preferred RPA tool	
5	**Different controls in UiPath:**	
	a. Implement the attach window activity	
	b. Automate using Anchor Base	
	c. Automate using Element Exists.	
	d. Automate using Find Children control.	
	e. Use Get Ancestor control	
	f. Use Find Relative control	

6	**Keyboard and Mouse Events:**	
	a. Demonstrate the following activities in UiPath:	
	i. Mouse (click, double click and hover)	
	ii. Type into	
	iii. Type Secure text	
	b. Demonstrate the following events in UiPath:	
	i. Element triggering event	
	ii. Image triggering event	
	iii. System Triggering Event	
	c. Automate the process of launching an assistant bot on a keyboard event.	
7	**Screen Scraping and Web Scraping methods:**	
	a. Automate the following screen scraping methods using UiPath:	
	a. Full Text	
	b. Native	
	c. OCR	
	b. Demonstrate Data Scraping and display values in Message box.	
	c. Demonstrate Screen Scraping for a pdf, web page and image file.	
8	**PDF Automation and Exception Handling:**	
	a. Read PDF With OCR	
	b. Merge PDF's into one	
	c. Get PDF Total Page count Using Regex	
	d. Extract data from a PDF or Excel file and populate it into a database or spreadsheet.	
	e. Extract data from a PDF or Excel file and populate it into a database or spreadsheet. Implement data manipulation techniques like filtering, sorting, or data validation.	
	f. Demonstrate Exception Handling using UiPath	
9	**Email Automation:**	
	a. Configure Email using UiPath	
	b. Read Emails	
	c. Send Email with Attachment	
	d. Save Email Attachments	
	e. Reply to Email	
10	**Orchestrator management and mini project:**	
	a. Deploy bots to Orchestrator	
	b. Run jobs from Orchestrator	
	c. Queue Introduction:	
	i. Add items to Queue	
	ii. Get Queue item from Orchestrator	
	d. Build UiPath Chatbot using Google dialogflow	

11	**RPA Applications:**	
	a. Automate the extraction of data from invoices, validate information, and update accounting systems.	
	b. Automate data entry tasks from various sources such as emails, forms, or documents, and validate data against predefined rules.	
	c. Automate the process of expense reporting by extracting data from receipts, categorizing expenses, and generating reports.	
	d. Automate inventory tracking and management processes, including stock updates, reordering, and inventory audits.	
	e. Automate sales order processing tasks such as order entry, validation, and fulfilment.	
12	**Prepare the Robotics process automation project on any one of the following domains:**	
	RPA in Logistics, Intelligent Process Automation, IT Process Automation Explained, RPA in Banking, RPA in Education, RPA in Telecommunications, RPA in Healthcare, RPA in Insurance, RPA in Accounting, RPA Challenges, RPA in Real Estate, RPA in BPO, RPA Security	

Practical : 1 Sequences and Flowcharts :

a. Create a simple sequence-based project. Code :

Print message hello

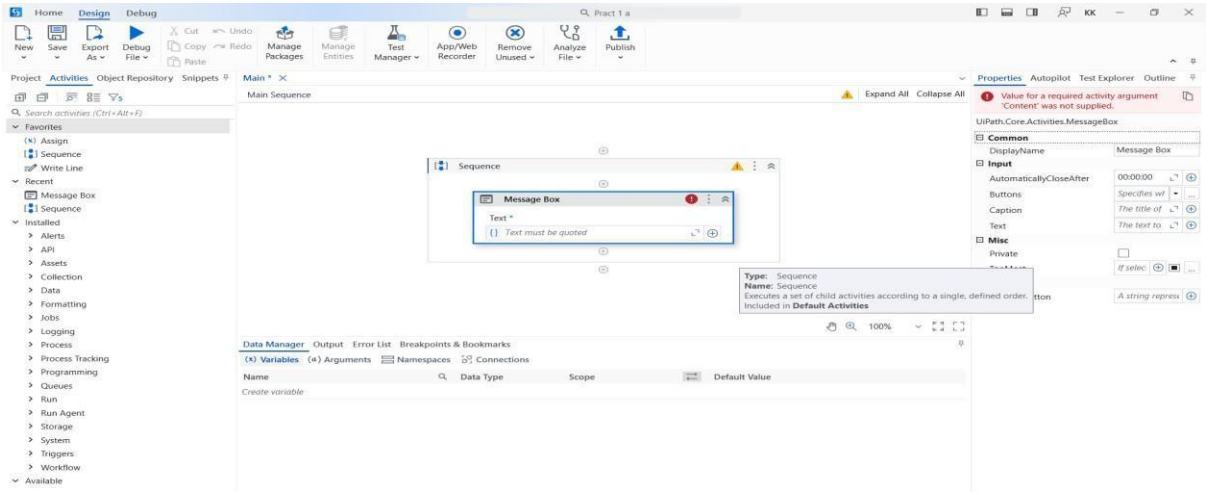
Step 1: Open a UiPath studio.

Step 2: Select blank process and give a name to create a project.

Step 3: Click on open main workflow.

Step 4: In activity panel search sequence and select sequence and drop in main workflow.

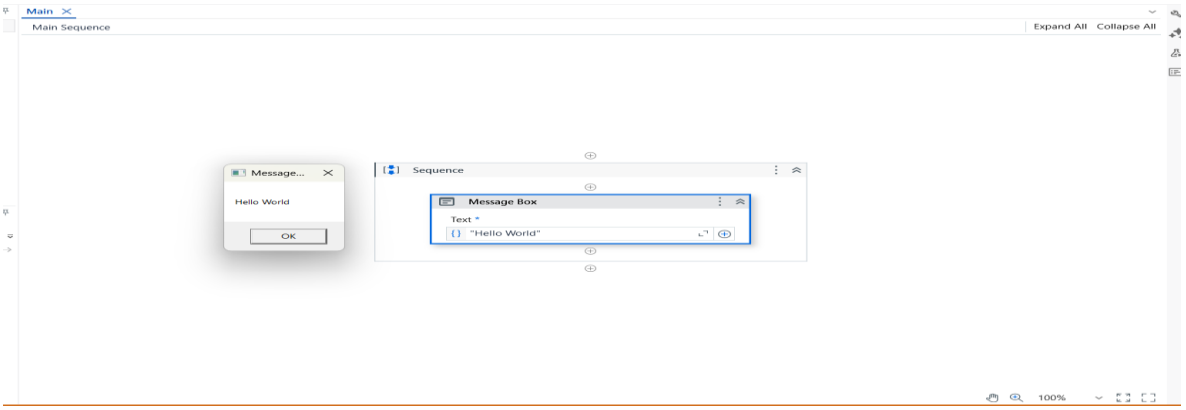
Step 5: Now search for message box and select message box and drop in main workflow.



Step 6: In message box type hello world.

Step 7: click on debug file and select run file.

Output :

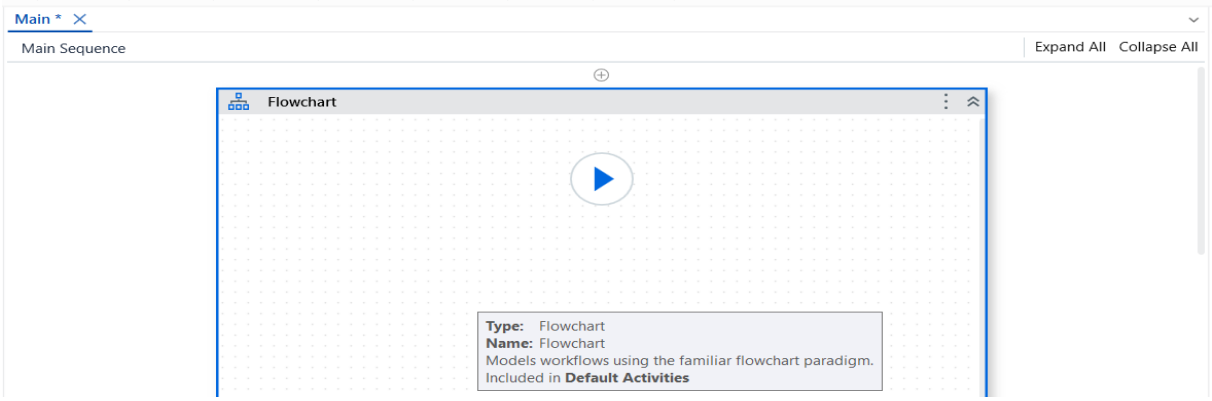


b. Create a flowchart-based project.

Code :

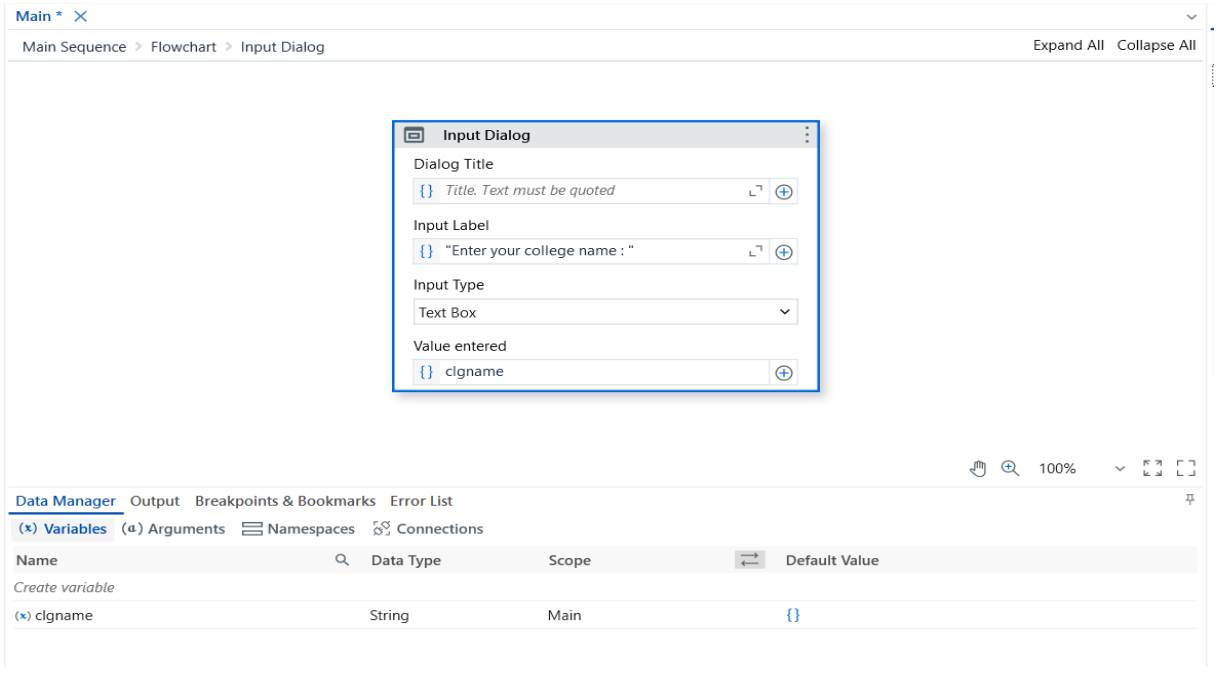
Step 1: Open UiPath studio and click on process give name to your project.

Step 2: Click on Open main workflow. Drag and drop flowchart from activities panel.



Step 3: Drag and drop input dailog box inside a flowchart.

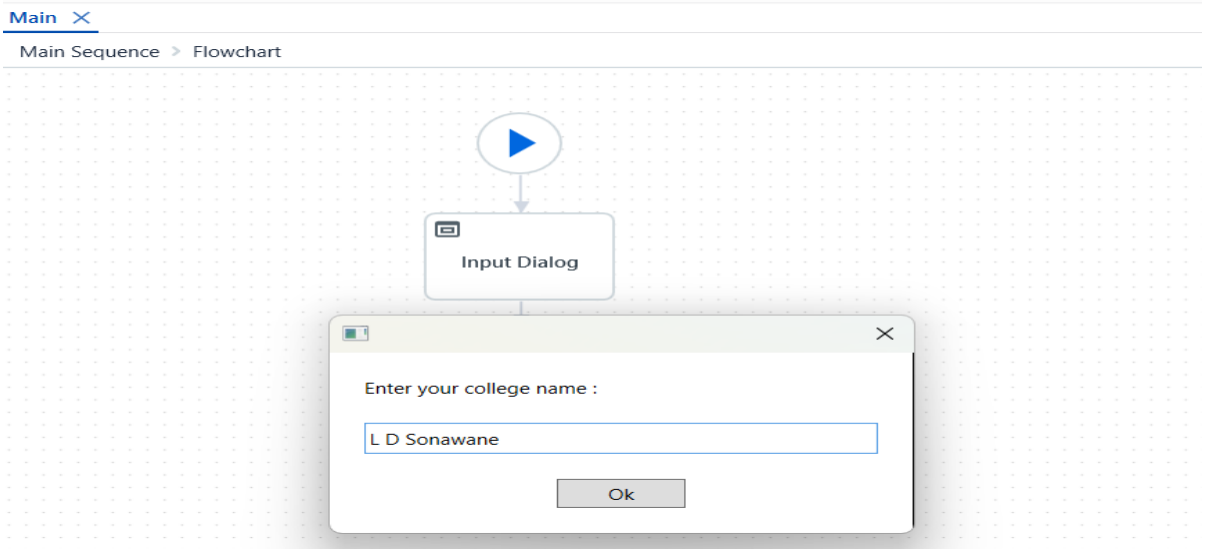
Step 4: Create a variable “clgname” and give your input label



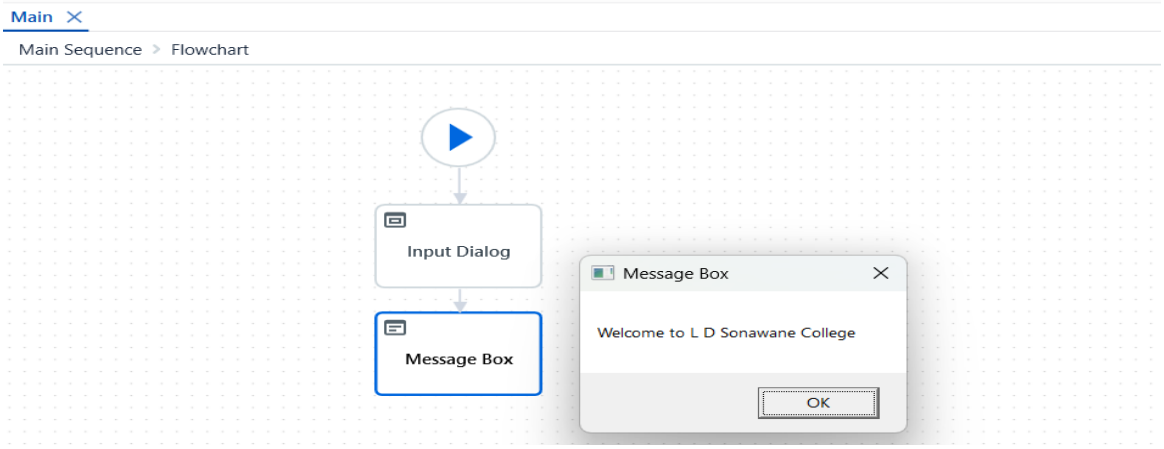
Step 5: Drag and drop a message box below input dialog box and enter your message inside message box.

Step 6: Click on debug file and select run file.

Output :



Actual Output :



c. Automate UiPath Number Calculation (Addition,Subtraction, Multiplication, Division of numbers).

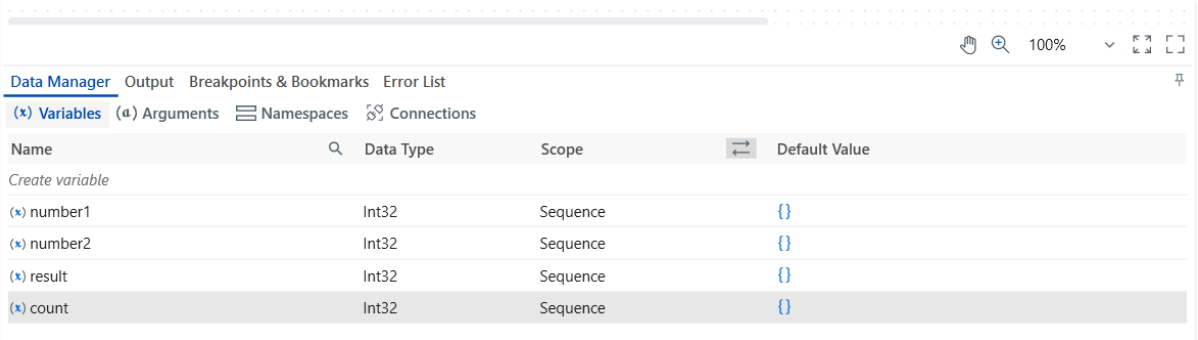
Code:

Step 1: Open UiPath studio select process and give name to your project and click on create.

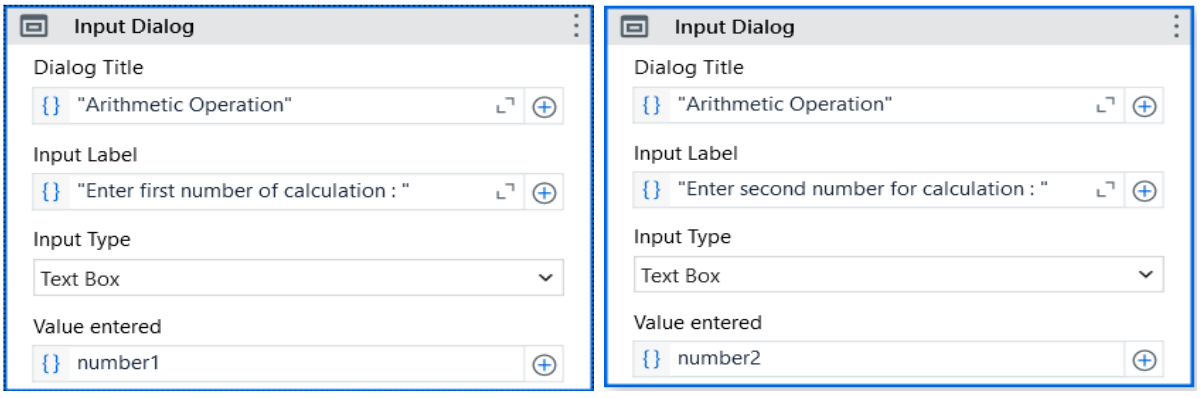
Step 2: Click on open main workflow

Step 3: go to activities and select sequence drag and drop.

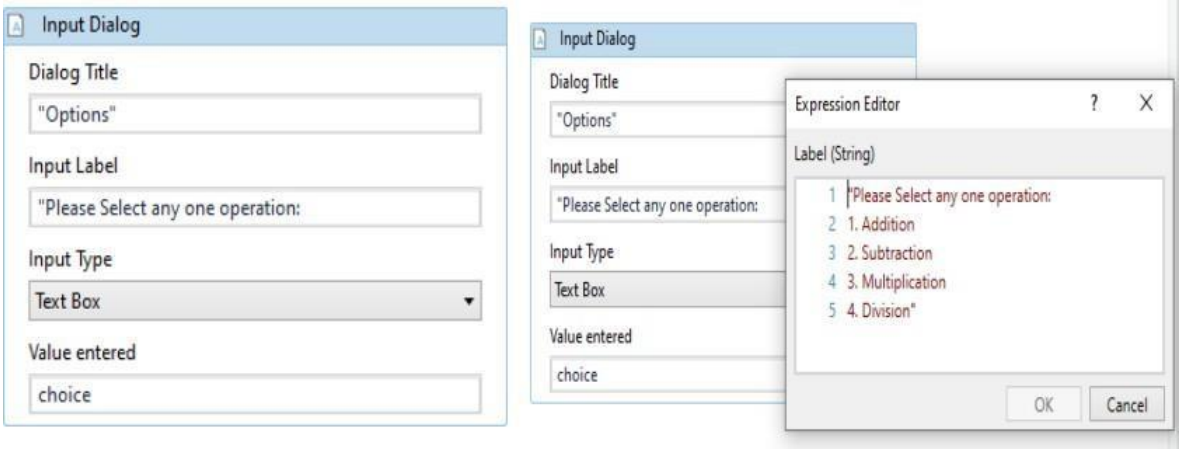
Step 4: Create 4 integer variable.



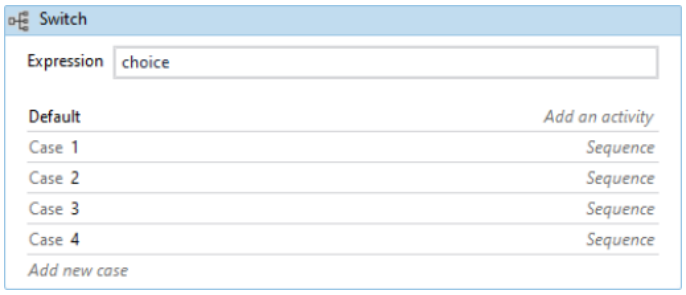
Step 5: Select 2 input dialogs from the activities drag and drop. Enter the values in dialog box.



