សាកលវិទ្យាល័យនរតុន
NORTON UNIVERSITY

Introduction to Expert Systems

By: SEK SOCHEAT

Advisor to DCS and Lecturer

Mobile: 017 879 967

Email: socheatsek@norton-u.com

socheat.sek@gmail.com

Expert System

2025 – 2026

Y3 – DCS – NU

Table of Contents:

1. Overview of Expert Systems (ES)

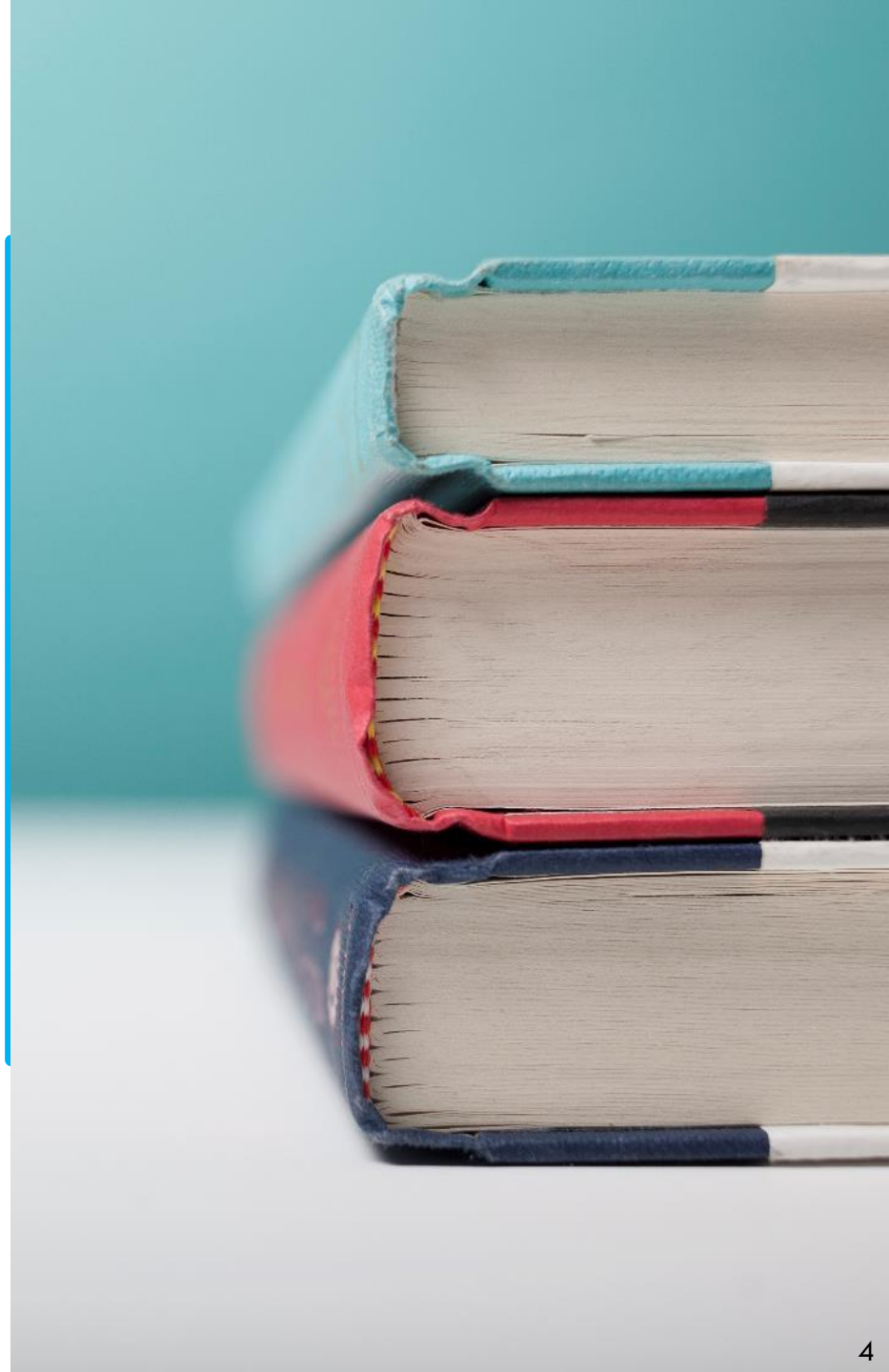
Expert Systems (ES) are AI systems that mimic expert decision-making using a knowledge base and inference engine, often applied in fields like healthcare and finance.

Theory

Definition of Expert Systems

What expert systems are: computer systems that emulate (mimic) decision-making abilities of human experts in specific domains.

