# Software Requirements Specification (SRS)

## Netflix Movie Recommendation System

* Version: 1.0.0
* Date: 2025-09-26
* Status: Draft

This SRS is derived from the implementation and workflow in the Jupyter notebook `netflix\_recommendation\_system.ipynb` and the CSV datasets in this repository. Where possible, it also anticipates alignment with any existing SRS context you’ve shared. A short “Alignment with external SRS PDF” section is provided for future reconciliation.

## 1. Introduction

### 1.1 Purpose

Define the functional and non-functional requirements for a data science system that predicts whether a Netflix user will like a movie/TV show based on user attributes, interaction history, and content metadata, and that supports comparative model training and evaluation within a reproducible notebook workflow.

### 1.2 Scope

* Data ingestion from three CSVs: movie metadata, user profiles, and user-movie interactions.
* Data preprocessing, cleaning, and feature engineering.
* Dataset splitting with safeguards against user cold-start leakage.
* Feature preprocessing and feature selection via multiple strategies.
* Model training (Logistic Regression, Random Forest, XGBoost, MLP), hyperparameter tuning, and evaluation.
* Reporting of metrics and feature importance; recommendation-ready output via like-probability scores.

Out of scope for this iteration:

* Production serving endpoints or microservices.
* Real-time personalization and A/B testing.
* UI/UX surfaces (web/app) for users.

### 1.3 Definitions, Acronyms, and Abbreviations

* Interaction: A user’s event with a show (e.g., watched X minutes, rated, like/dislike).
* Like: Binary target (1 liked, 0 not liked) for supervised learning.
* Feature Set: A specific subset/representation of features used to train models.
* Cold-start: Evaluating users/items unseen in training; mitigated by split policy.

### 1.4 References

* Notebook: `netflix\_recommendation\_system.ipynb`
* Datasets: `netflix\_movies\_detailed\_up\_to\_2025.csv`, `netflix\_user.csv`, `netflix\_user\_movie\_interaction.csv`
* Libraries: pandas, numpy, scikit-learn, xgboost, matplotlib, seaborn, plotly, imbalanced-learn

## 2. Overall Description

### 2.1 Product Perspective

This is an analytics/ML workflow defined in a Jupyter notebook. It operates offline, consuming static CSV files to produce trained models and evaluation artifacts.

### 2.2 Product Functions (High-Level)

* Load and validate datasets
* EDA with visual summaries
* Preprocess movie and user data with derived columns
* Merge interactions → feature engineering (user/movie stats, genre match, temporal)
* Train/validation/test split by user cohorts to reduce leakage
* Feature scaling/encoding and selection (ANOVA, RF importance, correlation filtering)
* Baseline model training and selection
* Hyperparameter tuning of top models
* Final evaluation and feature importance analysis

### 2.3 User Classes and Characteristics

* Data Scientist/ML Engineer: executes notebook, inspects results, iterates.
* Analyst/Stakeholder: reviews metrics and diagrams, interprets insights.

### 2.4 Operating Environment

* Executed in Jupyter notebook within a Python environment (refer to `requirement.txt`).
* Offline batch processing using local CSV files.

### 2.5 Design and Implementation Constraints

* Data volume limited by CSV memory footprint.
* Notebook execution order; cells must be run top-to-bottom.
* External libraries availability (e.g., xgboost, imbalanced-learn).

### 2.6 Assumptions and Dependencies

* CSV schemas match the fields used in the notebook code (see Section 4).
* Timestamps are parseable and in a standard format for `interaction\_date`.
* Reproducibility depends on fixed random seeds and library versions.

## 3. System Features (Functional Requirements)

Each feature includes an identifier (F-#), description, rationale, and acceptance criteria.

### F-1 Data Ingestion

* Description: Load three CSV files for movies, users, and user-movie interactions.
* Rationale: Provide raw inputs for the ML pipeline.
* Acceptance:
* Successfully reads files with shape logs.
* Missing/invalid date fields in `interaction\_date` are parseable or handled.

### F-2 Exploratory Data Analysis (EDA)

* Description: Produce summary statistics and plots for key distributions and genre breakdowns.
* Acceptance:
* Histograms for years, vote averages, popularity, user ages.
* Count of top genres and type mix (movie vs TV).

### F-3 Data Preprocessing

* Description: Clean movies (fill missing values, derived columns like ROI, age), clean users (age groups, watch-time categories, account maturity), normalize durations.
* Acceptance:
* No NaNs in critical numeric features used by models after preprocessing.
* Added columns: `duration\_minutes`, `is\_movie`, `has\_budget`, `has\_revenue`, `profit`, `roi`, `movie\_age`, `popularity\_rank`, `rating\_category`, user demographic buckets.

### F-4 Feature Engineering

* Description: Merge datasets; compute user/movie aggregates, genre dummies, genre match score, temporal features, deltas (e.g., user\_avg\_rating − vote\_average), popularity-vs-interactions.
* Acceptance:
* Feature matrix created with shape and column counts logged.
* `genre\_match\_score` computed for all merged rows.

