

# Project Title: System Verification and Validation Plan for Recommender System

Seyed Ali Mousavi

March 2, 2024

## Revision History

Date	Version	Notes
2024-02-19	1.0	Initial Release
2024-03-02	2.0	Second Draft

# Contents

<b>1</b>	<b>General Information</b>	<b>1</b>
1.1	Summary . . . . .	1
1.2	Objectives . . . . .	1
1.3	Relevant Documentation . . . . .	1
<b>2</b>	<b>Plan</b>	<b>2</b>
2.1	Verification and Validation Team . . . . .	2
2.2	SRS Verification Plan . . . . .	3
2.3	Design Verification Plan . . . . .	3
2.4	Verification and Validation Plan Verification Plan . . . . .	4
2.5	Implementation Verification Plan . . . . .	4
2.6	Automated Testing and Verification Tools . . . . .	4
2.7	Software Validation Plan . . . . .	5
<b>3</b>	<b>System Test Description</b>	<b>5</b>
3.1	Tests for Functional Requirements . . . . .	5
3.1.1	One Popular Movie . . . . .	5
3.1.2	Artificial Closed Users . . . . .	6
3.2	Tests for Nonfunctional Requirements . . . . .	7
3.2.1	Accuracy . . . . .	7
3.3	Traceability Between Test Cases and Requirements . . . . .	8
<b>4</b>	<b>Unit Test Description</b>	<b>8</b>

# 1 General Information

## 1.1 Summary

This project is building a recommendation system based on the MovieLens dataset, i.e. a Movie Recommender. Our movie recommender system utilizes a collaborative filtering method. Collaborative methods for recommender systems are methods that are based solely on the past interactions recorded between users and items to produce new recommendations. These interactions are stored in the so-called “user-item interactions matrix”. The rows in the matrix represent users indices and the columns represent items indices (movies, in our case). So the entry  $ij$  of the matrix represents the rating of the user  $i$  for the movie  $j$  (an integer number between 1 to 5). The main idea that rules collaborative methods is that these past user-item interactions are sufficient to produce good enough recommendations. After receiving a specified user of interest (target user index) for whom we want to make recommendations, the ANN or KNN method algorithm is executed to generate the top recommendations.

## 1.2 Objectives

The objectives outlined in this VnV are as follows:

- Validate the software’s capability to produce good recommendations (according to performance metrics in subsequent sections).
- Verify the proper functioning of the Movie Recommender.
- Confirm the correctness and Maintainability of the software

I avoid conducting usability testing for the Movie Recommender as this project lacks a defined user interface, and the required inputs are too straightforward to warrant user testing.

## 1.3 Relevant Documentation

The reader of this document will be referenced to the other documents of this project(for instance, SRS MG, MIS, etc.) whenever it’s needed.

## 2 Plan

This section outlines the approach for validating both the documentation and software of the Movie Recommender. The main elements scheduled for validation include the Software Requirements Specification (SRS), design, Verification and Validation (VnV) plan, and implementation.

### 2.1 Verification and Validation Team

The team members and their responsibilities are specified in [Table 1](#).

Name	Document	Role	Description
Seyed Ali Mousavi	All	Author	Generate and oversee document creation, develop the VnV plan, conduct VnV testing, and validate the implementation.
Dr. Spencer Smith	All	Instructor/Reviewer	Evaluate the documents for design and documentation style.
Nada Elmasry	All	Domain Expert Reviewer	Review all the documents.
Xinyu Ma	SRS	Secondary Reviewer	Review the SRS document
Valerie Vreugdenhil	VnV Plan	Secondary Reviewer	Review the VnV plan.
Gaofeng Zhou	MG + MIS	Secondary Reviewer	Review the MG and MIS document.

Table 1: Verification and Validation Team

## 2.2 SRS Verification Plan

The SRS document will be verified in the following way:

1. The assigned members (Dr. Spencer Smith, Nada Elmasry, and Xinyu Ma) will conduct the initial review. This will involve a manual assessment using the provided SRS Checklist, along with additional checklists gathered from various sources. SRS Checklist and following additional checklist are gathered from [here](#).
  - Do all requirements strictly pertain to requirements, and not to design or implementation solutions?
  - Has each requirement been uniquely and accurately identified?
  - Is each requirement verifiable by testing, demonstration, review, or analysis?
  - Do any requirements conflict with or duplicate other requirements?
  - Is each requirement in scope for the project?
  - Are all requirements written at a consistent and appropriate level of detail?
  - Do the requirements provide an adequate basis for design?
2. The reviewers will provide feedback to the author by creating an issue on GitHub.
3. Seyed Ali Mousavi, the author, is responsible for addressing the issues.

## 2.3 Design Verification Plan

The assigned members (Dr. Spencer Smith, Xinyu Ma, and Gaofeng Zhou) will conduct the initial review. This will involve a manual assessment using the provided checklist.

- Does the design meet all specified requirements outlined in the design specification document?
- Are all interfaces between design components specified clearly?
- Are all components of the design testable separately?

- Is the design scalable to accommodate future growth or changes in requirements?
- Are the design components separated as much as possible?
- Does each design component have a logical task?

## 2.4 Verification and Validation Plan Verification Plan

The specified members (Dr. Spencer Smith, Xinyu Ma, and Valerie Vreugdenhil) will perform the initial review, which will involve manually assessing the document using the following checklist.

