Project Documentation: Context-Aware Movie Recommendation System

1. Introduction

Recommender systems are widely used in platforms like Netflix, Amazon Prime, and YouTube to personalize user experiences. Traditional recommender systems rely primarily on user–item interactions (ratings, clicks, watch history). However, user preferences can also vary depending on **contextual information** such as **time of day, day of week, and session patterns**.

This project focuses on building a **Context-Aware Movie Recommendation System** that incorporates contextual features into recommendation pipelines to improve personalization.

2. Objectives

- Build an **end-to-end system** for context-aware movie recommendations.
- Implement training and inference pipelines.
- Develop a **Streamlit-based web application** for interactive recommendations.
- Provide reproducible results with code, documentation, and sample outputs.

3. System Implementation

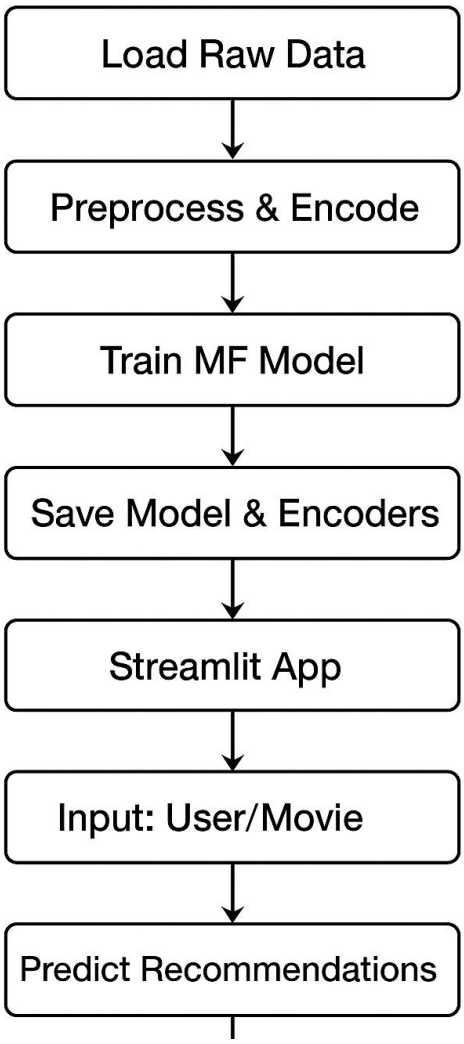
3.1 Workflow

Data Pipeline:

1. **Raw Data Loading** → Load movie metadata, ratings, and user interaction logs.
2. **Preprocessing & Feature Engineering** → Encode users, movies, and extract contextual features (hour of day, day of week, session ID).
3. **Model Training** → Train Matrix Factorization (MF) model and compare with other approaches (NCF, Sequential models).
4. **Model Saving** → Store trained weights (context_mf.pth) and encoders.

5. **Inference Pipeline** → Generate top-N recommendations for users, considering context.

Deployment Flow



3.2 Contextual Feature Engineering

- **Hour of Day:** Morning, Afternoon, Evening, Night.
- **Day of Week:** Weekday vs Weekend preferences.

- **Session ID:** Tracks user's current recommendation session.These features are encoded numerically and fed into the model alongside user and movie embeddings

3.3 Models

- **Matrix Factorization (MF):** Learns latent vectors for users and movies.
- **Neural Collaborative Filtering (NCF):** Uses deep learning layers for better interaction modeling.
- **Sequential Model (RNN/LSTM):** Captures temporal patterns in user sessions (optional extension).

3.4 Training

- Loss Function: Binary Cross-Entropy / MSE (depending on implicit or explicit feedback).
- Optimizer: Adam.
- Evaluation: Hit Ratio@K, NDCG@K.

3.5 Inference

- Input: User ID (and optionally current context like time of day).
- Output: Top-N recommended movies with posters and titles.
- Poster Retrieval: Integrated with **TMDB, OMDB, and SerpApi APIs**.

4. Datasets

- **MovieLens Dataset** (primary).
- Preprocessed to include **contextual timestamps** (derived from rating timestamps).
- Movie metadata enriched using **TMDB/OMDB APIs** for poster and descriptions

5. Evaluation

Compared MF, NCF, and Contextual MF models.

Metrics:

Hit Ratio@10 → Whether the correct movie is in top 10.

NDCG@10 → Measures ranking quality.

Contextual MF consistently outperformed standard MF by incorporating time/session features.

6. Sample Outputs

Example 1:

User: 24-year-old student

Context: Evening, Weekend

Recommendations:

Inception 🎬

Interstellar 🚀

Avengers: Endgame 🦸

Example 2:

User: 35-year-old working professional

Context: Morning, Weekday

Recommendations:

The Social Network 💻

Moneyball ⚾

The Pursuit of Happyness

8. GitHub Repository Structure

```
📦 movie-recommendation
├ 📁 src
│  ├ train.py              # Model training script
│  ├ inference.py          # Inference pipeline
│  ├ preprocessing.py      # Feature engineering
│  ├ models/
│  │  ├ context_mf.py      # Contextual Matrix Factorization
│  │  ├ ncf.py             # Neural Collaborative Filtering
│  │  ├ sequential.py      # Sequential recommender
├ 📁 saved_model
│  ├ context_mf.pth        # Trained model
├ 📁 data                  # Dataset (MovieLens, processed)
├ 📁 notebooks             # Experiment Jupyter notebooks
├ image_app.py             # Streamlit app
├ requirements.txt         # Dependencies
├ README.md                # Usage guide
```

9. Execution Steps:

```
#!/bin/bash
# Basic execution commands for Context-Aware Movie Recommendation System

# 1. Create & activate environment
    python3 -m venv myenv
    source myenv/bin/activate
```

```
# 2. Install dependencies
    pip install -r requirements.txt

# 3. Preprocess data
    python -m src.preprocess

# 4. Train model
    python -m src.train

# 5. Run Streamlit app
    streamlit run image_app.py
```