# University Of Asia Pacific

**Technical Report on**

## Animal Classification and Characteristics Knowledge Base in Prolog

***Course Title: Artificial Intelligence & Expert System Lab***
***Course Code: CSE 404***

**Prepared for:**
**Bidita Sarkar Diba**
**Lecturer, Dept. of CSE**
**University of Asia Pacific**

**Prepared by:**
**Anika Nawer Nabila**
**ID No:** 22101152
**Sec: D**

**Date**: **11 August 2025**

**Department of Computer Science and Engineering**
**University of Asia Pacific**

# i. Problem Title

**Animal Classification and Characteristics Knowledge Base in Prolog**

---

# ii. Problem Description

This project is designed to implement a **knowledge base** in **Prolog** for the classification of animals and for querying their characteristics through logical reasoning. The purpose is to demonstrate how **facts** and **rules** can be combined to form an intelligent system capable of answering queries based on stored information.

The knowledge base contains a collection of **facts** that define:

- Different categories of animals, such as **mammals**, **birds**, and **fish**.
- Specific traits of each animal, such as whether they **have fur**, **can fly**, or **lay eggs**.
- Special conditions or exceptions for certain animals (e.g., penguins are birds but cannot fly).

In addition to the facts, the project defines **Prolog rules** to determine:

1. Whether a given animal is classified as a mammal.
2. Whether an animal can truly fly, accounting for exceptions like penguins and other flightless birds.
3. Identification of animals that live in water but are **not fish** (e.g., dolphins).
4. Listing of mammals that have fur.
5. Determining whether two animals belong to the same classification category.

The rules in this knowledge base enable Prolog to **infer new information** from existing facts, allowing complex queries such as:

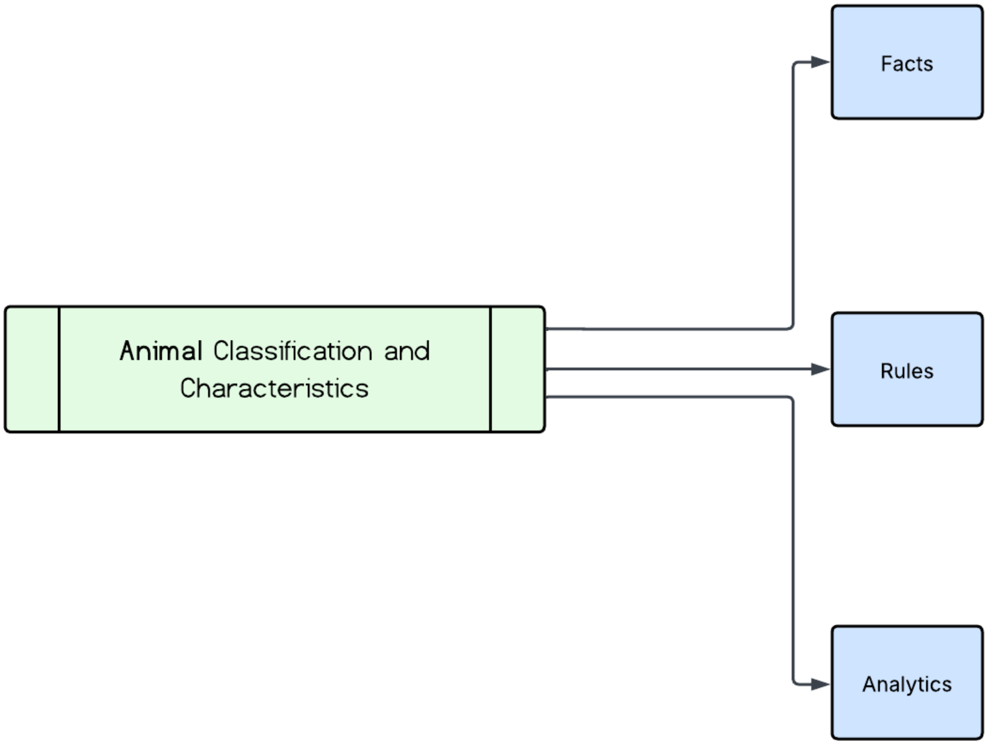
- "Is a dolphin a mammal?"
- "Which animals can fly?"
- "Which mammals have fur?"
- "What animals live in water but are not fish?"
- "Do a tiger and an elephant belong to the same class?"