- Does the VnV plan verify all the functional and non-functional requirements?
- Are all input and output pairs for test cases correct?
- Are roles and responsibilities clearly defined for individuals or teams involved in verification and validation activities?

## 2.5 Implementation Verification Plan

The Movie Recommender project will utilize the Python programming language for development purposes. Moreover, code walkthroughs involving domain expert Nada Elmasry will be conducted. Dynamic testing will include both system and unit level testing, elaborated further in sections 3 and 4.

## 2.6 Automated Testing and Verification Tools

- Pytest: A Python unit testing framework facilitating straightforward and scalable testing of Python code.
- GitHub: This platform will be used for version control and collaboration.
- GitHub CI workflow: This will automate regression tests and checks that Movie Recommender builds are passing before code is merged into a protected branch.

- Docker: A method employed for containerizing applications, ensuring their ease of installation and comprehensibility across the project.

## 2.7 Software Validation Plan

No plans for Software Validation yet.

# 3 System Test Description

## 3.1 Tests for Functional Requirements

### 3.1.1 One Popular Movie

Suppose we have a user-item interaction matrix like below:

$$\begin{array}{c} \text{USERS} \end{array} \begin{pmatrix} & \text{MOVIES} \\ 1 & 1 & 1 & \dots & 1 & 5 \\ 1 & 1 & 1 & \dots & 1 & 5 \\ 1 & 1 & 1 & \dots & 1 & 5 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & 1 & 5 \\ 1 & 1 & ? & \dots & ? & ? \end{pmatrix}$$

Figure 1: User-Item interaction Matrix for test-case 1

The final row represents the target user, whom we aim to provide recommendations for. It's evident from the user feedback, based on the ratings, that the last movie is the highest-rated, while all other movies are rated significantly lower. Therefore, it's clear that if the Movie Recommender is expected to generate a single recommendation, it should unequivocally be the last movie.

### One Popular Movie



1. test-popular movie-id1

Control: Manual

Initial State: N/A

Input: User-Item Interaction Matrix specified in the figure, in addition to the target user's index number

Output: The Last Movie Id

How test will be performed:

### 3.1.2 Artificial Closed Users

Suppose we have a user-item interaction matrix like below:

	MOVIES							
USERS	1	1	1	1	1	1	1	2
	1	2	3	2	2	1	5	5
	1	2	3	1	2	2	4	5
	1	1	1	1	1	5	1	2
	5	5	1	5	5	1	2	1
	5	4	4	3	3	3	1	1
	1	2	3	1	2	4	5	4
	1	2	3	1	?	?	?	?

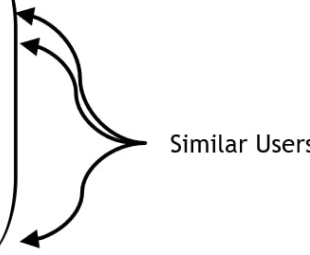


Figure 2: User-Item interaction Matrix for test-case 2

Once more, the final row represents the target user, for whom we strive to offer recommendations. While it might not be immediately apparent, user2, user3, and the target user exhibit similar feedback vectors, indicating their proximity in terms of preferences. Conversely, the other users are noticeably distant from the target user. Consequently, we anticipate that the recommendation should include the favorite movies of user2 and user3, which happen to be the last movies.

### Artificial Closed Users

1. test-Artificial Closed Users-id1

Control: Manual

Initial State: N/A

Input: User-Item Interaction Matrix specified in the figure, in addition to the target user's index number

Output: The Id of the last two movies

## 3.2 Tests for Nonfunctional Requirements

### 3.2.1 Accuracy

Various metrics can gauge the performance of a Recommender system, with one commonly used metric being Recall@K. This metric measures how many items in the top  $K$  recommendations match those in the user's actual preferences, where  $K$  represents the number of recommendations provided to the user. For example, suppose we have rating feedback from a user for 50 movies, and we aim to calculate Recall@10. We divide the list of viewed movies into two sublists, one with 30 movies serving as the basis for generating new recommendations, and the other with 20 movies acting as the test set. Following the generation of recommendations, we compare them to the top 10 movies in the test list (as per the actual ratings). The higher the overlap between these lists, the better the performance of our recommender. Recall@10 is computed as the number of common movies (between the top 10 movies in the recommendation list and the top 10 movies according to the real ratings) divided by the number of recommendations,  $K$  (where  $K = 10$ ).

After obtaining the Recall@10 for my Movie Recommender, I'll proceed to compare it with that of TFRS (TensorFlow Recommender System), an open-source Python library renowned for constructing cutting-edge recommenders. Currently, I deem a relative error of approximately 50 percent to be within an acceptable range for comparison purposes.

### Accuracy

1. test-accuracy-1

Type: Manual

Initial State: N/A

Input/Condition: a User-Item interaction matrix with removed 40 percent of ratings of the user of interest.

Output/Result: Recall@10 for Movie Recommender and Recall@10 for the TFRS.

### 3.3 Traceability Between Test Cases and Requirements

Provide a table that shows which test cases are supporting which requirements.

	R1	R2	R3	NFR1
test-1	X	X		
test-2	X	X	X	X

Table 2: Relation of Test Cases and Requirements.

## 4 Unit Test Description

This section will be finalized upon the completion of MIS.