Example: Highlight expert systems in healthcare (diagnosis systems), finance (loan approval), and customer support (troubleshooting).

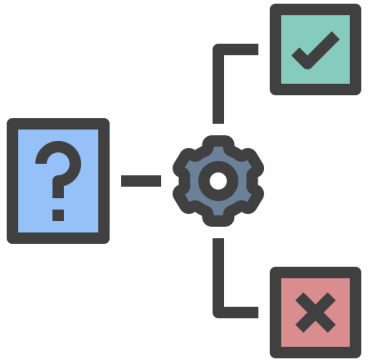


Types of Expert Systems

There are several types of Expert Systems (ES), each suited to specific tasks.

Each type helps emulate expert reasoning in diverse fields.

1. **Rule-Based Systems:** These use if-then rules to derive conclusions, commonly applied in diagnosis and troubleshooting.
2. **Case-Based Systems:** They solve new problems by comparing them to past cases, often used in legal and medical fields.
3. **Model-Based Systems:** These systems use mathematical or logical models to simulate complex processes, ideal for engineering and scientific applications.
4. **Hybrid Systems:** Combining two or more types, these systems offer flexibility, especially in environments needing both rules and case comparisons, like customer support.



1

2

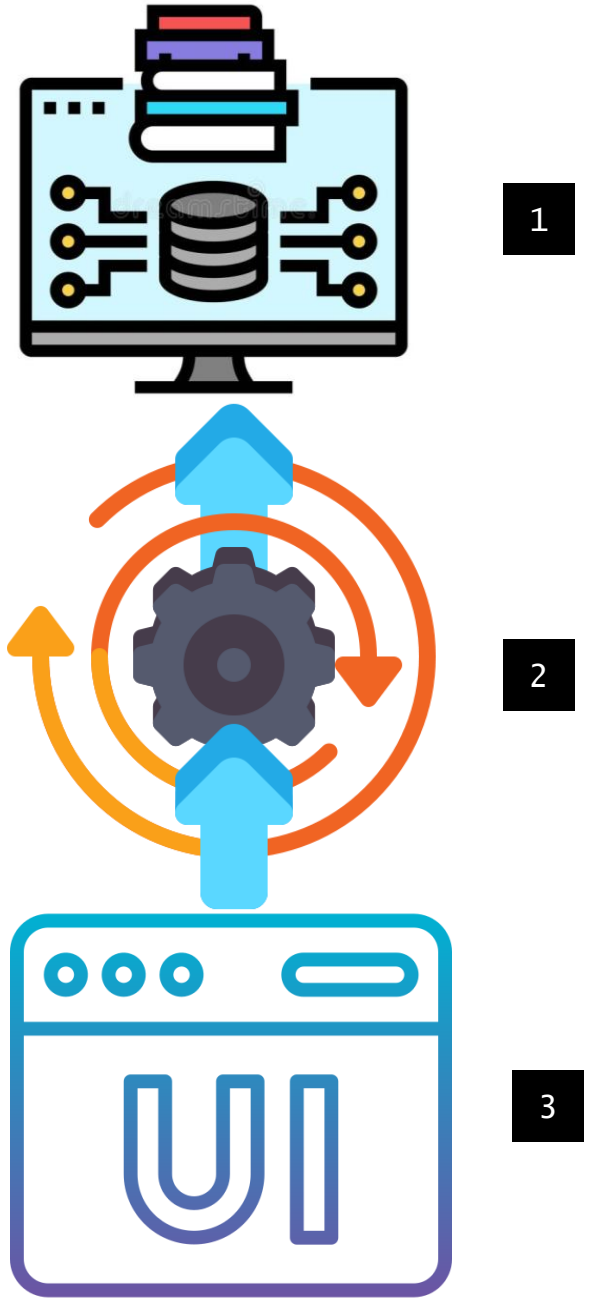
3

4

Components of an Expert System

An Expert System has three main components and these components work together to simulate expert decision-making.

1. **Knowledge Base:** Stores expert knowledge as rules and facts.
2. **Inference Engine:** Applies rules to known facts to draw conclusions.
3. **User Interface:** Allows users to interact with the system, providing inputs and receiving insights.



1

2

3

Real-World Applications

1. Healthcare

Diagnose diseases and recommend treatments (e.g., MYCIN for infections).

2. Finance

Assist in loan approvals, fraud detection, and investment advising.

3. Customer Support

- Troubleshoot issues and provide automated support for products.

4. Manufacturing

Monitor systems, predict equipment failures, and optimize processes.

5. Agriculture

Advise on crop management, pest control, and soil health.

