| Nr | Obszar | Wymaganie | KOD |  | Przyznane  pkt | Pkt  max |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | UI | JEST | import tkinter as tk | ☑ |  |  |
| Wprowadzanie  danych | def open\_add\_animal\_window(self):  name\_entry = ttk.Entry(add\_window, width=20) | ☑ |  | 2 |
| Wyświetlanie  danych | def refresh\_animals\_tree(self):  self.animals\_tree.insert("", "end", values=(  animal\_id, animal.name, animal.age, …) | ☑ |  | 2 |
| Zmiana danych | def open\_edit\_animal\_window(self):  def save\_changes():  name = name\_entry.get().strip() | ☑ |  | 2 |
| Wyszukiwanie  danych | def open\_animal\_search\_window(self):  …  def perform\_search():  …  filtered = [(k, v) for k, v in filtered if id\_query == k] | ☑ |  | 2 |
| Przedstawienie  wyników | def open\_animal\_search\_window(self):  …  def perform\_search():  …  results\_tree.insert("", "end", values=(  animal\_id, animal.name, animal.age, …) | ☑ |  | 2 |
| 2 | Podstawy | Zmienne | species = species\_var.get()  animal\_id = str(self.next\_id) | ☑ |  | 2 |
| typy danych | age = int(age\_entry.get().strip())  animal\_id = str(self.next\_id) | ☑ |  | 2 |
| komentarze | class ShelterApp:  # Inicjalizacja aplikacji z interfejsem i danymi  def \_\_init\_\_(self, root): | ☑ |  | 1 |
| operatory | assert name and age >= 0  if adoption["animal\_id"] == animal\_id:  self.next\_id += 1 | ☑ |  | 1,5 |
| Instrukcje warunkowe (if, elif,  else) | def get\_key(x):  if self.animals\_sort\_default:  return int(x[0])  if self.animals\_sort\_column == "ID":  return int(x[0])  elif self.animals\_sort\_column == "Imię":  …  else:  return "Adoptowane" if x[1].is\_adopted else "W schronisku" | ☑ |  | 3 |
| Instrukcje iteracyjne |  |  |  |  |
| for | for animal\_id, animal in sorted\_items: | ☑ |  | 2 |
| while | while row\_index <len(list(reader)): | ☑ |  | 2 |
| Operacje wejścia  (input) | def open\_add\_animal\_window(self):  name\_entry = ttk.Entry(add\_window, width=20) | ☑ |  | 1,5 |
| Operacje wyjścia  (print) | except Exception as e:  print(f"Błąd zapisu zwierząt: {e}") | ☑ |  | 1,5 |
| Funkcje z parametrami i wartościami  zwracanymi | def get\_feeding\_status(self):  last\_fed\_time = datetime.strptime(self.last\_fed, "%Y-%m-%d %H:%M:%S")  delta = datetime.now() - last\_fed\_time | ☑ |  | 2 |
| Funkcje  rekurencyjne | def count\_animals\_by\_type(self, animals, animal\_class, index=0):  if index >= len(animals):  return 0  count = 1 if isinstance(animals[index][1], animal\_class) and not animals[index][1].is\_adopted else 0  return count + self.count\_animals\_by\_type(animals, animal\_class, index + 1) | ☑ |  | 3 |
| Funkcje przyjmujące inne funkcje jako  argumenty | def log\_action(func):  @wraps(func)  def wrapper(\*args, \*\*kwargs):  return func(\*args, \*\*kwargs)  return wrapper | ☑ |  | 3 |
| Dekoratory | def log\_action(func):  @wraps(func)  def wrapper(\*args, \*\*kwargs):  return func(\*args, \*\*kwargs)  return wrapper | ☑ |  | 1,5 |
| 3 | Kontenery | Użycie listy | filtered = list(self.animals.items()) | ☑ |  | 2 |
| Użycie słownika | self.animals = {str(k): self.species\_map[v["species"]](str(k), v["name"], v["age"]) for k, v in data["animals"].items()} | ☑ |  | 2 |
| Użycie zbioru |  | ☐ |  | 1,5 |
| Użycie krotki | values=(  animal\_id, animal.name, animal.age | ☑ |  | 1,5 |
| 4 | Przestrzenie nazw | Zastosowano  zmienne lokalne | name = name\_entry.get().strip() | ☑ |  | 1,5 |
| Zastosowano  zmienne globalne | self.animals = self.data\_manager.animals | ☑ |  | 1,5 |
| Zastosowano  zakresy funkcji | def validate\_and\_add():  try:  name = name\_entry.get().strip()  age = int(age\_entry.get().strip())  assert name and age >= 0 | ☑ |  | 1,5 |
| Zastosowano  zakresy klas | self.animals = {} | ☑ |  | 1,5 |
| 5 | Moduły i pakiety | Projekt podzielony na moduły (import,  init ) | from data\_manager import DataManager  def \_\_init\_\_(self, root):  self.root = root  self.root.title("System Schroniska")  self.root.geometry("1200x800") | ☑ |  | 2 |