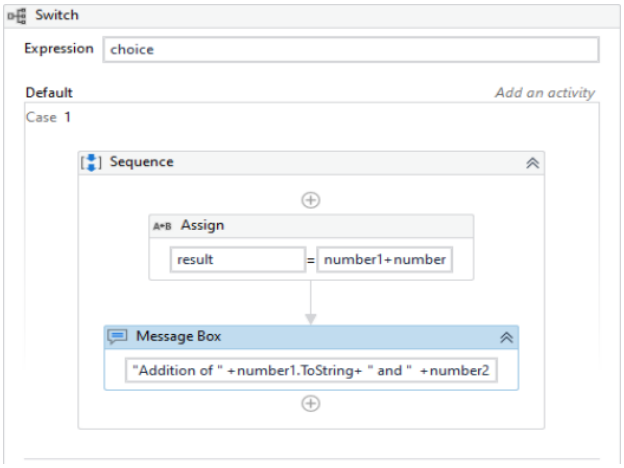
Step 6 : Select another dialog box for selecting options



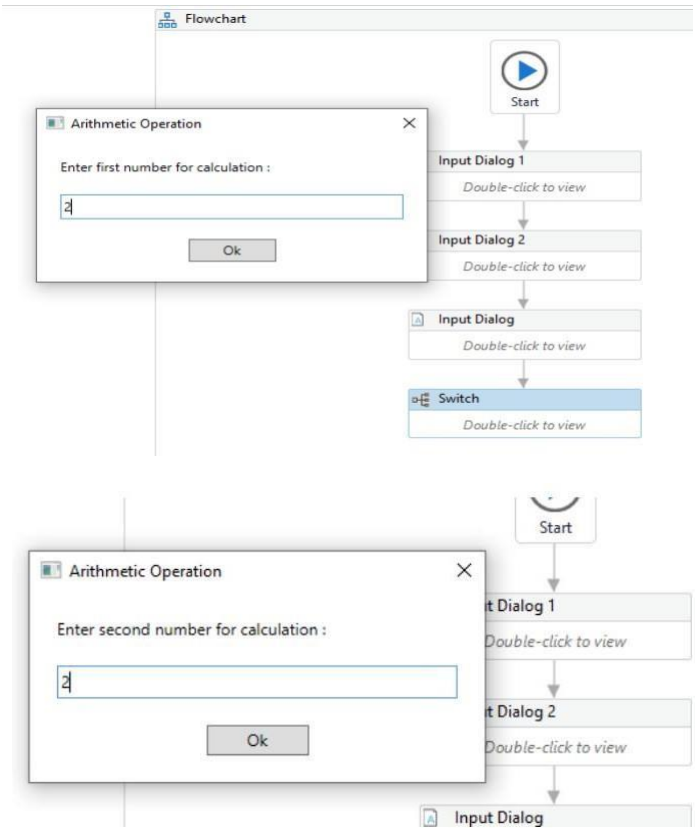
Step 7: Select switch activity and assign each case, required condition

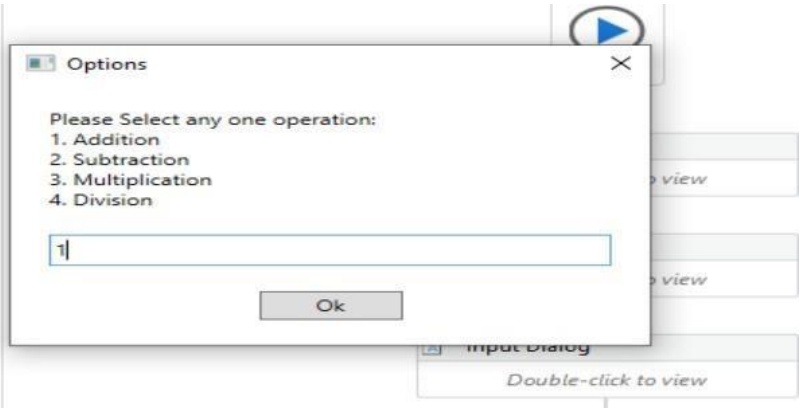


Step 8: Repeat the same step as below for all the case.



Step 9: Click on debug file and click on run file:




Options
Please Select any one operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
1
Ok
Double-click to view


Message Box
Addition of 2 and 2 is 4
OK

d. Create an automation UiPath project using different types of variables (number, datetime, Boolean, generic, array, data table)

Code :

Step 1: Create a blank project and give it a meaningful name.

Step 2: Drag and drop the Sequence activity and give it a name.

Step 3: Create two variables named variable1 and variable2 of int32.

Step 4: Create another variable named Result of int32.

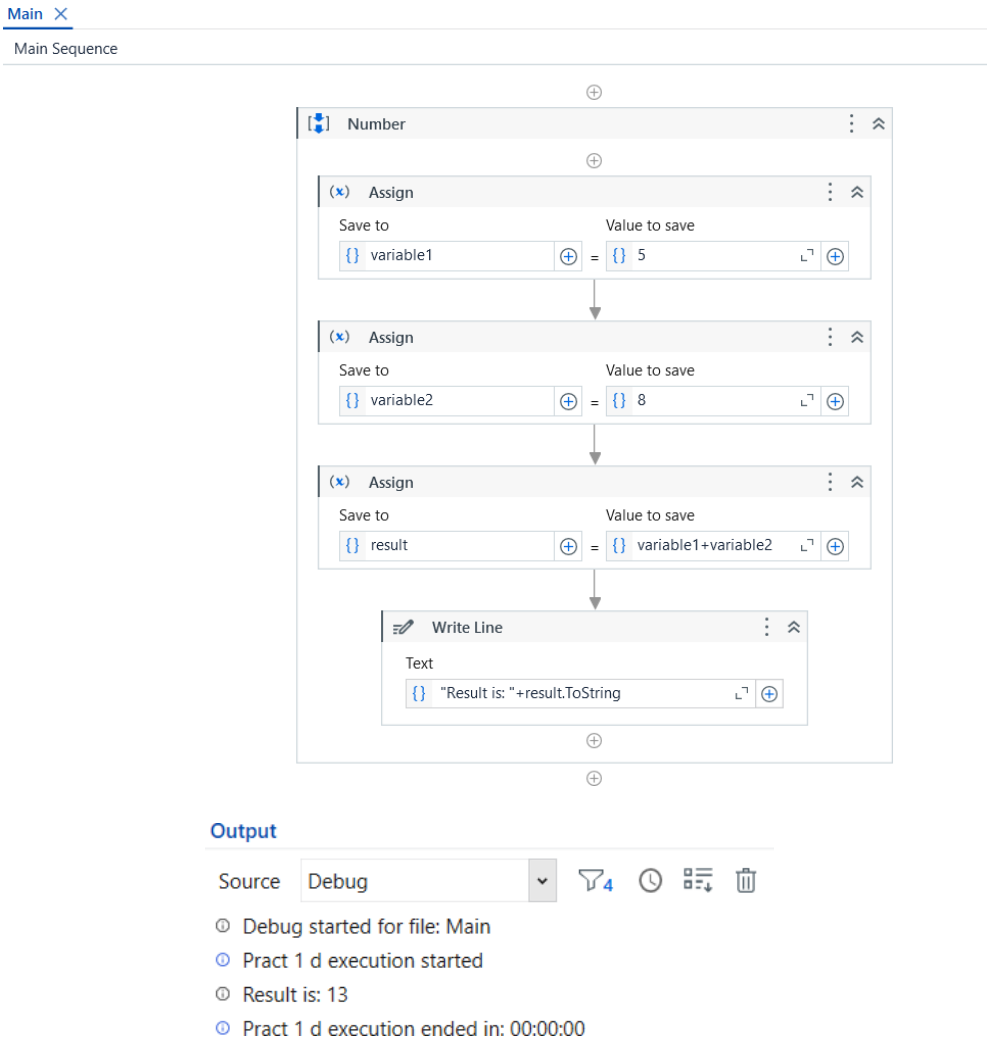
Step 5: Select the Assign activity and assign the variable1 as “5”.

Step 6: Select the Assign activity and assign the variable2 as “10”.

Step 7: Select the Assign activity and assign the Result as “variable1+variable2”.

Step 8: Select the Write Line activity and print the Result value.

Step 9: Run.



String:

Step 1: Create a blank project and give it a meaningful name.

Step 2: Drag and drop the Sequence activity and give it a name.

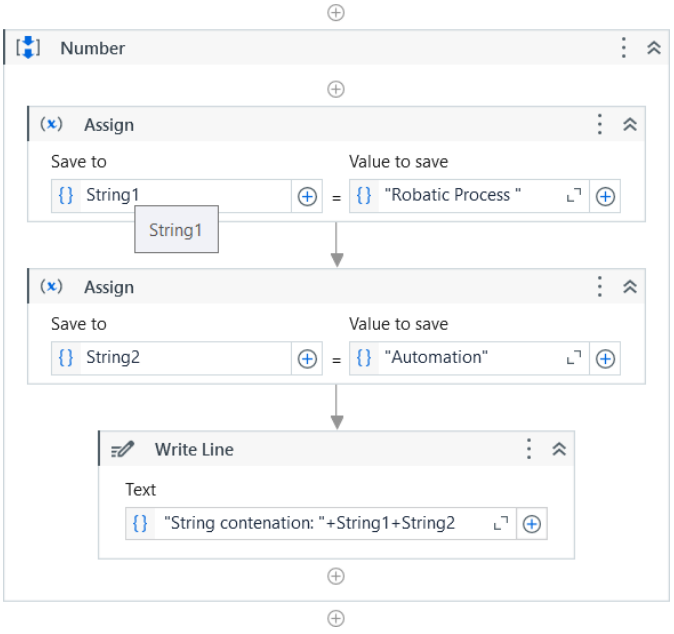
Step 3: Create two variables named String1 and String2 of String.

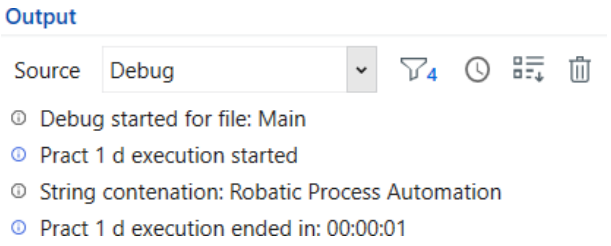
Step 4: Select the Assign activity and assign the String1 as “Robotic Process”.

Step 5: Select the Assign activity and assign the String2 as “Automation”.

Step 6: Select the Write Line activity and perform the concatenation method in it.

Step 7: Run.





DateTime:

Step 1: Create a blank project and give it a meaningful name.

Step 2: Drag and drop the Sequence activity and give it a name.

Step 3: Create a variables named Date1 of String.

Step 4: Select Assign activity and assign “Now.ToString” to the variable Date1.

Step 5: Select Write Line activity and Print the above.

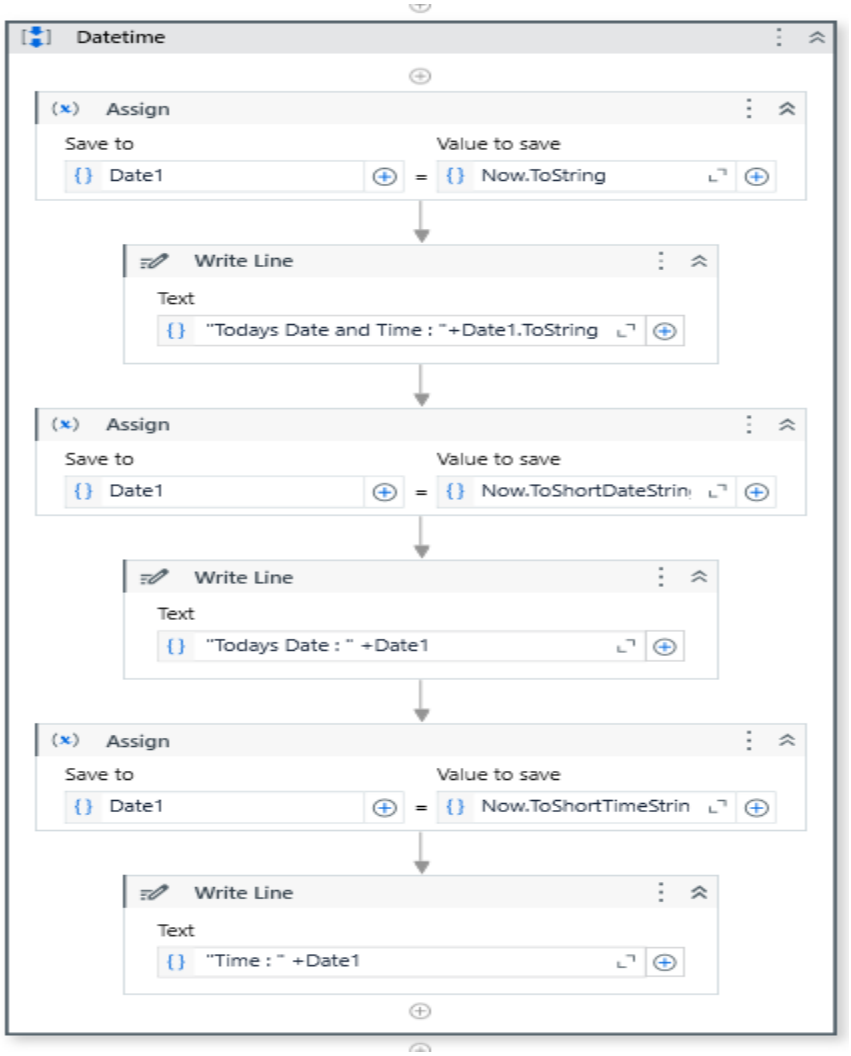
Step 6: Select Assign activity and assign “Now.ToShortDateString” to the variable Date1.

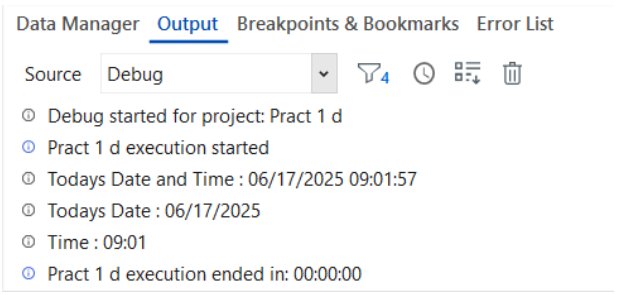
Step 7: Select Write Line activity and Print the above.

Step 8: Select Assign activity and assign “Now.ToShortTimeString” to the variable Date1.

Step 9: Select Write Line activity and Print the above.

Step 10: Run.





Practical 2 : Decision making and looping

a. **Consider an array of names. We have to find out how many of them start with the letter "a". Create an automation where the number of names starting with "a" is counted and the result is displayed.**

Code :

Step 1: Open uipath studio and give name to your project.

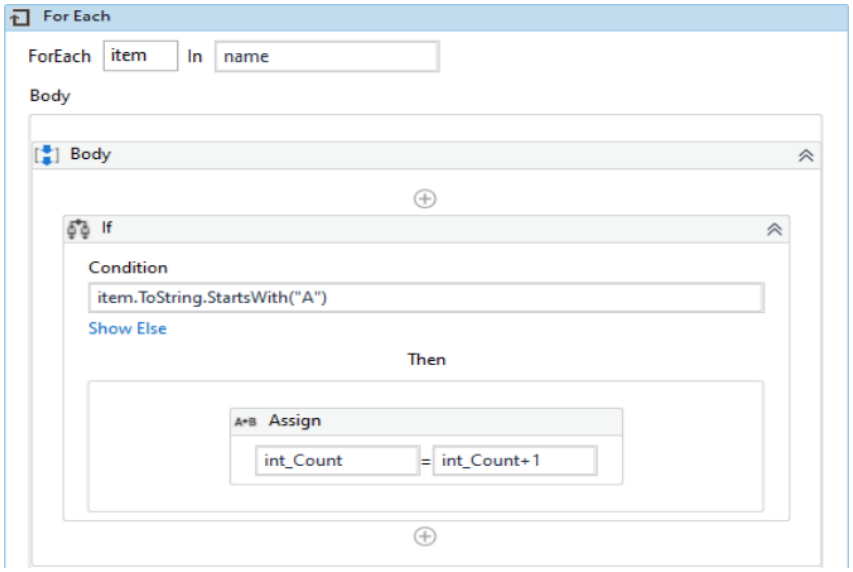
Step 2: Click on open main workflow. click on new and select sequence.

Step 3 : Select a flowchart and drag and drop multiple assign activity in it.

Step 4: Create 2 variables. One for storing the array of names and other for counting the number of names in array.

Step 5: Drag and drop for each activity

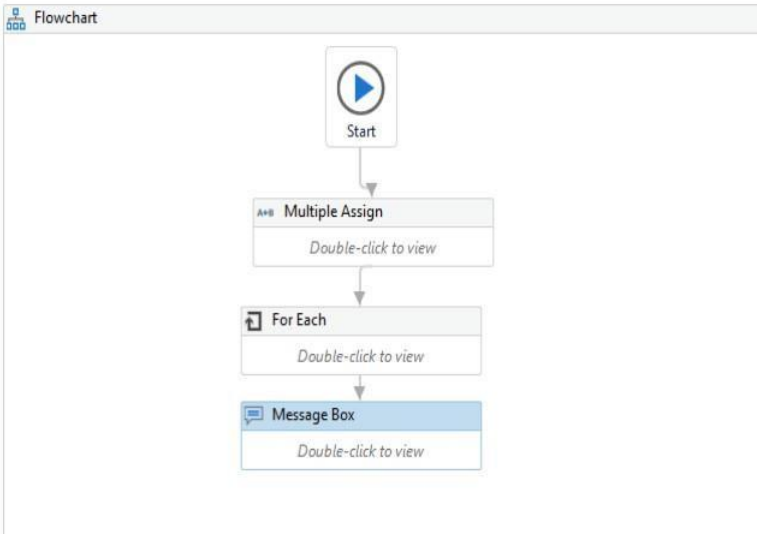
Step 6: Inside the body section of for each activity, drag and drop if activity and give the condition for counting the number of occurrence of character ‘a’.



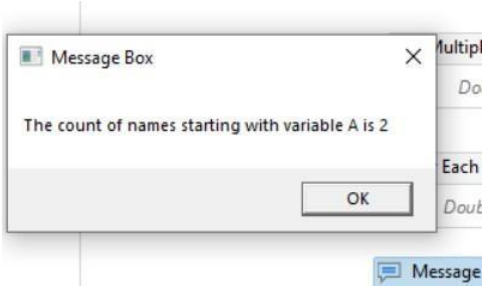
Step 7: Drag and drop a message box below for each activity. Enter the value to be displayed in the message box.

Output:

Flow of Activity



Result



b. Demonstrate switch statement with an example

The Switch activity The Switch activity can be used to make a choice. When we have various options available and want to execute one option, we frequently use the Switch activity. By default, the Switch activity takes an integer argument. If we want to take a desired argument, then we can change it from the Properties panel, from the TypeArgument list. The Switch activity is very useful in the categorization of data according to one's own choice.

Perform the following steps:

1. Add a Sequence activity.

2. Add an Input dialog activity inside the Sequence.

3. Now, create an integer type variable L.

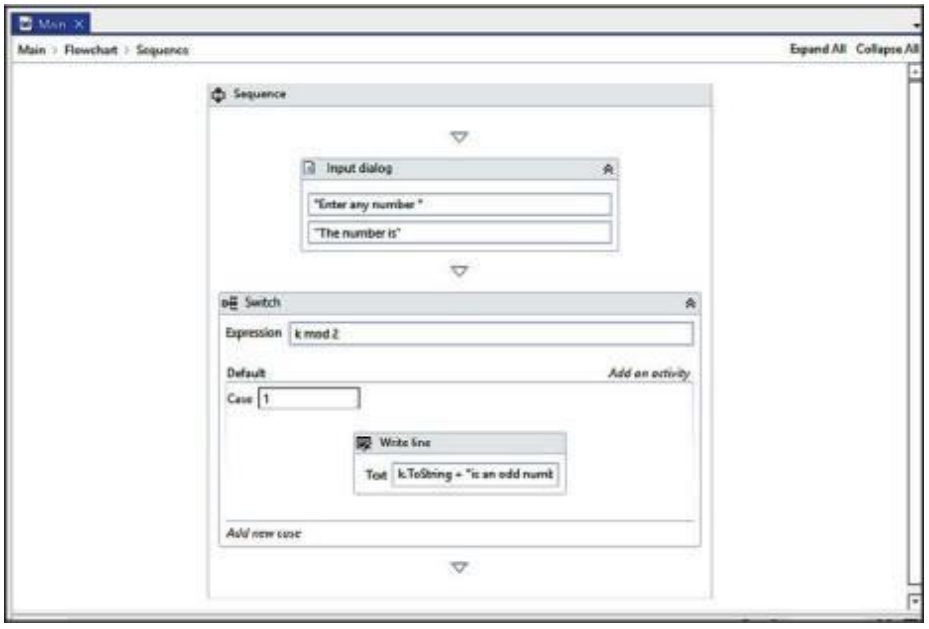
4. Specify the newly created variable's name in the Result property inside the Properties panel.

5. Add the Switch activity under the Input dialog activity.

6. In the Expression field, set LNPE to check whether the number is divisible by 2 or not.

7. Add a Write line activity to the Default section and type the L5P4USJOH JTBOFWFOOVNCFS in the text field.

8. Now, create Case 1, add the one other Write line activity to it, and type L5P4USJOH ——— JTBOPEEOVNCFS ——————————————————— in the text field:



e. Create an automation using Delay Activity between two writeline activities to separate their execution by 5 seconds

To better understand how the Delay activity works, let us see an example of an automation that writes two messages to the Output panel with a delay of 50 seconds.

Perform the following steps:

1. First, create a new Flowchart.

2. Add a Write line activity from the Activities panel and connect it to the Start node.

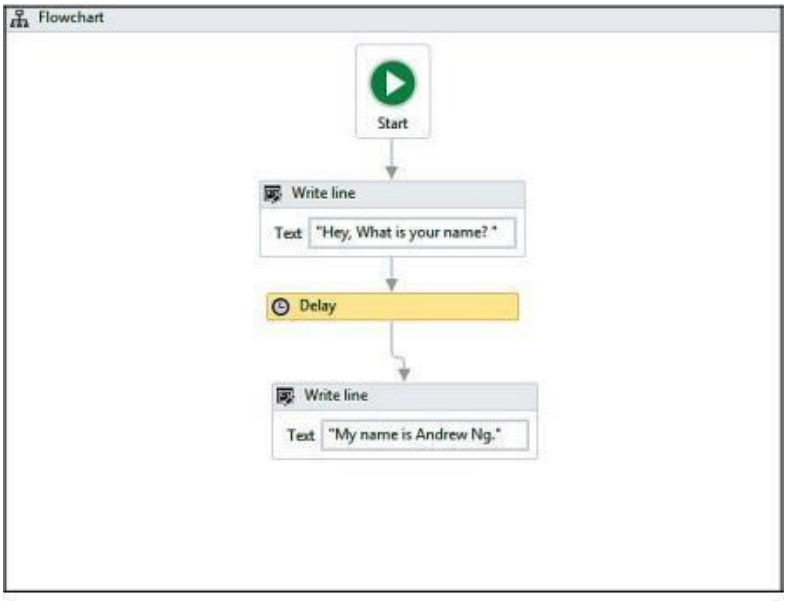
3. Select the Write line activity. Now, type the following text into the Text box:

“Hey, what is your name?”

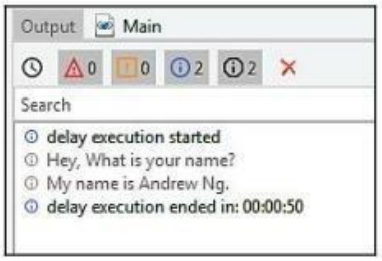
4. Next, add a Delay activity and connect it to the Write line activity.

5. Select the Delay activity and go to the Properties panel. In the Duration field, set 00:00:50. This is a 50-second delay between the two logged messages.

