Problem Statement:
Problem Statement:
PetPals, The Pet Adoption Platform scenario is a software system designed to facilitate the adoption of
pets, such as dogs and cats, from shelters or rescue organizations. This platform serves as a digital
marketplace where potential adopters can browse and select pets, shelters can list available pets, and
donors can contribute to support animal welfare.
Implement OOPs
Create SQL Schema from the pet and user class, use the class attributes for table column names.
1.Create and implement the mentioned class and the structure in your application.
Pet Class:
Attributes:
• Name (string): The name of the pet.
• Age (int): The age of the pet.
• Breed (string): The breed of the pet.
Methods:
• Constructor to initialize Name, Age, and Breed.
• Getters and setters for attributes.
• ToString() method to provide a string representation of the pet.
Dog Class (Inherits from Pet):
Additional Attributes:
• DogBreed (string): The specific breed of the dog.
Additional Methods:
• Constructor to initialize DogBreed.
• Getters and setters for DogBreed.
Cat Class (Inherits from Pet):
Additional Attributes:
• CatColor (string): The color of the cat.
Additional Methods:
• Constructor to initialize CatColor.
• Getters and setters for CatColor.
3.PetShelter Class:
Attributes:
• availablePets (List of Pet): A list to store available pets for adoption.
Methods:
• AddPet(Pet pet): Adds a pet to the list of available pets.
• RemovePet(Pet pet): Removes a pet from the list of available pets.
• ListAvailablePets(): Lists all available pets in the shelter.
4.Donation Class (Abstract):
Attributes:
• DonorName (string): The name of the donor.

• Amount (decimal): The donation amount.
Methods:
• Constructor to initialize DonorName and Amount.
• Abstract method RecordDonation() to record the donation (to be implemented in derived classes).
CashDonation Class (Derived from Donation):
Additional Attributes:
• DonationDate (DateTime): The date of the cash donation.
Additional Methods:
• Constructor to initialize DonationDate.
• Implementation of RecordDonation() to record a cash donation
ItemDonation Class (Derived from Donation):
Additional Attributes:
• ItemType (string): The type of item donated (e.g., food, toys).
Additional Methods:
• Constructor to initialize ItemType.
• Implementation of RecordDonation() to record an item donation.
5.IAdoptable Interface/Abstract Class:
Methods:
• Adopt(): An abstract method to handle the adoption process.
AdoptionEvent Class:
Attributes:
• Participants (List of IAdoptable): A list of participants (shelters and adopters) in the adoption event.
Methods:
• HostEvent(): Hosts the adoption event.
• RegisterParticipant(IAdoptable participant): Registers a participant for the event.
6.Exceptions handling
Create and implement the following exceptions in your application.
• Invalid Pet Age Handling:
o In the Pet Adoption Platform, when adding a new pet to a shelter, the age of the pet should be a positive integer. Write a program that prompts the user to input the age of a pet. Implement exception handling to ensure that the input is a positive integer. If the input is not valid, catch the exception and display an error message. If the input is valid, add the pet to the shelter.
• Null Reference Exception Handling:
o In the Pet Adoption Platform, when displaying the list of available pets in a shelter, it's important to handle situations where a pet's properties (e.g., Name, Age) might be null. Implement exception handling to catch null reference exceptions when accessing properties of pets in the shelter and display a message indicating that the information is missing.
• Insufficient Funds Exception:
o Suppose the Pet Adoption Platform allows users to make cash donations to shelters. Write a program that prompts the user to enter the donation amount. Implement exception handling to catch situations where the donation amount is less than a minimum allowed

amount (e.g., $10). If the donation amount is insufficient, catch the exception and display an error message. Otherwise, process the donation.

• File Handling Exception:

o In the Pet Adoption Platform, there might be scenarios where the program needs to read data from a file (e.g., a list of pets in a shelter). Write a program that attempts to read data from a file. Implement exception handling to catch any file-related exceptions (e.g., FileNotFoundException) and display an error message if the file is not found or cannot be read.

• Custom Exception for Adoption Errors:

o Design a custom exception class called AdoptionException that inherits from Exception. In the Pet Adoption Platform, use this custom exception to handle adoption-related errors, such as attempting to adopt a pet that is not available or adopting a pet with missing information. Create instances of AdoptionException with different error messages and catch them appropriately in your program.

7.Database Connectivity

Create and implement the following tasks in your application.

