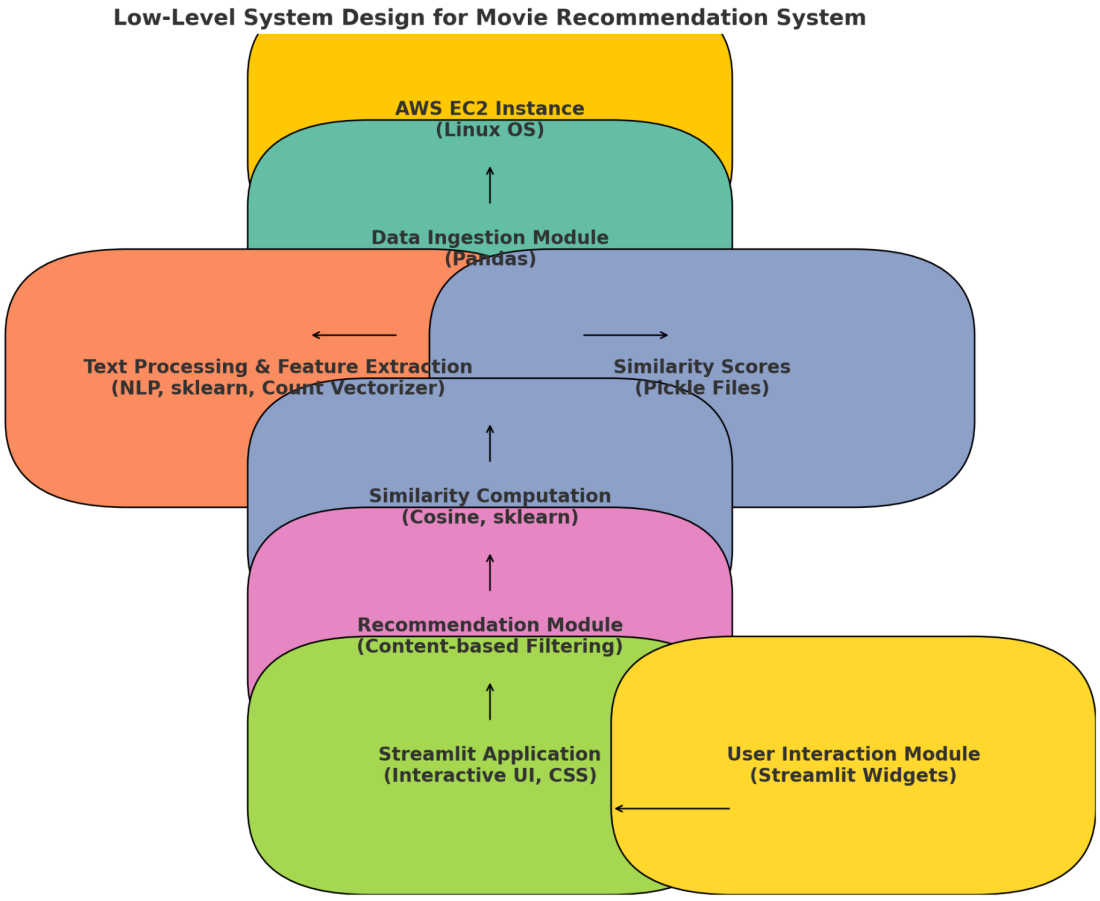

Diagram Explanation

- **AWS EC2 Instance**: The central node hosting the application, running on a Linux OS.
- **Data Ingestion Module**: Reads and loads movie metadata from CSV files into Pandas DataFrames.
- **Text Processing & Feature Extraction**: Applies NLP techniques for text preprocessing and extracts features using Countvectorizer.
- **Similarity Scores**: Stores precomputed cosine similarity matrices in Pickle files for efficient retrieval.
- **Similarity Computation**: Computes cosine similarity matrices using the Count Vectorizer.
- **Recommendation Module**: Uses content-based filtering to generate recommendations based on similarity scores.
- **Streamlit Application**: Hosts the interactive user interface, allowing users to explore recommendations.

- **User Interaction Module**: Captures and processes user inputs using Streamlit widgets.

Low-Level System Design for Movie Recommendation System

1. Data Ingestion and Storage

- **Movie Metadata (CSV Files)**:
  - **Structure**: Includes columns like movie_id, title, description, genres, cast, crew, etc.
  - **Storage**: Stored in a directory accessible by the EC2 instance.
- **Similarity Scores (Pickle Files)**:
  - **Structure**: Contains precomputed cosine similarity matrices.
  - **Storage**: Stored in a directory accessible by the EC2 instance.

2. Data Processing

- **Data Ingestion Module**:
  - **Function**: Reads CSV files and loads data into Pandas DataFrames.
  - **Components**: `pandas.read_csv()`.
- **Text Processing and Feature Extraction**:
  - **Function**: Cleans and preprocesses movie descriptions and reviews using NLP techniques.
  - **Components**: Tokenization, Lemmatization, Count Vectorization.
  - **Libraries**: `nltk`, `sklearn`.
- **Similarity Computation**:
  - **Function**: Computes cosine similarity between movies.
  - **Components**: Countvectorizer matrix, cosine similarity function.
  - **Libraries**: `sklearn`.

3. Recommendation Engine

- **Recommendation Module**:
  - **Function**: Generates recommendations based on user preferences.
  - **Components**: Fetches precomputed similarity scores, filters based on genres, cast, crew.
  - **Algorithm**: Content-based filtering using cosine similarity.

4. User Interface

- **Streamlit Application**:
  - **Components**:

        - **Main Dashboard**: Displays movie recommendations.
        - **Filters**: Allows users to filter recommendations by genre, cast, crew.
    - **Customization**: Custom CSS for styling.
- **Interaction Handling**:
    - **Components**: Streamlit widgets (sliders, dropdowns).
    - **Function**: Captures user inputs for personalised recommendations.

5. Deployment and Version Control

- **Deployment**:
    - **Platform**: AWS EC2 instance.
    - **Environment Setup**: Uses virtual environments (e.g., `venv`, `virtualenv`) and dependency management (`requirements.txt`).
- **Version Control**:
    - **Repository**: GitHub.
    - **Function**: Tracks changes, facilitates collaboration