6. Take another Write line activity and connect it to the Delay activity. In the Text field, write “My name is Andrew Ng”:



7. After clicking on the Run button, the Output panel shows the message that delays it by 50 seconds:



f. Create an automation to demonstrate use of decision statements (if)

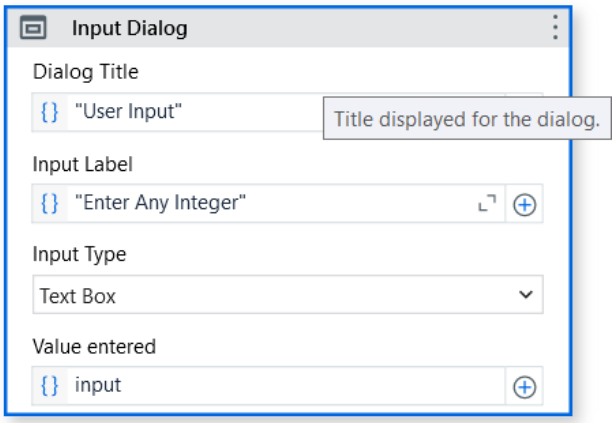
If-else Activity

Step 1: Create a blank project and give it a name.

Step 2: Drag and drop the Flowchart activity and give it a name.

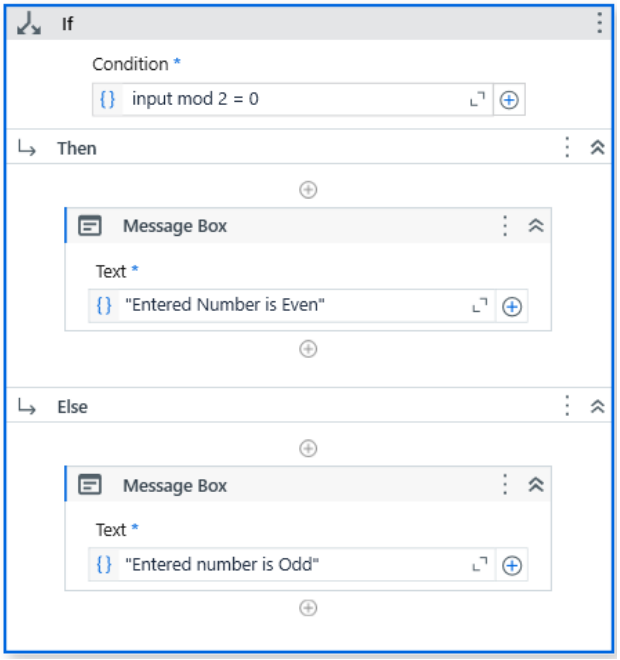
Step 3: Create a variable name for storing user input.

Step 4: Drag and drop input dialog box.



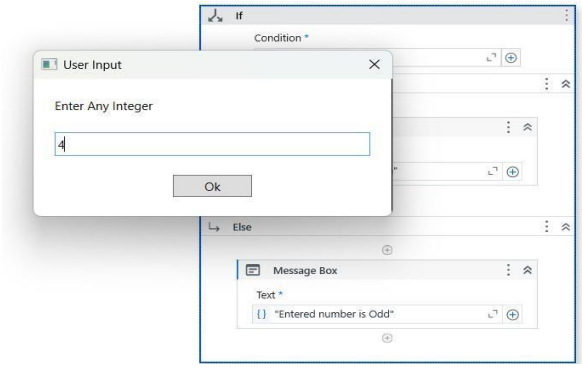
Step 5: Drag and drop if activity and give the condition.

Step 6: Drag and drop message box inside if activity for displaying the output.

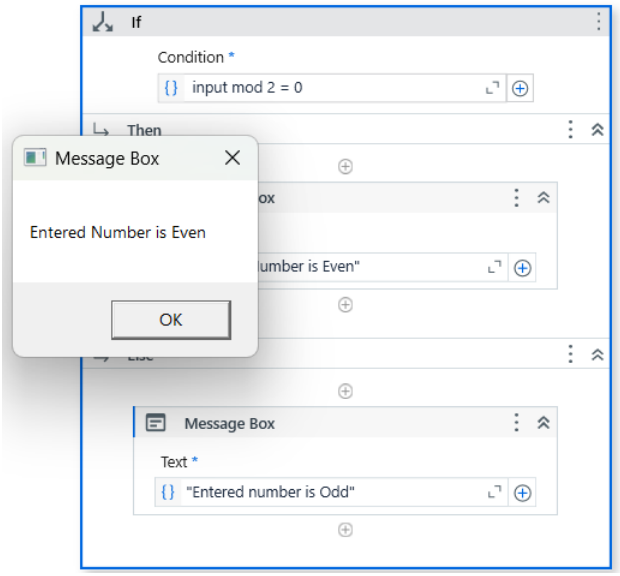


Step 8: Run.

Output:



Result of activity:

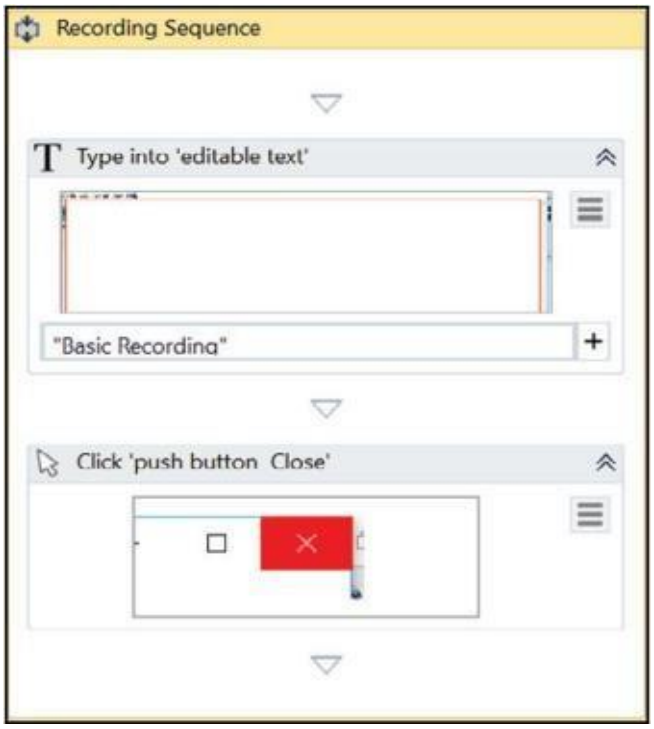


Practical 3 : Types of Recording

a. Basic Recording using Toolbar

This is used to record the actions of applications that have a single window. Basic Recording uses a full Selector. It works better for applications performing a single action. It is not suitable for applications with multiple windows.

There are two types of selectors, partial selectors and full selectors. A Full selector has all the attribute to recognize a control or application. The Basic recording uses full selectors

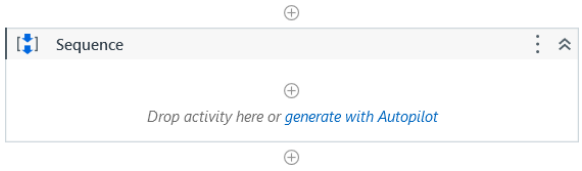


Please note that, in the preceding image that there are different activities but those activities are not wrapped inside containers, it is generated by Basic recorder. Basic recording generates different activities and places them directly in the sequence with full selector. You have already seen how to automate tasks using the Basic recorder; now, let us cover other recorders.

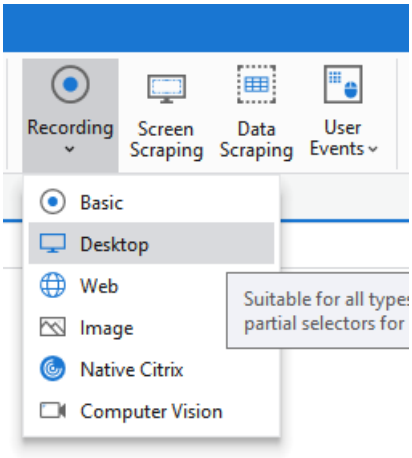
b. Basic Recording using Notepad

Steps:

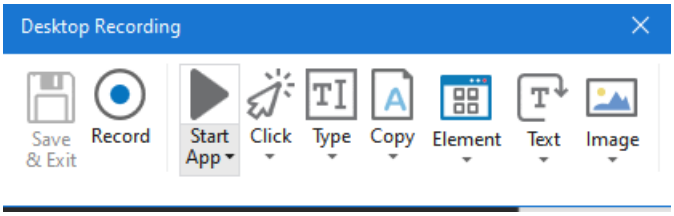
1. Create a new Blank Project and give it an appropriate name. Drag a Sequence activity from Activity tab.



2. Select Desktop Recording under Recording.



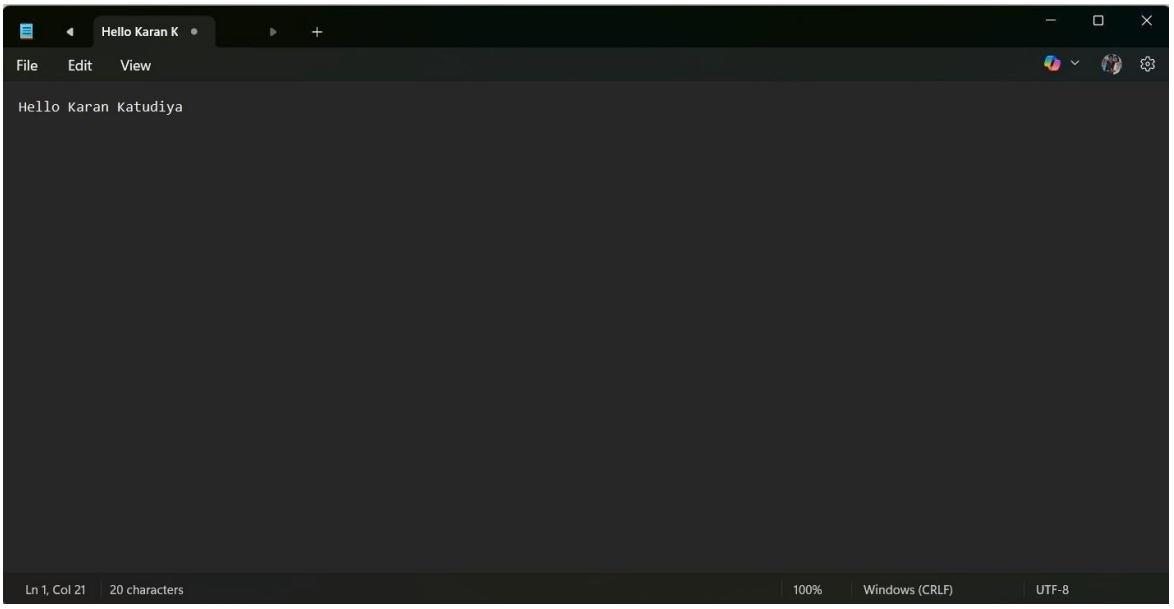
3. Open a new notepad, then on desktop recording click on open application and select notepad then click ok on the prompt that appears



4. Start recording by clicking on record button in recording panel, then click in the text area of notepad.

5. In the type into prompt enter the text you want to be typed.

Output:

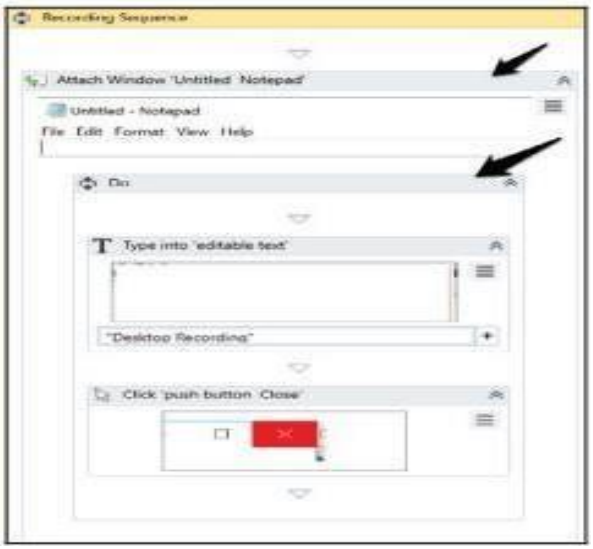


C. Desktop Recording using Tool bar

This is similar to Basic recording with the added advantage of working with multiple actions. It is most suitable for automating Desktop applications. Desktop recorder generates Partial selectors. The Partial selectors, have a hierarchical structure. They are split into parent child views for recognizing the UI element properly.

Please note in the preceding image there is a Attach Window activities and other activities are nested under it. This flow is generated Desktop recorder:

Output:



d. Desktop Recording by creating a workflow

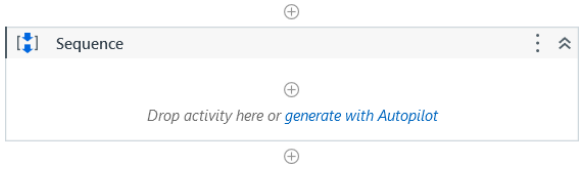
Use "double Ui" and automate stuff (Scrape Text, Input Text, Click Button)

Insert the data from the following excel file.

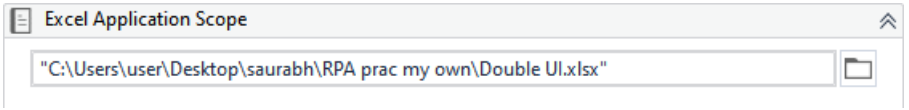
A	B	C	D	E	F	G
Sr no.	Cash In	Check_1	Check_2	Total	Transactic	Status
1	100	200	400			
2	200	300	600			
3	300	400	800			
4	400	500	1000			
5	500	600	1200			
6	600	700	1400			
7	700	800	1600			
8	800	900	1800			

Steps:

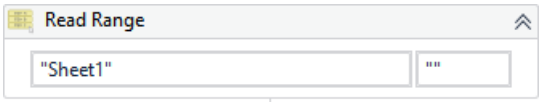
1. Create a new Blank Project and give it an appropriate name. Drag a Sequence activity from Activity tab.



2. Add Excel Application Scope Activity and enter the path of the excel file.

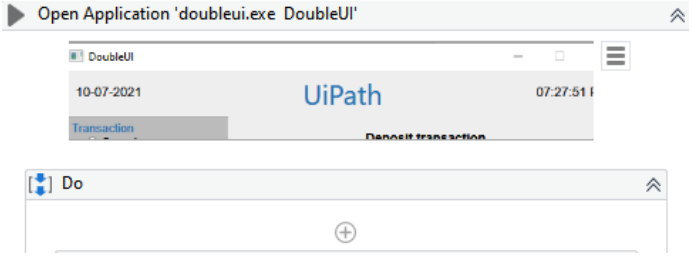


3. Add Read Range and create a DataTable to store the data that will be read by read range.

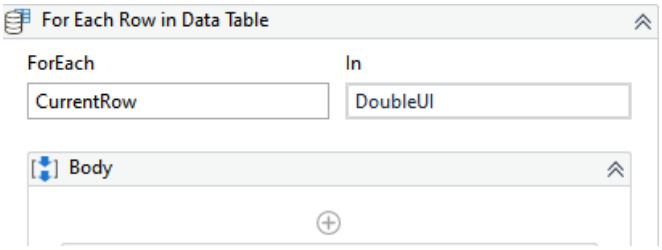


Name	Variable type	Scope	Default
DoubleUI	DataTable	Saurabh Yadav 4B Sequence	Enter a VB expression
transactionid	String	Saurabh Yadav 4B Sequence	Enter a VB expression
total	String	Saurabh Yadav 4B Sequence	Enter a VB expression

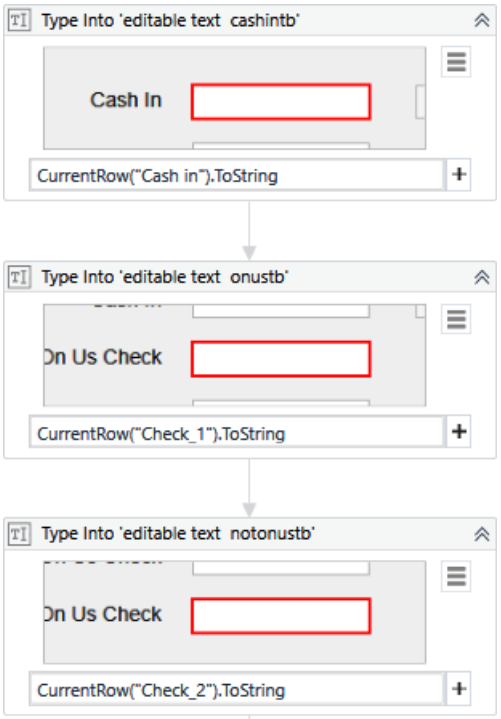
4. Add Open Application activity, start Double UI then click on “Indicate window on screen” option and select Double UI.



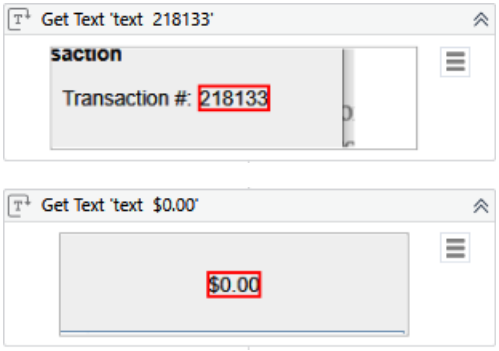
5. Add For each row in DataTable activity and iterate over DoubleUI datatable.



6. Add three type into activity in the for each activity. Indicate to the three textboxes in the Double UI app. Enter the DataTable element you wan to enter in the Text attribute.

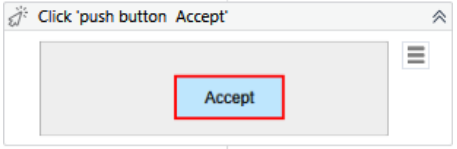


7. Add two get text activities and create two variables to store the transaction is andtotal. Indicate these elements to the Get text activities.



In case of Total remove the name attribute from the Edit Selector.

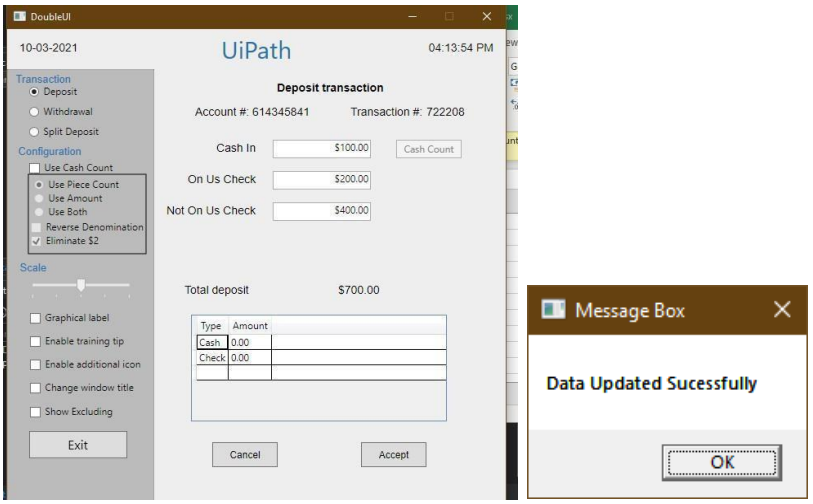
8. Add Click activity and indicate to Accept Button in Double UI.



9. Add three Write Cell Activities to enter the transaction id and total and to Update the status column. ("E"+(DoubleUI.Rows.IndexOf(CurrentRow)+2).ToString)

10. Finally add a Message Box to print the success of the updation of Excel file.

Output:

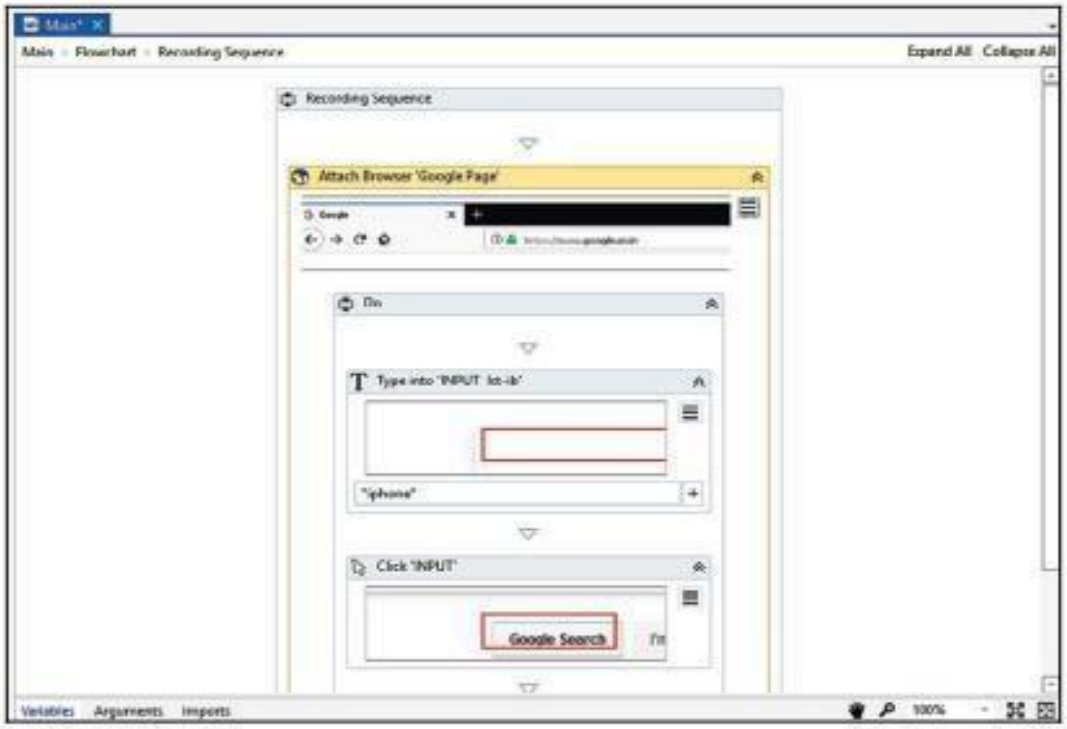


	A	B	C	D	E	F	G
1	Sr no.	Cash In	Check_1	Check_2	Total	Transactic	Status
2	1	100	200	400	700	722208	Done
3	2	200	300	600	1100	722209	Done
4	3	300	400	800	1500	722210	Done
5	4	400	500	1000	1900	722211	Done
6	5	500	600	1200	2300	722212	Done
7	6	600	700	1400	2700	722213	Done
8	7	700	800	1600	3100	722214	Done
9	8	800	900	1800	3500	722215	Done

e. Web Recording e.g. Find the rating of the movie from imdb web site

Web Recording can be done by using the Web recorder. For recording web actions, the UiPath extension for that browser should be installed. Otherwise, you will not able to automate tasks or actions using Web recording. You just have to click on the Setup icon and then click on Setup Extensions. Now, choose your browser and click on it. The UiPath extension will be added to your specified browser. Web Recording is similar to Desktop Recording. You just have to record the actions and save it.

