Faculty of Engineering

Ain Shams University

Black Box Testing Report

[CSE337s] Software Testing

Team 7

Youssef Medhat Mahmoud	2200626
Mostafa Al Hassan Salah	2200747
Mario Milad Helmy	2200540
Ahmed Sayed Abdullah	2200737
Ibrahim Mahmoud Ibrahim	2200182
Omar Ahmed Mohamed	2200267
Seif Eldin Mustafa Abdel Fattah	2200794
Ahmed Saeed Abd Elhamid	2200689

1. Introduction

This document provides black box test documentation. The testing strategy used are Equivalence Partitioning (EP), Boundary Value Analysis (BVA), and Decision Table testing.

2. Database Class Testing

2.1 Database Equivalence Partitioning

The following tests validate the Database class functionality using Equivalence Partitioning and Boundary Value Analysis techniques.

Scen #	Scenario Description	Req #	Test Data	Test Conditions/Steps	Expected Results	Post-Conditions	Actual Results	Pass/Fail
BB-1	Empty database initialization	EP1	new DataBase()	Create new DataBase instance	moviesDataBase and usersDataBase initialized as empty lists (size = 0)	Empty lists created	Lists initialized with size 0	**Pass**
BB-2	Insert 1 movie (boundary minimum)	BV1	Movie(id="TDK001", name="The Dark Knight")	Insert single movie into empty database	Database size should be 1, movie ID matches inserted movie	1 movie in database	size=1, id="TDK001"	**Pass**
BB-3	Insert multiple movies	EP2	Movie1(id="TDK001"), Movie2(id="INC002")	Insert 2 movies into database	Database size should be 2	2 movies in database	size=2	**Pass**
BB-4	Insert 100 movies (large quantity)	BV2	100 Movies (id="M000" to "M099")	Insert 100 movies in a loop	Database size should be 100	100 movies in database	size=100	**Pass**
BB-5	Set empty movie database	EP3	Empty ArrayList	setMoviesDataBase(empty list)	Database size should be 0	Empty database	size=0	**Pass**

3. ReadMovie Class Testing

3.1 Overview

The ReadMovie class testing uses Equivalence Class Partitioning and Decision Table techniques to validate movie file reading functionality. Tests cover file existence, data format validation, and required field verification.

3.2 Test Cases

Test ID	Test Name	Techniq ue	Test Data	Test Steps	Expected Result	Test File	Actual Result	Status
RM-1	ECP_File NotFou nd	ECP	Non-existent file: "wrong_path.txt"	Create ReadMovie with invalid path, call getMovies()	Throws FileNotFoundExcept ion	N/A	FileNotFoundEx ception	**Pass**
RM-2	ECP_Inv alidMov ieFile	ECP	File with invalid format: blackBoxInvalidM ovies.txt	Read file with malformed data	Throws IllegalArgumentExce ption	blackBoxInva lidMovies.txt	IllegalArgument Exception	**Pass**
RM-3	ECP_Val idMovie File	ECP	Valid file: movies.txt	Read properly formatted movie file	Successfully reads movies without exception	movies.txt	No exception	**Pass**
RM-4	testMiss ingNam e	Decision Table	Movie with missing name field (starts with comma)	Read file with movie name missing	Throws IllegalArgumentExce ption (name required)	BlackBoxMis singName.tx t	IllegalArgument Exception	**Pass**
RM-5	testMiss ingId	Decision Table	Movie with name but missing ID	Read file with movie ID missing	Throws IllegalArgumentExce ption (ID required)	BlackBoxMo vieWithMissi ngId.txt	IllegalArgument Exception	**Pass**
RM-6	testMiss ingGenr es	Decision Table	Movie with name and ID but missing genres	Read file with genre information missing	Throws IllegalArgumentExce ption (genres required)	BlackBoxMo vieWithMissi ngGener.txt	IllegalArgument Exception	**Pass**

4. ReadUser Class Testing

4.1 Overview

The ReadUser class testing validates user file reading functionality using Equivalence Class Partitioning and Decision Table techniques. Tests verify file handling, data validation, and required field enforcement.