### F-5 Dataset Splitting

* Description: User-level stratified split into train/val/test; ensures users in validation/test have representation without leaving all interactions to held-out sets.
* Acceptance:
* Logged counts of users and samples in each split.
* Like-rate distribution similar across splits (± 5 p.p.).

### F-6 Preprocessing & Feature Selection

* Description: Scale numeric features; encode categoricals; apply multiple selection methods: SelectKBest (ANOVA), RF importance, correlation filtering.
* Acceptance:
* Provide at least one feature set with ≥ 30 features.
* Report top features and removed highly correlated columns.

### F-7 Baseline Modeling

* Description: Train Logistic Regression, Random Forest, XGBoost (and optionally MLP) on each feature set; compute accuracy, precision, recall, F1, ROC-AUC.
* Acceptance:
* Print validation metrics per model-feature set.
* Persist best result summary table.

### F-8 Hyperparameter Tuning

* Description: Randomized search over RF, XGBoost, and MLP parameter grids using combined train+val folds.
* Acceptance:
* Best params and CV scores reported for each tuned model.

### F-9 Final Evaluation

* Description: Evaluate tuned models on the test set; show confusion matrix and comparison plots; compute feature importance from tree-based models.
* Acceptance:
* Test metrics table produced; figures rendered.
* Top-10 features plotted by importance when supported.

### F-10 Recommendation-Ready Scores

* Description: Output per-(user, show) probabilities to support ranking for recommendations.
* Acceptance:
* Probabilities accessible from the best model; can be exported for top-N ranking in a downstream step (outside this SRS’s core scope).

## 4. Data Requirements

### 4.1 Input Datasets and Expected Fields

* Movies (`netflix\_movies\_detailed\_up\_to\_2025.csv`):
* Identifiers: `show\_id`, `title`
* Metadata: `type` (Movie/TV Show), `genres`, `director`, `cast`, `country`, `language`, `duration`, `release\_year`, `date\_added`, `description`
* KPIs: `vote\_average`, `vote\_count`, `popularity`, `budget`, `revenue`
* Users (`netflix\_user.csv`):
* Identifiers: `user\_id`
* Demographics and profile: `age`, `gender`, `subscription\_type`, `preferred\_genres` (comma-separated), `preferred\_watch\_time`, `account\_age\_months`, `avg\_daily\_watch\_time`
* Interactions (`netflix\_user\_movie\_interaction.csv`):
* Keys: `user\_id`, `show\_id`
* Targets and signals: `liked` (0/1), `rating`, `watch\_percentage`, `minutes\_watched`, `interaction\_date`

### 4.2 Derived Fields (examples)

* Movies: `duration\_minutes`, `is\_movie`, `has\_budget`, `has\_revenue`, `profit`, `roi`, `movie\_age`, `popularity\_rank`, `rating\_category`
* Users: `age\_group`, `watch\_time\_category`, `account\_maturity`, `num\_preferred\_genres`
* Merged/Engineered: `user\_avg\_rating`, `user\_rating\_std`, `user\_total\_interactions`, `user\_total\_likes`, `user\_like\_rate`, `user\_avg\_watch\_pct`, `user\_avg\_minutes`, `movie\_avg\_rating`, `movie\_rating\_std`, `movie\_total\_interactions`, `movie\_like\_rate`, `movie\_avg\_watch\_pct`, genre one-hots, `genre\_match\_score`, `user\_movie\_rating\_diff`, `popularity\_vs\_user\_interactions`, `movie\_popularity\_rank\_norm`, `user\_selectivity`, `movie\_appeal`, `interaction\_year`, `interaction\_month`, `interaction\_dayofweek`, `movie\_age\_at\_interaction`.

### 4.3 Data Quality Requirements

* Missing numeric values imputed via median; categoricals by mode/constant.
* Consistent delimiters for genre fields (comma/`, `).
* All IDs non-null; join cardinalities preserved.

## 5. External Interface Requirements

* User Interface: Jupyter notebook cells (Markdown explanations + code).
* Reports: Printed tables and Matplotlib/Plotly figures within the notebook.
* Storage: CSV read/write on local filesystem.

## 6. System Architecture and Workflow

### 6.1 Data Flow Diagram

flowchart TD  
 M[Movies CSV\nnetflix\_movies\_detailed\_up\_to\_2025.csv] --> PM[Preprocess Movies]  
 U[Users CSV\nnetflix\_user.csv] --> PU[Preprocess Users]  
 I[Interactions CSV\nnetflix\_user\_movie\_interaction.csv] --> FE[Merge & Feature Engineering]  
 PM --> FE  
 PU --> FE  
 FE --> SPLIT[User-level Stratified Split]  
 SPLIT --> PREP[Scaling / Encoding]  
 PREP --> FS[Feature Selection\n(ANOVA, RF, Corr Filter)]  
 FS --> BM[Baseline Models]  
 BM --> TUNE[Hyperparameter Tuning]  
 TUNE --> EVAL[Final Evaluation]  
 EVAL --> OUT[Metrics, Plots, Feature Importance]

