Introduction to System Design & Architecture for Intelligent Systems

By: SEK SOCHEAT

Advisor to DCS and Lecturer

Mobile: 017 879 967

Email: socheat.sek@gmail.com

Intelligence System Development

2024 – 2025

Y4E1 – DCS – NU

Table of Contents

1. Problem Formulation

Identifying Real-World Problems Solvable by Intelligent Systems, focusing on how to recognize opportunities where AI and intelligent systems can be applied to make a significant impact.

What is Problem Formulation?

Problem formulation defines the initial state, goal state, and actions to solve a problem in AI.

Exploring Different Problem Domains

Understanding the different industries and fields where intelligent systems can be applied is essential. Here's how to approach it:

1. Healthcare:

IS can analyze medical images, predict diagnoses, recommend treatments, and manage data, like detecting cancer early or optimizing hospital resources.

2. Finance:

AI aids in fraud detection, trading, credit scoring, and risk management, like spotting fraud, automating support, or predicting stock trends.

3. Logistics and Supply Chain:

AI optimizes routing, inventory, demand forecasting, and sorting, like predicting demand, automating delivery, or optimizing warehouses.

4. Retail and E-Commerce:

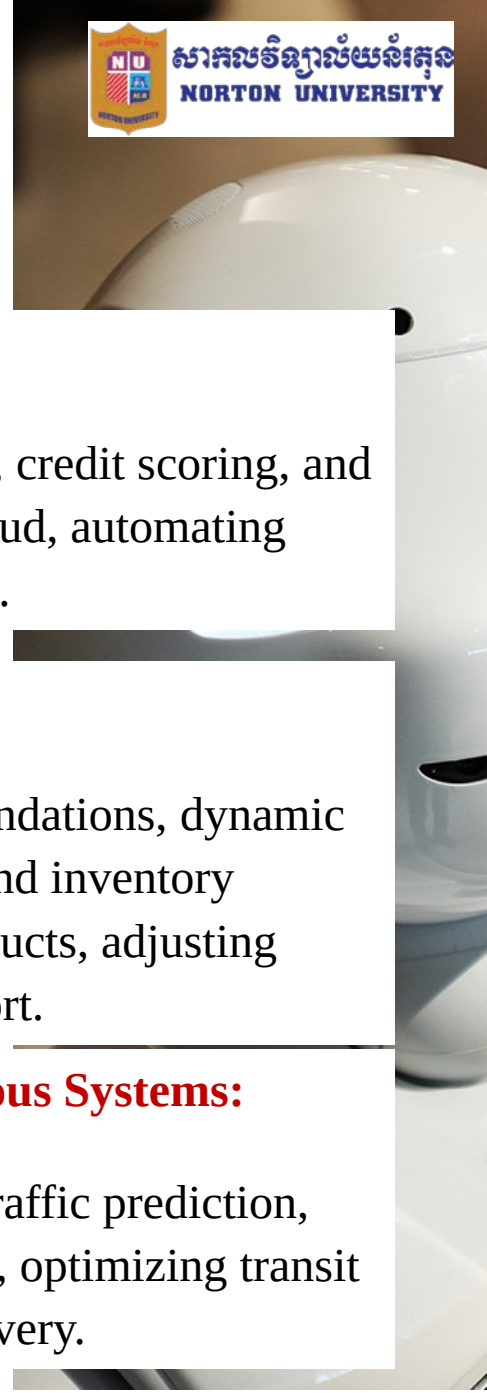
AI enables personalized recommendations, dynamic pricing, customer segmentation, and inventory management, like suggesting products, adjusting prices, or using chatbots for support.

5. Manufacturing:

AI supports predictive maintenance, quality control, and process optimization, like forecasting machine failures, inspecting defects, or optimizing schedules.

6. Transportation and Autonomous Systems:

AI powers autonomous vehicles, traffic prediction, and logistics, like self-driving cars, optimizing transit schedules, or using drones for delivery.



Types of Problems Solvable by IS

Understanding the types of problems that intelligent systems can solve helps in identifying suitable use cases:

1. Classification:

Classify data into set categories, used for spam detection, medical diagnosis, or sentiment analysis.

2. Regression:

Estimate continuous values using past data, like predicting stock prices, machinery lifespan, or sales.

3. Clustering:

Cluster similar data points without labels, used for customer segmentation, anomaly detection, or image compression.

4. Optimization:

Optimize solutions within constraints, like route planning, resource allocation, or manufacturing efficiency.

5. Anomaly Detection:

Detect outliers in data for uses like fraud detection, equipment failure, or security breaches.

6. Natural Language Understanding (NLU):

Analyze and understand language for tasks like sentiment analysis, text summarization, or chatbots.

7. Computer Vision:

Process visual data to recognize objects, faces, or enable autonomous driving.