4.2 Test Cases

Test ID	Test Name	Technique	Test Data	Test Steps	Expected Result	Test File	Actual Result	Status
RU-1	ECP_ValidUsers File	ECP	Valid file: users.txt	Read properly formatted user file, access size	Successfully reads users, size() operation completes	users.txt	No exception	**Pass**
RU-2	ECP_FileNotFound	ECP	Non-existent file: "not_exist.txt"	Create ReadUser with invalid path, call getUsers()	Throws FileNotFoundException	N/A	FileNotFoundException	**Pass**
RU-3	testFakeUserFile	ECP	Invalid file path: "fake.txt"	Attempt to read non-existent file	Throws FileNotFoundException	N/A	FileNotFoundException	**Pass**
RU-4	testMissingName	Decision Table	User with ID but missing name	Read file with user name missing	Throws IllegalArgumentException (name required)	BlackBoxUserMissingName.txt	IllegalArgumentException	**Pass**
RU-5	testMissingUserId	Decision Table	User with name but missing ID	Read file with user ID missing	Throws IllegalArgumentException (ID required)	BlackBoxUserWithMissingId.txt	IllegalArgumentException	**Pass**
RU-6	testUserWithoutMovies	Decision Table	User with name and ID but no movies	Read file with user movies missing	Throws IllegalArgumentException (movies required)	BlackBoxUserWithMissingMovies.txt	IllegalArgumentException	**Pass**

5. WriteFile Class Testing

5.1 Overview

The WriteFile class testing uses Equivalence Class Partitioning to validate file writing functionality. Tests focus on constructor behavior with various input sizes and data combinations.

5.2 Test Cases

Test ID	Test Name	Technique	Test Data	Test Steps	Expected Result	Post-Conditions	Actual Result	Status
WF-1	testMinUserListSize	ECP	Empty user list, empty movie list	Create WriteFile with empty collections	Constructor executes without exception	Minimal input boundary	No exception	**Pass**
WF-2	testMinMovieListsSize	ECP	One user (ID: "123456789", Name: "TestUser") with 1 searched and 1 recommended movie	Create WriteFile with single user and minimal movie data	Constructor handles single user successfully	Minimal valid data	No exception	**Pass**
WF-3	testMultipleValidUsers	ECP	Two users: TestUser (with movies) and AnotherUser (ID: "987654321")	Create WriteFile with multiple users	Constructor processes multiple users correctly	Multiple users scenario	No exception	**Pass**

6.ValidateUser class testing

6.1 overview

UserValidator tests are divided into two separate test classes to improve clarity and coverage. One class focuses on validating user names according to naming rules, while the other verifies user IDs, ensuring correct structure, length constraints, allowed character composition, and uniqueness.

6.2 test cases (User Name tests)

Test ID	Test Name	Technique	Test Data	Test Steps	Expected Result	Post-Conditions	Actual Result	Status
UN-1	testValidUserName_SingleWord	EP	User Name: "Ahmed"	Validate single word name	Returns true	No state change	Returns true	Pass
UN-2	testValidUserName_TwoWords	EP	User Name: "Ahmed Sayed"	Validate two word name	Returns true	No state change	Returns true	Pass
UN-3	testValidUserName_ThreeWords	EP	User Name: "Ahmed Ali Sayed"	Validate three word name	Returns true	No state change	Returns true	Pass
UN-4	testValidUserName_MultipleWords	EP	User Name: "Muhammad Ahmed Ibrahim Hassan Saeed"	Validate multiple words (5+)	Returns true	No state change	Returns true	Pass
UN-5	testValidUserName_SingleSpaceBetween	EP	User Name: "John Doe"	Validate name with single space between words	Returns true	No state change	Returns true	Pass
UN-6	testValidUserName_Uppercase	EP	User Name: "AHMED SAEED"	Validate uppercase name	Returns true	No state change	Returns true	Pass
UN-7	testValidUserName_Lowercase	EP	User Name: "ahmed saeed"	Validate lowercase name	Returns true	No state change	Returns true	Pass
UN-8	testValidUserName_MixedCase	EP	User Name: "Ahmed SAEED"	Validate mixed upper and lower case name	Returns true	No state change	Returns true	Pass
UN-9	testValidUserName_AlternatingCase	EP	User Name: "aHmEd SaEeD"	Validate alternating case	Returns true	No state change	Returns true	Pass
UN-10	testValidUserName_SingleChar	BVA	User Name: "A"	Validate single character (minimum length)	Returns true	No state change	Returns true	Pass