6. Education

Provide personalized tutoring and assessment, offering guidance and feedback tailored to each student's needs.

Practical Setup

Installing Python and VS Code

- **Guided Setup:** Walk through the installation of Python and VS Code on student devices.

- **Setup Verification:** Verify installations by running a simple print("Hello, Expert Systems!") in the VS Code terminal.

Practical Setup

Introduction to Python Basics

- **Variables and Data Types:** Show how to declare variables (e.g., integers, strings, lists).
- **Control Structures:** Cover if statements and loops (for, while).
- **Interactive Exercise:** Create a simple program that assigns values to variables and performs calculations (e.g., basic math operations).

2. Knowledge Representation in Expert Systems

Knowledge Representation in Expert Systems organizes information as rules, facts, and relationships to enable reasoning, often using formats like if-then rules, semantic networks, and frames to mimic expert decision-making.

Theory

Definition of Knowledge Representation

Knowledge Representation in Expert Systems is how information is structured to simulate expert reasoning.

This involves organizing **rules**, **facts**, and **relationships** to make decisions or provide solutions.

Theory

Key Forms of Knowledge Representation

1. **Rules:** The most common format, often using if-then statements to make logical conclusions. Example: "If a patient has a high fever, then suggest a flu test."
2. **Frames:** Data structures that represent stereotyped situations, such as objects and their attributes. Example: A “Car” frame with attributes like “make,” “model,” and “year.”
3. **Semantic Networks:** Graph-based representations that show relationships between concepts, helpful for visualizing hierarchies or dependencies. Example: Representing symptoms and their associated diseases.
4. **Logic and Probability:** Used in more advanced systems to handle uncertainty, such as Bayesian networks that calculate the likelihood of various outcomes.

Theory

Role of the Inference Engine

The inference engine processes this knowledge, applying rules to known facts to derive conclusions. It helps Expert Systems simulate human-like reasoning by navigating through organized knowledge to reach solutions.

This structured approach enables Expert Systems to solve complex problems by logically interpreting stored knowledge, enhancing their effectiveness in decision-making.

3. Practical Exercise

Implementing Rule-Based Decision-Making with **if-else** Constructs

Objective: Create a basic expert system in Python to diagnose minor health issues based on symptoms.

Implementing Rule-Based Decision-Making with if-else Constructs

Step-by-Step Guide

1. Define Symptoms as Variables

Defining **symptoms** as boolean variables (**True** or **False**) to indicate the presence or absence of each symptom.

has_fever = True

has_cough = False

has_body_ache = True

Implementing Rule-Based Decision-Making with if-else Constructs

Step-by-Step Guide

2. Implement Basic if-else Logic

Write a simple series of **if-else** statements to check for symptom combinations and diagnose a health condition.

```
if has_fever and has_body_ache:
    print("Diagnosis: You might have the flu.")
elif has_cough:
    print("Diagnosis: You might have a common cold.")
else:
    print("Diagnosis: Symptoms not specific enough.")
```

Implementing Rule-Based Decision-Making with if-else Constructs

Step-by-Step Guide

Enhance the System

Add More Conditions: Guide students to add new conditions to expand the system.

For instance, we can add rules for sinus infection and allergies:

```
# Ask the user about each symptom and convert the input to a boolean
has_fever = input("Do you have a fever? (yes/no): ").strip().lower() == "yes"
has_body_ache = input("Do you have body aches? (yes/no): ").strip().lower() == "yes"
has_headache = input("Do you have a headache? (yes/no): ").strip().lower() == "yes"
has_stuffy_nose = input("Do you have a stuffy nose? (yes/no): ").strip().lower() == "yes"

# Diagnose based on symptoms
if has_fever and has_body_ache:
    print("Diagnosis: You might have the flu.")
elif has_headache and has_stuffy_nose:
    print("Diagnosis: You might have a sinus infection.")
elif has_stuffy_nose and not has_fever:
    print("Diagnosis: You might have allergies.")
elif has_headache:
    print("Diagnosis: You might have a common cold.")
else:
    print("Diagnosis: Symptoms not specific enough.")
```

4. Homework: Additional Exercises on Rule-Based Decision-Making

Expanding the Diagnosis System

Task: Add new symptoms and diagnoses to the simple expert system from class. Try to include:

- **New Symptoms:** like sore throat, runny nose, headache.
- **New Diagnoses:** such as “strep throat” or “common allergies.”

Instructions:

- Update your system to include these symptoms.
- Create at least two new rules to handle these added symptoms.

Goal: Understand how adding more conditions enhances the system’s ability to make accurate diagnoses.

Thank You

SEK SocheaT

socheatsek@norton-u.com