| Nr | Obszar | Wymaganie | KOD |  | Przyznane  pkt | Pkt  max |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Własne pakiety/funkcje pomocnicze w osobnych plikach  .py | data\_manager.py:  import json  import csv  from datetime import datetime  from animal\_manager import Dog, Cat, Bird, Rabbit, Hamster, Turtle  # Klasa zarządzająca danymi zwierząt i adopcji  class DataManager:  # Inicjalizacja menedżera danych z nazwami plików  def \_\_init\_\_(self, animals\_filename, adoptions\_filename):  self.animals\_filename = animals\_filename  self.adoptions\_filename = adoptions\_filename  self.animals = {}  self.adoptions = {} | ☑ |  | 2 |
| 6 | Obsługa błędów | Obsługa wyjątków  (try, except, finally) | try:  with open(self.animals\_filename, 'r', encoding='utf-8') as f:  data = json.load(f)  except (FileNotFoundError, json.JSONDecodeError, KeyError): | ☑ |  | 2 |
| Użycie assert do  testów i walidacji | assert name and age >= 0, "Imię puste lub wiek ujemny" | ☑ |  | 1,5 |
| 7 | Łańcuchy znaków | Operacje na stringach (m.in. formatowanie, dzielenie,  wyszukiwanie) | name\_query = name\_entry.get().strip().lower() | ☑ |  | 2 |
| 8 | Obsługa plików | Odczyt z plików .txt,  .csv, .json, .xml (min. 1) | with open(file\_path, 'r', encoding='utf-8') as f:  reader = csv.reader(f, delimiter=';') | ☑ |  | 2 |
| Zapis do plików .txt,  .csv, .json, .xml (min. 1) | with open(file\_path, 'w', newline='', encoding='utf-8') as f:  writer = csv.writer(f, delimiter=';') | ☑ |  | 2 |
| 9 | OOP | Klasy | class Animal:  def \_\_init\_\_(self, id, name, age):  self.id = id  self.name = name  self.age = age | ☑ |  | 2 |
| Metody | def get\_feeding\_status(self):  if not self.last\_fed:  return "Brak danych" | ☑ |  | 2 |
| Konstruktory | def \_\_init\_\_(self, root):  self.root = root  self.root.title("System Schroniska")  self.root.geometry("1200x800") | ☑ |  | 2 |
| Dziedziczenie | class Dog(Animal):  #Dziedziczone  def get\_feeding\_status(self):  if not self.last\_fed:  return "Brak danych" | ☑ |  | 2 |
| 10 | Programowanie funkcyjne | map |  | ☐ |  | 1,5 |
| filter | vaccinated = len(list(filter(lambda x: x[1].is\_vaccinated and not x[1].is\_adopted, animals\_list))) | ☑ |  | 1,5 |
| lambda | get\_key = lambda x: int(x[0]) if col == "ID" else x[1].name | ☑ |  | 1,5 |
| reduce |  | ☐ |  | 1,5 |
| 11 | Wizualizacja danych | Wygenerowano wykres (np.  matplotlib, seaborn) | def generate\_report():  ...  if chart\_data\_var.get() == "Szczepienia":  labels, data = ["Zaszczepione", "Niezaszczepione"], [vaccinated, not\_vaccinated] | ☑ |  | 2 |
| Zapisano wykres do pliku graficznego  (.png lub .jpg) | def save\_chart():  fig.savefig(file\_path) | ☑ |  | 1,5 |
| T12 | Testowanie | Testy jednostkowe (assert, unittest,  pytest) | def test\_get\_feeding\_status(self):  animal = Dog("1", "Reksio", 5)  animal.last\_fed = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  self.assertTrue("godziny temu" in animal.get\_feeding\_status())  animal.last\_fed = None  self.assertEqual(animal.get\_feeding\_status(), "Brak danych")  animal.last\_fed = "invalid\_date"  self.assertEqual(animal.get\_feeding\_status(), "Nieprawidłowy format daty") | ☑ |  | 1,5 |
| Testy funkcjonalne | def test\_add\_animal(self):  self.app.animals = {}  self.app.next\_id = 1  animal = Dog("1", "Reksio", 5)  animal.is\_vaccinated = True  animal.admission\_date = "2025-06-20 02:00:00"  self.app.animals["1"] = animal  self.app.data\_manager.save\_animals(self.app.animals, 2, 1)  self.app.refresh\_animals\_tree()  items = self.app.animals\_tree.get\_children()  self.assertEqual(len(items), 1)  values = self.app.animals\_tree.item(items[0])["values"]  self.assertEqual(values[1], "Reksio")  self.assertEqual(values[2], 5)  self.assertEqual(values[3], "Pies") | ☑ |  | 1,5 |
