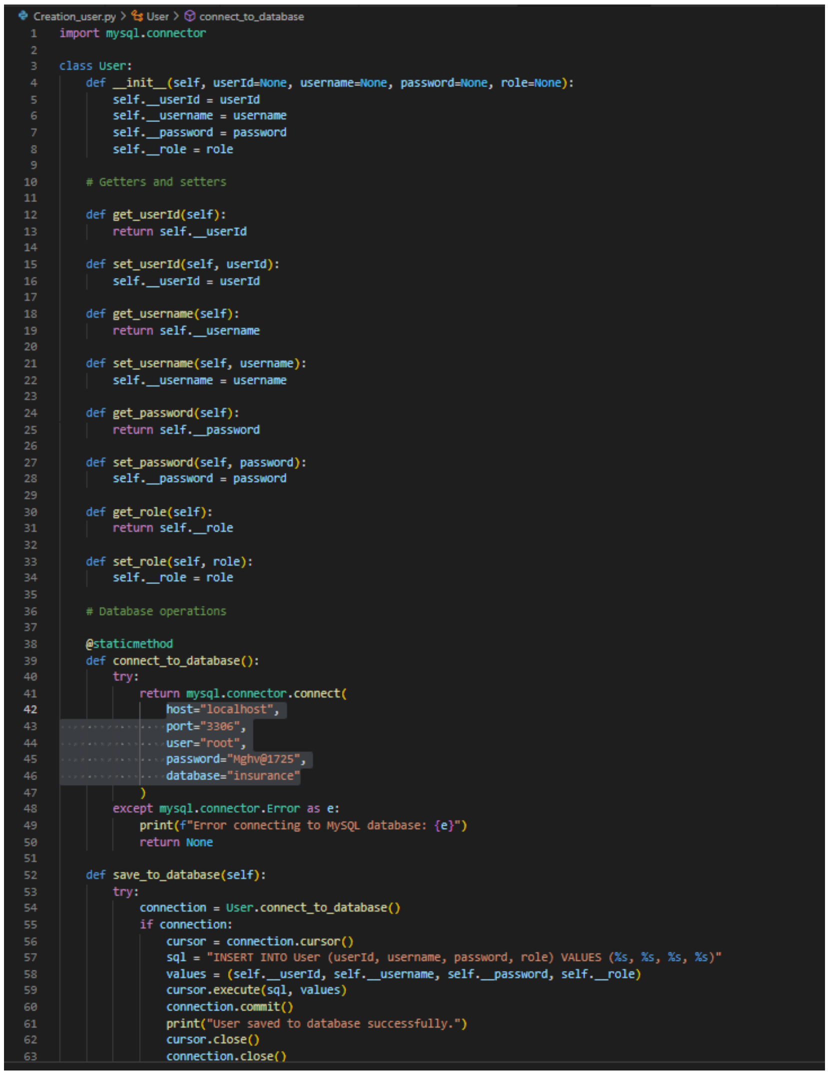
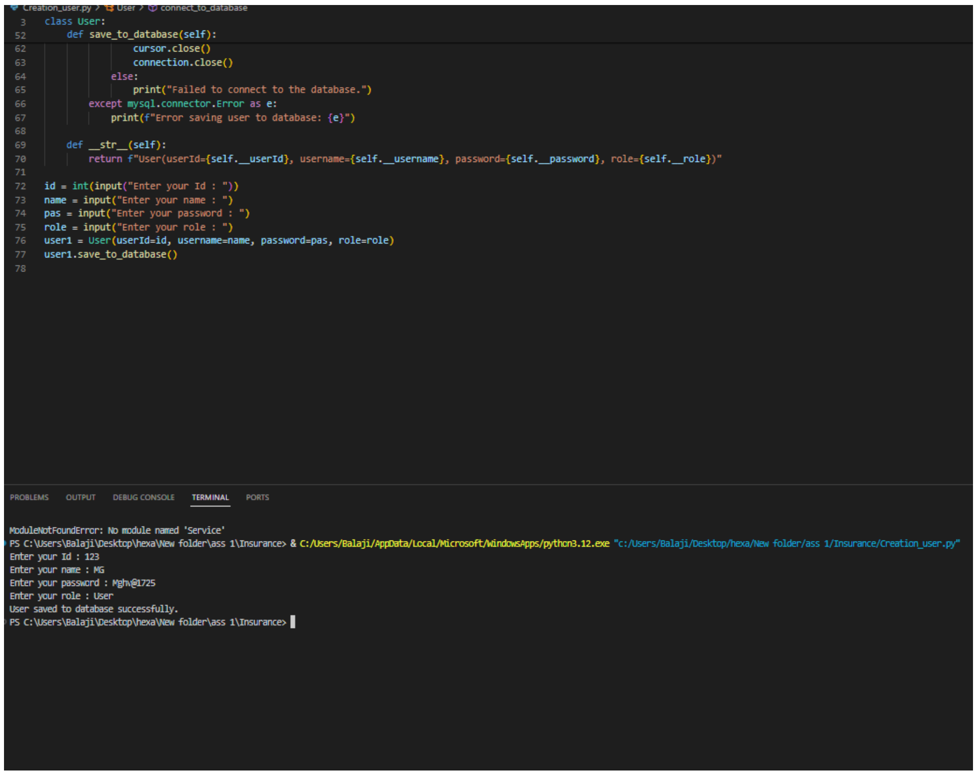
INSURANCE MANAGEMENT SYSTEM

1 Define `User ` class with the following confidential attributes: a. userId; b. username; c. password; d. role;



Creation_user.py > User > connect_to_database

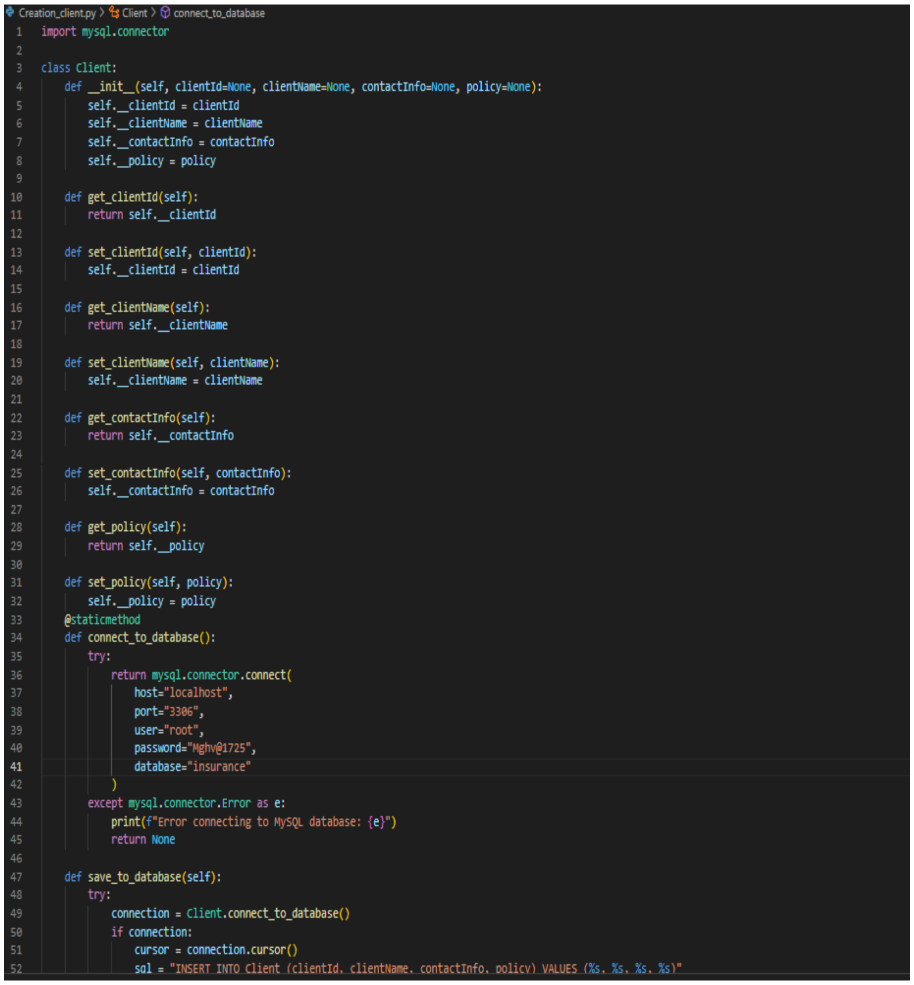
```python
import mysql.connector

class User:
    def __init__(self, userId=None, username=None, password=None, role=None):
        self.__userId = userId
        self.__username = username
        self.__password = password
        self.__role = role

    # Getters and setters

    def get_userId(self):
        return self.__userId

    def set_userId(self, userId):
        self.__userId = userId

    def get_username(self):
        return self.__username

    def set_username(self, username):
        self.__username = username

    def get_password(self):
        return self.__password

    def set_password(self, password):
        self.__password = password

    def get_role(self):
        return self.__role

    def set_role(self, role):
        self.__role = role

    # Database operations

    @staticmethod
    def connect_to_database():
        try:
            return mysql.connector.connect(
                host="localhost",
                port="3306",
                user="root",
                password="Mghv@1725",
                database="insurance"
            )
        except mysql.connector.Error as e:
            print(f"Error connecting to MySQL database: {e}")
            return None

    def save_to_database(self):
        try:
            connection = User.connect_to_database()
            if connection:
                cursor = connection.cursor()
                sql = "INSERT INTO User (userId, username, password, role) VALUES (%s, %s, %s, %s)"
                values = (self.__userId, self.__username, self.__password, self.__role)
                cursor.execute(sql, values)
                connection.commit()
                print("User saved to database successfully.")
                cursor.close()
                connection.close()
```



```python
class User:
    def save_to_database(self):
                cursor.close()
                connection.close()
            else:
                print("Failed to connect to the database.")
        except mysql.connector.Error as e:
            print(f"Error saving user to database: {e}")

    def __str__(self):
        return f"User(userId={self.__userId}, username={self.__username}, password={self.__password}, role={self.__role})"

id = int(input("Enter your Id : "))
name = input("Enter your name : ")
pas = input("Enter your password : ")
role = input("Enter your role : ")
user1 = User(userId=id, username=name, password=pas, role=role)
user1.save_to_database()