Criteria for Choosing a Problem

Selecting the right problem is crucial to the success of an AI project.
Consider these criteria:

- **Feasibility:** Ensure AI can solve the problem with current techniques and data.
- **Data Availability:** Sufficient quality data is crucial; labeled data is needed for supervised learning.
- **Impact Potential:** Focus on problems that significantly improve efficiency or reduce costs.
- **Scalability:** Check if the solution works across different settings and datasets.
- **Ethical and Regulatory Constraints:** Ensure compliance with ethical guidelines and legal regulations.

Case Studies and Use Cases

Studying real-world examples helps understand how intelligent systems can be applied effectively:

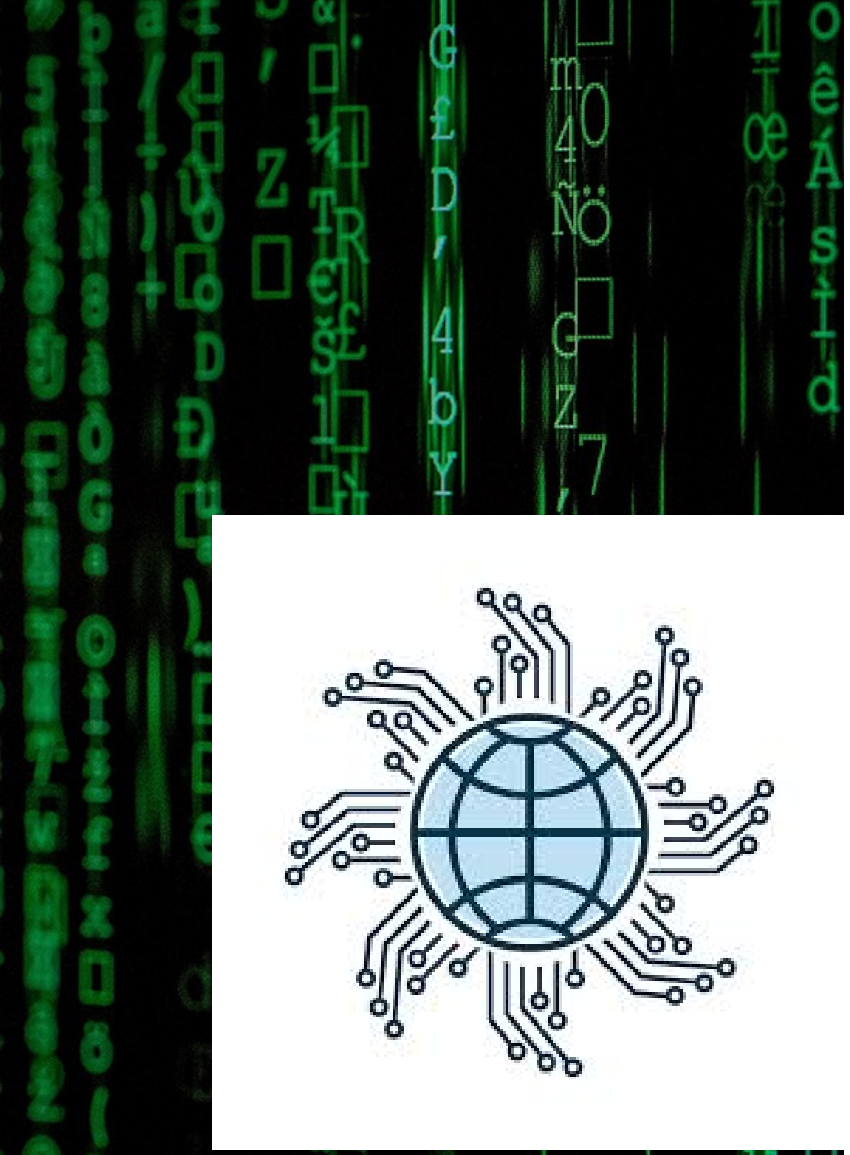
1. **Healthcare:** Use AI for early disease prediction and accurate medical image analysis.
2. **Finance:** Detect fraud through pattern recognition and automate trading decisions.
3. **Logistics:** Optimize delivery routes and forecast product demand with AI.
4. **Retail:** Enhance personalized product recommendations and implement dynamic pricing.
5. **Transportation:** Enable autonomous driving and predict traffic for efficient routing.

Summary

Identifying AI-solvable problems involves understanding industries, feasible challenges, and impactful solutions.

Real-world case studies show how AI improves efficiency, reduces costs, and drives innovation across sectors, ensuring AI's potential is maximized.