Create a Blank project. Drag and drop a Flowchart activity. Now, click on the Recording icon and choose Web recording. You can record your actions on the web on your own and then save it. In our case, we have opened a web page using Google Chrome and logged in to www.google.com Then, we started the recording by clicking on the Record button of the web recorder. Next, we typed some text in the search bar of Google and performed the Click activity. Then, we pressed the Esc key to exit the recording and clicked on the Save and Exit button. Now, a recording sequence is generated in our Designer panel. Connect this sequence to the Start node. Hit the Run button to see the result. In the following screenshot, you can see the sequence generated by the Web recorder:



Before running the UiPath workflow, make sure you are on the Google homepage.

We have seen Web recording and it is very easy. There is also another option to extract information from websites.

We can easily extract information from websites using data scraping.

Suppose we want to extract data from Amazon's website. Say we want to search for books on Amazon and extract the search results. Extracting data from websites becomes very easy with data scraping:

1. Create a blank project and give it a meaningful name. Click Create.

2. Log on to Amazon's website and search for books. A detailed list of books is listed on your screen:

3. Drag and drop a Flowchart activity on the Designer panel. Now, click on the Data Scraping icon. A window will pop up.

4. Click on the Next button.

5. You have to indicate the first book's entities. Entities can be name, price, author, and so on. It is your choice.

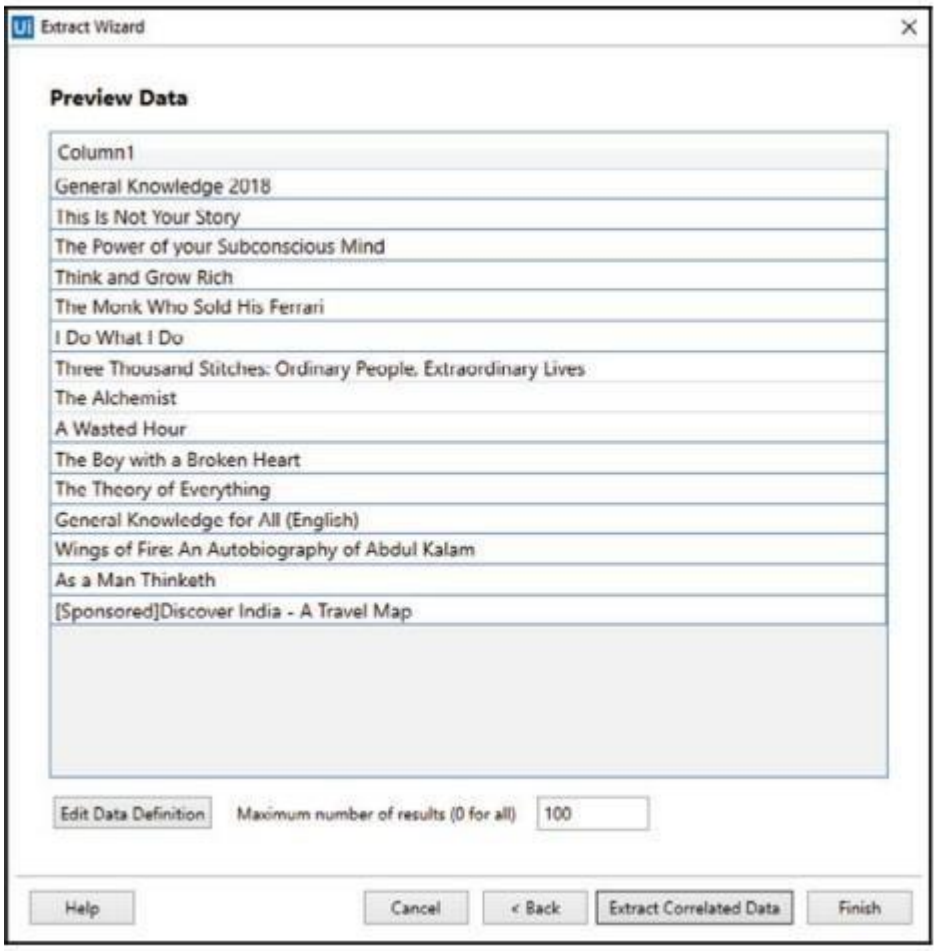
6. Lets's, indicate the book's name. After that, it will ask for the next book's entities. Indicate the second book entity as well. Click on Next.

7. This means you have to indicate the second book's entities: however, the entities will be the same. If you choose name as the first book's entity then you have to be specific and choose name as the second book's entity. You should not choose name as the first book's entity and then choose price as the second book's entity.

8. Again, a window will pop up asking you to configure the columns. You can also extract the URL. If you want to do this, check the Extract URL checkbox.

9. You can specify the column name as well. Click on the Next button.

10. As you can see, all the book names are extracted to a window. If you want to extract more columns or more entities, then click on Extract Correlated Data and you have to again indicate another entity of the book to extract more columns, as we have done previously. After that, all the data will be extracted and will be added to this table. Here, we have one column but if you extract more entities, then more columns will be added to this table:



11. Click on the Finish button. If the results of your query span multiple pages, it will ask you to indicate the page navigation link on the website (Next button of the website that we used to navigate to another/next page). If the results of your query span multiple pages, click on the Yes button and indicate the link, otherwise click on the No button.

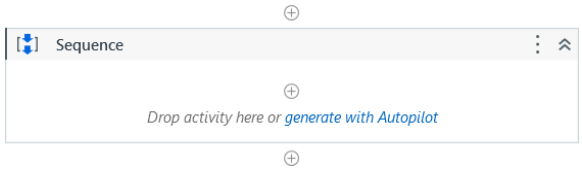
12. We have clicked on the No button. A data scraping sequence is generated in our Flowchart. It will also generate a data table. You can retrieve the information from the data table

f. Web Recording manually

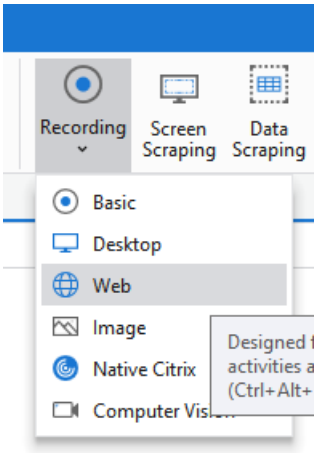
Use "google.com" and automate stuff (Scrape Text, Input Text, Click Button)

Steps:

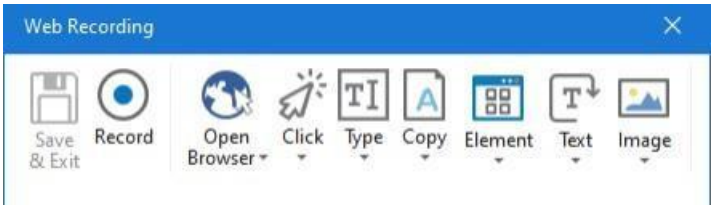
1. Create a new Blank Project and give it an appropriate name. Drag a Sequence activity from Activity tab.



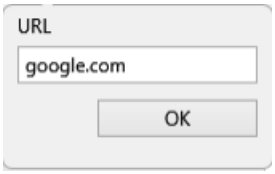
2. Select web recording under the recording option.



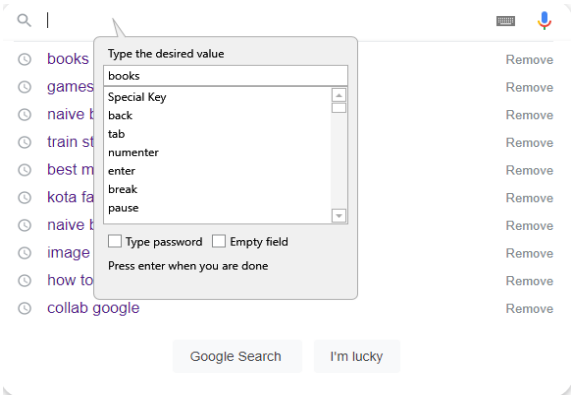
3. Open a Chrome window and from the web recording dialog select open browser and click on the Chrome window.

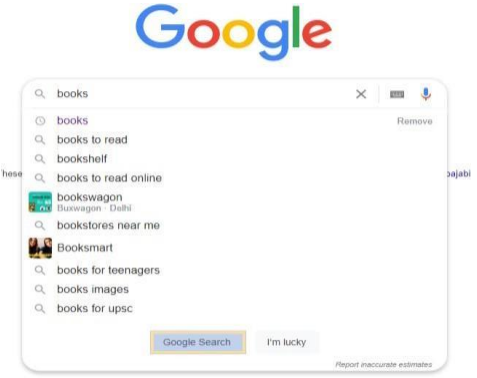


4. In the url dialog enter google.com and press enter.

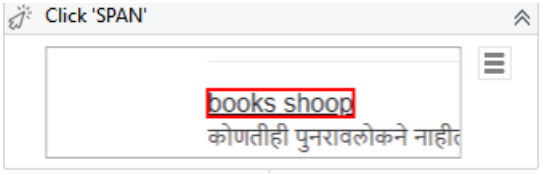


5. Now start recording by clicking on Record and click on the search button. Type the desired search and click on the Google search button. The press Esc.

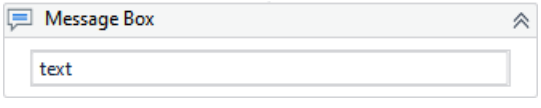


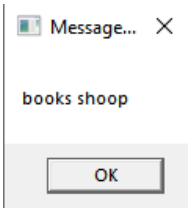


6. Now press save and exit. Add get text activity and select the text you want to scrape.



7. Add message box to print scrapped text.





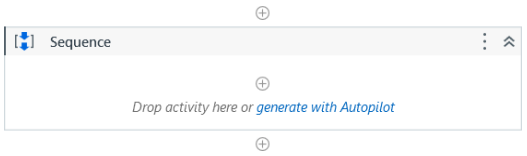
Practical 4 : Excel Automation

a. Automate the process to extract data from an excel file into a data table and vice versa

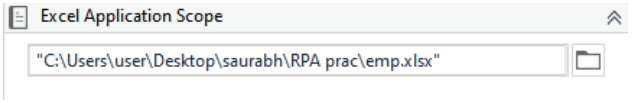
Emp_Id	F_name	L_name	Salary	Expenses	Saving
1001	Saurabh	Yadav	100000	10000	90000
1002	Pankaj	Gavali	200000	20000	180000
1003	Bharat	Bhagat	300000	30000	270000
1004	Virat	Kohli	400000	40000	360000
1005	Mahendra	Dhoni	500000	50000	450000
106	Raju	Pal	60000	1500	58500
107	Mihir	Gandhi	70000	9600	60400

Step:

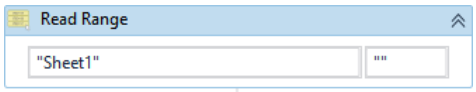
1. Start project with sequence.

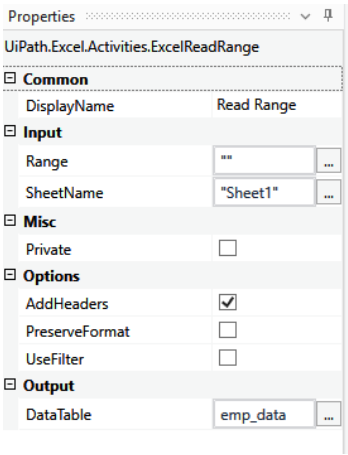


2. Add “Excel Application Scope” and enter the path of excel file

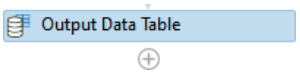


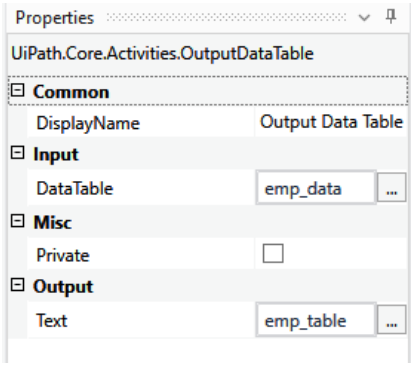
3. Inside body add “Read Range” variable name and datatype as datatable(emp_data) and range to extract data from excel (blank means entire sheet)





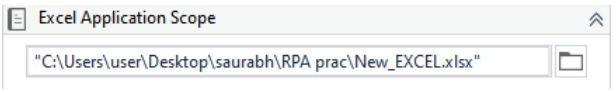
4. Add “Output Data Table” input as “emp_data” output as “emp_table”.



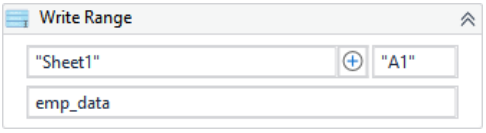


5. "Message Box" to print the result adding variable "emp_table" (here extracted data will be printed).

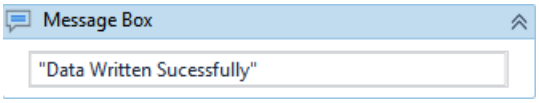
6. Insert "excel application scope" and new excel sheet path



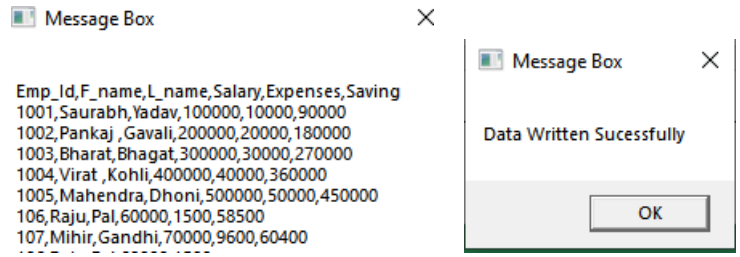
7. Insert "Write Range" range from where the data should be pasted and variable "emp_data".



8. Message Box to print confirmation.



Output :



	A	B	C	D	E	F	G
1	1001	Saurabh	Yadav	100000	10000	90000	
2	1002	Pankaj	Gavali	200000	20000	180000	
3	1003	Bharat	Bhagat	300000	30000	270000	
4	1004	Virat	Kohli	400000	40000	360000	
5	1005	Mahendra	Dhoni	500000	50000	450000	
6	106	Raju	Pal	60000	1500	58500	
7	107	Mihir	Gandhi	70000	9600	60400	
8	106	Raju	Pal	60000	1500		
9	107	Mihir	Gandhi	70000	9600		
10							

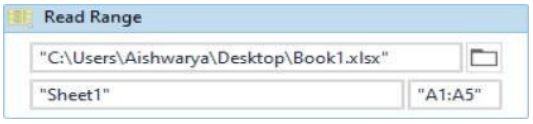
b,c,d . Create an application automating the read, write and append operation on excel file.

c. Read cell:

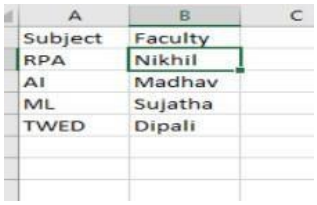
Step 1 : Drag and drop a flowchart activity.

Step 2: Drag and drop read range activity.

Step 3: Specify the path of excel sheet, sheet name as well as the range of cells to read.



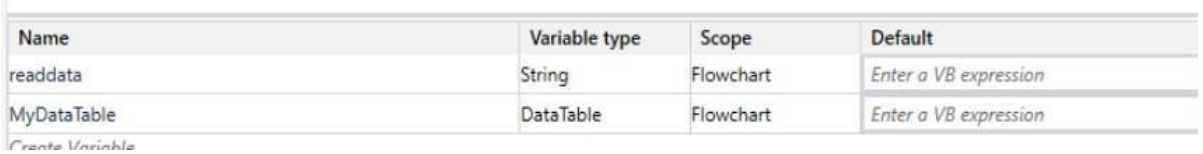
Value in excel:



	A	B	C
	Subject	Faculty	
	RPA	Nikhil	
	AI	Madhav	
	ML	Sujatha	
	TWED	Dipali	

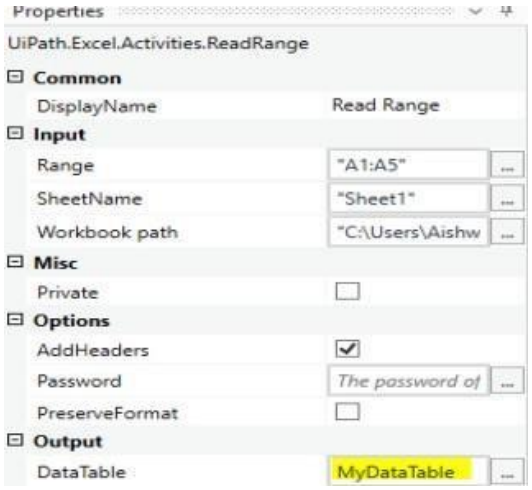
Step 4: Create a variable of type “string” to hold the read data.

Step 5: Create a datatable variable “MyDataTable” of DataTable type.



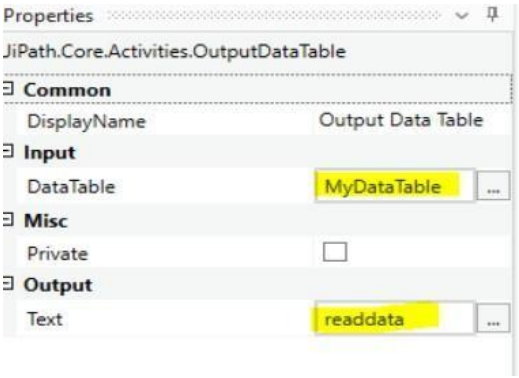
Name	Variable type	Scope	Default
readdata	String	Flowchart	Enter a VB expression
MyDataTable	DataTable	Flowchart	Enter a VB expression

Step 6: Set the output of Read Range activity to MyDataTable



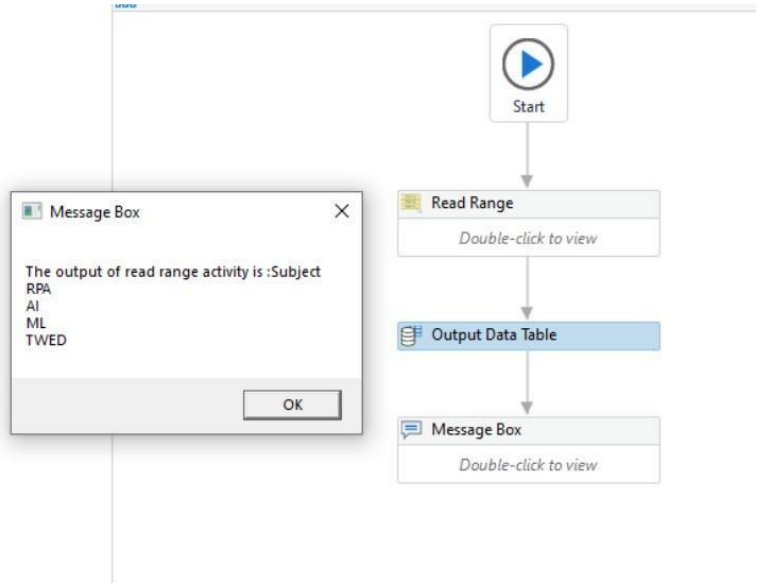
Step 7: Drag and drop Output Data table activity

Step 8: Select the Activity Output Data Table and set its input to “mydatatable” and output to “result”



Step 9: Select Message Box activity and configure it to display the “Result” variable contents.

Step 10: Run the file.



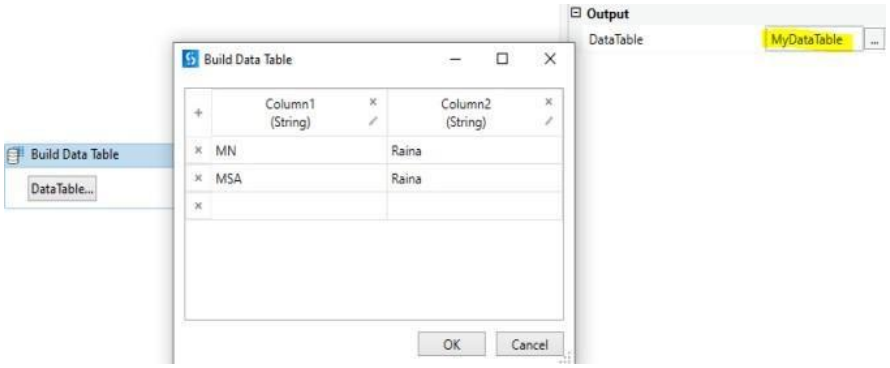
b. Write cell:

Step 1 : Drag and drop a flowchart activity.

Step 2: Select activity “Build Data Table” and enter some data into it

Step 3: Create a data table variable MyDataTable MyDataTable of DataTable type

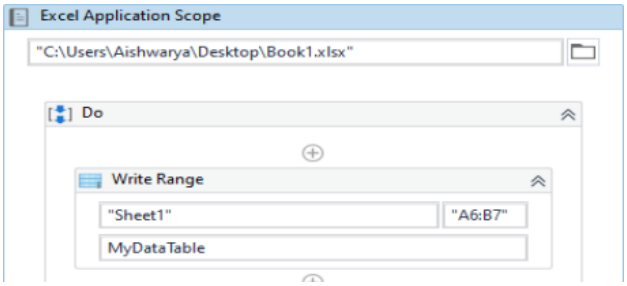
Step 4: Set the output of Build Data Table activity to MyDataTable



Step 5: Select Excel Application Scope .Specify the path of the excel sheet

Step 6: Select Write Range Activity.

Step 7: Specify the sheet name and range of cells to write Step 8: Set the data table value to MyDataTable



Step 9: Run

Output:

A	B
Subject	Faculty
RPA	Nikhil
AI	Madhav
ML	Sujatha
TWED	Dipali
MN	Raina
MSA	Raina

d. Append range:

Step 1 : Drag and drop a flowchart activity.

Step 2: Select activity “Build Data Table” and enter some data into it

Step 3: Create a data table variable MyDataTable MyDataTable of DataTable type

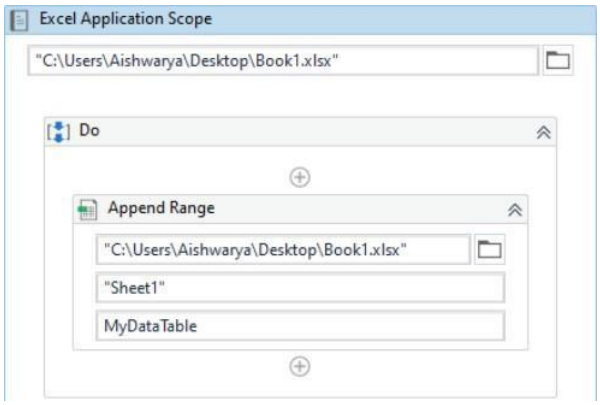
Step 4: Set the output of Build Data Table activity to MyDataTable

Step 5: Select Excel Application Scope .Specify the path of the excel sheet

Step 6: Select Append Range Activity.

