# System Design Document

Daniel David Cuellar, Henry Jhonmarcos Osorio Sánchez,
Juan Andrés Jimenez Palomino, Miguel Alejandro Chavez Porras

Universidad Distrital Francisco Jose de Caldas

**Abstract.** In the following document we did an exhaustive analysis of the system to see the interactions , components and behaviour of the system also doing an overview of the requierements that system needs. Addicionaly, what is expected to have as output, the sensitivity and chaos and lastly a Technical Stack for future applications.

**Keywords:** Music · System Analysis · Prediction model · Design · Sensitivity

## 1 System summary

The previous analysis examined the KKBOX Music Recommendation System, which predicts whether a user will replay a song within one month or not. The system involves users, songs, listening events, and contextual variables such as time and source type. Data characteristics: The available datasets contain millions of entries, but also show significant noise and inconsistency. Some columns (like user age or ISRC codes) include outliers and formatting errors, which increase system complexity. Constraints: The train/test split is strictly time-based, and several features (genre IDs, source screen) may contain missing or ambiguous values. Sensitivity: Model accuracy is highly sensitive to feature selection and preprocessing. Removing contextual variables or misinterpreting metadata drastically reduces AUC. The cold-start condition (new users or songs) introduces instability. Chaos and randomness: Unpredictable user behavior, feedback loops from recommendations, and rapid changes in trends introduce chaotic dynamics. External events (new releases, concerts) also cause unpredictable fluctuations. Implication for design: These findings highlight the need for a robust architecture capable of handling noise, monitoring drift, and adapting dynamically. The proposed system design will therefore include a preprocessing layer for data validation, a feedback monitoring module, and retraining pipelines to mitigate chaotic effects.

### System Requirements

The system design derives directly from the findings obtained in Workshop 1. During the analysis, several key challenges were identified, such as noisy datasets, outliers, unpredictable user behavior, and temporal instability. To address these issues, the following system requirements are established.

## 2 Functional Requirements

1. The system must ingest and preprocess all available datasets, including train.csv, songs.csv, members.csv, and song_extra_info.csv.
2. The system must perform feature extraction and transformation to prepare data for model training.
3. It must train a binary classification model to predict whether a user will replay a specific song within one month.
4. The design must include an evaluation stage that measures performance using the AUC (Area Under the ROC Curve) metric.
5. The system must include monitoring and retraining routines to maintain model performance over time.
6. A data validation process must detect and handle missing values, duplicates, and corrupted entries before model training.

## 3 Non-Functional Requirements

1. Data processing pipelines should be completed within a reasonable time frame (under five minutes for standard datasets).
2. The system architecture must support scalability to handle millions of user-song interactions.
3. Each module should be modular and maintainable, allowing independent updates to data preprocessing, modeling, and evaluation components.
4. The system must tolerate moderate levels of noise without significant loss of predictive accuracy.
5. Logging and tracking must be integrated to ensure traceability of all processing steps.

## 4 Sensitivity and Chaos-Related Requirements

1. The design must include mechanisms to manage high-sensitivity variables, such as outlier filtering for user age and normalization of ISRC codes.
2. The system should detect performance degradation caused by data drift or concept drift and trigger model retraining automatically when necessary.
3. Feedback loops must be monitored to prevent reinforcement of biased recommendation patterns.
4. The architecture should incorporate a validation layer capable of identifying abnormal input distributions before model inference.

## 5 User-Centric Requirements

1. The system must provide interpretable predictions through feature importance or SHAP-based explanations.

2. Interfaces or reports must allow analysts to inspect system outputs and data statistics easily.
3. Data privacy and secure handling of user attributes must be guaranteed throughout the entire process.
4. The overall design should ensure reliability, transparency, and adaptability to real-world streaming environments.

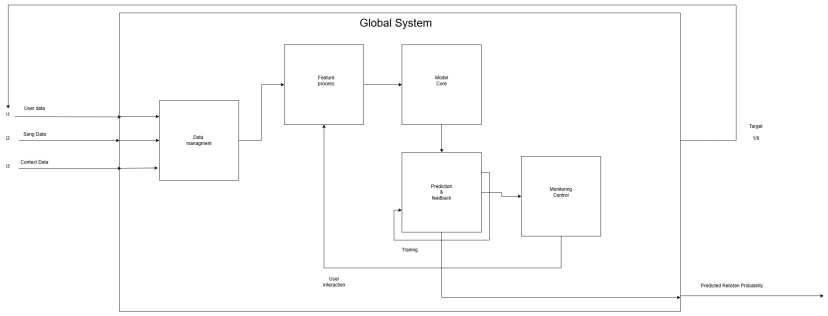


**Fig. 1.** Global System.

### Data ingestion

Considering the nature of the problem and the proposed datasets, it is encouraged to use batch-based processing data supported by an ETL pipeline (extract-transform-load). The following diagram shows how data ingestion will be executed.