```

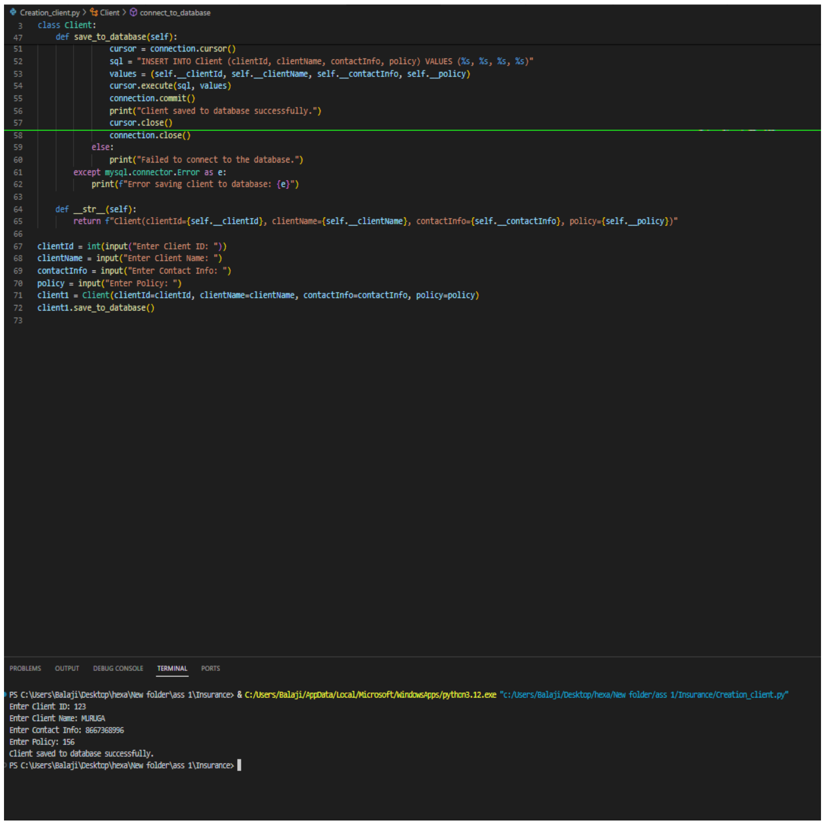
```
ModuleNotFoundError: No module named 'Service'
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/Creation_user.py"
Enter your Id : 123
Enter your name : MG
Enter your password : Mghv@1725
Enter your role : User
User saved to database successfully.
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance>
```

2. Define ` Client ` class with the following confidential attributes: a. clientId; b. clientName; c. contactInfo; d. policy;//Represents the policy associated with the client



Creation_client.py > Client > connect_to_database

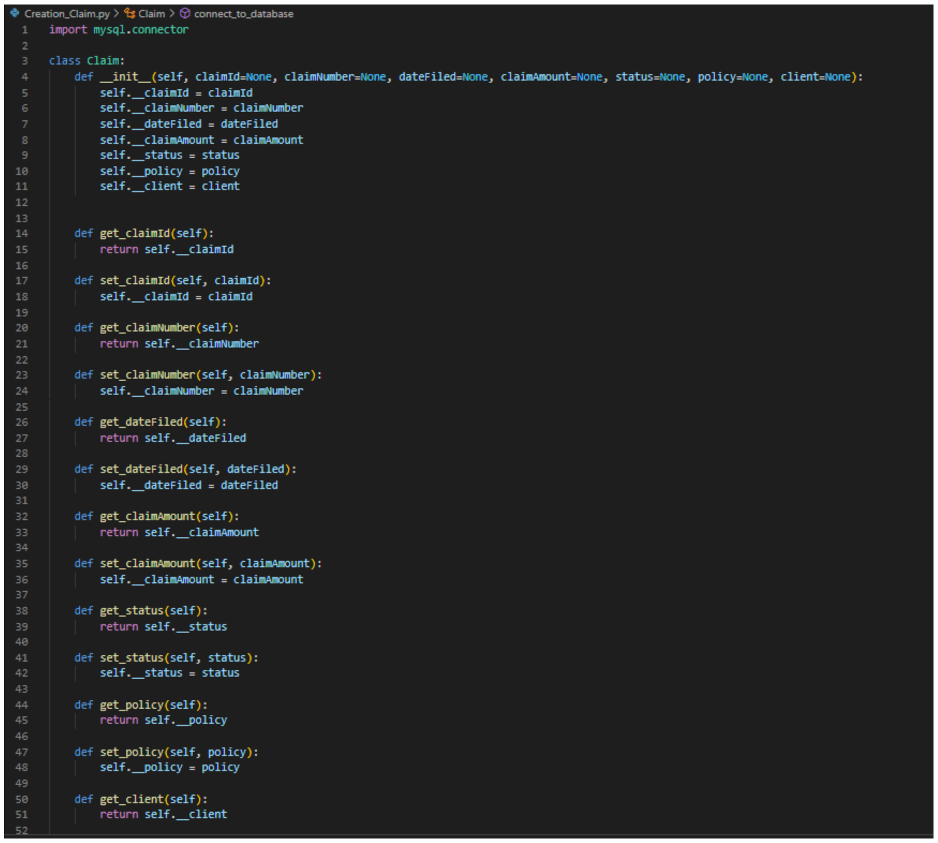
```python
import mysql.connector

class Client:
    def __init__(self, clientId=None, clientName=None, contactInfo=None, policy=None):
        self.__clientId = clientId
        self.__clientName = clientName
        self.__contactInfo = contactInfo
        self.__policy = policy

    def get_clientId(self):
        return self.__clientId

    def set_clientId(self, clientId):
        self.__clientId = clientId

    def get_clientName(self):
        return self.__clientName

    def set_clientName(self, clientName):
        self.__clientName = clientName

    def get_contactInfo(self):
        return self.__contactInfo

    def set_contactInfo(self, contactInfo):
        self.__contactInfo = contactInfo

    def get_policy(self):
        return self.__policy

    def set_policy(self, policy):
        self.__policy = policy
    @staticmethod
    def connect_to_database():
        try:
            return mysql.connector.connect(
                host="localhost",
                port="3306",
                user="root",
                password="Mghv@1725",
                database="insurance"
            )
        except mysql.connector.Error as e:
            print(f"Error connecting to MySQL database: {e}")
            return None

    def save_to_database(self):
        try:
            connection = Client.connect_to_database()
            if connection:
                cursor = connection.cursor()
                sql = "INSERT INTO Client (clientId, clientName, contactInfo, policy) VALUES (%s, %s, %s, %s)"
```



```python
class Client:
    def save_to_database(self):
                cursor = connection.cursor()
                sql = "INSERT INTO Client (clientId, clientName, contactInfo, policy) VALUES (%s, %s, %s, %s)"
                values = (self.__clientId, self.__clientName, self.__contactInfo, self.__policy)
                cursor.execute(sql, values)
                connection.commit()
                print("Client saved to database successfully.")
                cursor.close()
                connection.close()
            else:
                print("Failed to connect to the database.")
        except mysql.connector.Error as e:
            print(f"Error saving client to database: {e}")

    def __str__(self):
        return f"Client(clientId={self.__clientId}, clientName={self.__clientName}, contactInfo={self.__contactInfo}, policy={self.__policy})"

clientId = int(input("Enter Client ID: "))
clientName = input("Enter Client Name: ")
contactInfo = input("Enter Contact Info: ")
policy = input("Enter Policy: ")
client1 = Client(clientId=clientId, clientName=clientName, contactInfo=contactInfo, policy=policy)
client1.save_to_database()
```

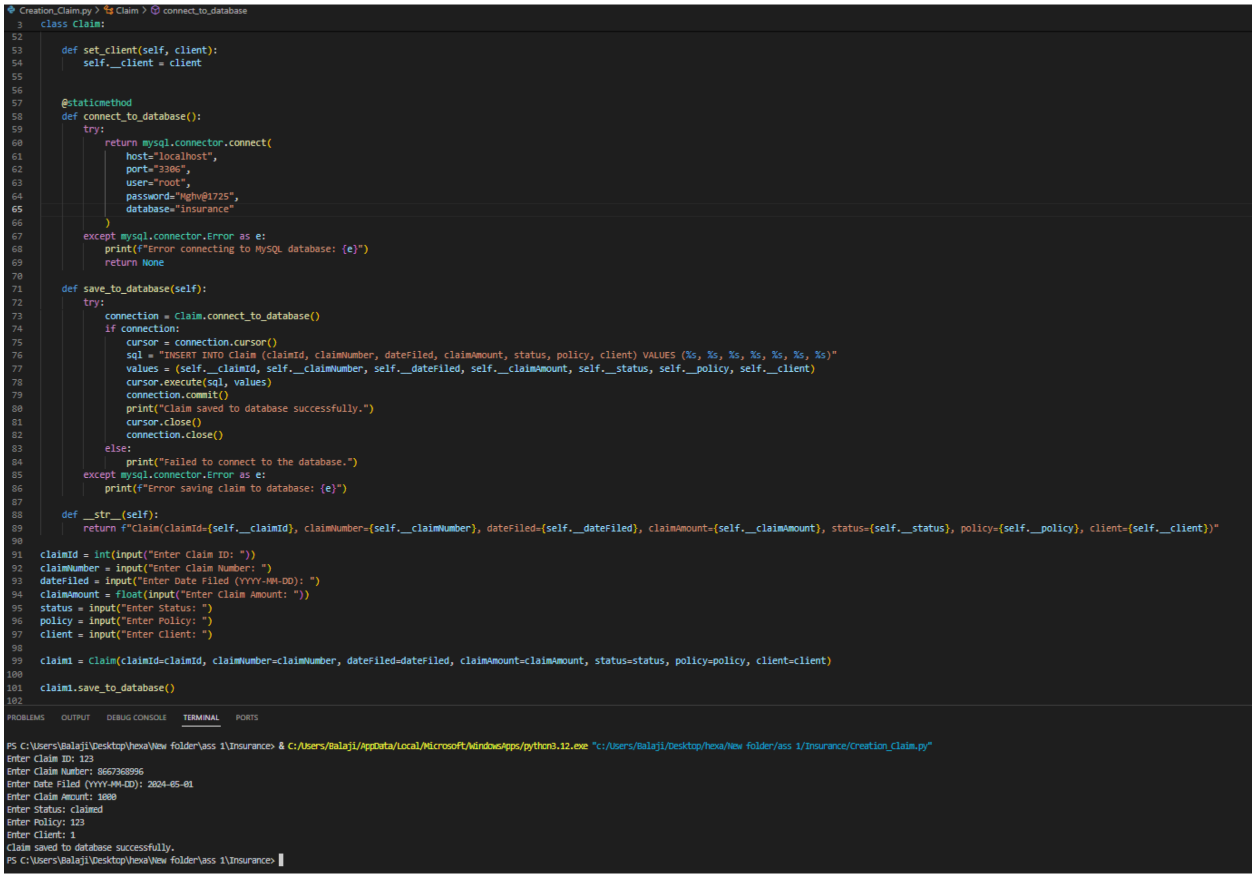
```
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/Creation_client.py"
Enter Client ID: 123
Enter Client Name: MURUGA
Enter Contact Info: 8667368996
Enter Policy: 156
Client saved to database successfully.
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> 
```

3. Define ` Claim ` class with the following confidential attributes: a. claimId; b. claimNumber; c. dateFiled; d. claimAmount; e. status; f. policy;//Represents the policy associated with the claim g. client; // Represents the client associated with the claim



Creation_Claim.py > Claim > connect_to_database

```python
import mysql.connector

class Claim:
    def __init__(self, claimId=None, claimNumber=None, dateFiled=None, claimAmount=None, status=None, policy=None, client=None):
        self.__claimId = claimId
        self.__claimNumber = claimNumber
        self.__dateFiled = dateFiled
        self.__claimAmount = claimAmount
        self.__status = status
        self.__policy = policy
        self.__client = client


    def get_claimId(self):
        return self.__claimId

    def set_claimId(self, claimId):
        self.__claimId = claimId

    def get_claimNumber(self):
        return self.__claimNumber

    def set_claimNumber(self, claimNumber):
        self.__claimNumber = claimNumber

    def get_dateFiled(self):
        return self.__dateFiled

    def set_dateFiled(self, dateFiled):
        self.__dateFiled = dateFiled

    def get_claimAmount(self):
        return self.__claimAmount

    def set_claimAmount(self, claimAmount):
        self.__claimAmount = claimAmount

    def get_status(self):
        return self.__status

    def set_status(self, status):
        self.__status = status

    def get_policy(self):
        return self.__policy

    def set_policy(self, policy):
        self.__policy = policy

    def get_client(self):
        return self.__client