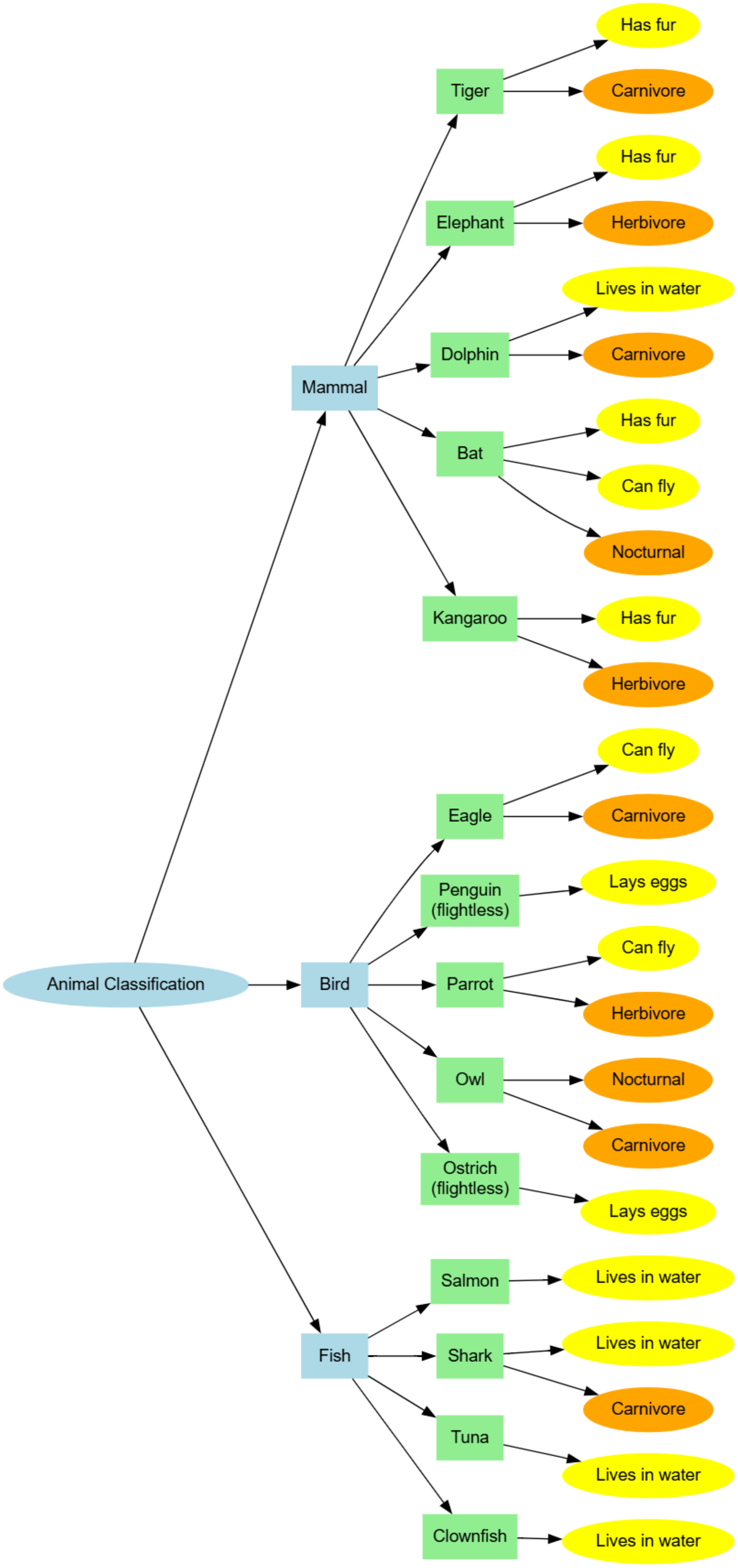
The **main objective** of this project is to illustrate the use of **declarative programming** with Prolog to represent knowledge logically, handle exceptions, and apply inference to answer queries. It showcases how **symbolic AI** can be applied in classification problems and highlights Prolog’s strength in pattern matching and logical deduction.