2. System Design



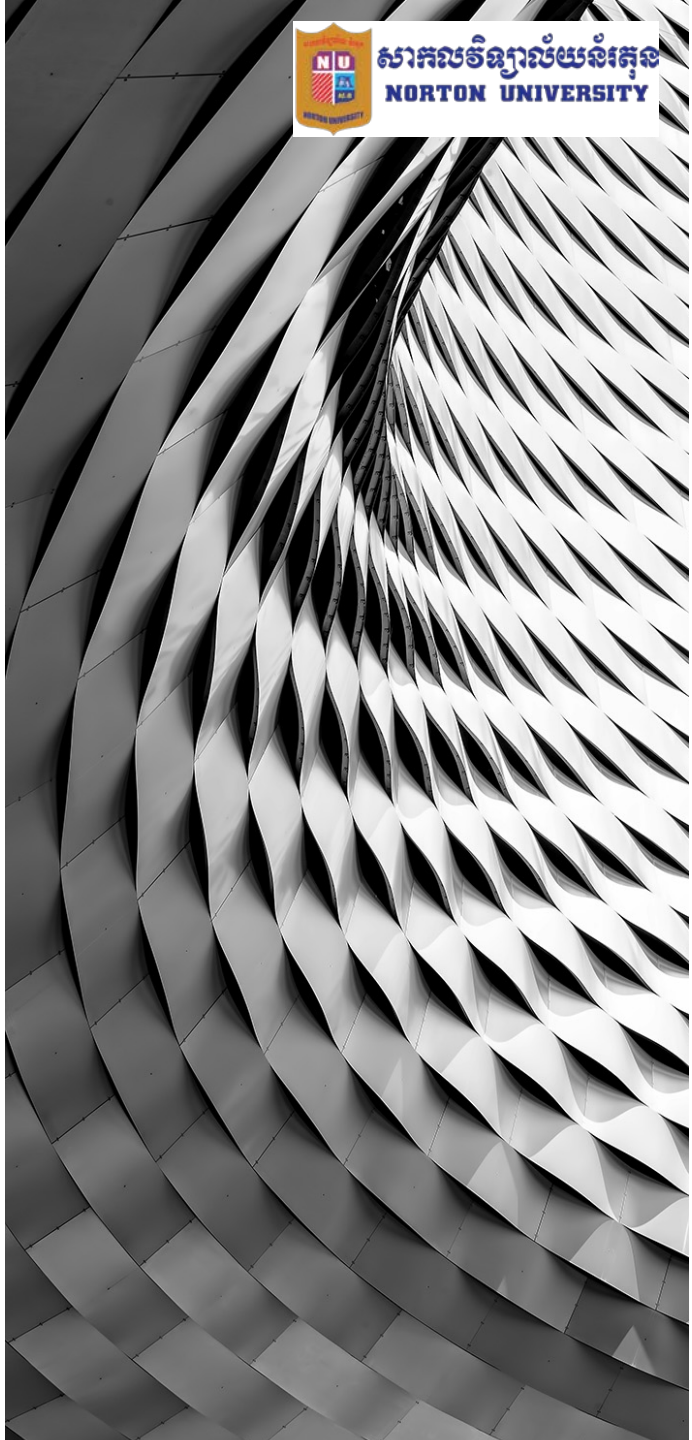
What is System Design?

System design involves creating a system's architecture, modular components, scalability, maintainability, and performance. It includes data flow management, deployment planning, ethical considerations, and uses UML for visual representation.

Basic Concepts of System Design

System design for intelligent systems involves planning architecture and component interactions. Key concepts include:

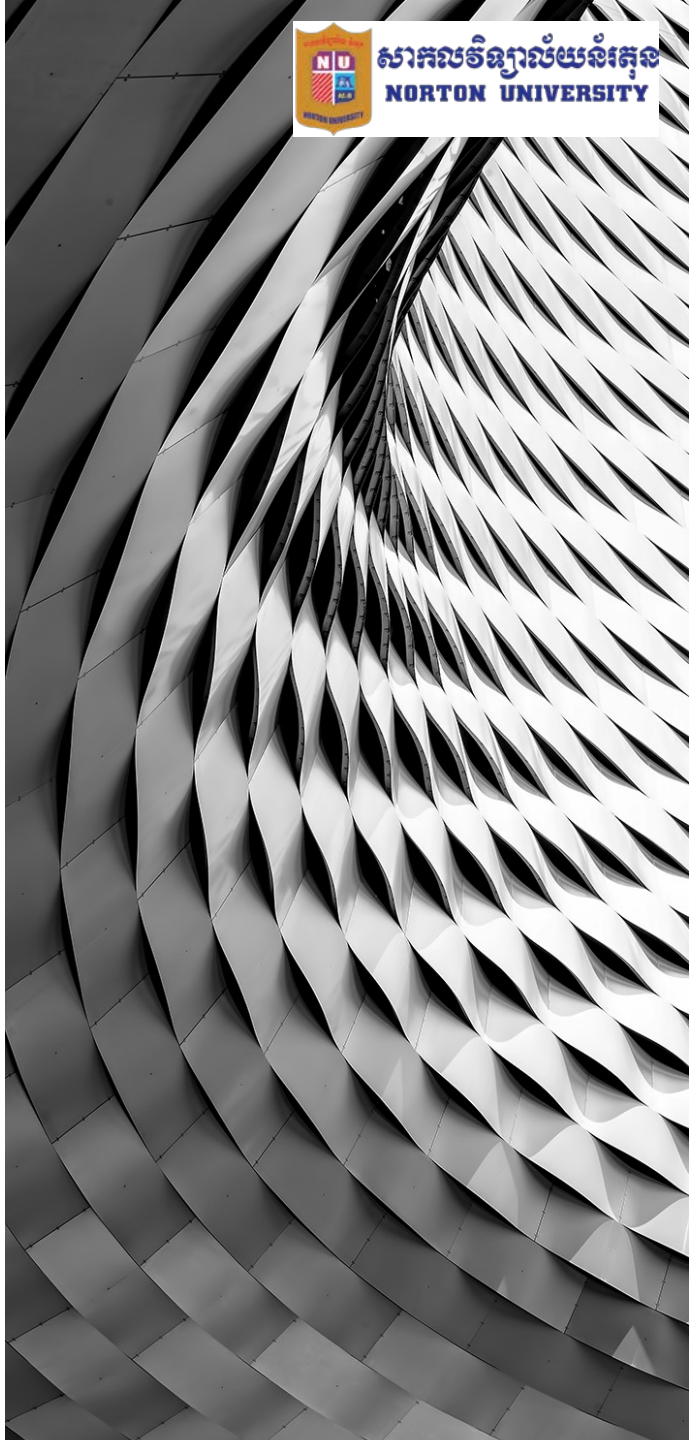
- **Modularity:** Design with independent modules for easier maintenance (e.g., separate data processing).
- **Scalability:** Ensure the system handles growth via horizontal or vertical scaling (e.g., add servers for more users).
- **Maintainability:** Make the system easy to update and test with clear documentation (e.g., modular data processing).
- **Performance:** Optimize for fast, efficient operation (e.g., real-time recommendations).