```

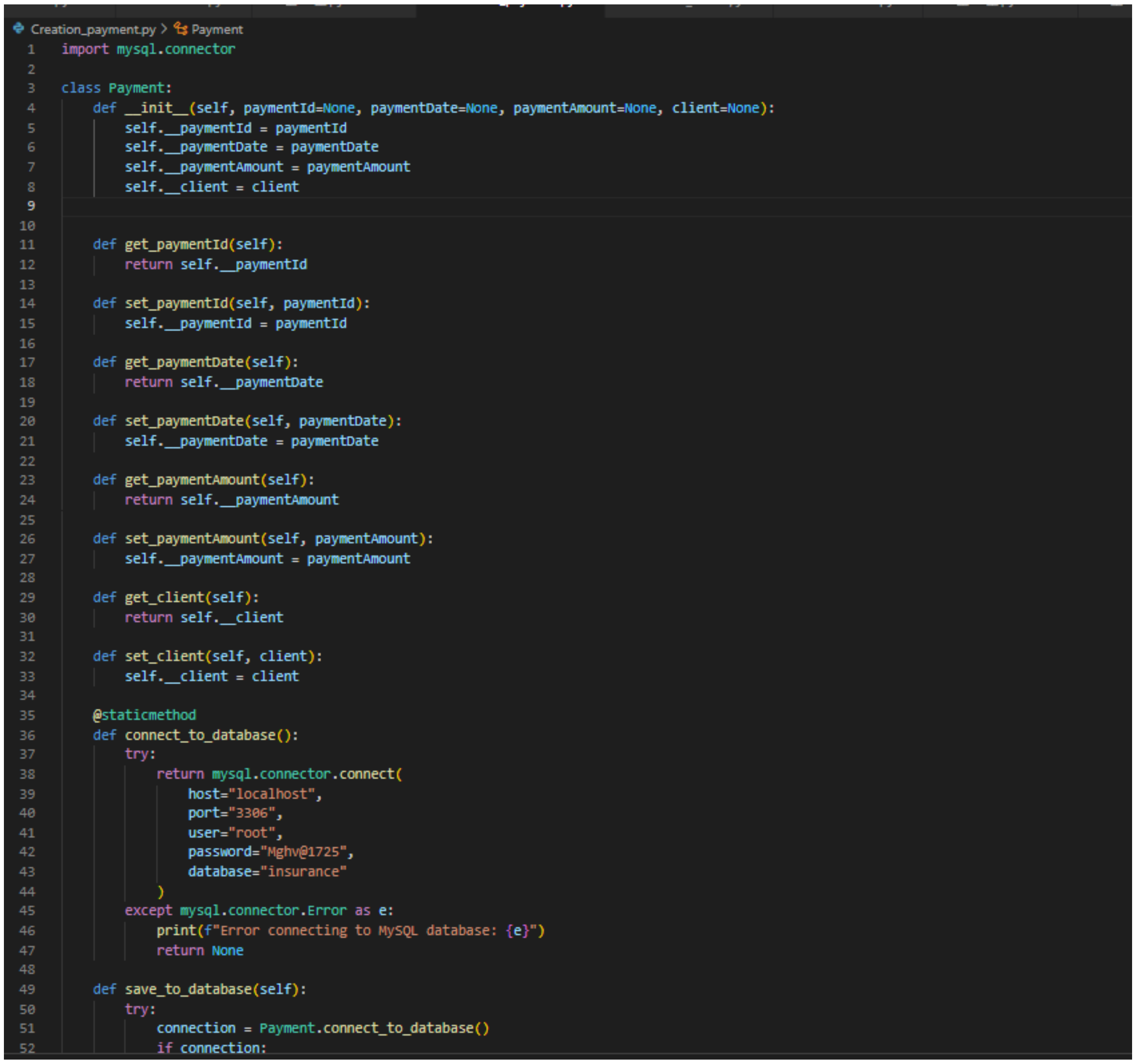


Creation_Claim.py > Claim > connect_to_database

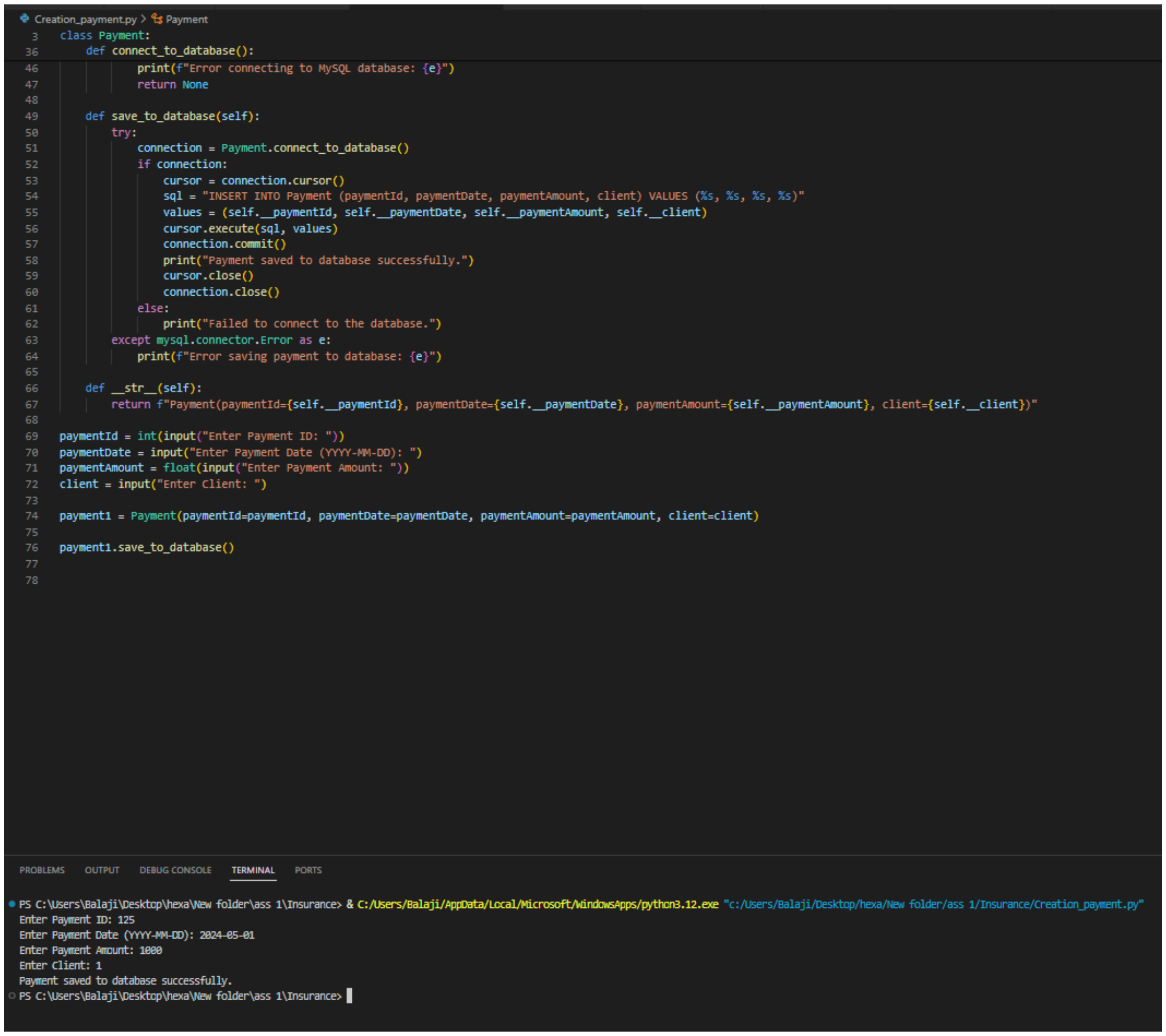
```python
class Claim:

    def set_client(self, client):
        self.__client = client


    @staticmethod
    def connect_to_database():
        try:
            return mysql.connector.connect(
                host="localhost",
                port="3306",
                user="root",
                password="Mghv@1725",
                database="insurance"
            )
        except mysql.connector.Error as e:
            print(f"Error connecting to MySQL database: {e}")
            return None

    def save_to_database(self):
        try:
            connection = Claim.connect_to_database()
            if connection:
                cursor = connection.cursor()
                sql = "INSERT INTO Claim (claimId, claimNumber, dateFiled, claimAmount, status, policy, client) VALUES (%s, %s, %s, %s, %s, %s, %s)"
                values = (self.__claimId, self.__claimNumber, self.__dateFiled, self.__claimAmount, self.__status, self.__policy, self.__client)
                cursor.execute(sql, values)
                connection.commit()
                print("Claim saved to database successfully.")
                cursor.close()
                connection.close()
            else:
                print("Failed to connect to the database.")
        except mysql.connector.Error as e:
            print(f"Error saving claim to database: {e}")

    def __str__(self):
        return f"Claim(claimId={self.__claimId}, claimNumber={self.__claimNumber}, dateFiled={self.__dateFiled}, claimAmount={self.__claimAmount}, status={self.__status}, policy={self.__policy}, client={self.__client})"

claimId = int(input("Enter Claim ID: "))
claimNumber = input("Enter Claim Number: ")
dateFiled = input("Enter Date Filed (YYYY-MM-DD): ")
claimAmount = float(input("Enter Claim Amount: "))
status = input("Enter Status: ")
policy = input("Enter Policy: ")
client = input("Enter Client: ")

claim1 = Claim(claimId=claimId, claimNumber=claimNumber, dateFiled=dateFiled, claimAmount=claimAmount, status=status, policy=policy, client=client)

claim1.save_to_database()

```

```
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/Creation_Claim.py"
Enter Claim ID: 123
Enter Claim Number: 8667368996
Enter Date Filed (YYYY-MM-DD): 2024-05-01
Enter Claim Amount: 1000
Enter Status: claimed
Enter Policy: 123
Enter Client: 1
Claim saved to database successfully.
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance>
```

4.. Define ` Claim ` class with the following confidential attributes: a. paymentId; b. paymentDate; c. paymentAmount; d. client; // Represents the client associated with the payment



```python
import mysql.connector

class Payment:
    def __init__(self, paymentId=None, paymentDate=None, paymentAmount=None, client=None):
        self.__paymentId = paymentId
        self.__paymentDate = paymentDate
        self.__paymentAmount = paymentAmount
        self.__client = client


    def get_paymentId(self):
        return self.__paymentId

    def set_paymentId(self, paymentId):
        self.__paymentId = paymentId

    def get_paymentDate(self):
        return self.__paymentDate

    def set_paymentDate(self, paymentDate):
        self.__paymentDate = paymentDate

    def get_paymentAmount(self):
        return self.__paymentAmount

    def set_paymentAmount(self, paymentAmount):
        self.__paymentAmount = paymentAmount

    def get_client(self):
        return self.__client

    def set_client(self, client):
        self.__client = client

    @staticmethod
    def connect_to_database():
        try:
            return mysql.connector.connect(
                host="localhost",
                port="3306",
                user="root",
                password="Mghv@1725",
                database="insurance"
            )
        except mysql.connector.Error as e:
            print(f"Error connecting to MySQL database: {e}")
            return None

    def save_to_database(self):
        try:
            connection = Payment.connect_to_database()
            if connection:
```