UN-11	testValidUserName_VeryLongName	BVA	User Name: "Muhammad Ahmed Ibrahim Hassan Mahmoud Ali Omar Khalid"	Validate very long name (50+ chars)	Returns true	No state change	Returns true	Pass
UN-12	testInvalidUserName_Null	EP	User Name: null	Validate null user name	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-13	testInvalidUserName_Empty	EP	User Name: ""	Validate empty string	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-14	testInvalidUserName_OnlySpaces	EP	User Name: " "	Validate only multiple spaces	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-15	testInvalidUserName_LeadingSpace	EP	User Name: " Ahmed"	Validate single leading space	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-16	testInvalidUserName_MultipleLeadingSpaces	EP	User Name: " Ahmed"	Validate multiple leading spaces	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-17	testInvalidUserName_LeadingTab	EP	User Name: "\tAhmed"	Validate tab at start	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-18	testInvalidUserName_DigitAtEnd	EP	User Name: "Ahmed1"	Validate single digit at end	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-19	testInvalidUserName_DigitAtStart	EP	User Name: "1Ahmed"	Validate single digit at beginning	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-20	testInvalidUserName_DigitInMiddle	EP	User Name: "Ah3med"	Validate digit in middle	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-21	testInvalidUserName_MultipleDigits	EP	User Name: "Ahmed123"	Validate multiple digits	Throws IllegalAr	No state change;	Throws IllegalArg	Pass

					gumentException	exception thrown	umentException	
UN-22	testInvalidUserName_DigitsBetweenWords	EP	User Name: "Ahmed 123 Saeed"	Validate digits between words	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-23	testInvalidUserName_OnlyDigits	EP	User Name: "123456"	Validate only digits	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-24	testInvalidUserName_AtSign	EP	User Name: "Ahmed@Saeed"	Validate at sign (@)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-25	testInvalidUserName_Hash	EP	User Name: "Ahmed#Saeed"	Validate hash (#)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-26	testInvalidUserName_DollarSign	EP	User Name: "Ahmed$Saeed"	Validate dollar sign ($)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-27	testInvalidUserName_Percent	EP	User Name: "Ahmed%Saeed"	Validate percent (%)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-28	testInvalidUserName_Ampersand	EP	User Name: "Ahmed&Saeed"	Validate ampersand (&)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-29	testInvalidUserName_Asterisk	EP	User Name: "Ahmed*Saeed"	Validate asterisk (*)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-30	testInvalidUserName_Plus	EP	User Name: "Ahmed+Saeed"	Validate plus (+)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-31	testInvalidUserName_Equals	EP	User Name: "Ahmed=Saeed"	Validate equals (=)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-32	testInvalidUserName_	EP	User Name:	Validate	Throws IllegalAr	No state change;	Throws IllegalArg	Pass

	Underscore		"Ahmed_Saeed"	underscore (_)	gumentException	exception thrown	umentException	
UN-33	testInvalidUserName_Hyphen	EP	User Name: "Ahmed-Saeed"	Validate hyphen/dash (-)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-34	testInvalidUserName_Period	EP	User Name: "Ahmed.Saeed"	Validate period/dot (.)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-35	testInvalidUserName_Comma	EP	User Name: "Ahmed,Saeed"	Validate comma (,)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-36	testInvalidUserName_Semicolon	EP	User Name: "Ahmed;Saeed"	Validate semicolon (;)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-37	testInvalidUserName_Colon	EP	User Name: "Ahmed:Saeed"	Validate colon (:)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-38	testInvalidUserName_Quote	EP	User Name: "Ahmed"Saeed"	Validate quote (")	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-39	testInvalidUserName_Apostrophe	EP	User Name: "Ahmed'Saeed"	Validate apostrophe (')	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-40	testInvalidUserName_ForwardSlash	EP	User Name: "Ahmed/Saeed"	Validate forward slash (/)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-41	testInvalidUserName_Backslash	EP	User Name: "Ahmed\Saeed"	Validate backslash (\)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-42	testInvalidUserName_Pipe	EP	User Name: "Ahmed|Saeed"	Validate pipe (|)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-43	testInvalidUserName_	EP	User Name:	Validate brackets	Throws IllegalAr	No state change;	Throws IllegalArg	Pass