## iii. Tools and Languages Used

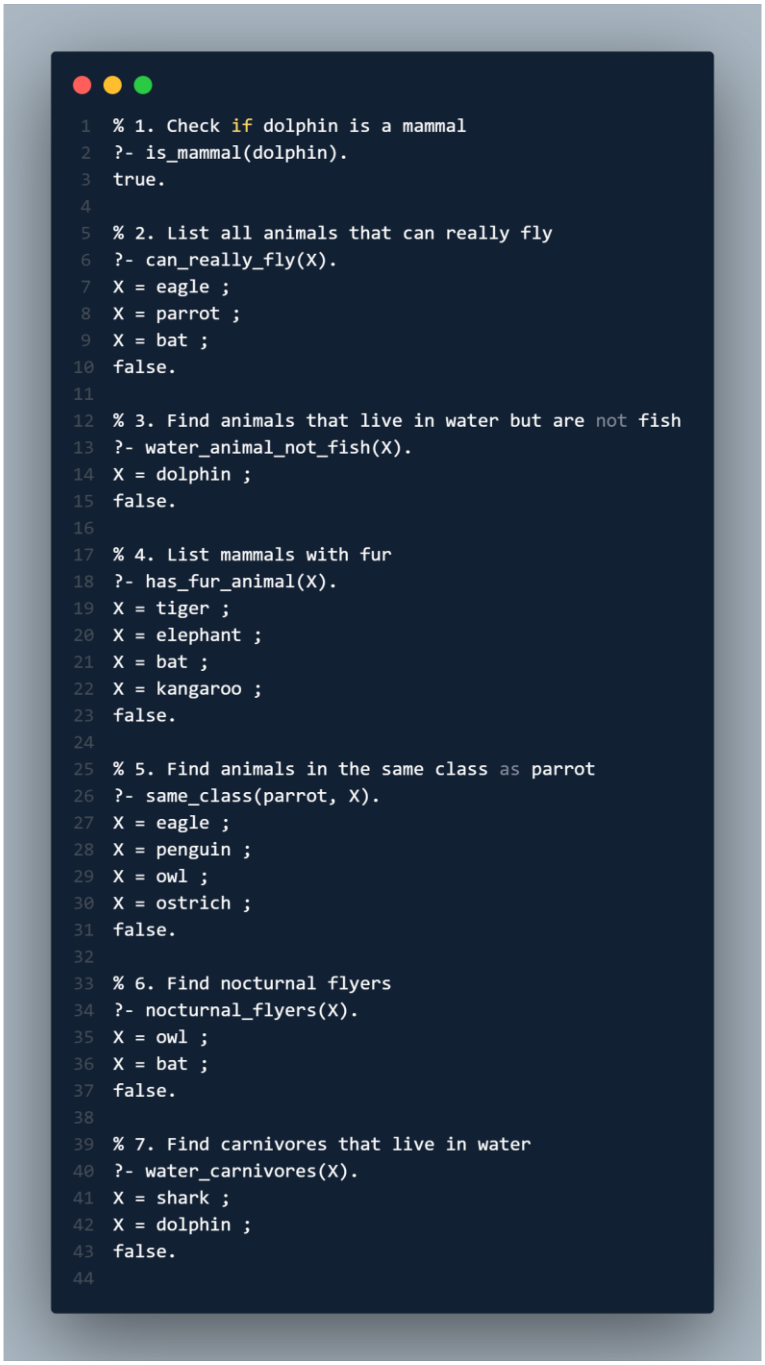
- **Language:** Prolog (*Logic Programming Language*) — Chosen for its declarative style, allowing knowledge representation in the form of facts and rules, and for its strength in logical reasoning and pattern matching.
- **Interpreter: SWI-Prolog** — A free, open-source Prolog environment widely used for educational and research purposes, providing built-in predicates for data processing and mathematical calculations.