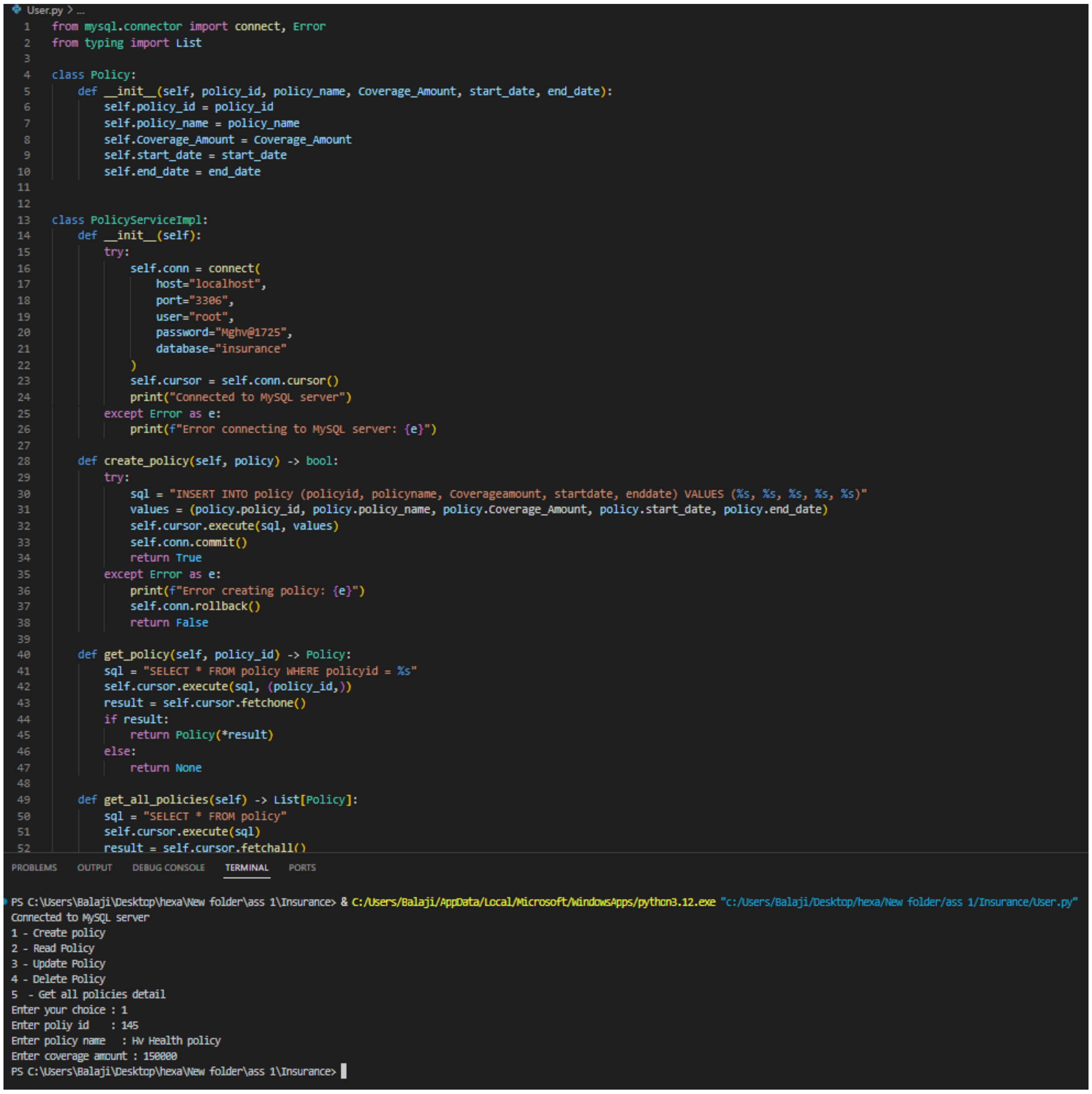


```python
class Payment:
    def connect_to_database():
            print(f"Error connecting to MySQL database: {e}")
            return None

    def save_to_database(self):
        try:
            connection = Payment.connect_to_database()
            if connection:
                cursor = connection.cursor()
                sql = "INSERT INTO Payment (paymentId, paymentDate, paymentAmount, client) VALUES (%s, %s, %s, %s)"
                values = (self.__paymentId, self.__paymentDate, self.__paymentAmount, self.__client)
                cursor.execute(sql, values)
                connection.commit()
                print("Payment saved to database successfully.")
                cursor.close()
                connection.close()
            else:
                print("Failed to connect to the database.")
        except mysql.connector.Error as e:
            print(f"Error saving payment to database: {e}")

    def __str__(self):
        return f"Payment(paymentId={self.__paymentId}, paymentDate={self.__paymentDate}, paymentAmount={self.__paymentAmount}, client={self.__client})"

paymentId = int(input("Enter Payment ID: "))
paymentDate = input("Enter Payment Date (YYYY-MM-DD): ")
paymentAmount = float(input("Enter Payment Amount: "))
client = input("Enter Client: ")

payment1 = Payment(paymentId=paymentId, paymentDate=paymentDate, paymentAmount=paymentAmount, client=client)

payment1.save_to_database()
```

```
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/Creation_payment.py"
Enter Payment ID: 125
Enter Payment Date (YYYY-MM-DD): 2024-05-01
Enter Payment Amount: 1000
Enter Client: 1
Payment saved to database successfully.
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance>
```

5. Define IPolicyService interface/abstract class with following methods to interact with database Keep the interfaces and implementation classes in package dao

a. createPolicy() I. parameters: Policy Object II. return type: boolean

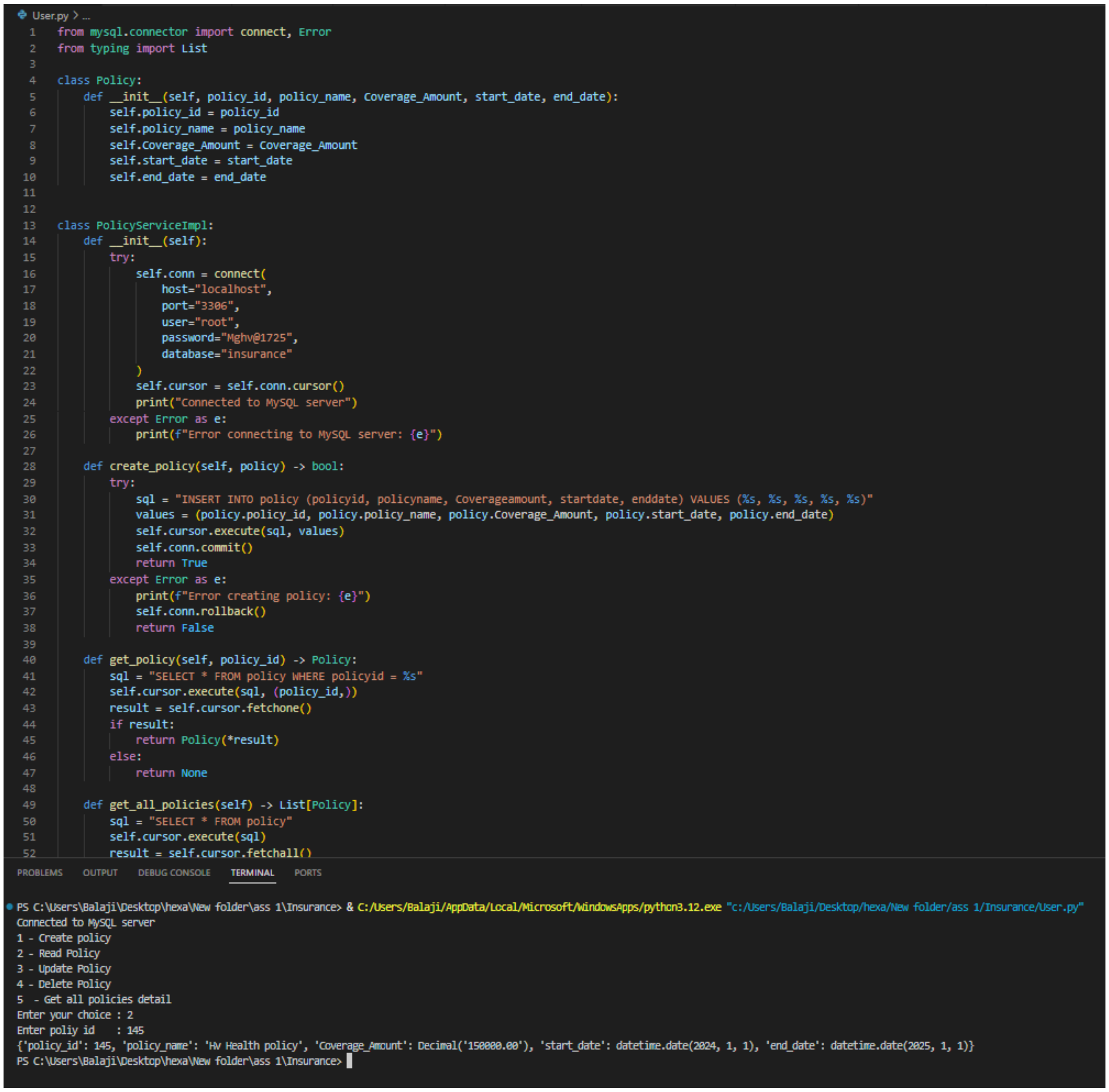


User.py > ...

```python
from mysql.connector import connect, Error
from typing import List

class Policy:
    def __init__(self, policy_id, policy_name, Coverage_Amount, start_date, end_date):
        self.policy_id = policy_id
        self.policy_name = policy_name
        self.Coverage_Amount = Coverage_Amount
        self.start_date = start_date
        self.end_date = end_date


class PolicyServiceImpl:
    def __init__(self):
        try:
            self.conn = connect(
                host="localhost",
                port="3306",
                user="root",
                password="Mghv@1725",
                database="insurance"
            )
            self.cursor = self.conn.cursor()
            print("Connected to MySQL server")
        except Error as e:
            print(f"Error connecting to MySQL server: {e}")

    def create_policy(self, policy) -> bool:
        try:
            sql = "INSERT INTO policy (policyid, policyname, Coverageamount, startdate, enddate) VALUES (%s, %s, %s, %s, %s)"
            values = (policy.policy_id, policy.policy_name, policy.Coverage_Amount, policy.start_date, policy.end_date)
            self.cursor.execute(sql, values)
            self.conn.commit()
            return True
        except Error as e:
            print(f"Error creating policy: {e}")
            self.conn.rollback()
            return False

    def get_policy(self, policy_id) -> Policy:
        sql = "SELECT * FROM policy WHERE policyid = %s"
        self.cursor.execute(sql, (policy_id,))
        result = self.cursor.fetchone()
        if result:
            return Policy(*result)
        else:
            return None

    def get_all_policies(self) -> List[Policy]:
        sql = "SELECT * FROM policy"
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
```

```
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/User.py"
Connected to MySQL server
1 - Create policy
2 - Read Policy
3 - Update Policy
4 - Delete Policy
5  - Get all policies detail
Enter your choice : 1
Enter poliy id    : 145
Enter policy name   : Hv Health policy
Enter coverage amount : 150000
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance>
```