Step 7: Specify the sheet name and range of cells to write

Step 8: Set the data table value to MyDataTable



Step 9: Run

Output:

A	B
Subject	Faculty
RPA	Nikhil
AI	Madhav
ML	Sujatha
TWED	Dipali
MN	Raina
MSA	Raina
MN	Raina
MSA	Raina

Practical 5 : Different controls in UiPath

a. Implement the attach window activity.

Step 1. Create a blank project and give it a meaningful name.

Step 2: Drag and drop a Sequence activity on the Designer panel. Also, drag and drop a Click activity inside the Designer panel.

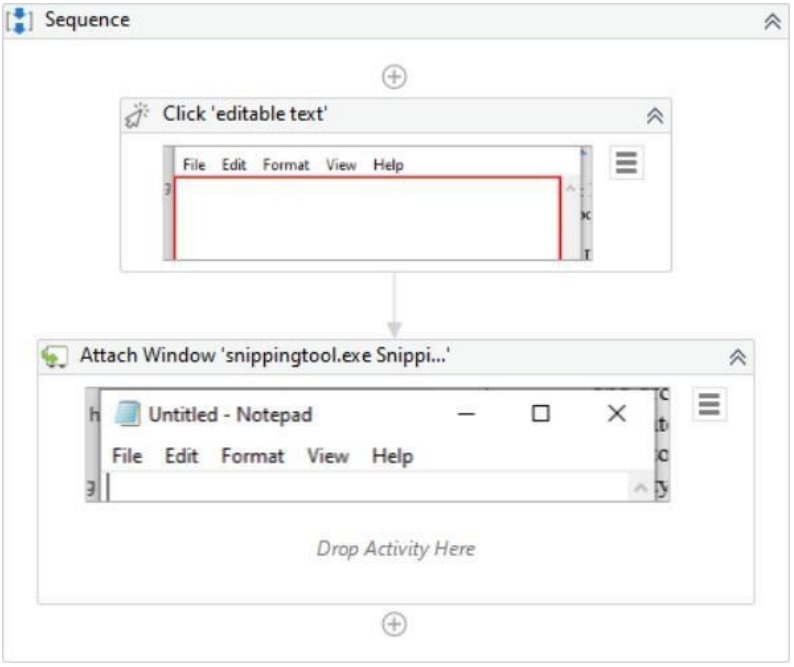
Step 3 : Open a notepad

Step 4: Double-click on the Click activity and then click on Indicate on screen. Locate the Notepad icon.

Step 5: Drag and drop the Attach Window activity on the main Designer panel.

Step 6: Double-click on the Attach Window activity. Click on Click Window on Screen and indicate the Notepad window

Output:



b. Automate using Anchor Base

This control is used for locating the UI element by looking at the UI element next to it. This activity is used when we have no control over the selector. That means when we do not have a reliable selector, then we should use the Anchor base control to locate the UI element.

We can use the Anchor base control as explained in the following section:

1. Drag and drop a Flowchart activity on the Designer panel of a blank project. Also, drag and drop an Anchor base control from the Activities panel. Connect the Anchor base control with Start.

2. Double-click on the Anchor base control:

3. There are two activities that we have to supply to the Anchor base control: Anchor and action activities.

4. Drag and drop the Anchor base activity (for example; Find Element activity) in the Anchor field and Action activity (for example; Type into) in the Drop Action Activity Here field of the Anchor base control.

The Anchor base activity will find the relative element nearby the element on which you want to perform the Action, and the Action activity will perform the appropriate action that you have specified

C. Automate using Element Exists.

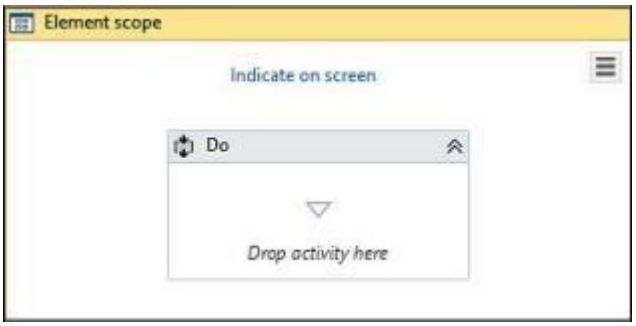
This control is used to check the availability of the UI element. It checks if the UI element Exists or not. It also returns a Boolean result if the UI Element Exists, then it returns true: otherwise, it returns false.

You can use this control to check for the UI element. In fact, it is good practice to use this control for UI elements whose availability is not confirmed or those that change frequently

Just drag and drop the Element Exists control from the Activities panel. Double-click on it. You can see there is an Indicate on screen option. Click on it to indicate the UI element. It returns a Boolean result, which you can retrieve later from the Exists property. You just have to supply a Boolean variable in the Exists property in the Properties panel.

Element scope

This control is used to attach a UI element and perform multiple actions on it. You can use a bunch of actions within a single UI element. Drag and drop the Element scope control and double-click on this control:

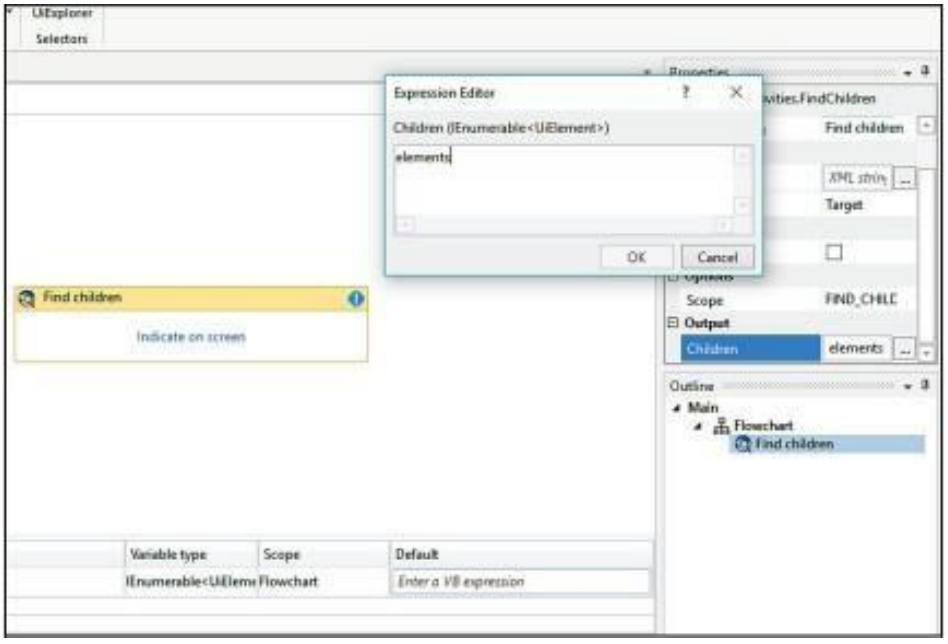


You can clearly see that you have to indicate the UI element by clicking on Indicate on screen and specifying all the actions that you want to perform in the Do sequence. You can add many activities inside the Do sequence.

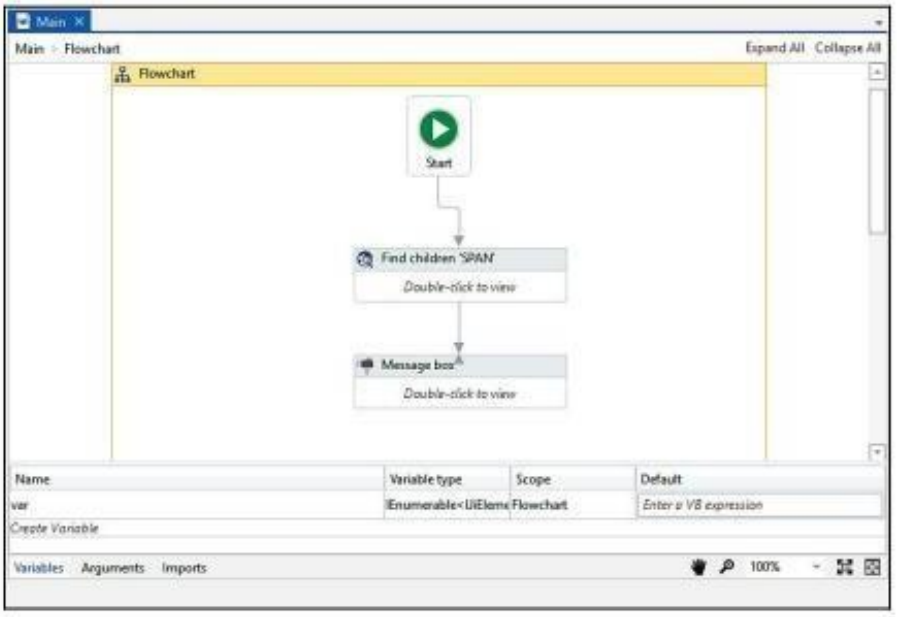
d. Automate using Find Children control

This control is used to find all the children UI elements of a specified UI element. It also retrieves a collection of children UI elements. You can use a loop to inspect all the children UI elements or set up some filter criteria to filter out the UI elements.

Drag and drop the Find children control from the Activities panel. Double-click on it to indicate the UI element that you want to specify. You can indicate it by clicking on Indicate on screen:



You have to supply a variable of type *&OVNFSBCMF-6*&MFNFOUT in the children property, as mentioned in the preceding screenshot. This variable is then used for retrieving the UI elements:



Find element

This control is used to find a particular UI element. It waits for that UI element to appear on the screen and returns it back.

You can use this control in the same way that you used the other controls. Just drag and drop this control, and indicate the UI element by clicking on Indicate on screen.

You can specify the variable of type UI element in the Found element property of the Find Element control to receive the UI element as output.

Find relative element

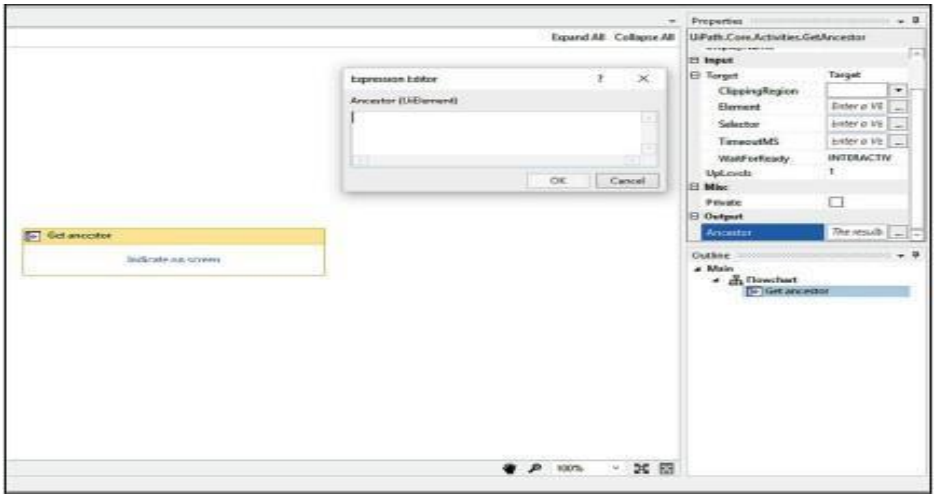
This control is similar to the Find element control. The only difference is that it uses the relative fixed UI element to recognize the UI element properly. This control can be used in scenarios where a reliable selector is not present. Just drag and drop this control, and indicate the UI element by clicking on Indicate on screen. You can also look for its selector property after indicating the UI element for better analysis.

e. Use Get Ancestor control

Get ancestor

This control is used to retrieve the ancestor of the specified UI element. You have to supply a variable to receive the ancestor element as output. You can specify the variable name in the Ancestor property of the Get ancestor control.

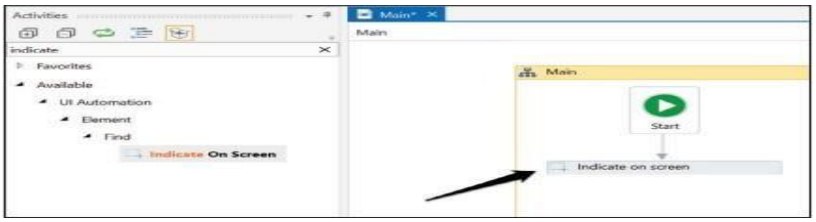
After receiving the ancestor element, you can retrieve its attributes, properties, and so on for further analysis.



Just drag and drop this control and indicate the UI element by clicking on Indicate on screen.

Indicate on screen

This control is used to indicate and select the UI element or region at runtime. It gives flexibility to indicate and select the UI element or region while running the workflow. You just have to drag and drop this control in your project:



Do not confuse this with Indicate on screen written inside any activity like Type into. In previous examples, we have used Indicate on screen inside various controls (as shown in the following screenshot). This button is used to locate the region or UI element before the execution of the workflow, while the Indicate on screen control executes its process after the execution of the workflow:

Practical 6 : Keyboard and Mouse Events

a. Demonstrate the following activities in UiPath:

i. Mouse (click, double click and hover)
ii. Type into
iii. Type Secure text

.i. Mouse (click, double click and hover)

1. Click Activity

Step 1: Go to uipath studio

Step 2: Click on Workflow and Drag Flow Chart from Activities panal.

Step 3: Double Click on FlowChart.

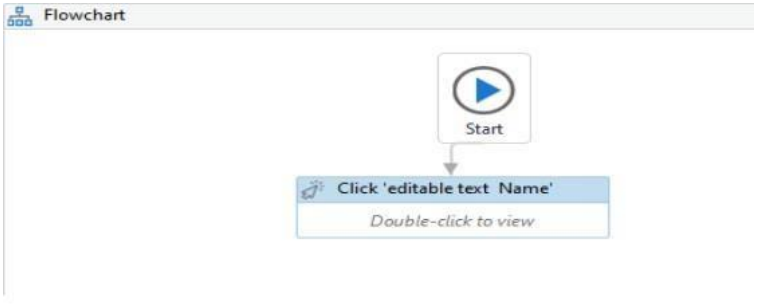
Step 4 : Drag and drop the Click activity

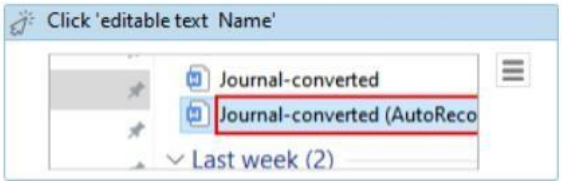
Step 5: Double click on click activity.

Step 6: Click on indicate on screen and indicate the UI element you want to click on.

Step 7: Now click on run.

Output:





2. Double Click Activity

Step 1: Go to uipath studio

Step 2: Click on Workflow and Drag Flow Chart from Activities panal.

Step 3: Double Click on FlowChart.

Step 4 : Drag and drop the Double Click activity

Step 5: Double click on double click activity.

Step 6: Click on indicate on screen and indicate the UI element you want to click on.

Step 7: Now click on run.

Output:

3. Hover Activity

Step 1: Go to uipath studio

Step 2: Click on Workflow and Drag Flow Chart from Activities panal.

Step 3: Double Click on FlowChart.

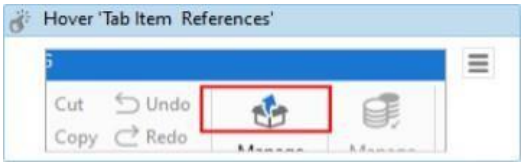
Step 4 : Drag and drop the Hover activity

Step 5: Double click on hover activity.

Step 6: Click on indicate on screen and indicate the UI element you want to click on.

Step 7: Now click on run.

Output:



ii Type into

Step1: Add a new sequence and name it as Type into Activity

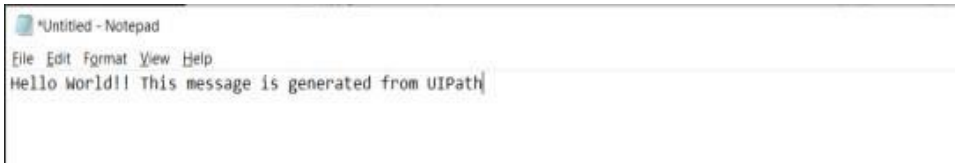
Step2: Search for Type into Activity in the activity panel and drag it inside the sequence.

Step3: Click on Indicate on Screen and indicate the pointer towards notepad editor.

Step4: Type the message to be printed on the notepad in the editable text section

Step5: Hit the Run Button to see the results.

Output:



iii Type secure text

Step1: Add a new flowchart

Step2: Search for Type secure text Activity in the activity panel and drag it inside the sequence.

Step3: Click on Indicate on Screen and indicate the pointer towards notepad editor.

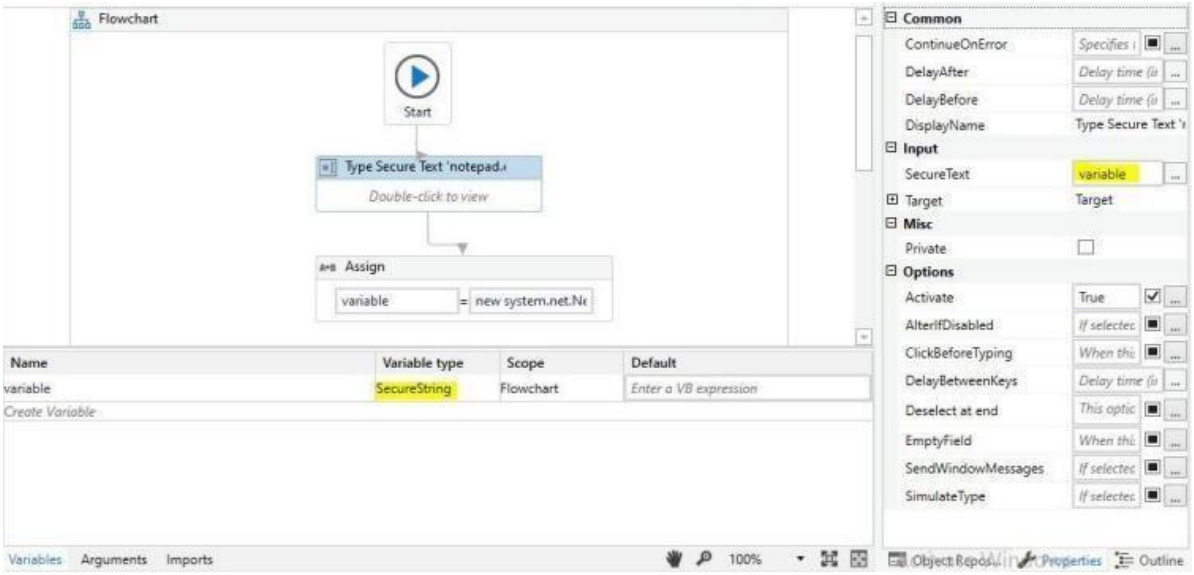
Step4: Create a variable of type “securestring” and assign it to SecureText property in the properties.

Step5:Drag and drop a assign activity.

Step6: Assign the variable created with this value “new system.net.NetworkCredential(String.Empty,"Test@123").SecurePassword”

Step7: Hit the Run Button to see the results.

Output:



b. Demonstrate the following events in UiPath:

i. Element Trigger

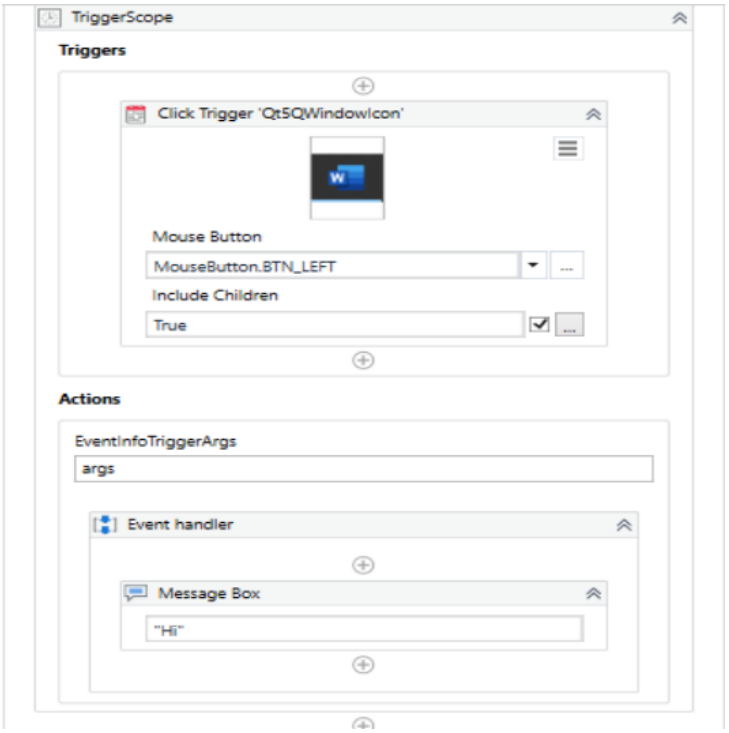
On click element

Steps1: Click on user events select on click element

Step 2: Single click on icon present in the toolbar.

Step 3: Drag and drop message box inside event handler.

Step 4: Write any message you want to display.



Step 5: Run the file.

Step 6: Click the icon which you selected in user event.

Output:

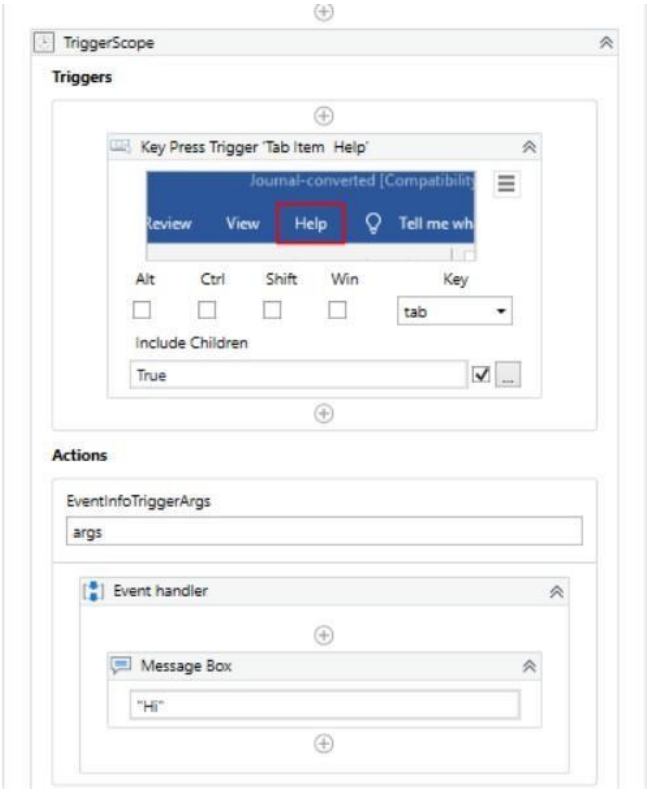
On keypress event

Step1: Click on user events select on key press element

Step 2: Indicate on screen and select the key.

Step 3: Drag and drop message box inside event handler.

