Software Requirements Specification

for

Movie Recommendation System

Version 1.0 approved

Prepared by Alex Horton, David Ho, Marko Kosoric

McMaster University

November 13, 2025

Table of Contents

Table of Contents ii
Revision History ii
1. Introduction 1
1.1 Purpose 1
1.2 Intended Audience and Reading Suggestions 1
1.3 Product Scope 1
1.4 References 1
2. Overall Description 2
2.1 Product Perspective 2
2.2 Product Functions 2
2.3 User Classes and Characteristics 2
2.4 Operating Environment 2
2.5 Design and Implementation Constraints 2
2.6 Assumptions and Dependencies 2
2.7 Use Cases 3
3. External Interface Requirements 4
3.1 User Interfaces 4
4. System Features 4
4.1 File Processing 4
4.2 Recommendation Engine 4
5. Other Nonfunctional Requirements 5
5.1 Performance Requirements 5
5.2 Software Quality Attributes 5
Appendix A: Glossary 6
Appendix B: To Be Determined List 6

Revision History

Name	Date	Reason For Changes	Version
Alex, David, Marko	11/13/2025	First Final Version	1.0

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the requirements for the Movie Recommendation System (MRS) developed for COMPSCI 2ME3 at McMaster University. The purpose of this SRS is to clearly define the functional and non-functional requirements of the MRS software, which provides users with personalized movie recommendations based on content similarity and user ratings.

1.2 Intended Audience and Reading Suggestions

This document is intended for any developers wishing to replicate this software application. It serves as a guideline to follow to ensure all requirements are met.

1.3 Product Scope

MRS is a Java-based application that recommends movies to users based on the similarity of movie content and past ratings. The MRS serves an educational purpose within the COMPSCI 2ME3:Introduction to Software Development course by providing experience with the Waterfall and Agile software processes. The main goals of the MRS are to accurately read and parse movie and rating data, enable users to rate and update movie ratings, generate personalized recommendations based on genre similarity, and handle missing data or malformed data without causing program crashes.

1.4 References

Movies.txt - A comma-separated value (CSV) file containing the movies.
Each line includes:
movie_id, title, director, year, genres, avg_rating, num_ratings
movie_id is unique. Multiple genres are separated by semicolons

My_ratings.txt - A comma-separated values (CSV) file containing user movie ratings.
Each line includes:
movie_id, rating, timestamp
Ratings are integers from 1-5, and timestamps follow the YYY-MM-DD format.

2. Overall Description

2.1 Product Perspective

Standalone Java application which aims to give users content-based recommendations for movies supported by users ratings of movies. Offers best movie suggestions based on the users movie preferences.

2.2 Product Functions

- File parsing and Data Management
- Rating Functionality
- Content-Based Recommendations

2.3 User Classes and Characteristics

Classes:

- Movie
  - Store data for each individual movie
- MovieDatabase
  - Store every movie from movies.txt
- Review
  - Store review data for each review
- User
  - Store all of the users reviews from reviews.txt
- RecommendationEngine
  - House the recommendation algorithm
- FileHandler
  - Handle file processing logic
- Main
  - Main class that combines every part of the system to allow the product to work

Most important classes to satisfy: RecommendationEngine, FileHandler, Main

2.4 Operating Environment

Must be run on a computer with Java installed, and the two files movies.txt and my_Ratings.txt must be present.

2.5 Design and Implementation Constraints

Must be programmed in Java, must only use standard libraries, CSV file format only.