• Displaying Pet Listings:

o Develop a program that connects to the database and retrieves a list of available pets from the "pets" table. Display this list to the user. Ensure that the program handles database connectivity exceptions gracefully, including cases where the database is unreachable.

• Donation Recording:

o Create a program that records cash donations made by donors. Allow the user to input donor information and the donation amount and insert this data into the "donations" table in the database. Handle exceptions related to database operations, such as database errors or invalid inputs.

• Adoption Event Management:

o Build a program that connects to the database and retrieves information about upcoming adoption events from the "adoption_events" table. Allow the user to register for an event by adding their details to the "participants" table. Ensure that the program handles database connectivity and insertion exceptions properly.

CODE:

```
import sqlite3
from datetime import datetime

class Pet:
    def __init__(self, name, age, breed):
        self.name = name
        self.age = age
        self.breed = breed

    def __str__(self):
        return f"{self.name} - {self.age} years old ({self.breed})"

class Dog(Pet):
    def __init__(self, name, age, breed, dog_breed):
        super().__init__(name, age, breed)
        self.dog_breed = dog_breed

    def __str__(self):
        return f"{super().__str__()} - {self.dog_breed}"

class Cat(Pet):
    def __init__(self, name, age, breed, cat_color):
        super().__init__(name, age, breed)
        self.cat_color = cat_color

    def __str__(self):
        return f"{super().__str__()} - {self.cat_color}"

class PetShelter:
    def __init__(self):
        self.available_pets = []

    def add_pet(self, pet):
        self.available_pets.append(pet)

    def remove_pet(self, pet):
        self.available_pets.remove(pet)

    def list_available_pets(self):
        for pet in self.available_pets:
            print(pet)

class Donation:
    def __init__(self, donor_name, amount):
        self.donor_name = donor_name
        self.amount = amount

    def record_donation(self):
        pass

class CashDonation(Donation):
    def __init__(self, donor_name, amount, donation_date):
        super().__init__(donor_name, amount)
        self.donation_date = donation_date
    def record_donation(self):
        pass

class ItemDonation(Donation):
    def __init__(self, donor_name, amount, item_type):
        super().__init__(donor_name, amount)
        self.item_type = item_type

    def record_donation(self):
        pass

class IAdoptable:
    def adopt(self):
        pass

class AdoptionEvent:
    def __init__(self):
        self.participants = []

    def host_event(self):
        pass

    def register_participant(self, participant):
        self.participants.append(participant)

class InvalidPetAgeException(Exception):
    pass

class NullReferenceException(Exception):
    pass

class InsufficientFundsException(Exception):
    pass

class FileHandlingException(Exception):
    pass

class AdoptionException(Exception):
    pass

class DatabaseConnector:
    def __init__(self, database_name):
        self.conn = sqlite3.connect(database_name)
        self.create_tables()

    def create_tables(self):
        pass

    def get_available_pets(self):
        pass

    def record_cash_donation(self, donor_name, amount, donation_date):
        pass

    def get_upcoming_adoption_events(self):
        pass

    def register_for_event(self, participant):
        pass

shelter = PetShelter()

try:
    pet = Pet("Buddy", -2, "Labrador")
except InvalidPetAgeException:
    print("Invalid pet age. Please enter a positive integer for the pet's age.")

try:
    pet_with_null_properties = Pet(None, 3, "German Shepherd")
except NullReferenceException:
    print("Pet properties cannot be null. Please provide valid information.")

try:
    donation_amount = 5
    if donation_amount < 10:
        raise InsufficientFundsException("Donation amount must be at least $10.")
except InsufficientFundsException as e:
    print(e)

try:
    with open("nonexistent_file.txt", "r") as file:
        data = file.read()
except FileNotFoundError:
    print("File not found. Please check the file path.")

try:
    adoption_event = AdoptionEvent()
    participant = "InvalidParticipant"
    adoption_event.register_participant(participant)
except AdoptionException as e:
    print(e)

db_connector = DatabaseConnector("pet_adoption_database.db")

try:
    available_pets = db_connector.get_available_pets()
except Exception as e:
    print(f"Error connecting to the database: {e}")

try:
    db_connector.record_cash_donation("John Doe", 20, datetime.now())
except Exception as e:
    print(f"Error recording cash donation: {e}")

try:
    upcoming_events = db_connector.get_upcoming_adoption_events()
    event = AdoptionEvent()
    db_connector.register_for_event(event)
except Exception as e:
    print(f"Error registering for adoption event: {e}")
```