1. Groups must consist of 3 members.
2. We will have a presentation on your proposed idea on 27th January 2025.
3. Each group will get a maximum of 10 minutes for the presentation.
4. Every team member must contribute to the presentation in some way.
5. The project must consist of an Artificial Intelligence feature.
    a. The feature doesn't have to be within the course outline.
    b. But the feature should be implemented by yourselves.
6. You have to finish the project before your final exams.
7. Some example projects, but are not limited to:

1. Chatbot

- **Description**: Create a chatbot that can answer basic questions or simulate a conversation.
- **Libraries**: `NLTK`, `ChatterBot`, or `Transformers`
- **Steps**:
    - Use a rule-based approach or train a chatbot model.
    - Integrate a pre-trained model like GPT for better responses.
    - Add a graphical user interface (GUI) with `Tkinter` or a web app using `Flask`.

---

2. Image Classifier

- **Description**: Build a model to classify images into categories (e.g., cats vs. dogs).
- **Libraries**: `TensorFlow` or `PyTorch`, `OpenCV`
- **Steps**:
    - Use pre-trained models like MobileNet or ResNet.
    - Train on a small dataset (e.g., CIFAR-10).
    - Create a simple GUI to upload and classify images.

---

3. Handwriting Digit Recognition

- **Description**: Recognize handwritten digits (0-9).
- **Libraries**: `Scikit-learn`, `Keras`, or `TensorFlow`
- **Steps**:
    - Use the MNIST dataset for training.
    - Train a Convolutional Neural Network (CNN).
    - Create a tool to draw digits and recognize them in real-time.

---

4. Sentiment Analysis

- **Description**: Analyze the sentiment of text data (positive, negative, or neutral).
- **Libraries**: `NLTK`, `TextBlob`, or `HuggingFace Transformers`
- **Steps**:
    - Preprocess text data (tokenization, stopword removal).
    - Train a simple model or use a pre-trained one like BERT.
    - Apply it to reviews, tweets, or custom datasets.

---

5. Face Recognition System

- **Description**: Build a face recognition system to detect and identify faces.
- **Libraries**: `OpenCV`, `face_recognition`, `dlib`
- **Steps**:
    - Use Haar Cascades or deep learning models for face detection.
    - Integrate a pre-trained face recognition library.
    - Add a feature to store and recognize faces in a database.

---

6. AI-Powered Calculator

- **Description**: A calculator that interprets natural language inputs (e.g., "add 5 and 3").
- **Libraries**: `NumPy`, `NLTK`
- **Steps**:
    - Parse natural language using NLTK or spaCy.
    - Convert input to mathematical expressions and compute results.
    - Enhance it with voice input using `SpeechRecognition`.

---

7. Spam Email Classifier

- **Description**: Classify emails as spam or not spam.
- **Libraries**: `Scikit-learn`, `NLTK`
- **Steps**:
    - Use a labeled dataset (e.g., the Enron email dataset).
    - Preprocess text (e.g., TF-IDF vectorization).
    - Train a Naive Bayes classifier or a simple neural network.

---

8. AI Game Bot

- **Description**: Create an AI bot that can play games like Tic-Tac-Toe or Snake.
- **Libraries**: `PyGame`, `Keras`
- **Steps**:
  - Use reinforcement learning for the bot's decision-making.
  - Implement a GUI for the game using PyGame.
  - Train the AI to learn and improve its gameplay.

---

9. Voice Assistant

- **Description**: Develop a basic voice assistant for tasks like searching the web or setting reminders.
- **Libraries**: `SpeechRecognition`, `Pyttsx3`, `Google Assistant SDK`
- **Steps**:
  - Use `SpeechRecognition` to process voice input.
  - Integrate APIs for actions (e.g., Google Search API).
  - Convert the assistant's responses to speech with `Pyttsx3`.

---

10. AI-Powered Personal Finance Tracker

- **Description**: Create an app to track expenses and provide financial insights.
- **Libraries**: `Pandas`, `Matplotlib`, `Scikit-learn`
- **Steps**:
  - Build a model to classify transactions into categories.
  - Visualize spending patterns using Matplotlib.
  - Add AI predictions for future expenses based on historical data.

---

11. Recommendation System

- **Description**: Suggest movies, books, or products based on user preferences.
- **Libraries**: `Scikit-learn`, `Surprise`, or `TensorFlow`
- **Steps**:
  - Use collaborative filtering or content-based filtering.
  - Train a recommendation model with a dataset like MovieLens.
  - Deploy it as a web application using `Flask`.

---

12. Real-Time Object Detection

- **Description**: Detect objects in real-time using a webcam.
- **Libraries**: `OpenCV`, `YOLO`, `TensorFlow`
- **Steps**:
  - Use pre-trained YOLO or SSD models.
  - Integrate with a webcam feed using OpenCV.
  - Add functionality to identify specific objects (e.g., animals, vehicles).

---

13. Weather Forecasting App

- **Description**: Create a weather forecasting app with AI predictions.
- **Libraries**: `OpenWeatherMap API`, `Scikit-learn`
- **Steps**:
  - Train a time-series forecasting model on historical weather data.
  - Use OpenWeatherMap API for real-time updates.
  - Build a web or mobile-friendly interface.

---

14. Optical Character Recognition (OCR)

- **Description**: Extract text from images or scanned documents.
- **Libraries**: `Tesseract`, `OpenCV`, `Pytesseract`
- **Steps**:
  - Use Pytesseract for text extraction.
  - Enhance the model with pre-processing using OpenCV.
  - Integrate with a PDF reader for batch processing.