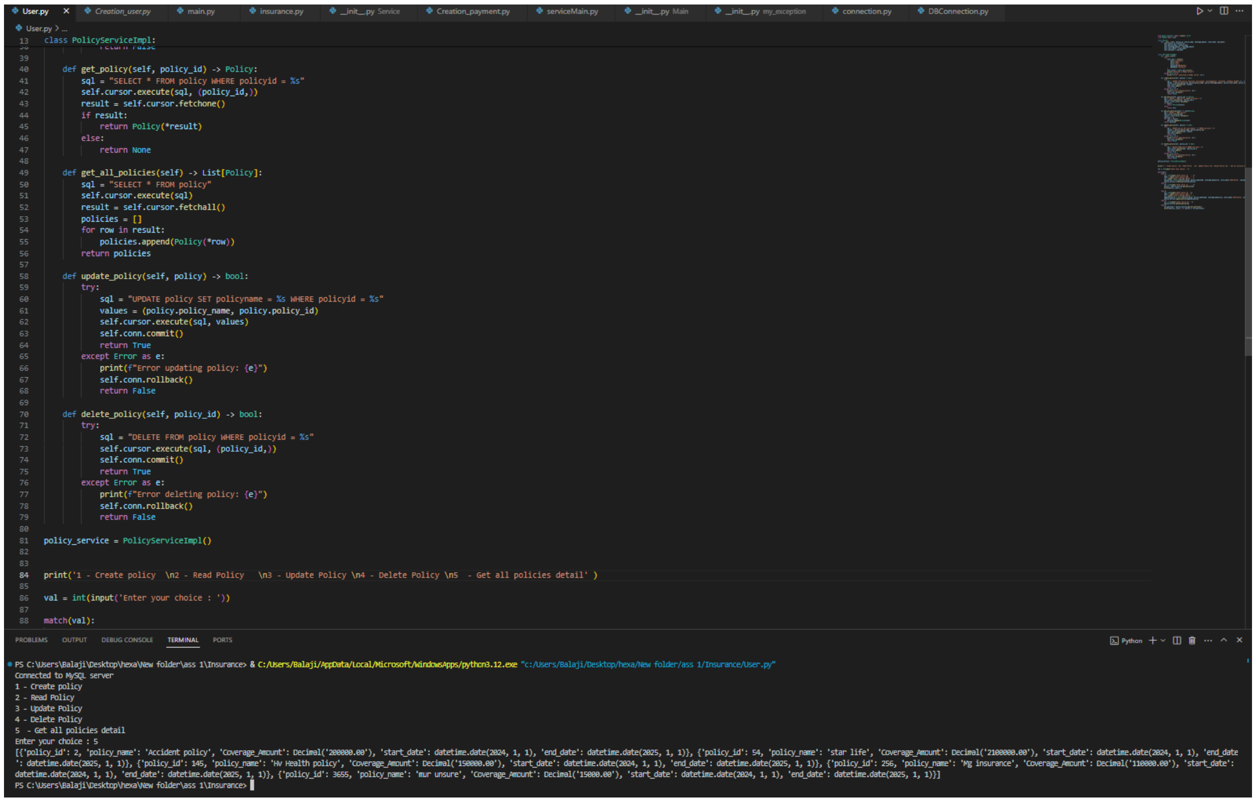
b. getPolicy() I. parameters: policyId II. return type: Policy Object



```python
from mysql.connector import connect, Error
from typing import List

class Policy:
    def __init__(self, policy_id, policy_name, Coverage_Amount, start_date, end_date):
        self.policy_id = policy_id
        self.policy_name = policy_name
        self.Coverage_Amount = Coverage_Amount
        self.start_date = start_date
        self.end_date = end_date


class PolicyServiceImpl:
    def __init__(self):
        try:
            self.conn = connect(
                host="localhost",
                port="3306",
                user="root",
                password="Mghv@1725",
                database="insurance"
            )
            self.cursor = self.conn.cursor()
            print("Connected to MySQL server")
        except Error as e:
            print(f"Error connecting to MySQL server: {e}")

    def create_policy(self, policy) -> bool:
        try:
            sql = "INSERT INTO policy (policyid, policyname, Coverageamount, startdate, enddate) VALUES (%s, %s, %s, %s, %s)"
            values = (policy.policy_id, policy.policy_name, policy.Coverage_Amount, policy.start_date, policy.end_date)
            self.cursor.execute(sql, values)
            self.conn.commit()
            return True
        except Error as e:
            print(f"Error creating policy: {e}")
            self.conn.rollback()
            return False

    def get_policy(self, policy_id) -> Policy:
        sql = "SELECT * FROM policy WHERE policyid = %s"
        self.cursor.execute(sql, (policy_id,))
        result = self.cursor.fetchone()
        if result:
            return Policy(*result)
        else:
            return None

    def get_all_policies(self) -> List[Policy]:
        sql = "SELECT * FROM policy"
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
```

```
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/User.py"
Connected to MySQL server
1 - Create policy
2 - Read Policy
3 - Update Policy
4 - Delete Policy
5  - Get all policies detail
Enter your choice : 2
Enter poliy id    : 145
{'policy_id': 145, 'policy_name': 'Hv Health policy', 'Coverage_Amount': Decimal('150000.00'), 'start_date': datetime.date(2024, 1, 1), 'end_date': datetime.date(2025, 1, 1)}
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance>
```

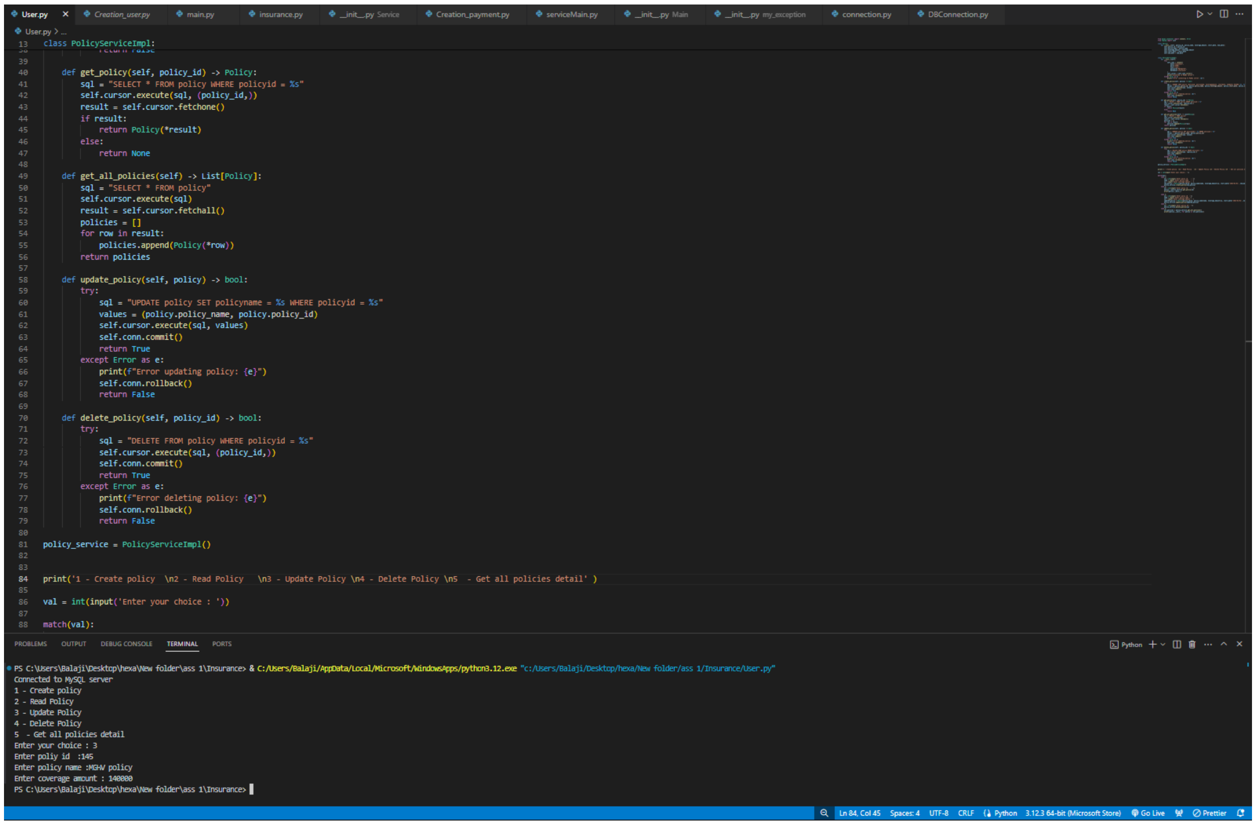
c.getAllPolicies() I. parameters: none II. return type: Collection of Policy Objects



```python
class PolicyServiceImpl:

    def get_policy(self, policy_id) -> Policy:
        sql = "SELECT * FROM policy WHERE policyid = %s"
        self.cursor.execute(sql, (policy_id,))
        result = self.cursor.fetchone()
        if result:
            return Policy(*result)
        else:
            return None

    def get_all_policies(self) -> List[Policy]:
        sql = "SELECT * FROM policy"
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
        policies = []
        for row in result:
            policies.append(Policy(*row))
        return policies

    def update_policy(self, policy) -> bool:
        try:
            sql = "UPDATE policy SET policyname = %s WHERE policyid = %s"
            values = (policy.policy_name, policy.policy_id)
            self.cursor.execute(sql, values)
            self.conn.commit()
            return True
        except Error as e:
            print(f"Error updating policy: {e}")
            self.conn.rollback()
            return False

    def delete_policy(self, policy_id) -> bool:
        try:
            sql = "DELETE FROM policy WHERE policyid = %s"
            self.cursor.execute(sql, (policy_id,))
            self.conn.commit()
            return True
        except Error as e:
            print(f"Error deleting policy: {e}")
            self.conn.rollback()
            return False

policy_service = PolicyServiceImpl()


print('1 - Create policy  \n2 - Read Policy   \n3 - Update Policy \n4 - Delete Policy \n5  - Get all policies detail' )

val = int(input('Enter your choice : '))

match(val):
```

```
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/User.py"
Connected to MySQL server
1 - Create policy
2 - Read Policy
3 - Update Policy
4 - Delete Policy
5  - Get all policies detail
Enter your choice : 5
[{'policy_id': 2, 'policy_name': 'Accident policy', 'Coverage_Amount': Decimal('200000.00'), 'start_date': datetime.date(2024, 1, 1), 'end_date': datetime.date(2025, 1, 1)}, {'policy_id': 54, 'policy_name': 'star life', 'Coverage_Amount': Decimal('2100000.00'), 'start_date': datetime.date(2024, 1, 1), 'end_date
': datetime.date(2025, 1, 1)}, {'policy_id': 145, 'policy_name': 'Hv Health policy', 'Coverage_Amount': Decimal('150000.00'), 'start_date': datetime.date(2024, 1, 1), 'end_date': datetime.date(2025, 1, 1)}, {'policy_id': 256, 'policy_name': 'Mg insurance', 'Coverage_Amount': Decimal('110000.00'), 'start_date':
datetime.date(2024, 1, 1), 'end_date': datetime.date(2025, 1, 1)}, {'policy_id': 3655, 'policy_name': 'mur unsure', 'Coverage_Amount': Decimal('15000.00'), 'start_date': datetime.date(2024, 1, 1), 'end_date': datetime.date(2025, 1, 1)}]
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance>
```

