Practical – 08

Designing Test Suites

Project Title: Movie Recommendation System

---

Aim of the Experiment

To design test cases for the Movie Recommendation System that verify its core functionalities and ensure reliability using different software testing techniques.

---

Introduction

In this experiment, we focus on ensuring the accuracy and effectiveness of our Movie Recommendation System by using software testing strategies. Testing helps in identifying bugs, validating user expectations, and improving software quality. For this project, we perform unit testing, integration testing, and system testing to ensure that all modules such as login, recommendation engine, and movie database interaction function as expected.

---

Objectives

After completing this experiment, you will be able to:

Understand and apply different software testing techniques.

Design unit and integration test cases for key modules.

Identify and fix bugs in the Movie Recommendation System.

Create a test suite that validates system accuracy.

---

Theory

Software Testing

Software testing is the process of evaluating a software system to detect differences between given and expected output. It ensures that the system performs as intended.

Verification and Validation

Verification: Ensures the product is built correctly (e.g., code review, static analysis).

Validation: Ensures the right product is built, meeting user needs (e.g., functional testing).

Standards for Software Test Documentation

Test Plan

Test Case ID

Preconditions

Input Data

Expected Output

Actual Output

Pass/Fail Result

Testing Frameworks

Popular frameworks include:

JUnit for Java-based testing.

PyTest or Unittest for Python.

Selenium for automated UI testing (if applicable).

Need for Software Testing

Ensures functionality is working as expected.

Detects bugs and helps maintain code quality.

Prevents failure in production.

Helps validate user satisfaction.

---

Test Cases and Test Suite for Movie Recommendation System

---

Types of Software Testing

1. Unit Testing

Tests individual functions like login check or recommendation logic.

Example: Testing if the validateUser() function works properly.

2. Integration Testing

Ensures that components like Login → RecommendationEngine → MovieDatabase work seamlessly together.

3. System Testing

Tests the entire system flow: login, fetch recommendation, display movies.

Validates that the system behaves correctly for end-to-end user scenarios.