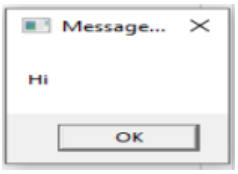
Step 4: Write any message you want to display.



Step 5: Run the file.

Step 6: Click the key which you selected in user event.

Output:



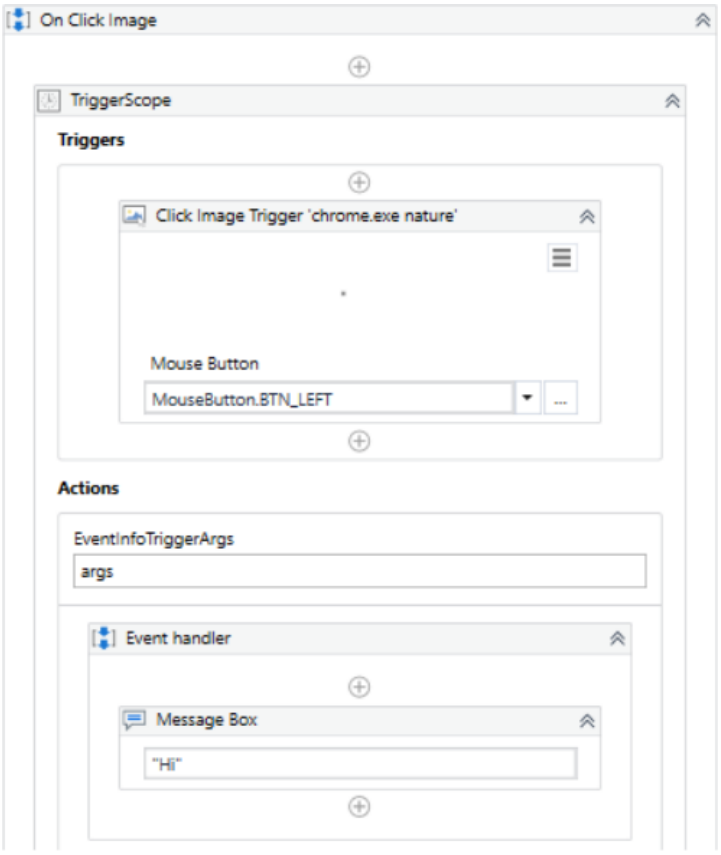
ii. Image trigger

Steps1: Click on user events select on click image element

Step 2: Single click on any image present in the screen.

Step 3: Drag and drop message box inside event handler.

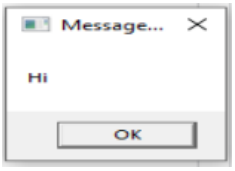
Step 4: Write any message you want to display.



Step 5: Run the file.

Step 6: Click the image which you selected in user event.

Output:



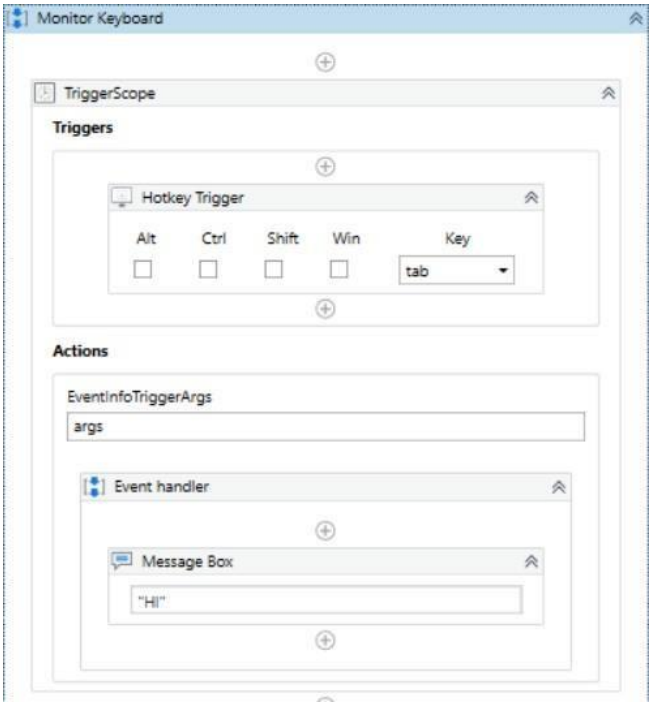
iii. System trigger

Step1: Click on user events select on monitor keyboard element

Step 2: Select any key.

Step 3: Drag and drop message box inside event handler.

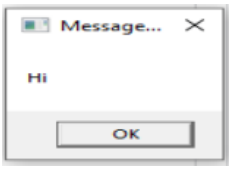
Step 4: Write any message you want to display.



Step 5: Run the file.

Step 6: Click the key which you selected in user event.

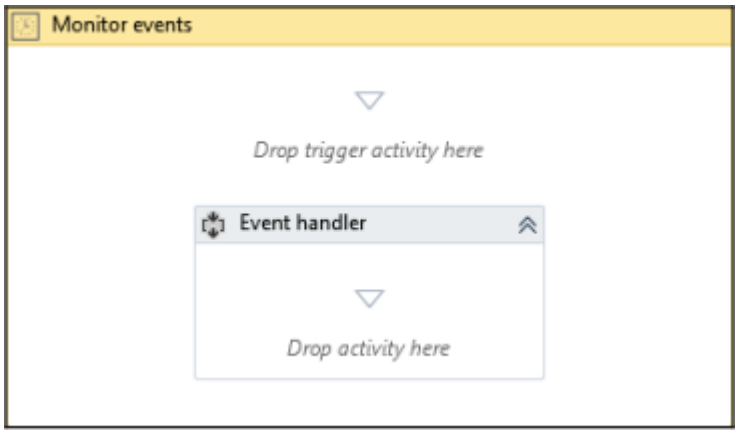
Output:



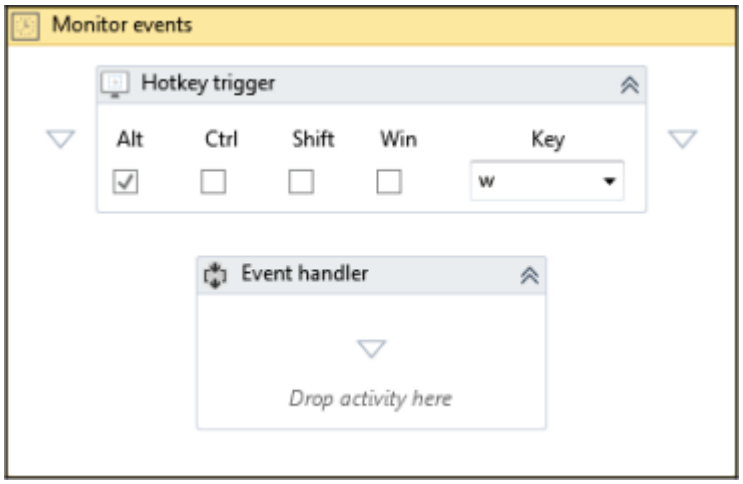
C. Automate the process of launching an assistant bot on a keyboard event.

Let us say we want our assistant bot to start automating only when we trigger an event. For example, the user wants his Robot to open and start typing in the Notepad window when he presses Alt + W. This can be achieved using the Hotkey trigger. Also, inside the Event handler, just create or record the sequence of steps to be followed. The detailed procedure has been explained in the following sections:

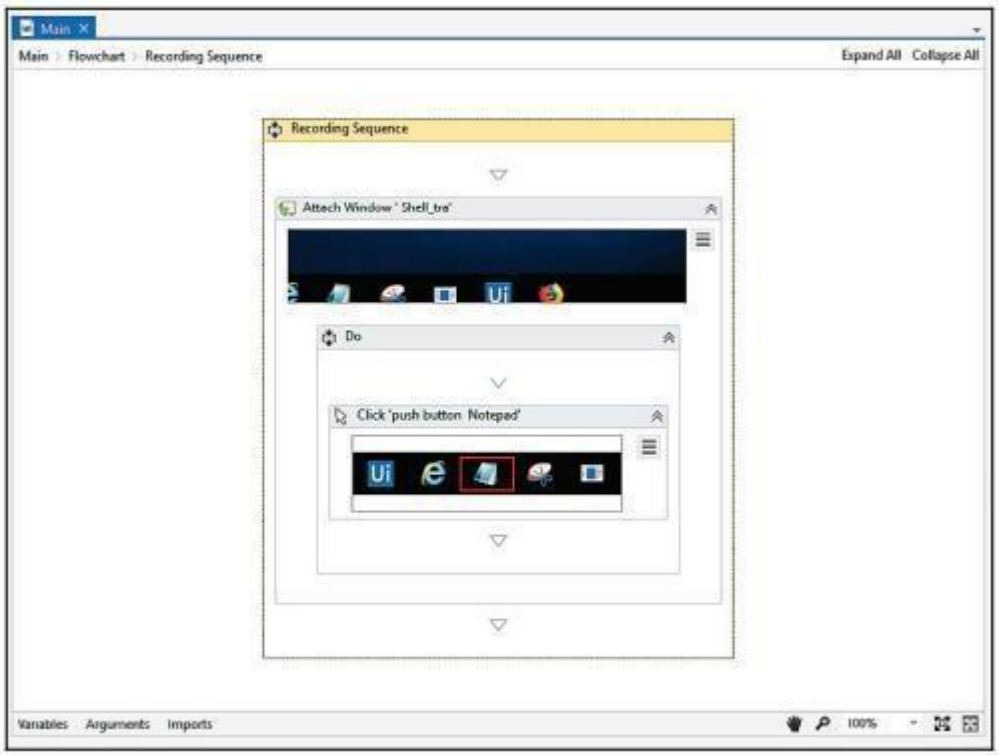
1. Drag and drop the Monitor events activity: In this step, we will just drag and drop the Monitor events activity into the workflow. When we double-click on it, it will look like this:



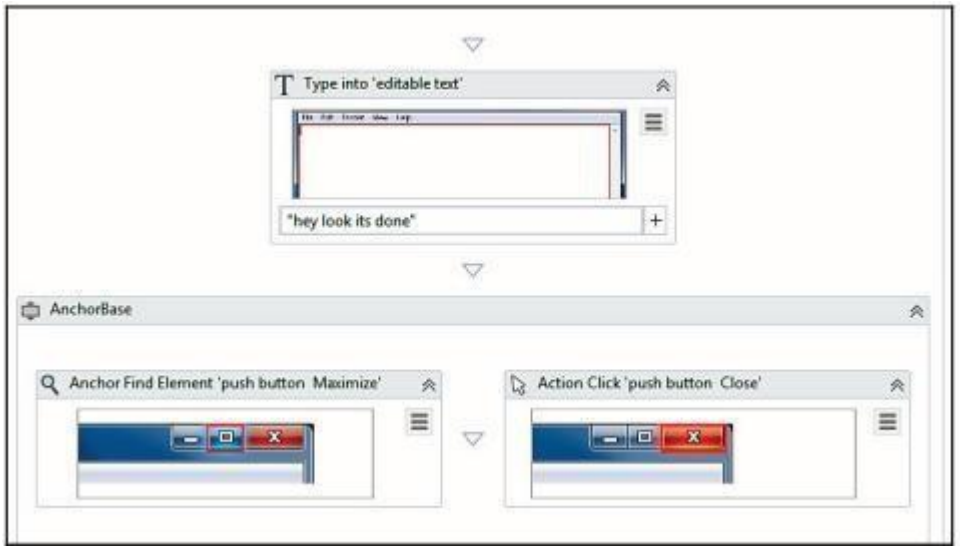
2. Drag the Hotkey trigger activity: In the next step, we will use the Hotkey trigger activity for the user to start the automation process. Assign Alt + W to the hotkey so that, when the user presses this hotkey, the event will be executed:



3. Open Notepad and type into it: Our final step is to record the sequence of the steps to be performed. In this case, this is to open Notepad and then type into it. For that just use the help of the Desktop recorder. First, we double-click on the Notepad application in the window as shown in the screenshot. Select the ClickType as CLICK_DOUBLE from the Properties panel:



After that, we record the typing action and close the Notepad window. Then click on Do not Save because you do not want to save your file. The sequence is shown in the following screenshot:



We have also indicated the anchor to recognize the correct button to be clicked (in this case, the close window button's anchor is the maximize button). This makes it easier for the Robot to find the UI element.

Now, on pressing Alt + W the Robot will start executing the sequence.

Practical 7 : Screen Scraping and Web Scraping methods

a. **Automate the following screen scraping methods using UiPath:**
   A. **Full Text**
   B. **Native**
   C. **OCR**

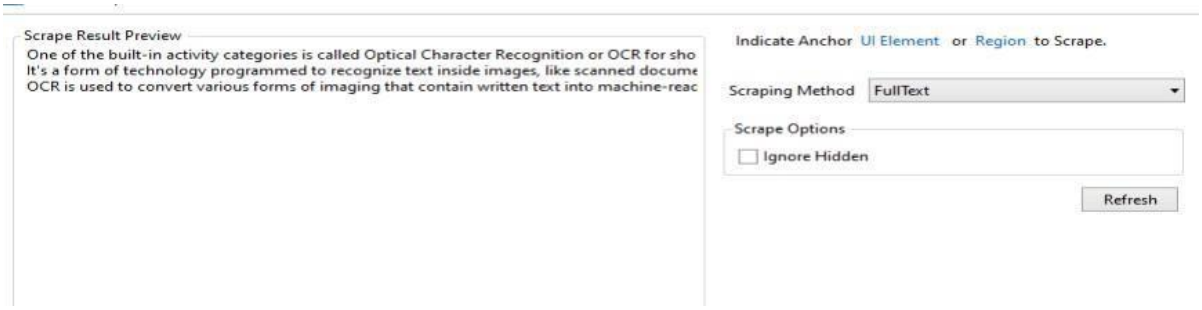
Full Test

Step1: First we select a new project and give it a name.

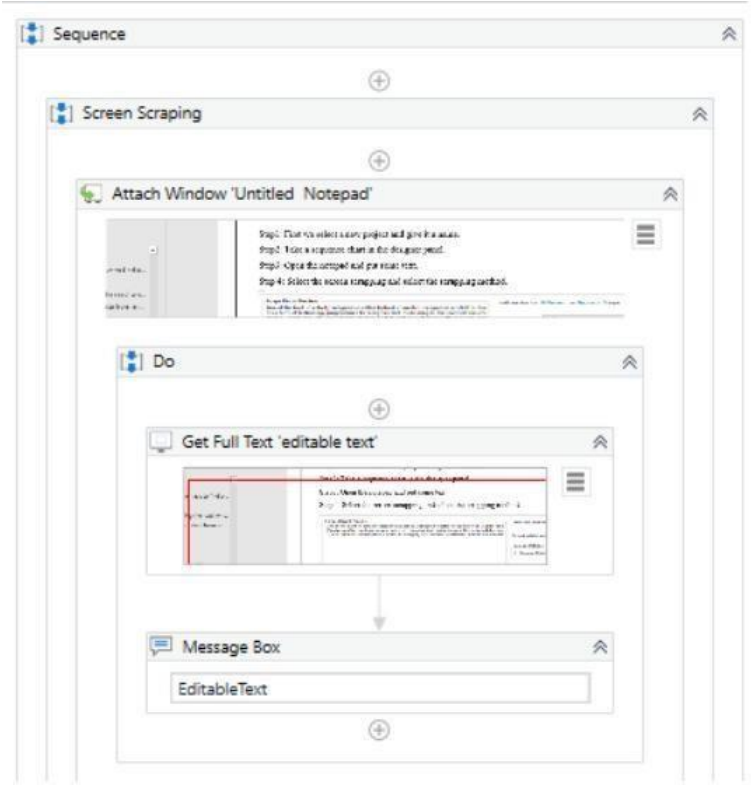
Step2: Take a sequence chart in the designer panel.

Step3: Open the notepad and put some text.

Step 4: Select the screen scrapping and select the scrapping method.

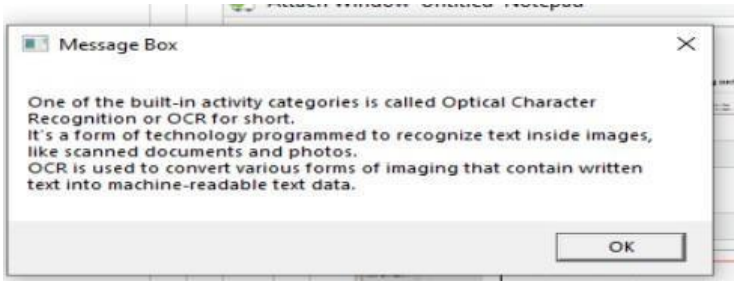


Step 5: Drag and drop a message box and give the variable name automatically created.



Step 6: Run the file.

Output:



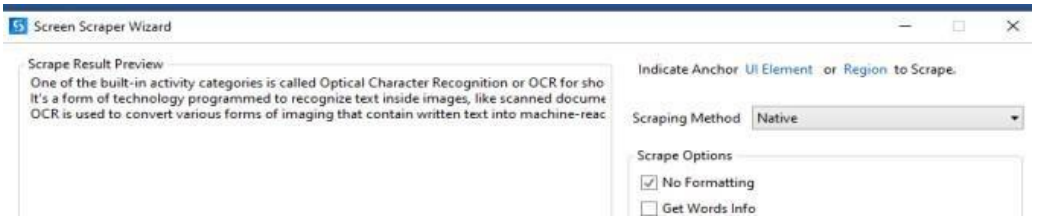
Native

Step1: First we select a new project and give it a name.

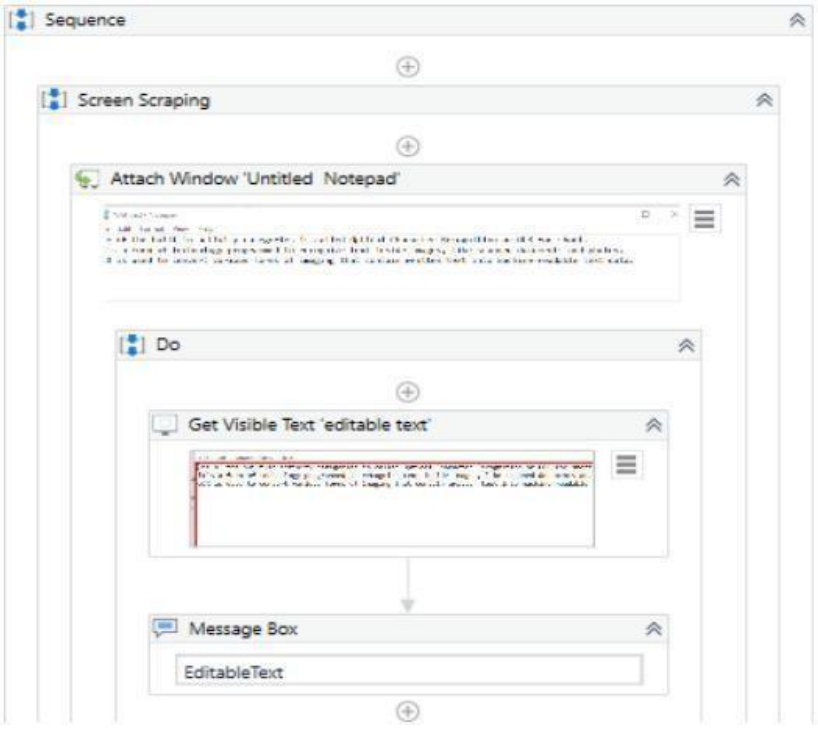
Step2: Take a sequence chart in the designer panel.

Step3: Open the notepad and put some text.

Step 4: Select the screen scrapping and select the scrapping method

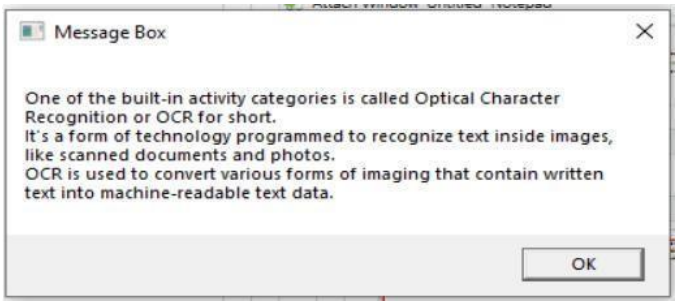


Step 5: Drag and drop a message box and give the variable name automatically created.



Step 6: Run the file.

Output:



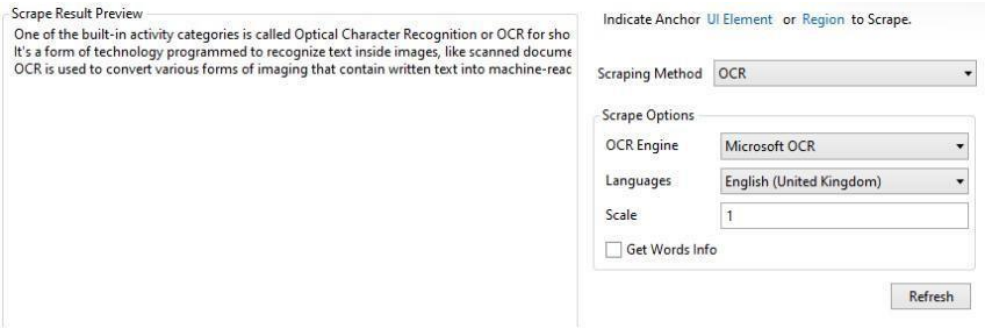
OCR

Step1: First we select a new project and give it a name.

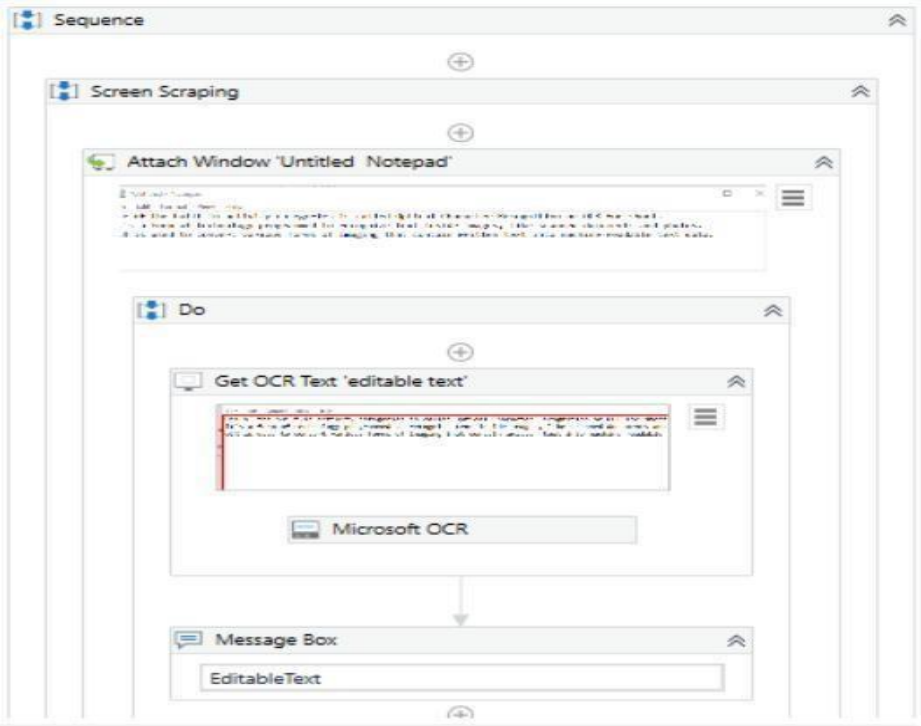
Step2: Take a sequence chart in the designer panel.