| Testy Integracyjne | def test\_export\_import\_animals(self):  animal = Dog("1", "Reksio", 5)  animal.is\_vaccinated = True  self.data\_manager.animals = {"1": animal}  self.data\_manager.save\_animals(self.data\_manager.animals, 2, 1)  self.data\_manager.export\_animals\_csv(self.test\_csv)  self.data\_manager.animals = {}  self.data\_manager.import\_animals\_csv(self.test\_csv, replace=True)  self.assertIn("1", self.data\_manager.animals)  self.assertEqual(self.data\_manager.animals["1"].name, "Reksio")  self.assertTrue(self.data\_manager.animals["1"].is\_vaccinated) | ☑ |  | 1,5 |
| Testy graniczne /  błędne dane | with open(self.test\_csv, "w", encoding="utf-8") as f:  f.write("ID;Imię;Wiek;Gatunek;Zaszczepione;Ostatnie karmienie;Data przyjęcia;Status\n")  f.write("1;;-1;Nieznany;Tak;invalid\_date;invalid\_date;Adoptowane\n")  errors = self.data\_manager.import\_animals\_csv(self.test\_csv, replace=True)  self.assertIn("Imię nie może być puste", errors[0]) | ☑ |  | 1,5 |
| Testy wydajności  (np. czas wykonania, timeit) | @pytest.mark.performance  def test\_performance\_save\_animals(self):  duration = timeit(lambda: self.dm.save\_animals(self.animals, 1001, 1), number=100)  self.assertLess(duration, 1.0) # 100 zapisów < 1s | ☑ |  | 1,5 |
| Testy pamięci  memory\_profiler | @profile  def test\_memory\_save\_animals(self):  animals = {str(i): Dog(str(i), f"Animal{i}", 5) for i in range(1000)}  self.data\_manager.save\_animals(animals, 1001, 1) | ☑ |  | 1,5 |
| Test jakości kodu  (flake8, pylint) |  | ☐ |  | 1,5 |
| 13 | Wersjonowanie | Repozytorium GIT | .idea  Wgranie projektu na GitHub  last week  Test  Wgranie projektu na GitHub  last week  dist  Dodanie pliku wykonywalnego  last week  Adopcje.json  Wgranie projektu na GitHub  last week  Adopcje\_przyk.csv  Przykladowe dane  yesterday  README.md  Aktualizacja README  last week  Zwierzeta.json  Wgranie projektu na GitHub  last week  Zwierzeta\_przyk.csv  Przykladowe dane  last week  animal\_manager.py  Wgranie projektu na GitHub  last week  data\_manager.py  Wgranie projektu na GitHub  last week  decorators.py  Wgranie projektu na GitHub  last week  main.py  Wgranie projektu na GitHub  last week  wykres.png  Wgranie projektu na GitHub  last week | ☑ |  | 1 |
| Historia commitów | Dokumentacja i formularz  KosaK27pushed 1 commit • 954bfab…4cec4e5 • yesterday  Aktualizacja README  KosaK27pushed 1 commit • 9ada48f…954bfab • 6 days ago  Merge remote-tracking branch 'origin/master'  KosaK27pushed 2 commits • c0930b6…9ada48f • 6 days ago  Delete Aplikacja directory  KosaK27pushed 1 commit • 6ed06ed…c0930b6 • 6 days ago  Dodanie pliku wykonywalnego  KosaK27pushed 1 commit • d2413eb…6ed06ed • 6 days ago  Przykladowe dane  KosaK27pushed 1 commit • 0a20cf8…d2413eb • 6 days ago  Create README.md  KosaK27pushed 1 commit • df1492d…0a20cf8 • 6 days ago  Wgranie projektu na GitHub  KosaK27created this branch • df1492d • 6 days ago | ☑ |  | 1 |

| Nr | Obszar | Wymaganie | KOD |  | Przyznane  pkt | Pkt  max |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Link do GitHub | https://github.com/KosaK27/ProjektJS | ☑ |  | 1 |
| Opis commitów | Dokumentacja i formularz  KosaK27pushed 1 commit to master • 954bfab…4cec4e5 • yesterday  Aktualizacja README  KosaK27pushed 1 commit to master • 9ada48f…954bfab • 6 days ago | ☑ |  | 1 |
| 14 | Dokumentacja | Plik README.md  (cel, autorzy, uruchamianie) | Cel projektu  Aplikacja ułatwiająca prowadzenie opieki nad zwierzętami w schronisku  Autorzy  Jakub Kosatka, Oleksandr Stankevych  Uruchamianie  Uruchom plik 'Schronisko.exe' znajdujący się w katalogu 'dist' | ☑ |  | 1,5 |
| Przykładowe dane wejściowe i  wyjściowe | ID;Imię;Wiek;Gatunek;Zaszczepione;Ostatnie karmienie;Data przyjęcia;Status  1;Burek;5;Pies;Tak;2025-06-10 09:15:47;2025-01-20 12:30:11;Adoptowane  2;Goku;3;Kot;Nie;2025-06-05 16:22:19;2025-02-15 08:45:33;Adoptowane  ID;ID zwierzęcia;Nazwisko;PESEL;Numer telefonu;Data adopcji  1;1;Kowalski;80051234567;123456789;2025-05-15 14:22:35  2;2;Kowalski;80051234567;123456789;2025-05-20 16:10:47 | ☑ |  | 2 |
| Diagram klas lub  struktura modułów | ProjektJS/  │  ├── main.py  ├── animal\_manager.py  ├── data\_manager.py  ├── decorators.py  │  └── test/  └── test.py | ☑ |  | 2 |
|  | | SUMA | |  | |  |