Unified Modeling Language (UML)

UML visually represents system architecture, components, and interactions for better design understanding.

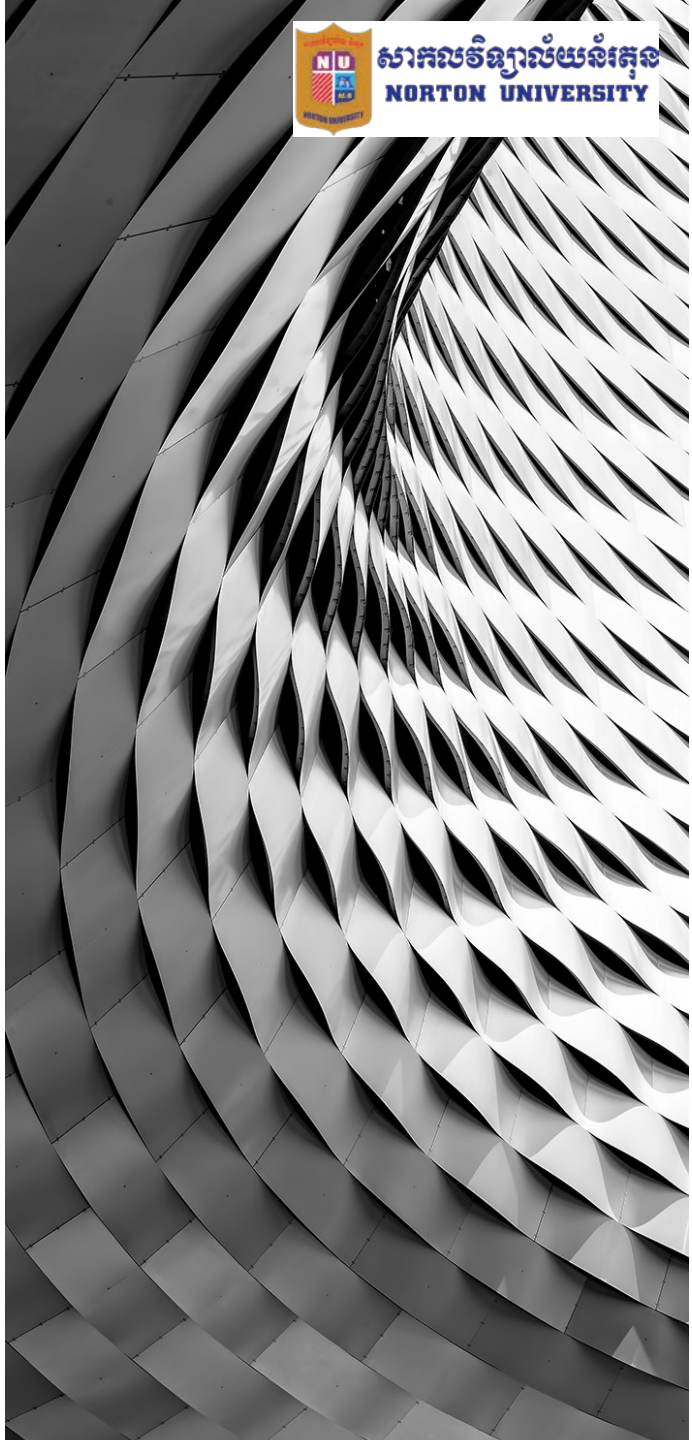
- **Use Case Diagrams:** Show system functions and user interactions (e.g., requesting recommendations).
- **Class Diagrams:** Outline system structure, including classes and relationships (e.g., text data, model).
- **Sequence Diagrams:** Display interactions over time (e.g., data ingestion to alert generation).