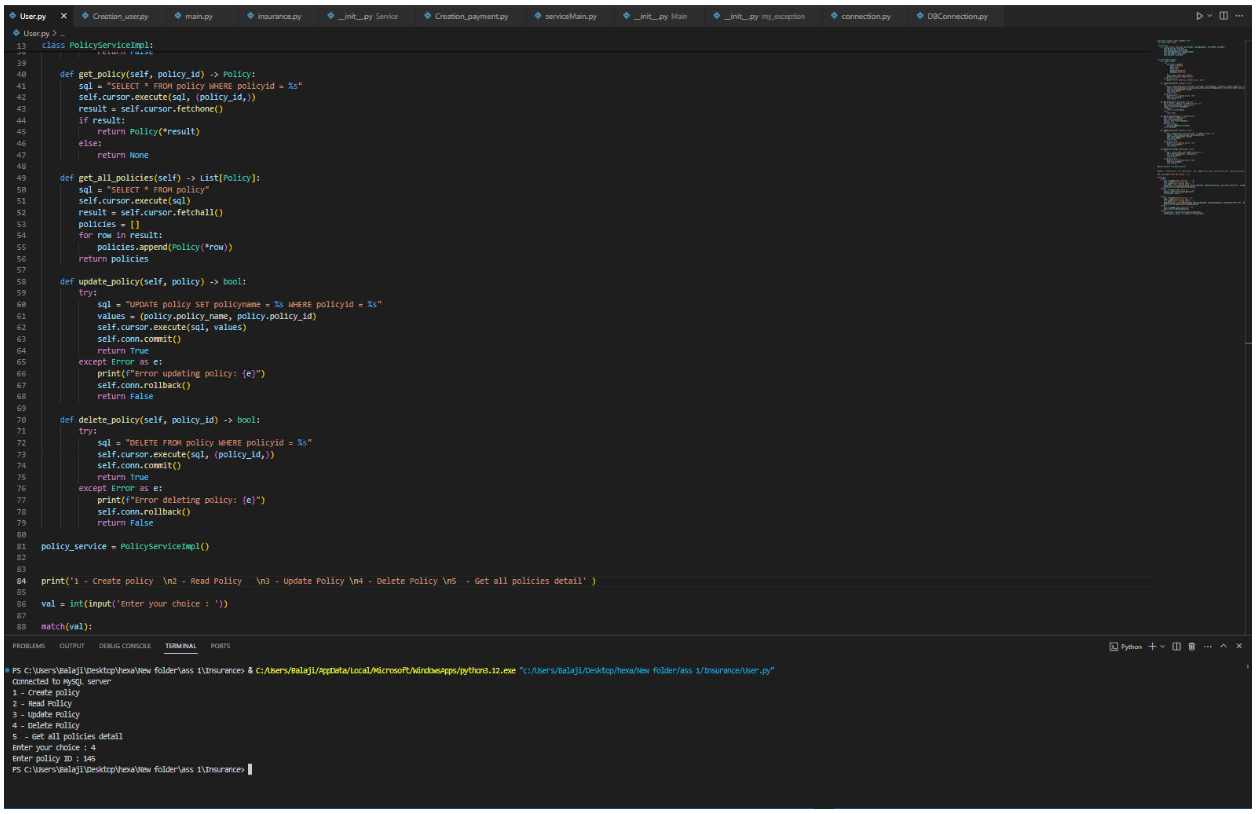
d.updatePolicy() I. parameters: Policy Object II. return type: Boolean



```python
class PolicyServiceImpl:

    def get_policy(self, policy_id) -> Policy:
        sql = "SELECT * FROM policy WHERE policyid = %s"
        self.cursor.execute(sql, (policy_id,))
        result = self.cursor.fetchone()
        if result:
            return Policy(*result)
        else:
            return None

    def get_all_policies(self) -> List[Policy]:
        sql = "SELECT * FROM policy"
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
        policies = []
        for row in result:
            policies.append(Policy(*row))
        return policies

    def update_policy(self, policy) -> bool:
        try:
            sql = "UPDATE policy SET policyname = %s WHERE policyid = %s"
            values = (policy.policy_name, policy.policy_id)
            self.cursor.execute(sql, values)
            self.conn.commit()
            return True
        except Error as e:
            print(f"Error updating policy: {e}")
            self.conn.rollback()
            return False

    def delete_policy(self, policy_id) -> bool:
        try:
            sql = "DELETE FROM policy WHERE policyid = %s"
            self.cursor.execute(sql, (policy_id,))
            self.conn.commit()
            return True
        except Error as e:
            print(f"Error deleting policy: {e}")
            self.conn.rollback()
            return False

policy_service = PolicyServiceImpl()


print('1 - Create policy  \n2 - Read Policy   \n3 - Update Policy \n4 - Delete Policy \n5  - Get all policies detail' )

val = int(input('Enter your choice : '))

match(val):
```

```
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/User.py"
Connected to MySQL server
1 - Create policy
2 - Read Policy
3 - Update Policy
4 - Delete Policy
5  - Get all policies detail
Enter your choice : 3
Enter poliy id  :145
Enter policy name :MGHV policy
Enter coverage amount : 140000
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance>
```

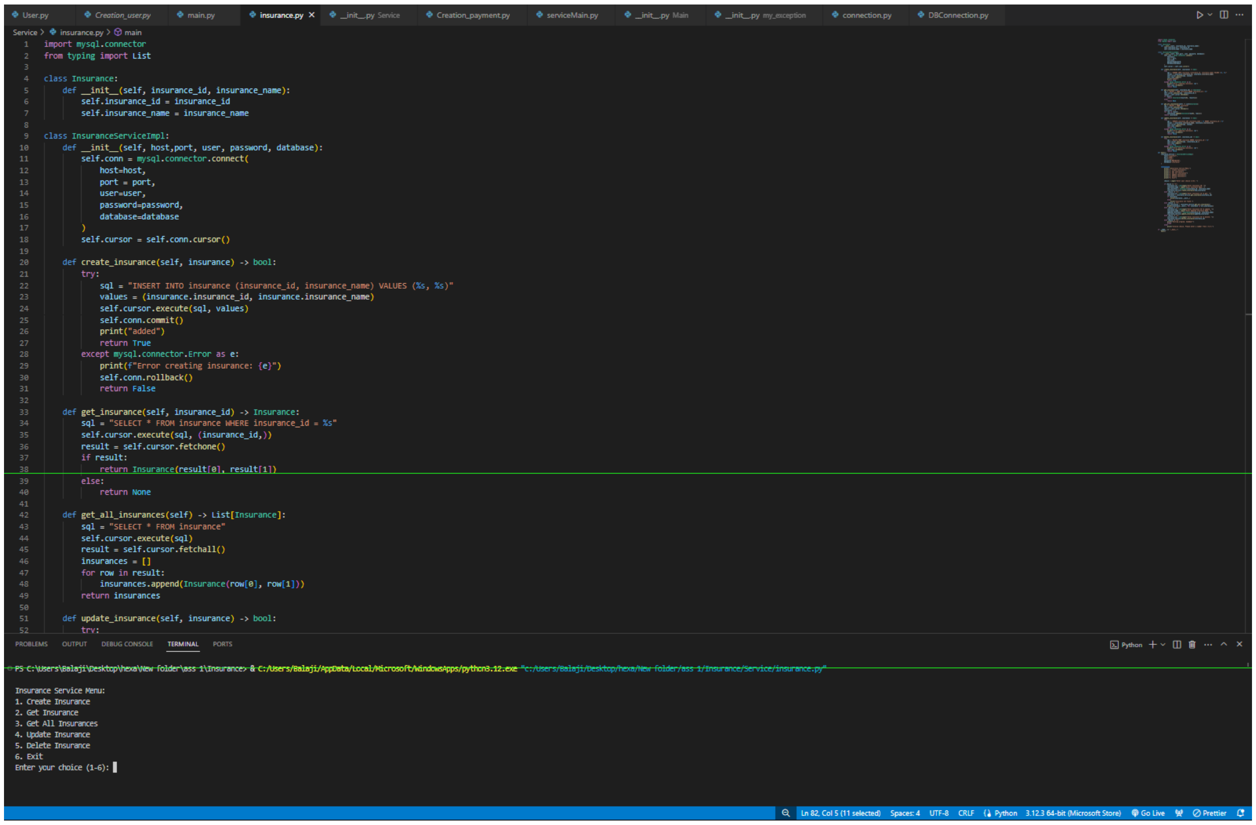
e. deletePolicy() I. parameters: PolicyId II. return type: Boolean



```python
class PolicyServiceImpl:

    def get_policy(self, policy_id) -> Policy:
        sql = "SELECT * FROM policy WHERE policyid = %s"
        self.cursor.execute(sql, (policy_id,))
        result = self.cursor.fetchone()
        if result:
            return Policy(*result)
        else:
            return None

    def get_all_policies(self) -> List[Policy]:
        sql = "SELECT * FROM policy"
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
        policies = []
        for row in result:
            policies.append(Policy(*row))
        return policies

    def update_policy(self, policy) -> bool:
        try:
            sql = "UPDATE policy SET policyname = %s WHERE policyid = %s"
            values = (policy.policy_name, policy.policy_id)
            self.cursor.execute(sql, values)
            self.conn.commit()
            return True
        except Error as e:
            print(f"Error updating policy: {e}")
            self.conn.rollback()
            return False

    def delete_policy(self, policy_id) -> bool:
        try:
            sql = "DELETE FROM policy WHERE policyid = %s"
            self.cursor.execute(sql, (policy_id,))
            self.conn.commit()
            return True
        except Error as e:
            print(f"Error deleting policy: {e}")
            self.conn.rollback()
            return False

policy_service = PolicyServiceImpl()


print('1 - Create policy  \n2 - Read Policy   \n3 - Update Policy \n4 - Delete Policy \n5  - Get all policies detail' )

val = int(input('Enter your choice : '))

match(val):
```

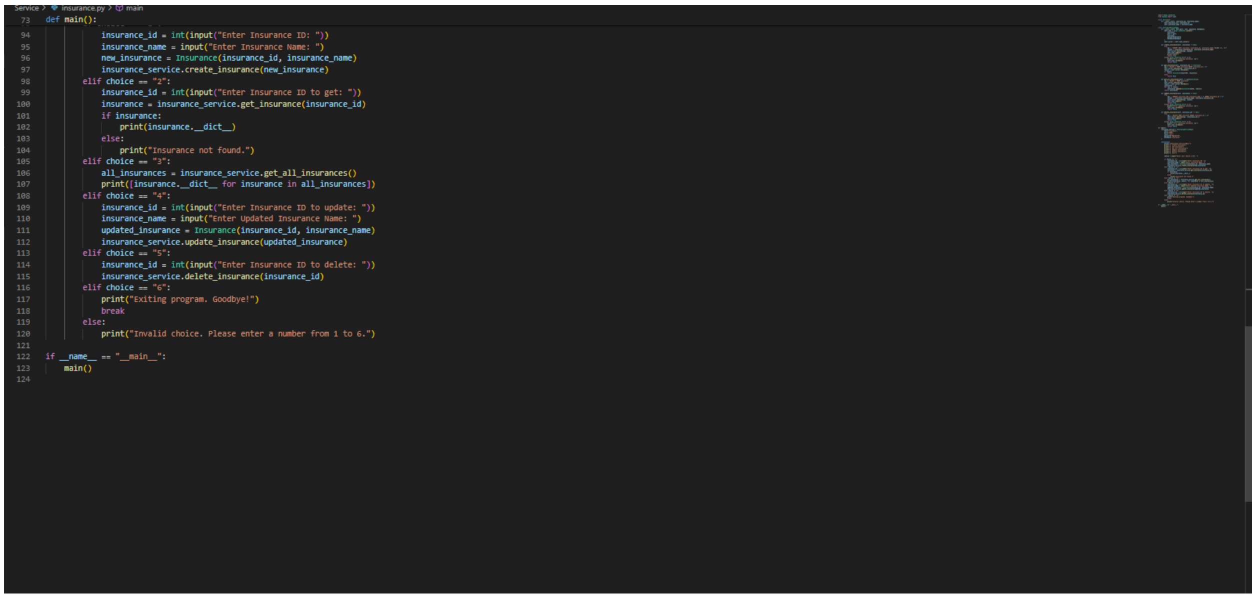
```
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/User.py"
Connected to MySQL server
1 - Create policy
2 - Read Policy
3 - Update Policy
4 - Delete Policy
5  - Get all policies detail
Enter your choice : 4
Enter policy ID : 145
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance>
```