### 6.2 Pipeline Activity Diagram

sequenceDiagram  
 participant DS as Data Scientist  
 participant NB as Notebook  
 DS->>NB: Run cell: Load libraries & data  
 NB-->>DS: Shapes, info, previews  
 DS->>NB: Run EDA  
 NB-->>DS: Charts & summaries  
 DS->>NB: Run preprocessing  
 NB-->>DS: Cleaned datasets  
 DS->>NB: Run feature engineering  
 NB-->>DS: Feature matrix + logs  
 DS->>NB: Create splits and preprocessing pipeline  
 NB-->>DS: Train/Val/Test sets  
 DS->>NB: Train baseline models  
 NB-->>DS: Validation metrics  
 DS->>NB: Hyperparameter tuning (best set)  
 NB-->>DS: Best params + CV scores  
 DS->>NB: Final evaluation on test  
 NB-->>DS: Test metrics, confusion matrix, importance

### 6.3 Components Overview

graph LR  
 subgraph Data  
 D1[Movies CSV]  
 D2[Users CSV]  
 D3[Interactions CSV]  
 end  
 subgraph Processing  
 P1[Preprocess Movies]  
 P2[Preprocess Users]  
 P3[Feature Engineering]  
 P4[Split by User]  
 P5[Scaling/Encoding]  
 P6[Feature Selection]  
 end  
 subgraph Modeling  
 M1[Logistic Regression]  
 M2[Random Forest]  
 M3[XGBoost]  
 M4[MLP]  
 end  
 subgraph Evaluation  
 E1[Validation Metrics]  
 E2[Hyperparameter Tuning]  
 E3[Test Metrics]  
 E4[Feature Importance]  
 end  
 D1-->P1-->P3  
 D2-->P2-->P3  
 D3-->P3  
 P3-->P4-->P5-->P6  
 P6-->M1-->E1  
 P6-->M2-->E1  
 P6-->M3-->E1  
 P6-->M4-->E1  
 E1-->E2-->E3-->E4

## 7. Non-Functional Requirements (NFRs)

* Performance
* NFR-P1: End-to-end notebook run should complete within a reasonable time on a mid-range laptop (e.g., < 30 minutes for datasets of current size). Actual time depends on hardware and data volume.
* Scalability
* NFR-S1: Pipeline should avoid O(N^2) where possible; use vectorized pandas and scikit-learn.
* Reliability
* NFR-R1: Determinism with fixed random seeds; repeatable splits and metrics.
* Maintainability
* NFR-M1: Code modularized into functions for preprocessing, feature engineering, split, training, tuning, evaluation.
* Security & Privacy
* NFR-SP1: User data is local; no PII beyond provided columns, and no external transmission.
* Observability
* NFR-O1: Logs printed for shapes, feature counts, and metric summaries.

## 8. Constraints and Risks

* Library compatibility issues (e.g., xgboost binary availability) may block training.
* Class imbalance may lead to biased metrics; consider SMOTE/undersampling if needed.
* Correlated features can inflate importance; addressed via correlation filtering.
* Cold-start challenges for truly new users/items are not fully modeled here.

## 9. Acceptance Criteria

* AC-1: Notebook executes top-to-bottom without errors using the provided CSVs.
* AC-2: At least three model families trained and compared on validation metrics.
* AC-3: Hyperparameter tuning performed for two or more top models.
* AC-4: Final test metrics table and at least one comparison visualization generated.
* AC-5: Feature importance table/plot produced for tree-based models.

## 10. Future Enhancements

* Implement top-N recommendation ranking per user using predicted probabilities and business rules (diversity, freshness).
* Add cross-validation at the user-group level for robust estimates.
* Persist pipelines and models (joblib) and provide a minimal API for batch scoring.
* Explore matrix factorization / implicit feedback models to complement supervised approach.

## 11. Alignment with external SRS PDF (Placeholder)

Note: The provided PDF content appears to be a generic SRS for another product (e.g., “DineOut”). To finalize alignment with your intended SRS, please provide the authoritative PDF text or confirm sections to inherit (scope, actors, constraints). We will reconcile terminology and add a traceability section mapping PDF requirements → this SRS’s features (F-1..F-10) and NFRs.

Proposed reconciliation checklist:

* Actors/User classes
* Functional scope and exclusions
* Data classification and privacy constraints
* Performance SLAs
* Acceptance and sign-off criteria

## 12. Traceability Matrix (Excerpt)

* Business Goal: Improve recommendation accuracy → F-7, F-8, F-9; NFR-P1, NFR-R1
* Reduce leakage/cold-start bias → F-5; NFR-R1
* Explainability via feature importance → F-9; Maintainability (NFR-M1)

## 13. Appendix

* Library versions are printed in the notebook upon import.
* Dataset field lists derived from code inspection in the notebook.