Step3: Open the notepad and put some text.

Step 4: Select the screen scrapping and select the scrapping method.

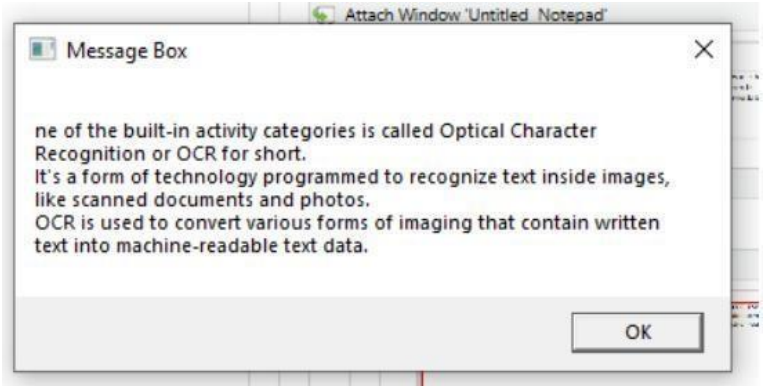


Step 5: Drag and drop a message box and give the variable name automatically created.



Step 6: Run the file.

Output:



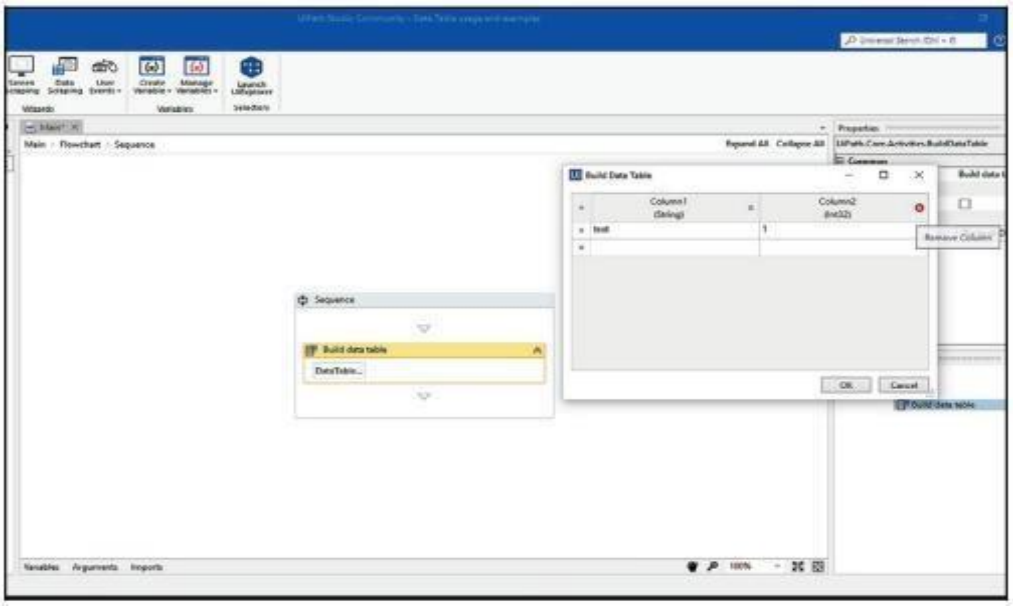
b. Demonstrate Data Scraping and display values in Message box.

Let us see, how to build a data table can be built. First, create an empty project. Give it a proper name:

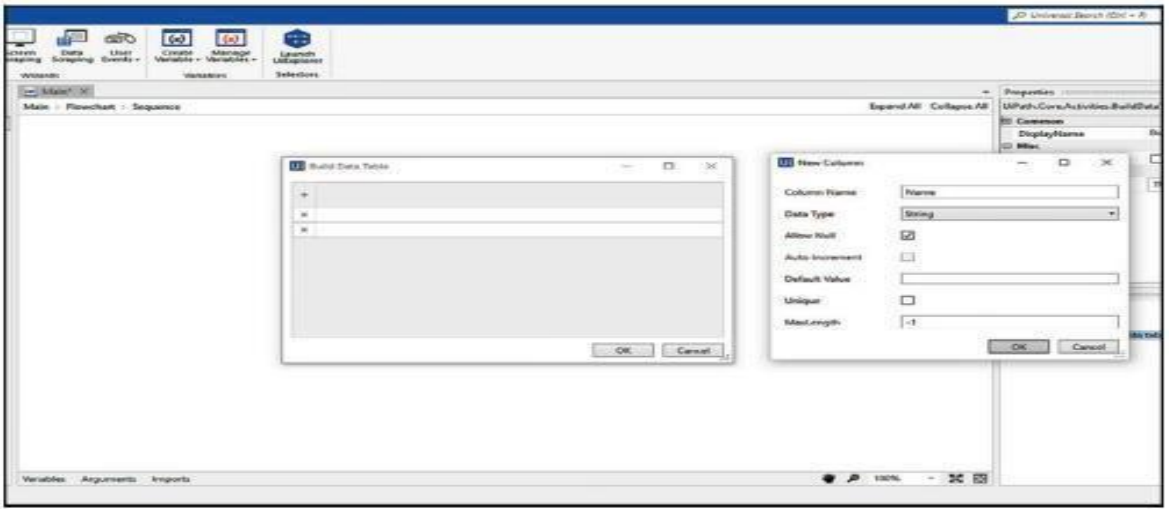
1. Drag and drop a Flowchart activity on the Designer panel. Also, drag and drop a Sequence activity and set it as the Start node.

2. Double click on the Sequence and drag and drop the Build Data Table activity inside the Sequence activity.

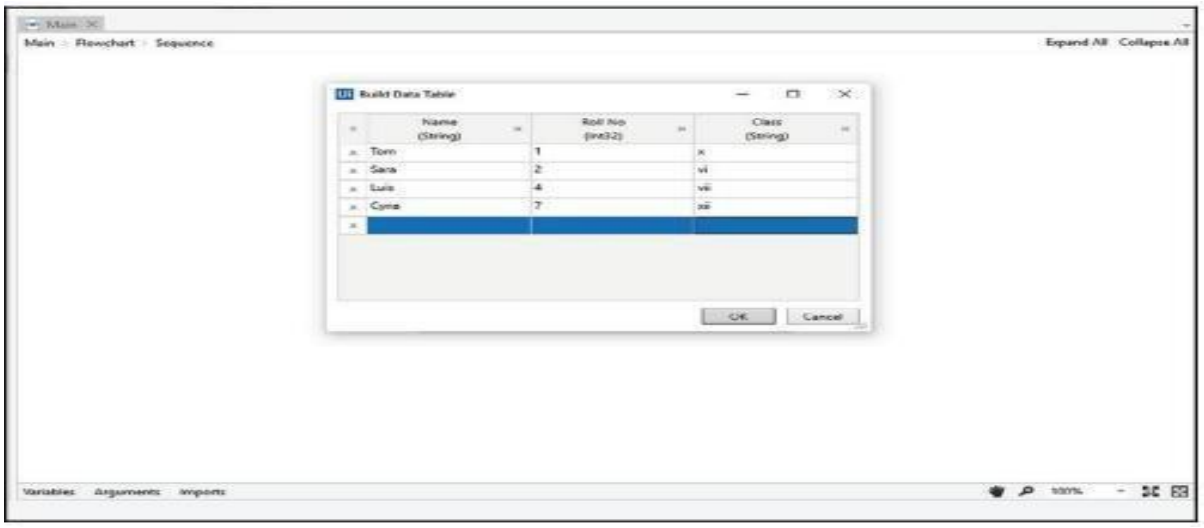
3. Click on the Data Table button. A pop-up window will appear on the screen. Remove both the columns (auto generated by the Build Data Table activity) by clicking on the Remove Column icon:



Now, we will add three columns by simply clicking on the + symbol. Specify the column names and select the appropriate data types from the drop-down list. Click on the OK button. We will add column /BNF of String Data Type, 3PMM@/P of Int32 type and finally Class of string type:



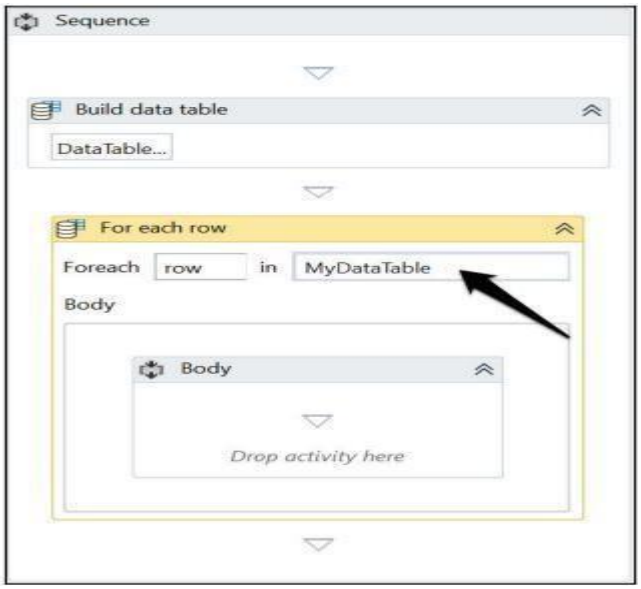
Now enter some random values just to insert the data into the rows:



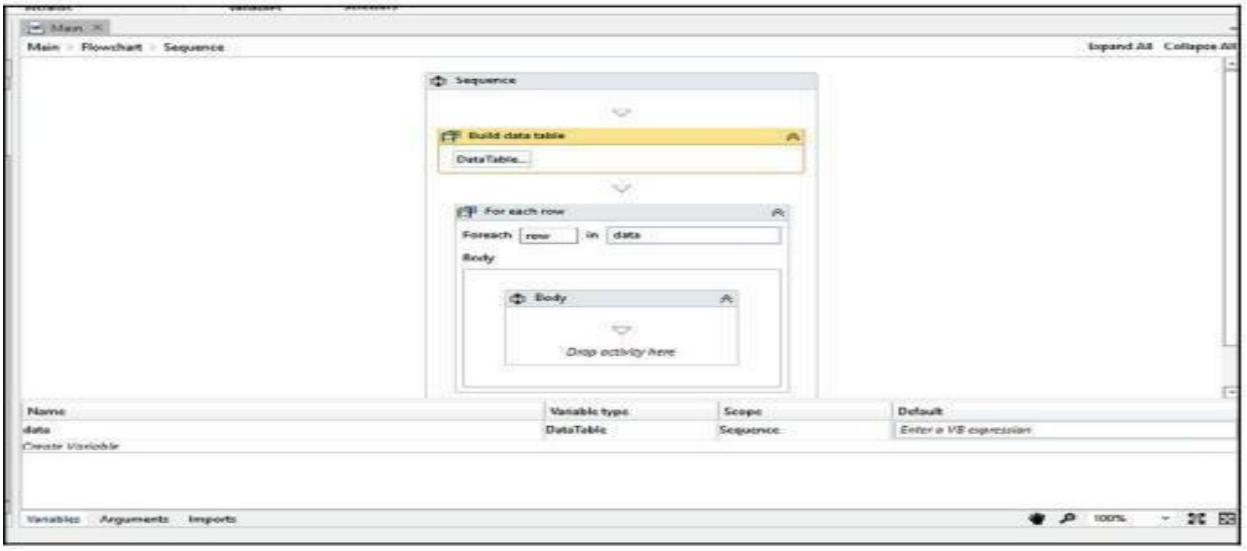
Click on the OK button and our data table is ready. We have to iterate over the data table's rows to make sure everything works correctly.

5. In order to store the Data Table created by Build Data Table activity, we have to create a data table variable Mydatatable of DataTable type and in order to store the result of the data table that we have dynamically built. Also, specify assign the Output property of the Build Data Table activity with this variable. Specify the data table variable's name there.

6. After our data table is ready, we will iterate the data table's rows to make sure everything works correctly. Drag and drop the For each row activity from the Activities panel inside the Sequence activity. Specify the data table variable's name (MyDataTable) in the expression text box of the For each row activity:

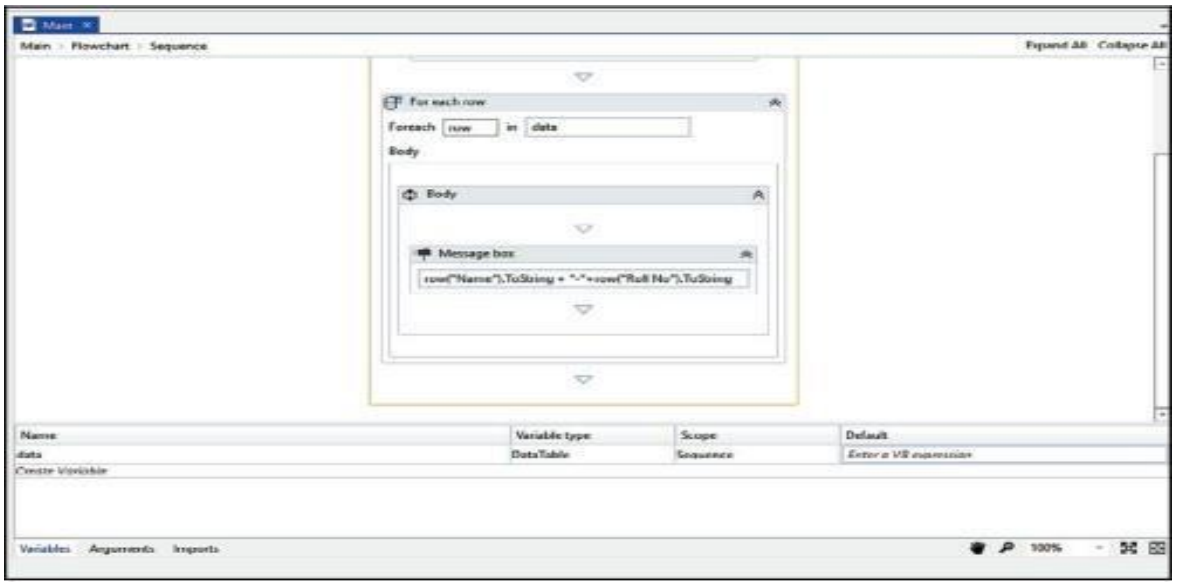


7. Drag and drop the For each row activity from the Activities panel inside the Sequence activity. Specify the data table variable's name in the expression text box of the For each row activity:



For each and For each row are two different activities. For each is used to iterate over the collections, while the For each row activity is used to iterate over the data table rows.

Drag and drop a Message box activity inside the For each row activity. In the Message box activity, Inside the message box we have to write following string:



This row variable contains all the columns of a particular row. Hence, we have to specify which column value we want to retrieve by specifying the column name. Instead of the column name, we can also specify the column index (the column index always starts from zero). Hit the Run button to see the result.

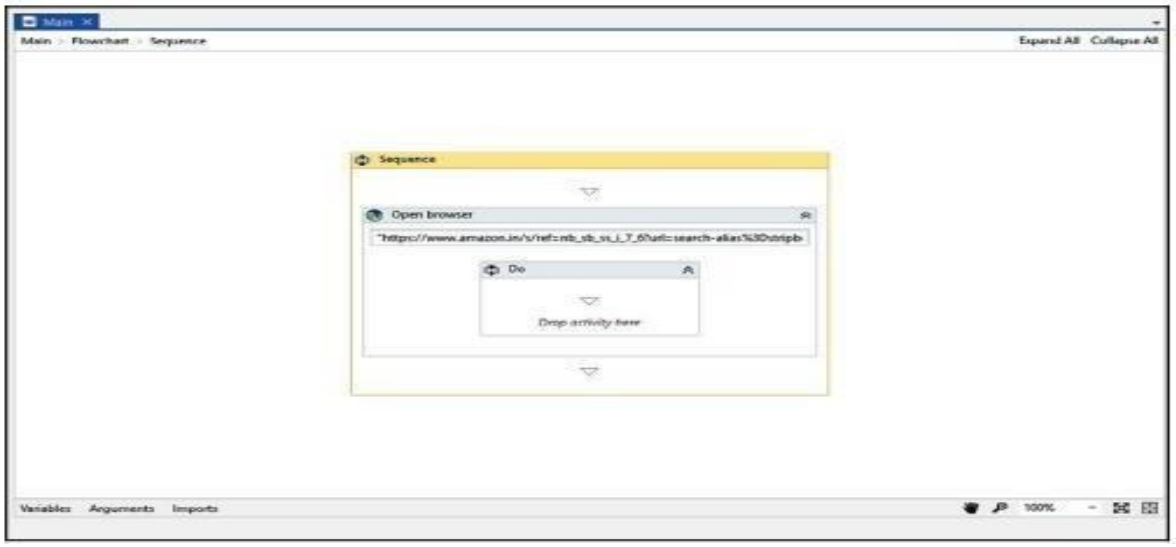
c. **Demonstrate Screen Scraping for a pdf, web page and image file**.

Using data scraping, we can build the data table at runtime. Let us consider an example of extracting data from Amazon's website. Perform the following steps:

1. Drag and drop the Flowchart activity from the Activities panel, and drag and drop the Sequence activity inside the Flowchart activity.

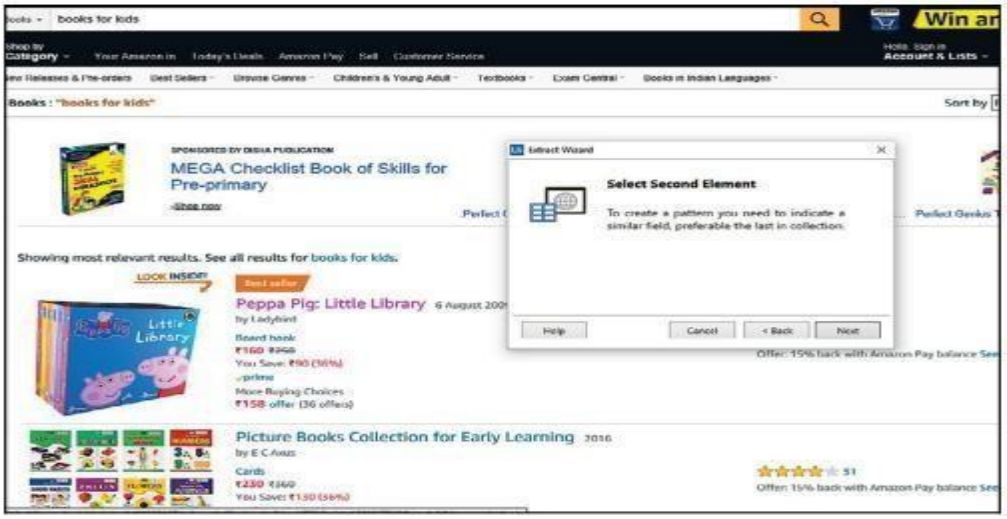
2. Double-click on the Sequence activity.

3. Drag and drop the Open Browser activity inside the Sequence activity. Specify the URL in the text box:

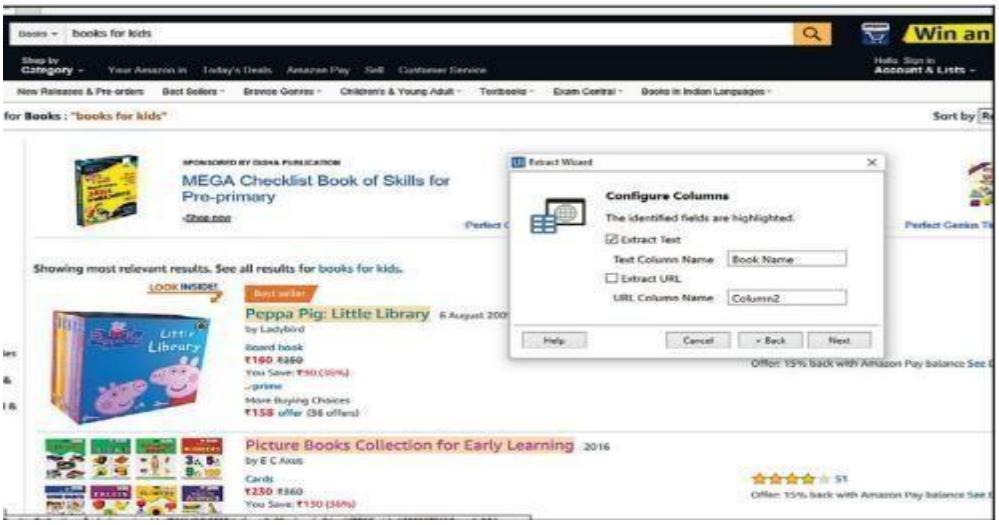


4. Click on the Data Scraping icon on the top left corner of UiPath Studio. A window will pop up. Click on the Next button.