6. Define InsuranceServiceImpl class and implement all the methods InsuranceServiceImpl



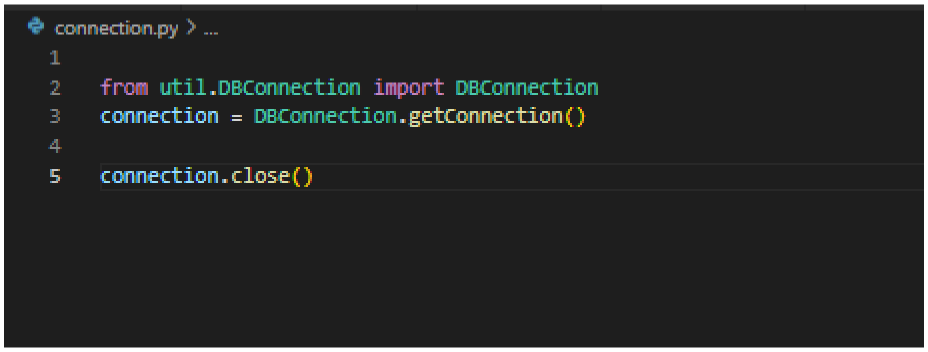
```python
import mysql.connector
from typing import List

class Insurance:
    def __init__(self, insurance_id, insurance_name):
        self.insurance_id = insurance_id
        self.insurance_name = insurance_name

class InsuranceServiceImpl:
    def __init__(self, host,port, user, password, database):
        self.conn = mysql.connector.connect(
            host=host,
            port = port,
            user=user,
            password=password,
            database=database
        )
        self.cursor = self.conn.cursor()

    def create_insurance(self, insurance) -> bool:
        try:
            sql = "INSERT INTO insurance (insurance_id, insurance_name) VALUES (%s, %s)"
            values = (insurance.insurance_id, insurance.insurance_name)
            self.cursor.execute(sql, values)
            self.conn.commit()
            print("added")
            return True
        except mysql.connector.Error as e:
            print(f"Error creating insurance: {e}")
            self.conn.rollback()
            return False

    def get_insurance(self, insurance_id) -> Insurance:
        sql = "SELECT * FROM insurance WHERE insurance_id = %s"
        self.cursor.execute(sql, (insurance_id,))
        result = self.cursor.fetchone()
        if result:
            return Insurance(result[0], result[1])
        else:
            return None

    def get_all_insurances(self) -> List[Insurance]:
        sql = "SELECT * FROM insurance"
        self.cursor.execute(sql)
        result = self.cursor.fetchall()
        insurances = []
        for row in result:
            insurances.append(Insurance(row[0], row[1]))
        return insurances

    def update_insurance(self, insurance) -> bool:
        try:
```

```
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/Service/insurance.py"

Insurance Service Menu:
1. Create Insurance
2. Get Insurance
3. Get All Insurances
4. Update Insurance
5. Delete Insurance
6. Exit
Enter your choice (1-6): 
```

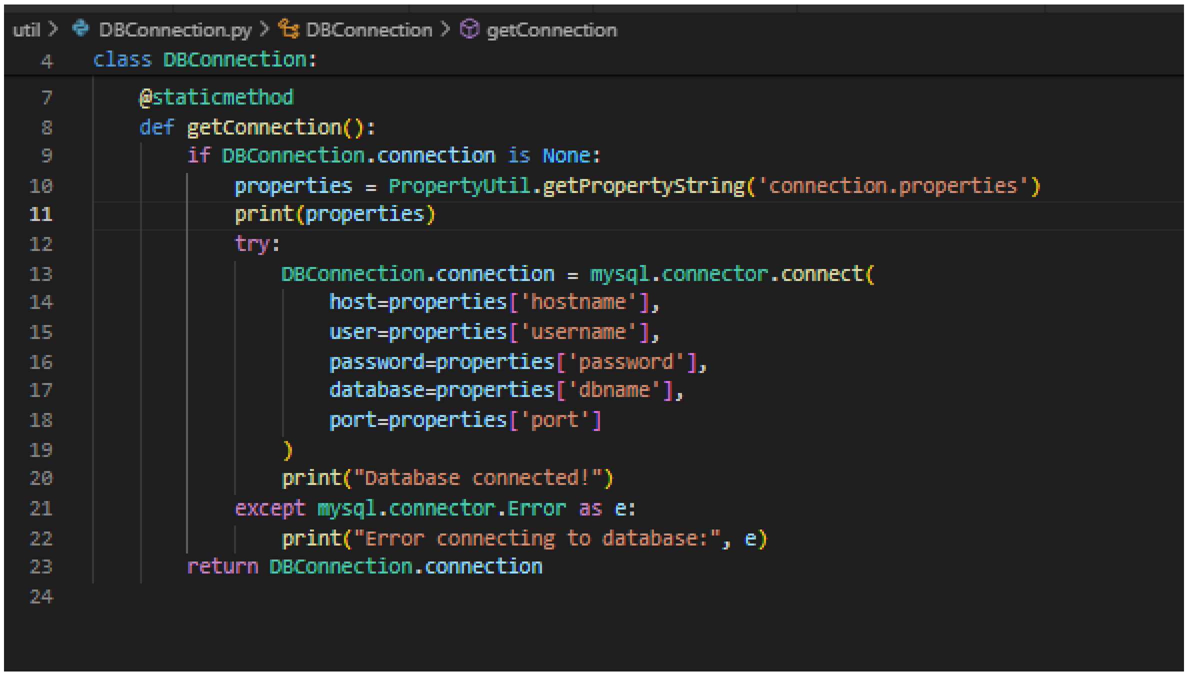


```python
def main():
            insurance_id = int(input("Enter Insurance ID: "))
            insurance_name = input("Enter Insurance Name: ")
            new_insurance = Insurance(insurance_id, insurance_name)
            insurance_service.create_insurance(new_insurance)
        elif choice == "2":
            insurance_id = int(input("Enter Insurance ID to get: "))
            insurance = insurance_service.get_insurance(insurance_id)
            if insurance:
                print(insurance.__dict__)
            else:
                print("Insurance not found.")
        elif choice == "3":
            all_insurances = insurance_service.get_all_insurances()
            print([insurance.__dict__ for insurance in all_insurances])
        elif choice == "4":
            insurance_id = int(input("Enter Insurance ID to update: "))
            insurance_name = input("Enter Updated Insurance Name: ")
            updated_insurance = Insurance(insurance_id, insurance_name)
            insurance_service.update_insurance(updated_insurance)
        elif choice == "5":
            insurance_id = int(input("Enter Insurance ID to delete: "))
            insurance_service.delete_insurance(insurance_id)
        elif choice == "6":
            print("Exiting program. Goodbye!")
            break
        else:
            print("Invalid choice. Please enter a number from 1 to 6.")

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

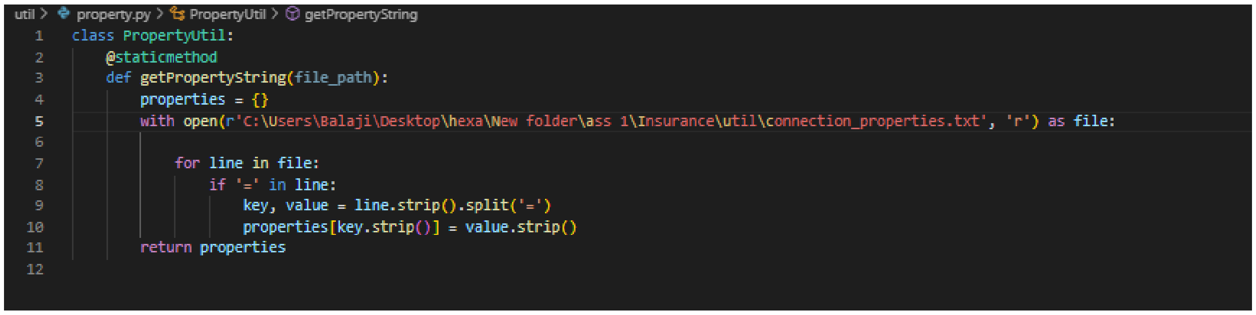
7. Create a utility class DBConnection in a package util with a static variable connection of Type Connection and a static method getConnection() which returns connection. Connection properties supplied in the connection string should be read from a property file. Create a utility class PropertyUtil which contains a static method named getPropertyString() which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.



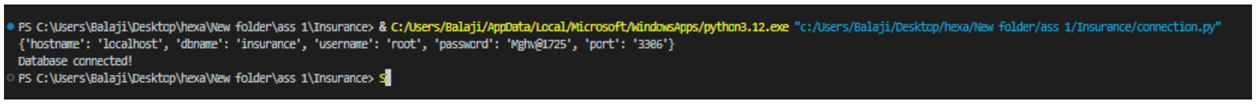
```python

from util.DBConnection import DBConnection
connection = DBConnection.getConnection()

connection.close()
```



```python
class DBConnection:
    @staticmethod
    def getConnection():
        if DBConnection.connection is None:
            properties = PropertyUtil.getPropertyString('connection.properties')
            print(properties)
            try:
                DBConnection.connection = mysql.connector.connect(
                    host=properties['hostname'],
                    user=properties['username'],
                    password=properties['password'],
                    database=properties['dbname'],
                    port=properties['port']
                )
                print("Database connected!")
            except mysql.connector.Error as e:
                print("Error connecting to database:", e)
        return DBConnection.connection

```



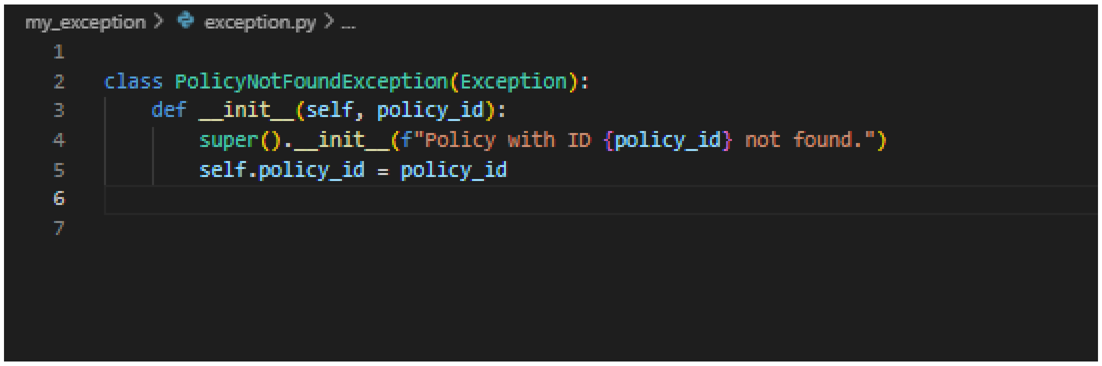
```python
class PropertyUtil:
    @staticmethod
    def getPropertyString(file_path):
        properties = {}
        with open(r'C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance\util\connection_properties.txt', 'r') as file:

            for line in file:
                if '=' in line:
                    key, value = line.strip().split('=')
                    properties[key.strip()] = value.strip()
        return properties

```



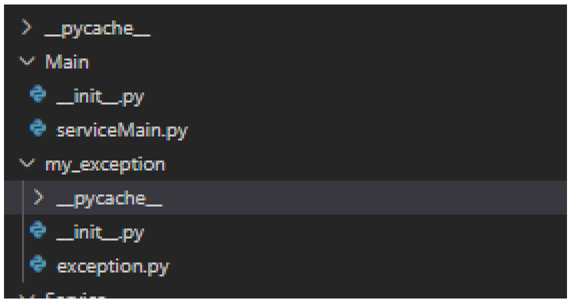
```
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> & C:/Users/Balaji/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Balaji/Desktop/hexa/New folder/ass 1/Insurance/connection.py"
{'hostname': 'localhost', 'dbname': 'insurance', 'username': 'root', 'password': 'Mghv@1725', 'port': '3306'}
Database connected!
PS C:\Users\Balaji\Desktop\hexa\New folder\ass 1\Insurance> $
```

8. Create the exceptions in package myexceptions Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method, 1. PolicyNotFoundException :throw this exception when user enters an invalid patient number which

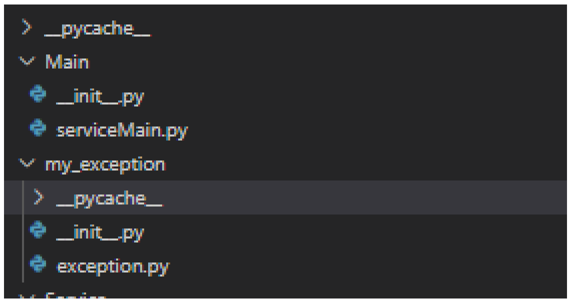
doesn't exist in db

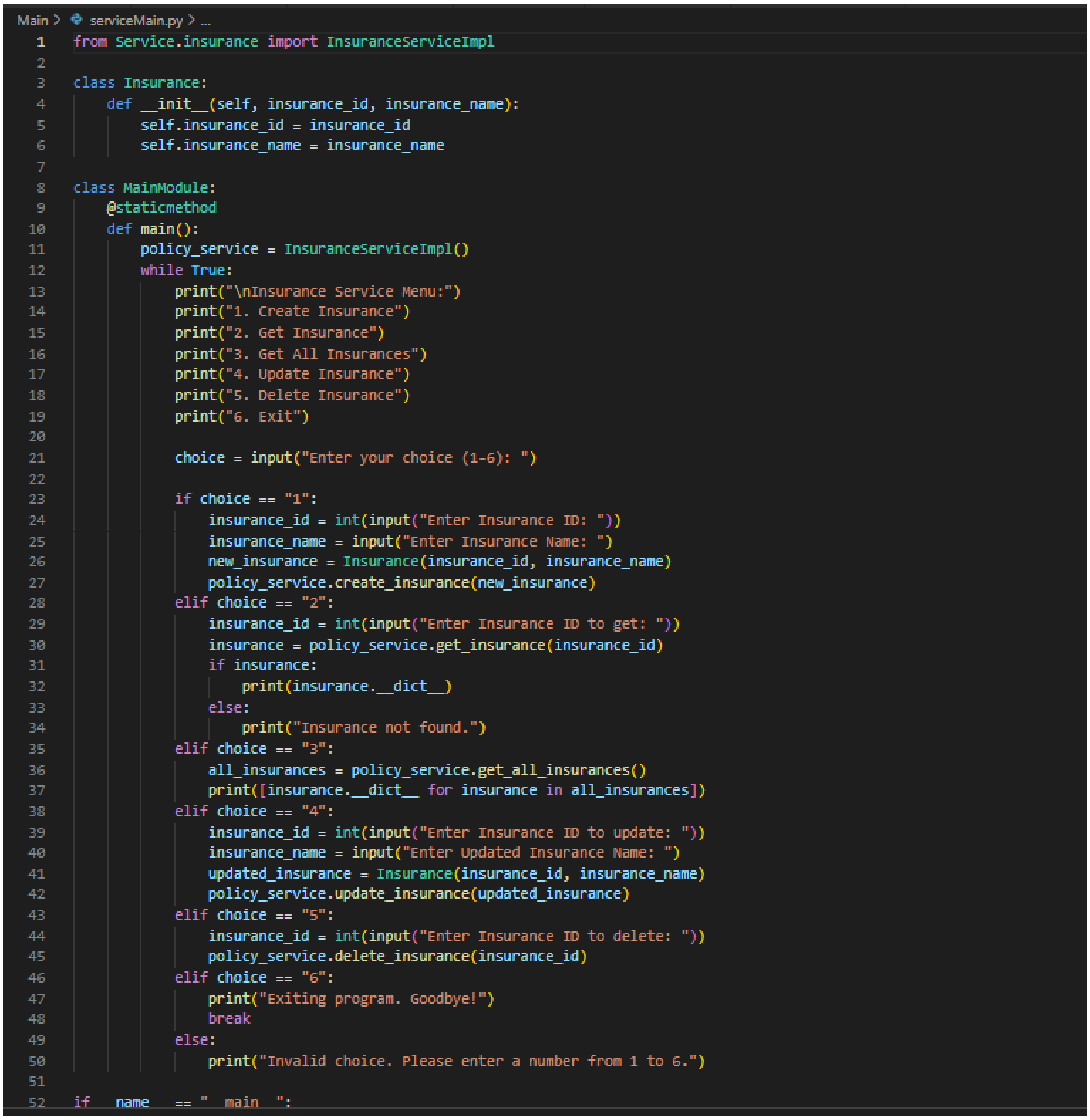

my_exception > exception.py > ...

```python

class PolicyNotFoundException(Exception):
    def __init__(self, policy_id):
        super().__init__(f"Policy with ID {policy_id} not found.")
        self.policy_id = policy_id

```

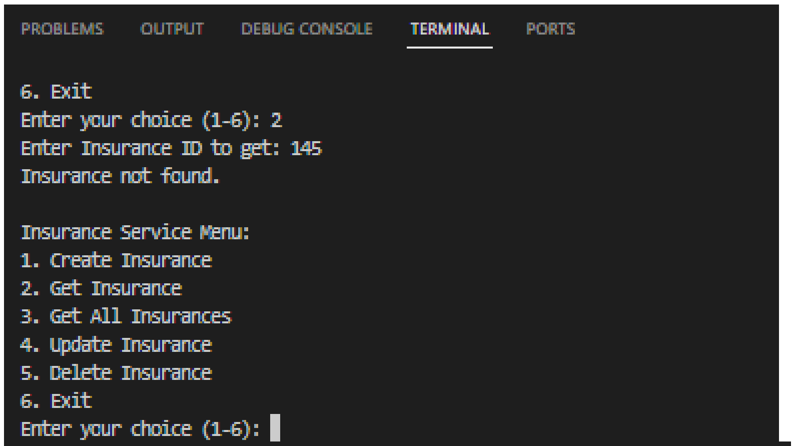

```
> __pycache__
v Main
  __init__.py
  serviceMain.py
v my_exception
  > __pycache__
  __init__.py
  exception.py
```

9. Create class named MainModule with main method in package mainmod. Trigger all the methods in service implementation class.

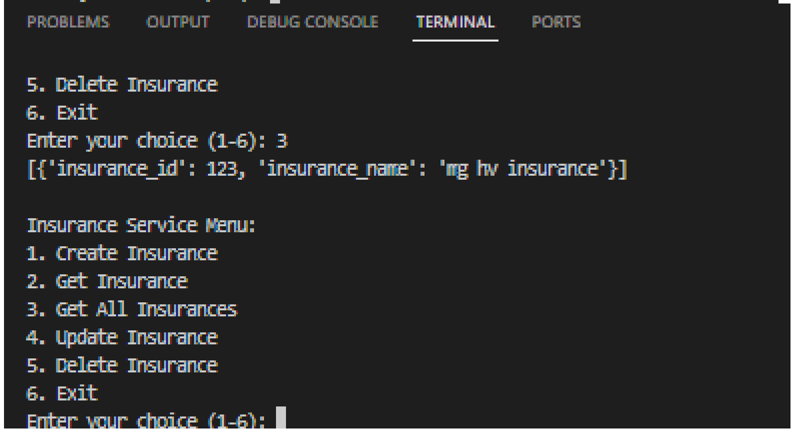

```
> __pycache__
v Main
  __init__.py
  serviceMain.py
v my_exception
  > __pycache__
  __init__.py
  exception.py
```



```python
from Service.insurance import InsuranceServiceImpl

class Insurance:
    def __init__(self, insurance_id, insurance_name):
        self.insurance_id = insurance_id
        self.insurance_name = insurance_name

class MainModule:
    @staticmethod
    def main():
        policy_service = InsuranceServiceImpl()
        while True:
            print("\nInsurance Service Menu:")
            print("1. Create Insurance")
            print("2. Get Insurance")
            print("3. Get All Insurances")
            print("4. Update Insurance")
            print("5. Delete Insurance")
            print("6. Exit")

            choice = input("Enter your choice (1-6): ")

            if choice == "1":
                insurance_id = int(input("Enter Insurance ID: "))
                insurance_name = input("Enter Insurance Name: ")
                new_insurance = Insurance(insurance_id, insurance_name)
                policy_service.create_insurance(new_insurance)
            elif choice == "2":
                insurance_id = int(input("Enter Insurance ID to get: "))
                insurance = policy_service.get_insurance(insurance_id)
                if insurance:
                    print(insurance.__dict__)
                else:
                    print("Insurance not found.")
            elif choice == "3":
                all_insurances = policy_service.get_all_insurances()
                print([insurance.__dict__ for insurance in all_insurances])
            elif choice == "4":
                insurance_id = int(input("Enter Insurance ID to update: "))
                insurance_name = input("Enter Updated Insurance Name: ")
                updated_insurance = Insurance(insurance_id, insurance_name)
                policy_service.update_insurance(updated_insurance)
            elif choice == "5":
                insurance_id = int(input("Enter Insurance ID to delete: "))
                policy_service.delete_insurance(insurance_id)
            elif choice == "6":
                print("Exiting program. Goodbye!")
                break
            else:
                print("Invalid choice. Please enter a number from 1 to 6.")

if __name__ == "__main__":
```



```
6. Exit
Enter your choice (1-6): 2
Enter Insurance ID to get: 145
Insurance not found.

Insurance Service Menu:
1. Create Insurance
2. Get Insurance
3. Get All Insurances
4. Update Insurance
5. Delete Insurance
6. Exit
Enter your choice (1-6):
```



```
5. Delete Insurance
6. Exit
Enter your choice (1-6): 3
[{'insurance_id': 123, 'insurance_name': 'mg hv insurance'}]

Insurance Service Menu:
1. Create Insurance
2. Get Insurance
3. Get All Insurances
4. Update Insurance
5. Delete Insurance
6. Exit
Enter your choice (1-6):
```