Principles of Design Thinking

Design thinking is an iterative, user-focused approach to problem-solving with key principles:

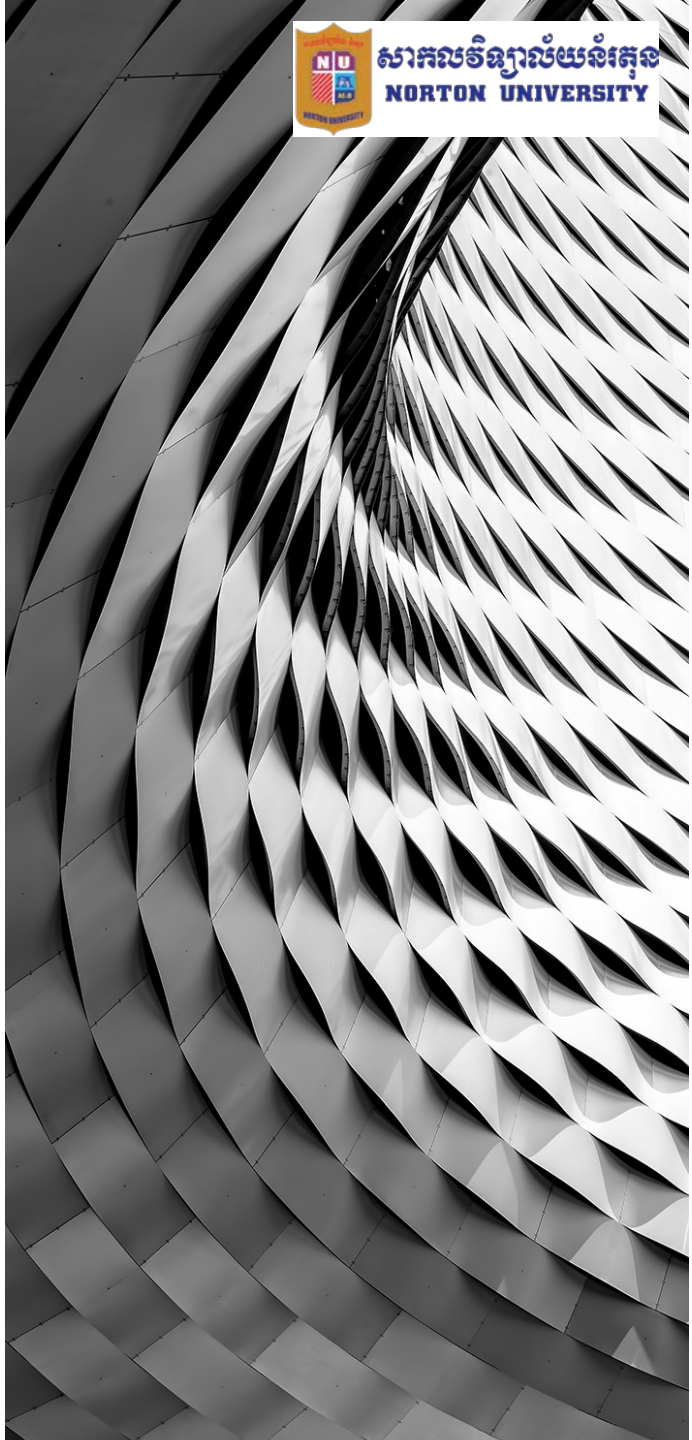
- **User-Centered Design:** Understand user needs for intuitive, functional systems (e.g., healthcare AI).
- **Iterative Development:** Continuously prototype, test, and refine based on feedback (e.g., improving an NLP chatbot).
- **Collaboration:** Engage cross-functional teams for diverse perspectives (e.g., developing an intelligent transport system).



Architecture Patterns for Intelligent Systems

Architecture patterns for intelligent systems include:

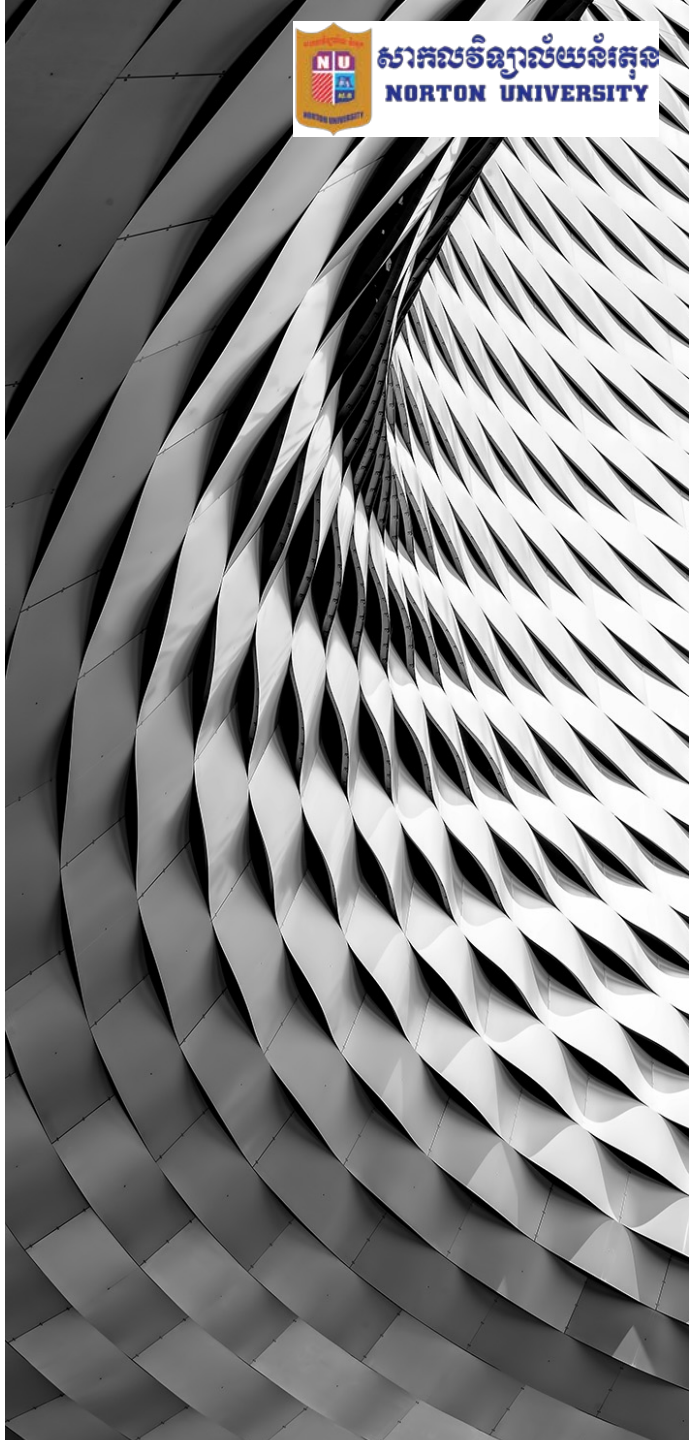
- **Client-Server:** Clients send requests to a server for processing (e.g., AI search engine).
- **Microservices:** Independent services handle specific tasks (e.g., e-commerce platform).
- **Layered:** Separate layers for UI, business logic, and data (e.g., predictive maintenance).
- **Event-Driven:** Components respond to real-time events (e.g., fraud detection).



Data Flow and Integration Considerations

Key data flow and integration considerations include:

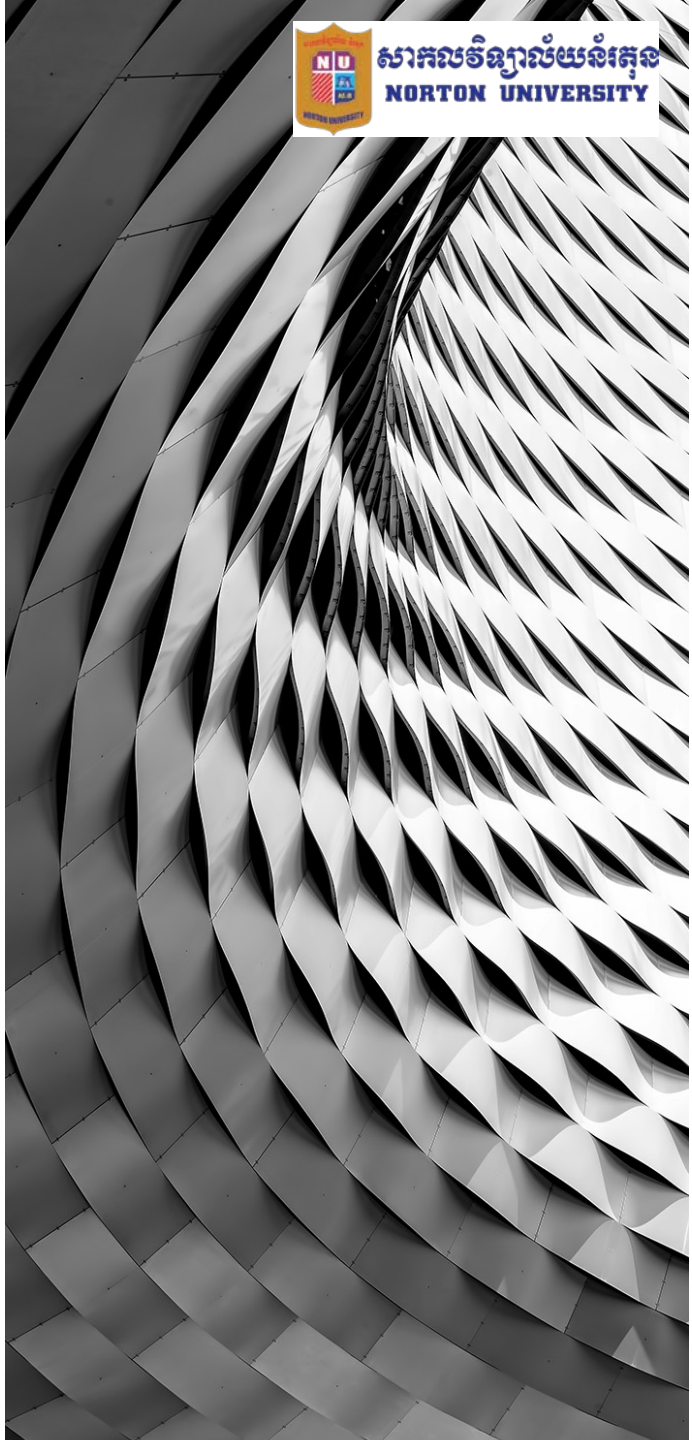
- **Data Ingestion:** Plan data acquisition methods (e.g., batch or real-time).
- **Data Preprocessing:** Clean and transform data for training (e.g., text tokenization).
- **Data Storage:** Select storage solutions based on needs (e.g., data lakes for unstructured data).
- **Model Integration:** Ensure models are easily integrated and scalable for real-time use (e.g., recommendation engines).