## iv. Knowledgebase Structure Diagram




Animal Classification
Mammal
Tiger
Has fur
Carnivore
Elephant
Has fur
Herbivore
Dolphin
Lives in water
Carnivore
Bat
Has fur
Can fly
Nocturnal
Kangaroo
Has fur
Herbivore
Bird
Eagle
Can fly
Carnivore
Penguin (flightless)
Lays eggs
Parrot
Can fly
Herbivore
Owl
Nocturnal
Carnivore
Ostrich (flightless)
Lays eggs
Fish
Salmon
Lives in water
Shark
Lives in water
Carnivore
Tuna
Lives in water
Clownfish
Lives in water

## v. Sample Input/Output



```
% 1. Check if dolphin is a mammal
?- is_mammal(dolphin).
true.

% 2. List all animals that can really fly
?- can_really_fly(X).
X = eagle ;
X = parrot ;
X = bat ;
false.

% 3. Find animals that live in water but are not fish
?- water_animal_not_fish(X).
X = dolphin ;
false.

% 4. List mammals with fur
?- has_fur_animal(X).
X = tiger ;
X = elephant ;
X = bat ;
X = kangaroo ;
false.

% 5. Find animals in the same class as parrot
?- same_class(parrot, X).
X = eagle ;
X = penguin ;
X = owl ;
X = ostrich ;
false.

% 6. Find nocturnal flyers
?- nocturnal_flyers(X).
X = owl ;
X = bat ;
false.

% 7. Find carnivores that live in water
?- water_carnivores(X).
X = shark ;
X = dolphin ;
false.
```

## vi. Conclusion and Challenges

**Conclusion:**
This knowledge base successfully demonstrates the potential of **Prolog** as a tool for logical reasoning, classification, and knowledge representation. By storing facts and defining rules, the system can answer complex queries, such as identifying animals by their traits, finding relationships between categories, and applying conditions to exclude exceptions. The implementation highlights how Prolog’s declarative nature allows a focus on **what** the problem is rather than **how** to solve it.

A key achievement of this project is the handling of **exceptions**, such as the case where penguins are birds but cannot fly. This shows that Prolog rules can be written in a flexible way to reflect real-world nuances. Additionally, the project demonstrates how multiple facts can be combined to **infer new knowledge**—for example, determining that a dolphin is a mammal even though it lives in water.

Overall, this project illustrates the strength of Prolog in **symbolic AI**, where reasoning is based on explicitly stated knowledge and logical relationships, making it well-suited for educational purposes and small-scale expert systems.

**Challenges:**

1. **Handling Exceptions** – Designing rules that account for exceptions requires careful logic construction. For example, the use of the \= operator is necessary to ensure that specific cases like penguins not flying are correctly handled without breaking general rules.
2. **Representing Real-World Complexities** – In reality, some animals may belong to multiple categories (e.g., platypus as an egg-laying mammal). Representing such overlapping classifications increases the complexity of the rules and may require more advanced logical constructs.
3. **Knowledge Base Growth and Readability** – As the number of facts and rules increases, maintaining clarity and readability becomes challenging. Large knowledge bases can become difficult to debug and extend without proper organization, naming conventions, and documentation.
4. **Balancing Specificity and Generality** – Rules need to be general enough to cover multiple cases but specific enough to avoid incorrect inferences. Finding this balance can be tricky, especially when dealing with exceptions or incomplete information.
5. **Scalability** – While Prolog is powerful for small to medium-sized classification problems, scaling up to very large datasets may require additional optimization or integration with other systems.