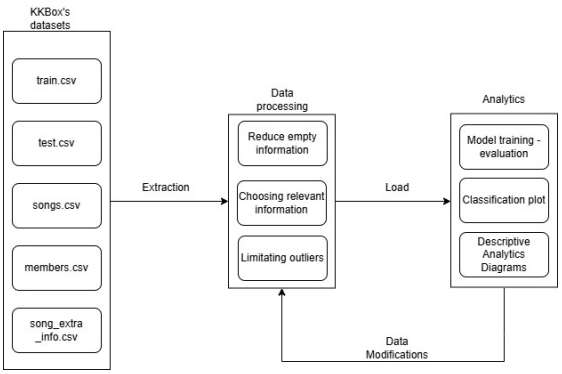


**Fig. 2.** Data Ingestion.

## 6 Addressing Sensitivity and Chaos

1. The system architecture mitigates high sensitivity and chaotic behavior by embedding controlled feedback loops and continuous monitoring throughout the data pipeline. These mechanisms ensure that unpredictable changes in user behavior, data inconsistencies, or random variations do not propagate unchecked across the system.
2. In the Data Management Layer, sensitivity is reduced through strict validation and normalization routines. Inconsistent or missing entries are detected, corrected, or removed before they reach the processing stages, preventing random data noise from influencing model training.
3. The Feature Processing Layer stabilizes variability by applying standardized scaling and encoding to all relevant attributes. It also generates aggregated and temporal features that capture broader patterns, reducing the impact of isolated or erratic user actions.
4. Within the Model Core, ensemble learning and time-aware validation are used to counteract chaotic fluctuations. These strategies distribute prediction variance across multiple weak learners and protect the model from overfitting to transient data anomalies. The model registry adds another stability mechanism by tracking versions and enabling rollback if unexpected behavior emerges.
5. The Prediction and Feedback Layer incorporates an internal feedback mechanism that compares the model's Predicted Relisten Probability with actual user outcomes (target 1/0). This comparison provides continuous performance signals to the monitoring system, transforming user interactions into structured diagnostic information.
6. The Monitoring and Control Layer is responsible for detecting and correcting system drift. It observes statistical deviations in data distributions and key metrics. When instability is detected, it triggers retraining cycles in the Model Core or sends recalibration signals to the Feature Processing Layer. In extreme cases, it reverts the model to the latest verified version to maintain operational consistency.
7. Error handling and resilience are addressed through automated exception logging and recovery routines. Each module is capable of restarting independently in response to faults, minimizing downtime and ensuring operational continuity even under unpredictable or incomplete input conditions.
8. Through this combination of validation, feature stability, ensemble modeling, and closed-loop monitoring, the architecture transforms a chaotic data environment into a self-regulating system that sustains reliable performance under real-world variability.

## 7 Technical Stack

1. The system will be implemented entirely in Python, chosen for its simplicity, readability, and strong support for data processing and machine learning tasks.

2. Pandas and NumPy will manage data ingestion, cleaning, and transformation within the Data Management and Feature Processing layers. Their efficiency with structured datasets ensures smooth handling of user, song, and context information.
3. The Model Core will use a Decision Tree Classifier from Scikit-learn, selected for its interpretability and robustness under noisy or nonlinear data conditions. Its transparent decision paths make it suitable for analyzing sensitivity and chaotic variations in user behavior.
4. Predictions will be served through a lightweight Flask API, which connects the trained model to incoming data and returns the Predicted Relisten Probability. This same layer will log user feedback for later retraining.
5. Matplotlib will be used to generate performance and drift visualizations, supporting monitoring and analysis in the Control Layer.
6. The overall design follows a modular pipeline pattern, where each component operates independently but communicates through defined interfaces. This structure simplifies integration, debugging, and the inclusion of feedback loops to maintain model stability .

## References

1. https://www.kaggle.com/competitions/kkbox-music-recommendation-challenge/overview
2. https://github.com/EngAndres/ud-public/tree/main/courses/systems-analysis
3. Systems Analysis and Design, by Alan Dennis, Barbara Haley Wixom, and Roberta M. Roth