Deployment and Infrastructure

Key deployment and infrastructure considerations:

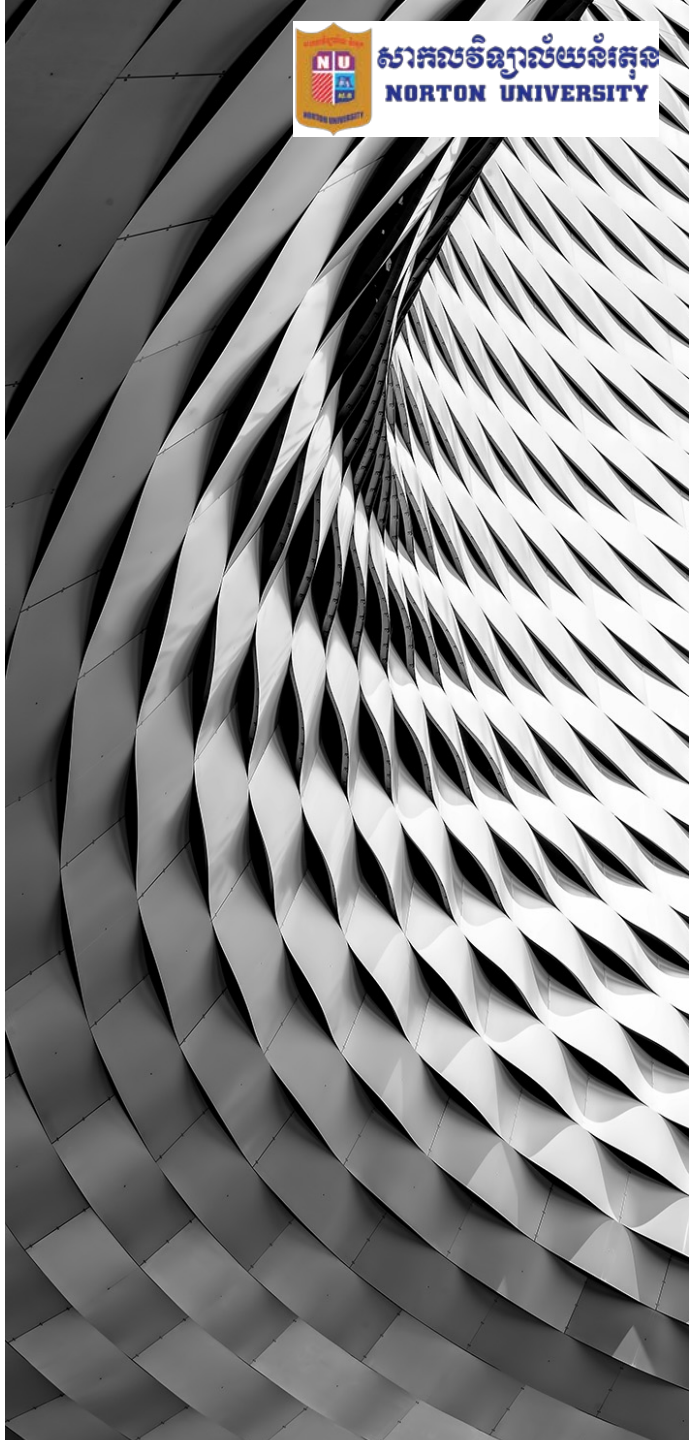
- **Cloud vs. On-Premises:** Choose based on scalability, cost, and security needs (e.g., cloud for large-scale AI).
- **Containerization & Orchestration:** Use Docker and Kubernetes for efficient scaling (e.g., microservices).
- **Continuous Integration (CI) and Continuous Deployment (CD):** Automate testing and deployment to keep the system updated (e.g., ML models).
- **Monitoring & Maintenance:** Use tools to track performance and detect issues (e.g., model accuracy).



