❖ Complete Test policy of my Movie Recommendation System

1. Introduction: The main goal of this testing phase was to verify the integration of the model with the web application, and assess the overall user experience.. To guarantee its reliability and effectiveness, we implemented a comprehensive testing policy.

2.1 Testing Methodologies: The following testing methodologies were employed to ensure the system's functionality and performance:

2.2 Unit testing: Unit testing validated the correctness of individual components, such as data preprocessing, collaborative filtering algorithms, and utility functions, by testing them in isolation. White-box testing was also applied to focus on individual components and their interactions.

White-box testing: White-box testing involves testing the internal structure, design, and implementation of the software. This approach is useful for ensuring the code's correctness, efficiency, and maintainability.

2.3 Black-box testing: This approach focuses on testing the system's functionality without considering its internal structure or implementation. It helps validate that the system behaves as expected and meets user requirements.

2.4 Integration testing: Provided in video.

2.5 System testing: System testing verified the overall functionality and performance of the integrated system in a real-world environment. This included testing the recommendation quality, response times, user experience, and system reliability.

2.6 Acceptance testing:Actual users or stakeholders interacted with the system during acceptance testing, simulating real-world usage scenarios. Feedback from this testing helped identify any issues or improvements needed before the system's release.

2.7. Usability Testing: Usability testing was conducted to evaluate the system's user interface and overall user experience. This testing helped identify areas for improvement in the system's design, layout, and interactions, ensuring that the application was user-friendly and met user expectations.

❖FIRST LEVEL TESTING:

#	Module	Input	Actual Output	Expected Output	Status
1.	Collaborative filtering algorithm	User preference	Recommendations	Recommendations	Pass
2.	Ranking in recommendation according to score	User preference	Recommendations based on rank (cosine similarity score)	Recommendations based on rank	Pass
3,	Recommendations based on year	Year input	Top 5 movies	Top 5 movies	Pass
4.	Recommendations based on movie title	Movie title input	Top 5 movies	Top 5 movies	Pass
5.	Recommendations based on genre	Genre input	Top 5 movies	Top 5 movies	Pass
6.	Recommendations based on genre & cast	Select 5 cast and 3 genre	Top 3 movies	Top 5 movies	FAIL
7.	Input Validation	Invalid input in forms	Error Message	Error Message	Pass
8.	Navigation to other pages from any page	Navigate to different pages from each page	Seamless navigation	Seamless navigation	Pass
9.	API Integration	API request (GET, POST)	API request (Json Data)	API request (Json Data)	Pass

These first-order test cases provide an overview of the individual components of the Movie Recommendation System, their inputs, actual outputs, expected outputs, and whether the test case has passed or not.

❖SECOND LEVEL TESTING:

#	Module	Input	Actual Output	Expected Output	Status
1.	Collaborative filtering performance	Large user-item matrix	Stable performance, response time	Stable performance, response time	Pass
2.	Ranking Algorithm Robustness	Various user preferences	Recommendations based on rank (cosine similarity score)	Recommendations based on rank	Pass
3,	Year-Based Recommendation Accuracy	Various year inputs	Top 5 movies	Top 5 movies	Pass
4.	Title-Based Recommendation Accuracy	Various movie title inputs	Top 5 movies	Top 5 movies	Pass
5.	Genre-Based Recommendation Accuracy	Genre input	Top 5 movies	Top 5 movies	Pass
6.	Recommendations based on genre & cast	Select 5 cast and 3 genre	Top 3 movies	Top 5 movies	FAIL (to be fixed)
7.	Input Validation Security	Invalid input in forms	Error Message	Error Message	Pass
8.	Seamless Navigation Robustness.	Navigate through all pages	Consistent navigation experience	Consistent navigation experience	Pass
9.	API Integration	API request (GET, POST)	API request (Json Data)	API request (Json Data)	Pass
10.	System performance under stress	High traffic, concurrent users		Stable performance, response time	To be tested

❖ Test Limitations and Unperformed Tests:

Due to constraints, such as lack of testing knowledge, time and insufficient testing tools, some testing methods could not be performed comprehensively. This hindered the testing process hence testing tasks such as Test environment and tools and test process improvement could not be completedThe project scope limitations also meant that certain test cases, such as user authentication and concurrent users, were not performed. The following testing methods could not be carried out in this project:

(i). Load Testing: This is used to evaluate the system's performance under normal/peak load conditions which could not be performed. This testing would have helped identify bottlenecks and ensure that the system can handle the expected number of users.

(ii). Stress Testing: It is used to evaluate the system's performance under extreme conditions, such as high traffic or limited resources could not be performed. This is on my to-do list.

❖ Higher-order test cases for movie recommendation system:

I designed and executed higher-order test cases for integration testing, keeping software testing principles in mind, such as testing early and frequently, and ensuring the system operates smoothly with all components combined. The test cases included: Higher-order test cases designed for the movie recommendation system included:

<u>Recommendation generation:</u> Integrated and tested the ML model, data preprocessing, and recommendation engine. Verified accurate and diverse recommendations based on user preferences.

<u>Data update and synchronization:</u> Tested the interaction between data storage, data preprocessing, and the recommendation engine during data updates. Ensured timely processing and incorporation of new data (e.g., movies, ratings, user preferences) into the recommendation engine.

By designing and executing these higher-order test cases during integration testing, you can ensure that the movie recommendation system operates smoothly when all components are combined and that it can handle complex scenarios and interactions.