	Brackets		"Ahmed[Saeed]"	([])	gumentException	exception thrown	umentException	
UN-44	testInvalidUserName_Parentheses	EP	User Name: "Ahmed(Saeed)"	Validate parentheses	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-45	testInvalidUserName_Exclamation	EP	User Name: "Ahmed!Saeed"	Validate exclamation (!)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-46	testInvalidUserName_QuestionMark	EP	User Name: "Ahmed?Saeed"	Validate question mark (?)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-47	testInvalidUserName_NumbersAndSpecialChars	EP	User Name: "Ahmed123@#$"	Validate numbers and special characters	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-48	testInvalidUserName_SpecialCharAtStart	EP	User Name: "@Ahmed"	Validate special character at start	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-49	testInvalidUserName_SpecialCharAtEnd	EP	User Name: "Ahmed!"	Validate special character at end	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UN-50	testValidUserName_MultipleSpacesBetween	BVA	User Name: "Ahmed Saeed"	Validate multiple spaces between words	Returns true	No state change	Returns true	Pass
UN-51	testValidUserName_TrailingSpace	BVA	User Name: "Ahmed Saeed "	Validate trailing space	Returns true	No state change	Returns true	Pass
UN-52	testValidUserName_MultipleTrailingSpaces	BVA	User Name: "Ahmed Saeed "	Validate multiple trailing spaces	Returns true	No state change	Returns true	Pass
UN-53	testValidUserName_RepeatedLetter	EP	User Name: "AAAA"	Validate single letter repeated	Returns true	No state change	Returns true	Pass
UN-54	testValidUserName_RepeatedWord	EP	User Name: "Ahmed Ahmed Ahmed"	Validate all same word repeated	Returns true	No state change	Returns true	Pass

6.3 User Id test cases

Test ID	Test Name	Technique	Test Data	Test Steps	Expected Result	Post-Conditions	Actual Result	Status
UID-1	testValidUserId_AllDigits	EP	User ID: "122456789"	Validate user ID with all 9 digits	Returns true	No state change	Returns true	Pass
UID-2	testValidUserId_EightDigitsOneLetter	EP	User ID: "12345678A"	Validate user ID with 8 digits and 1 uppercase letter at end	Returns true	No state change	Returns true	Pass
UID-3	testValidUserId_EightDigitsLowercaseLetter	EP	User ID: "12345678z"	Validate user ID with 8 digits and lowercase letter at end	Returns true	No state change	Returns true	Pass
UID-4	testValidUserId_StartingWithZero	EP	User ID: "012345678"	Validate user ID starting with 0	Returns true	No state change	Returns true	Pass
UID-5	testValidUserId_AllZeros	EP	User ID: "000000000"	Validate user ID with all zeros	Returns true	No state change	Returns true	Pass
UID-6	testValidUserId_ZerosWithLetter	EP	User ID: "00000000B"	Validate user ID with 8 zeros and letter	Returns true	No state change	Returns true	Pass
UID-7	testValidUserId_MixedDigitsUppercase	EP	User ID: "98765432Z"	Validate user ID with mixed digits and uppercase at end	Returns true	No state change	Returns true	Pass
UID-8	testValidUserId_ExactlyNineChars	BVA	User ID: "123456789"	Validate user ID with exactly 9 characters	Returns true	No state change	Returns true	Pass
UID-9	testInvalidUserId_EightChars	BVA	User ID: "12345678"	Validate user ID with 8 characters (below boundary)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-10	testInvalidUserId_TenChars	BVA	User ID: "1234567890"	Validate user ID with 10 characters (above boundary)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-11	testInvalidUserId_OneChar	BVA	User ID: "1"	Validate user ID with 1 character (minimum boundary)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-12	testInvalidUserId_Empty	BVA	User ID: ""	Validate empty user ID	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-13	testInvalidUserId_St	EP	User ID:	Validate user	Throws	No state	Throws	Pass

	artsWithUppercase		"A23456789"	ID starting with uppercase letter	IllegalArgumentException	change; exception thrown	IllegalArgumentException	
UID-14	testInvalidUserId_StartsWithLowercase	EP	User ID: "a23456789"	Validate user ID starting with lowercase letter	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-15	testInvalidUserId_StartsWithSpecialChar	EP	User ID: "@23456789"	Validate user ID starting with special character	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-16	testInvalidUserId_StartsWithSpace	EP	User ID: " 23456789"	Validate user ID starting with space	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-17	testInvalidUserId_SpecialCharAtSign	EP	User ID: "123@56789"	Validate user ID with @ character in middle	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-18	testInvalidUserId_SpecialCharHash	EP	User ID: "123#56789"	Validate user ID with # character in middle	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-19	testInvalidUserId_SpaceInMiddle	EP	User ID: "123 56789"	Validate user ID with space in middle	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-20	testInvalidUserId_DashInMiddle	EP	User ID: "123-56789"	Validate user ID with dash in middle	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-21	testInvalidUserId_UnderscoreChar	EP	User ID: "123_56789"	Validate user ID with underscore character	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-22	testInvalidUserId_DotChar	EP	User ID: "123.56789"	Validate user ID with dot character	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-23	testInvalidUserId_LetterInMiddle	EP	User ID: "1234A6789"	Validate user ID with letter in middle position	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-24	testInvalidUserId_LetterInSecondPosition	EP	User ID: "1A3456789"	Validate user ID with letter in second position	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-25	testInvalidUserId_T	EP	User ID:	Validate user	Throws	No state	Throws	Pass