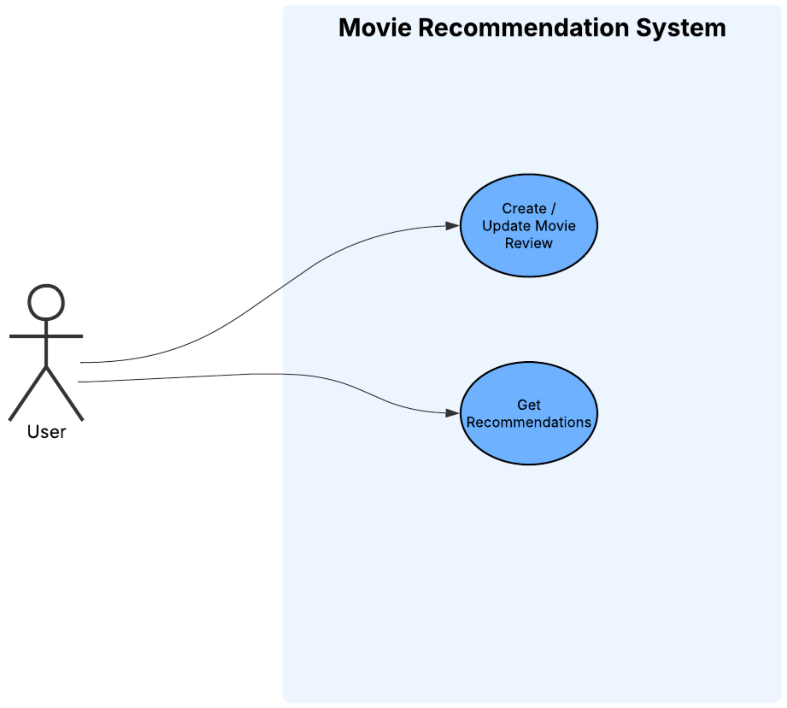
2.6 Assumptions and Dependencies

- The user must be able to run Java
- Files must be in the right location and accessible
- Files must be in the right format.

2.7 Use Cases

Use Case 1:
Name: Rate Movie
Actors: User
Description: User has the ability to rate movies on a score of 1-5 based on how much they liked the movie.

Use Case 2:
Name: Get Recommendations
Actors: User
Description: System recommends User movies based on movies they rated highly (4-5 stars). Orders movies based on recommendation algorithm and returns top results to User.



3. External Interface Requirements

3.1 User Interfaces

CLI done through the users terminal.

4. System Features

4.1 File Processing

4.1.1 Description and Priority

System must be capable of reading data from and writing to files. This is a high. priority feature and is required for the system to function.

4.1.2 Stimulus/Response Sequences

Movie and user reviews file must be read when user connects to system.
User creating or updating a review must write this new/updated review to the my_ratings.txt file

4.1.3 Functional Requirements

FR-FP1: The system shall be able to read and parse data from .txt file (CSV format)

Rationale: The system needs to be able to read the available movies from movies.txt and the users current ratings from my_ratings.txt (if any) to be able to calculate any recommendations based on those available movies and ratings.

Invalid input: The system will return meaningful error messages due to invalid input. i.e Missing files, invalid formatting…
Will continue running unless it is incapable of performing an action due to invalid input.

FR-FP2: The system shall be able to write to and update entries in a .txt file (CSV format)

Rationale: The user needs to be able to add any new ratings, or update any reviews they have made.

Invalid input: The system will return meaningful error messages due to invalid input. i.e Missing files, invalid formatting…
Will continue running unless it is incapable of performing an action due to invalid input.

4.2 Recommendation Engine

4.1.1 Description and Priority

Handles recommending the user movies based on their previous ratings. This is a high priority feature and is required for the system to function.

4.1.2 Stimulus/Response Sequences

User requests movie recommendations.

4.1.3 Functional Requirements

FR-RE1: The system shall be able to analyze the user's movie rating history and calculate similarity scores between their highly rated movies (4-5 stars) and ones they haven't watched (rated).

Rationale: The system needs to be able to recommend users movies based on ones they already enjoyed. Recommend movies with the highest similarity scores.

Invalid input: If the user has no ratings (my_ratings.txt) is empty, recommend popular movies instead (high average rating, and high number of ratings, exact threshold TBD)

5. Other Nonfunctional Requirements

5.1 Performance Requirements

NF-1: The system shall handle file operations robustly and gracefully handle errors for missing files and malformed data.
Rationale: Users should not experience any crashed due to file errors or malformed data. Also should handle cold start problem properly. Experience should be seamless for the user

NF-2: The system shall have a reasonable response time to any user action
Rationale: The system should not make the user wait long for any action performed. It should provide real time feedback with low latency to enhance user satisfaction.

NF-3: The system shall handle malformed or improper requests from the user gracefully without crashing
Rationale: Users should not experience any crashes because of an invalid request. The system should handle any user error gracefully to ensure their satisfaction.

5.2 Software Quality Attributes

NF-4: The systems code shall be modular with a clear separation of concerns between data management, recommendation logic, and user interface components.
Rationale: The system's components should be well organized and separated so it is easier to maintain, scale, and test.

NF-5: The systems code shall be reasonably commented and follow consistent Java best practices.
Rationale: The code should be easy to read and follow so each components job can be understood

Appendix A: Glossary

- FR-FPi : Functional Requirement – File Processing Number i
- FR-REi: Functional Requirement – Recommendation Engine Number i
- NF-i: Non-functional Requirement Number i

Appendix B: To Be Determined List

- Exact threshold of what popular movies score to recommend