5. Now, there will be a pointer pointing to the UI elements of the web page. Click on the name of the book:



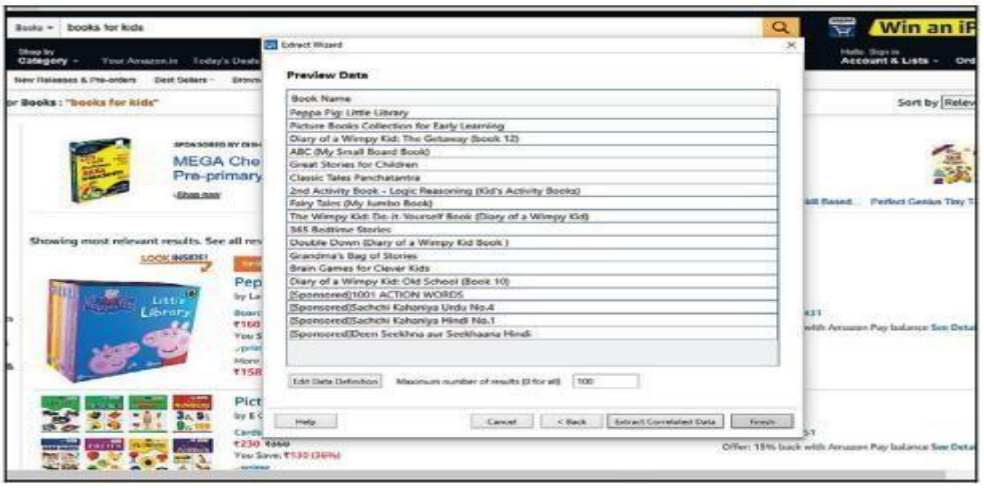
It will ask you to point to a second similar element on the web page:



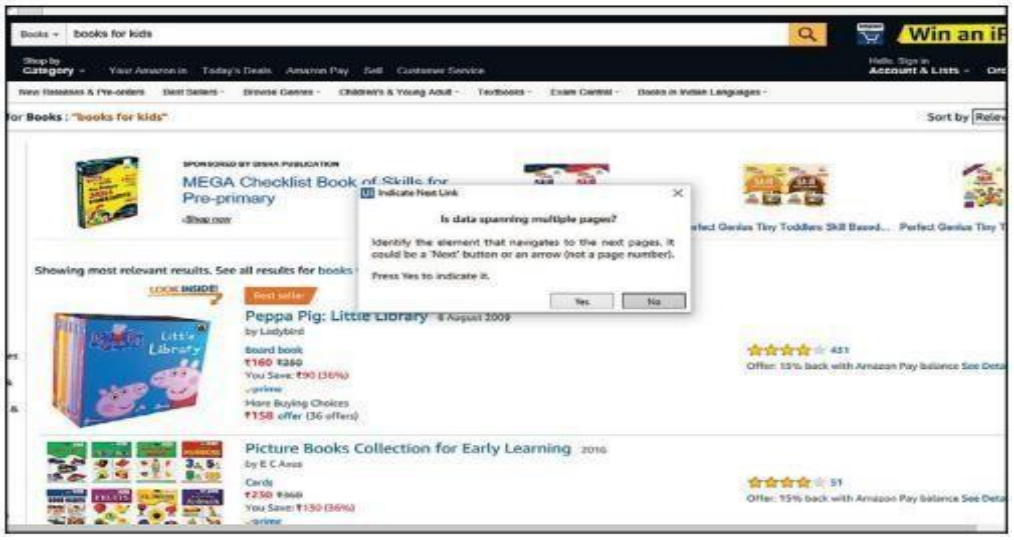
6. Point to a second similar element on that web page. Specify the name that you want to give for that extracted data column. (It will become the column name of the extracted data). Click on the Next button.

7. A list of names will appear in a separate window.

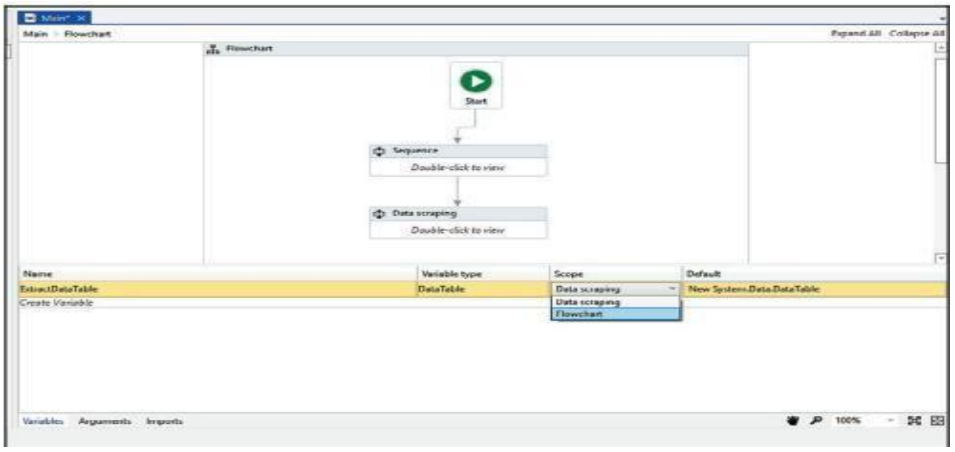
If you want to extract more information, then click on the Extract correlated data button and repeat the same process once again (just as we extracted the name of the book from Amazon's website). Otherwise, click on the Finish Button:



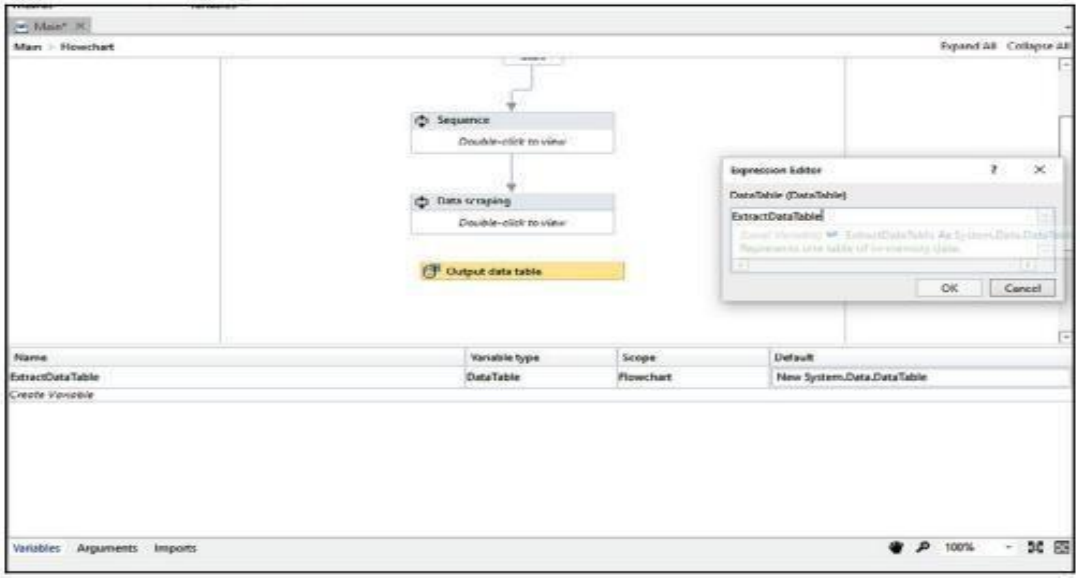
8. It will ask you to locate the next page's button/link. If you want to extract more information about the product and it spans across multiple pages, then click on the Yes button and point to the next page's button/link. Then, click on it. If you want to extract only the current page's data, click on the No button, (you can also specify the number of rows that you want to extract data from: By default it is 100):



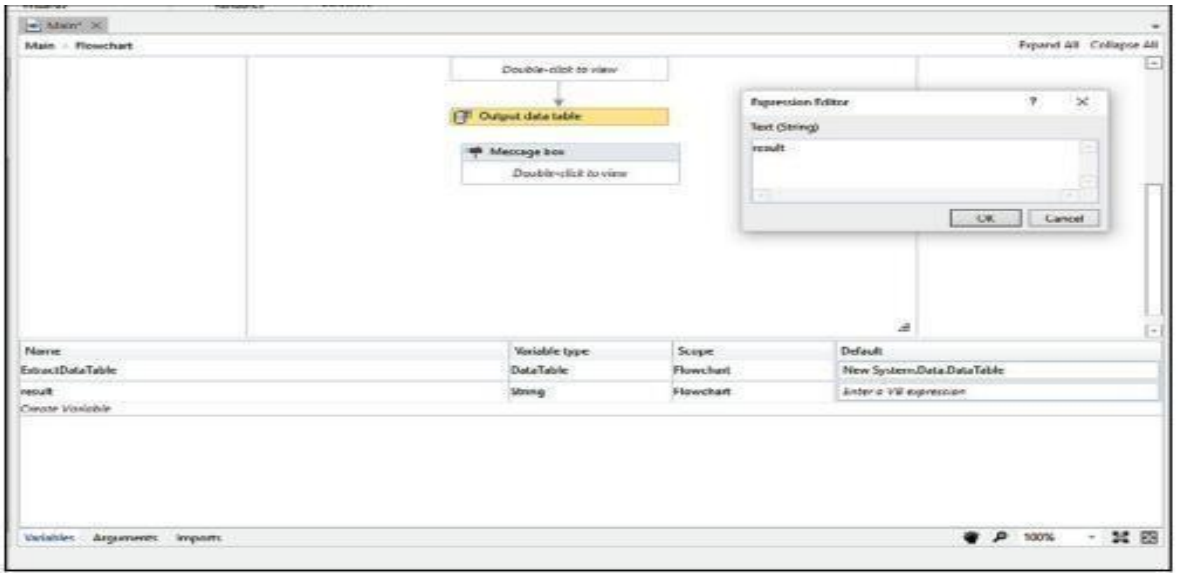
9. Data scraping generates a data table. (In this case, Extractiondatatable is generated.) Change the scope of Extractiondatatable to the Flowchart so that it is accessible within the Flowchart activity:



10. Drag and drop the Output data table activity on the Flowchart. Set the Output property of the Output data table activity as: Extractiondatatable:



11. Connect the Output data table activity to the Data Scraping activity. Drag and drop the Message box activity on the Designer window. Also create a string variable to receive the text from the Output data table activity (in our case, we have created a result variable). Specify the text property of the Output data table activity as the result variable to receive the text from the Output data table:



12. Connect the Message box activity to the Output data table activity. Double-click on the Message box and specify the text property as result variable (the variable that you created to receive the text from the Output data table activity).

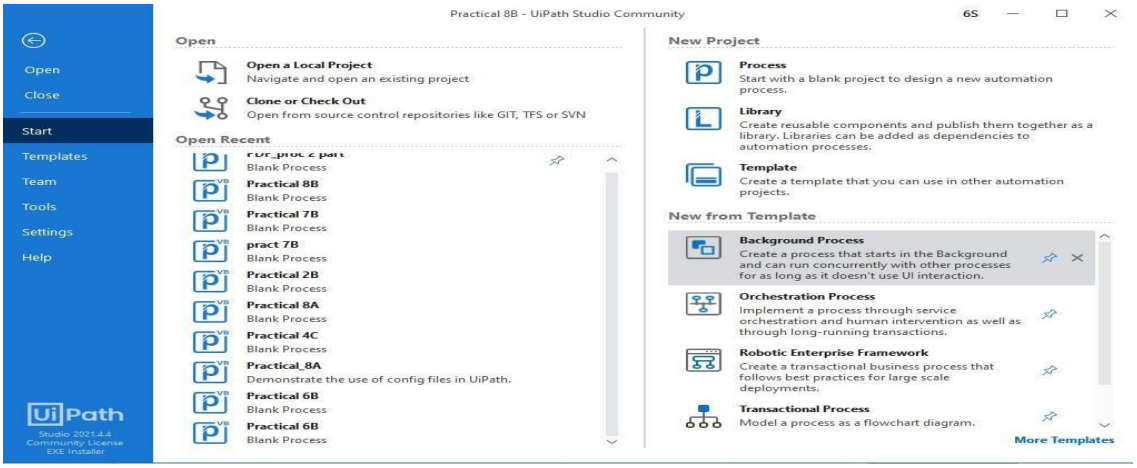
13. Hit the Run button and see the result.

Practical 8 : PDF Automation and Exception Handling

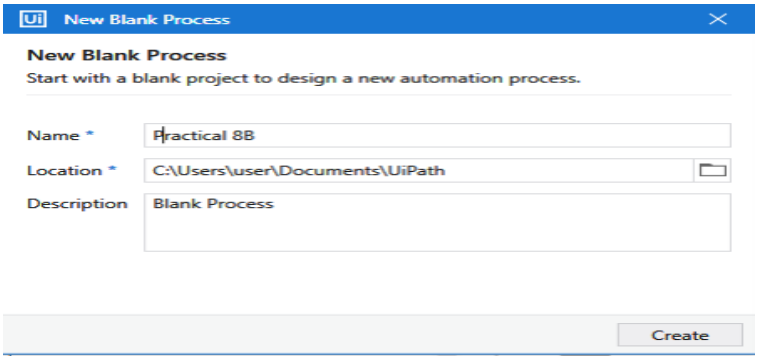
f. Demonstrate Exception Handling using UiPath

Steps:

1. Create new project: Open UiPath and create new project by clicking on “Process” option at the right side of the window.



2. In “New Blank Process” window insert the project name you want in the “Name” textbox. In “Location” textbox insert the path in which folder you want to save the project (In normal case you will see the default location of project). In “Description” textbox insert the project description. Then click on “Create” button.

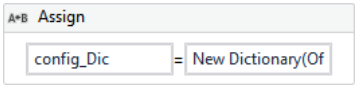


3. Create a Variable config_Dic of Variable types and Dictionary<String,Object>.

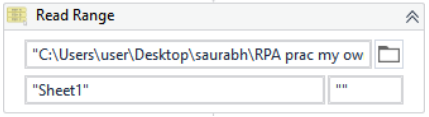
Name	Variable type	Scope	Default
config_DT	DataTable	Saurabh Yadav 8B Sequence	Enter a VB expression
config_Dic	Dictionary<String,Object>	Saurabh Yadav 8B Sequence	Enter a VB expression

4. Select Activity Try Catch from Activities and Insert sequence in it.

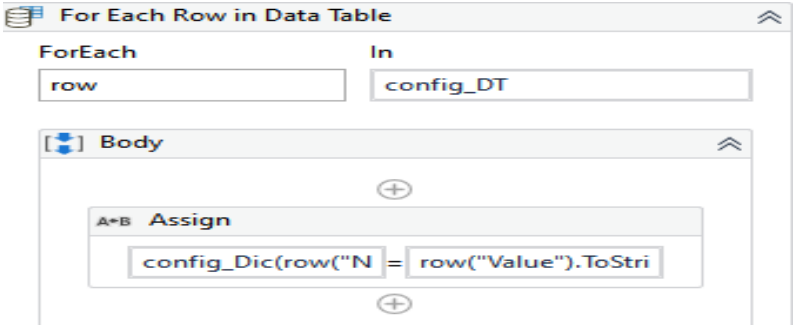
5. Select Activity Assign from Activities and initialize the dictionary.



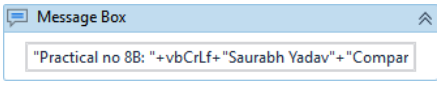
6. Select activity Read Range from activities to read Excel sheet.



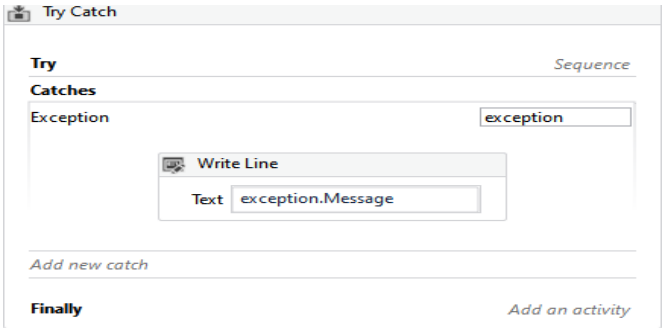
7. Select For Each Row from activities.



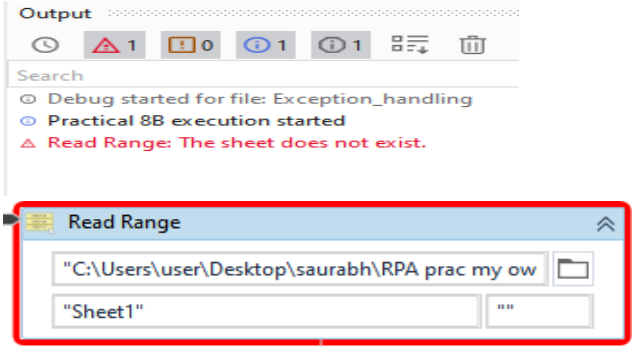
8. Select Message Box activity to print Output



9. Add a WriteLine Activity in the Catch Block.



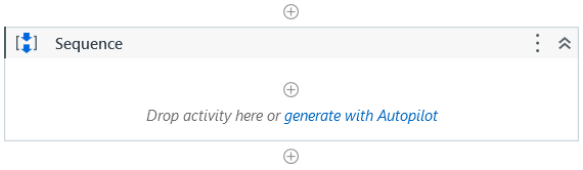
Output:



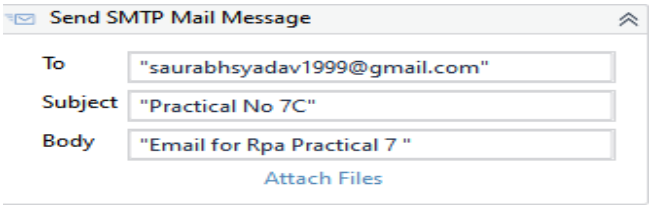
Practical 9 : Email Automation

c. Send Email with Attachment

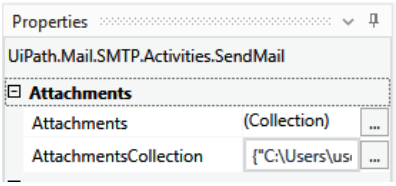
SMTP server address: smtp.gmail.com.
Gmail SMTP port (TLS): 587.
Add 3 different attachments as input.
Step:
1. Create a new Blank Project and give it an appropriate name. Drag a Sequence activity from Activity tab.



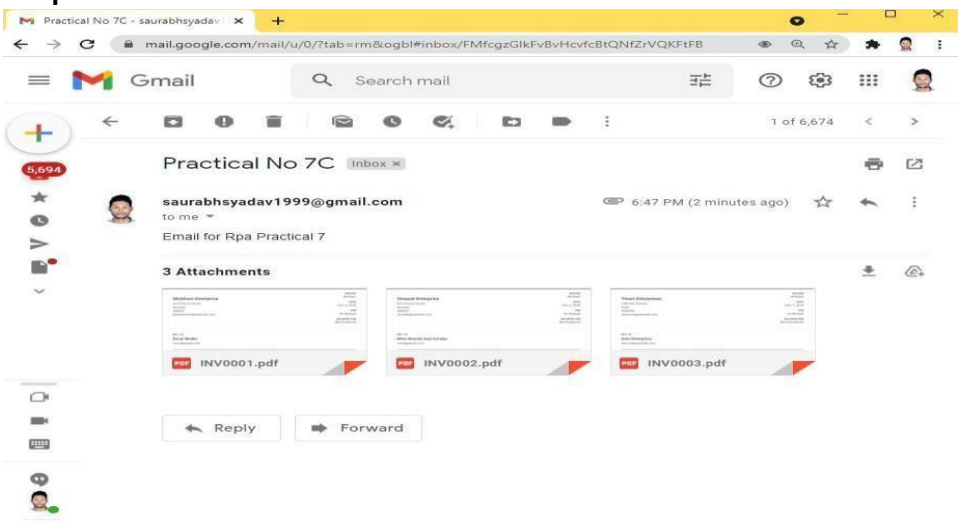
2. Add Send SMTP Mail Message Activity enter the recipient email, subject and body of the email to be sent.



3. Enter the files variable in AttachmentsCollection Attribute. Enter smtp port number in port
attribute of Host and the hostname in server field. Enter the email and password of the sender in the Logon email and password.



Output:

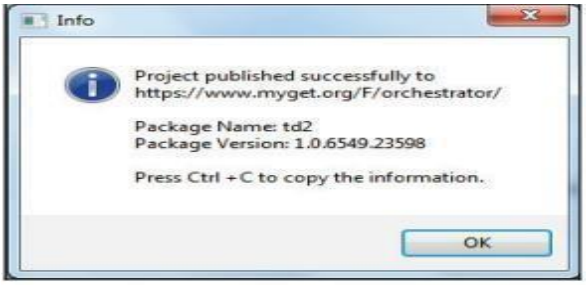

Practical No 7C - saurabhsyadav
mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzGlkFvBvHcvfcBtQNfZrVQKFtFB
Gmail
Search mail
1 of 6,674
5,694
Practical No 7C
Inbox
saurabhsyadav1999@gmail.com
to me
6:47 PM (2 minutes ago)
Email for Rpa Practical 7
3 Attachments
INV0001.pdf
INV0002.pdf
INV0003.pdf
Reply
Forward

Practical 10 : Orchestrator management and mini project

a. Deploy bots to Orchestrator

Deploy the Robot to Orchestrator To deploy our Robot, first of all, it must be connected to Orchestrator. Ensure that our bot is connected to Orchestrator then follow the given steps to deploy it:

1. First of all, install UiPath on the machine.
2. Provision the Robot machine and take the Robot key from Orchestrator.
3. After receiving the key go to the Robot configuration panel and enter the key here.
4. Also, you need to enter the Robot key into the configuration URL, which can be found from the admin section of Orchestrator.
5. Publish the project with publish utility from UiPath. When it is published successfully, it will display the information shown in the following screenshot:



6. The project has been published in the Orchestrator.
7. To create the environment, go to the home page, click on the ROBOTS option, and click on the Environments Tab. Then click on the + button:

8. Once the details are filled in, click on Create:

9. After creating the environment, a small window will appear as shown in the following screenshot, where we can manage the Robot within the environment:

10. After clicking on the + button, a will window pop-up where we can choose the Published package, as shown in the following screenshot, and then click on the CREATE button:

11. After clicking the Deploy Process button, a window popup will appear where we can choose the published package, as shown in the following screenshot, and then click on the CREATE button or

12. Packages can be manually uploaded from the local directory after clicking the View Packages option and then clicking the Upload button as given in the following screenshot: PROCESS | View Packages | Upload Packages:

13. Now the package has been deployed to the Orchestrator and is ready to be executed through the web.

14. Next, click the JOBS option for execution and click on the Start icon as shown:

15. After clicking the Start Job button, the Robot will execute over Orchestrator.

c. Queue Introduction:

i. Add items to Queue.

ii. Get Queue item from Orchestrator

Queues work as a container that stores tasks that need to be implemented. Simply imagine a group of boys standing in a queue in front of a ticketing counter. The logic is that the person who goes in first gets out first. First In First Out (FIFO).

Similarly, in the case of Robots, when we have a number of operations that are to be performed and when the server is busy, then tasks are moved in a queue and they are implemented on the same logic First In First Out (FIFO).

To create a new queues, search for the Queue option in the Orchestrator Server listed on the left-hand side and then inside the Queue page, you can add one. It also allows you to access all those Queues that have already been created. It contains some information
about the task such as the remaining time, progress time, average time, description, and so on, as listed in the following screenshot:

We can also add queue items from UiPath Studio and there are various activities that support this feature, which are listed as follows:

- Add Queue Item: This activity is used to add a new item to the queue in Orchestrator. The status of the item will be New.
- Add Transaction Item: This activity is used to add an item to the queue to begin transaction and set the status as In Progress. Here we can add custom reference for each respective transaction.
- Get Transaction Item: This activity is used to get an item from the queue to process it and set its status as In Progress.
- Postpone Transaction Item: This activity is used to define time parameters between which transaction should be processed. Here, basically, we will specify the time interval after which a process will start.
- Set Transaction Progress: Used to assist and create custom progress statuses for In Progress transactions. To notify its progress if the process crashes. This activity plays a significant role in tackling troubleshooting processes.

- Set Transaction Status: Used to modify the status of the transaction item; whether it fails or is successful.