Ethical and Regulatory Considerations

Key ethical and regulatory considerations:

- **Fairness & Bias Mitigation:** Prevent biased decisions (e.g., fair loan approvals).
- **Data Privacy & Security:** Protect data and comply with regulations (e.g., General Data Protection Regulation or GDPR).
- **Explain-ability:** Make outputs interpretable (e.g., explain credit decisions).



Practical Activity: Designing an Intelligent Customer Support Chatbot

Problem Formulation:

Aim to automate 60% of queries, using natural language understanding (NLU) for intent recognition and dialogue management. Feasibility confirmed with existing chat data and NLU libraries.

Ethical Considerations:

- Ensure data privacy and secure storage.
- Regularly review for bias in chatbot responses.

System Design:

Modular Architecture:

- Data Ingestion, NLU, Dialogue Management, and Escalation modules.

UML Diagram:

- **Actors:** Customer, Chatbot, Human Agent.
- **Use Cases:** "Ask a Question," "Troubleshoot," "Request Support."

Data Flow:

- User input → NLU → Dialogue → Response → (optional) Escalation.

Deployment:

- Use cloud platform and Docker for scalability.

Solution: Designing an Intelligent Customer Support Chatbot


```python
import spacy
import re
import json

# Load spaCy's English model
nlp = spacy.load('en_core_web_sm')

# Load patterns and responses from a JSON file
def load_data():
    """
    Load patterns and responses from the 'responses.json' file in the project directory.
    """
    try:
        with open('responses.json', 'r') as file:
            data = json.load(file)
            patterns = {key: re.compile(value, re.IGNORECASE) for key, value in data['patterns'].items()}
            responses = data['responses']
            return patterns, responses
    except FileNotFoundError:
        print("Error: 'responses.json' file not found.")
        return {}, {}

INTENT_PATTERNS, RESPONSES = load_data()

def recognize_intent(user_input):
    """
    Recognize intent based on regex patterns.
    """
    for intent, pattern in INTENT_PATTERNS.items():
        if pattern.search(user_input):
            return intent
    return "fallback"

```

Solution: Designing an Intelligent Customer Support Chatbot


```
34 def handle_user_input(user_input):
35     """
36     Handle the user input by recognizing intent and returning the appropriate response.
37     """
38     intent = recognize_intent(user_input)
39     return RESPONSES.get(intent, RESPONSES.get("fallback", "I didn't understand that."))
40 
41 def main():
42     """
43     Main function to run the chatbot console application.
44     """
45     print("Welcome to the Chatbot Console! Type 'exit' to quit.")
46 
47     while True:
48         user_input = input("You: ")
49         if user_input.lower() == 'exit':
50             print("Goodbye!")
51             break
52 
53         response = handle_user_input(user_input)
54         print(f"Bot: {response}")
55 
56 if __name__ == '__main__':
57     main()
58 
```

Homework:

Design a Predictive Maintenance System

1. Problem Formulation:

- **Objectives:** Reduce machinery downtime by 30% through proactive maintenance scheduling.
- **Problem Type:** Time-series prediction for equipment failure and anomaly detection.
- **Data Needed:** Sensor readings (e.g., temperature, vibration), maintenance logs, equipment usage data.

Homework: 2. System Design:

Modular Architecture:

- ***Data Ingestion Module:*** Collect and preprocess sensor and usage data.
- ***Feature Extraction Module:*** Extract relevant features like temperature trends, vibration patterns.
- ***Model Training Module:*** Train predictive models using historical data.
- ***Prediction Module:*** Provide real-time predictions on machinery health.

UML Diagram:

- ***Main Components:*** Data Ingestion, Feature Extraction, Model Training, Prediction.
- ***Interactions:*** Show data flowing from sensors to prediction output.

Data Flow:

- Sensor readings → Data Ingestion → Feature Extraction → Model Training → Prediction → Maintenance Alerts.

Deployment Plan:

- ***Cloud-Based:*** Suitable for scalability and remote monitoring, use Docker for containerization.

Homework: 3. Ethical and Regulatory Considerations

- ***Data Privacy:*** Ensure data collected from machines is anonymized and securely stored.
- ***Worker Safety:*** Use the system to enhance safety protocols by predicting hazardous equipment conditions.
- ***Regulatory Compliance:*** Adhere to industry standards for data security and operational safety (e.g., ISO standards).

Deliverables:

- A document with the problem formulation, system architecture, UML diagram, and data flow plan.
- A brief discussion on ethical considerations, including data privacy and worker safety measures.

សាកលវិទ្យាល័យនរតុន
NORTON UNIVERSITY

Thank you