	woLettersAtEnd		"1234567AB"	ID with two letters at end	IllegalArgumentException	change; exception thrown	IllegalArgumentException	
UID-26	testInvalidUserId_LetterNotAtEnd	EP	User ID: "12345678A9"	Validate user ID with letter at position 8 (not last)	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-27	testInvalidUserId_MultipleLettersScattered	EP	User ID: "1A3B5C7D9"	Validate user ID with multiple letters scattered	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-28	testInvalidUserId_StartWithLetter	EP	User ID: "A12345678"	Validate user ID starting with letter	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-29	testInvalidUserId_Null	EP	User ID: null	Validate null user ID	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-30	testInvalidUserId_OnlySpaces	EP	User ID: " "	Validate user ID with only spaces	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-31	testInvalidUserId_OnlyLetters	EP	User ID: "ABCDEFGHI"	Validate user ID with only letters	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-32	testInvalidUserId_AllSpecialChars	EP	User ID: "@#$%^&*()"	Validate user ID with all special characters	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-33	testInvalidUserId_Duplicate	EP	User ID: "123456789" (duplicate)	Validate duplicate user ID	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-34	testInvalidUserId_DuplicateWithLetter	EP	User ID: "12345678A" (duplicate)	Validate duplicate user ID with letter	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
UID-35	testValidUserId_CaseSensitiveUnique	EP	First: "12345678A", Second: "12345678a"	Validate case-sensitive uniqueness	Returns true	No state change	Returns true	Pass
UID-36	testValidUserId_MaxNumeric	BVA	User ID: "999999999"	Validate maximum numeric sequence	Returns true	No state change	Returns true	Pass
UID-37	testValidUserId_MinNumeric	BVA	User ID: "000000000"	Validate minimum numeric sequence	Returns true	No state change	Returns true	Pass
UID-38	testValidUserId_MaxWithZ	BVA	User ID: "99999999Z"	Validate maximum with ending Z	Returns true	No state change	Returns true	Pass
UID-39	testValidUserId_MaxWithA	BVA	User ID: "99999999A"	Validate maximum with ending A	Returns true	No state change	Returns true	Pass
UID-40	testValidUserId_SequentialDigits	EP	User ID: "123456789"	Validate sequential digits	Returns true	No state change	Returns true	Pass
UID-41	testValidUserId_Rev	EP	User ID:	Validate	Returns	No state	Returns true	Pass

	erseSequential		"987654321"	reverse sequential digits	true	change		
UID-42	testValidUserId_AlternatingDigits	EP	User ID: "101010101"	Validate alternating digits	Returns true	No state change	Returns true	Pass
UID-43	testValidUserId_RandomPattern	EP	User ID: "192837465"	Validate random valid pattern	Returns true	No state change	Returns true	Pass

7. MovieValidator tests

7.1 overview

MovieValidator tests are divided into two separate test classes to improve clarity and coverage. One class focuses on validating movie names according to naming rules, while the other verifies movie IDs, ensuring correct formatting, prefix consistency with the movie name, and adherence to numeric constraints.

7.2 test cases (Movie name tests)

Test ID	Test Name	Technique	Test Data	Test Steps	Expected Result	Post-Conditions	Actual Result	Status
MN-1	testValidMovieName_SingleCapitalWord	EP	Movie Name: "Avatar"	Validate single word movie name with capital letter	Returns true	No state change	Returns true	Pass
MN-2	testValidMovieName_MultipleCapitalWords	EP	Movie Name: "The Dark Knight"	Validate multiple words with capital letters	Returns true	No state change	Returns true	Pass
MN-3	testValidMovieName_MinimumLength	BVA	Movie Name: "A"	Validate minimum valid movie name (single character)	Returns true	No state change	Returns true	Pass
MN-4	testValidMovieName_LongName	BVA	Movie Name: "The Chronicles Of Narnia The Lion The Witch And The Wardrobe"	Validate long movie name	Returns true	No state change	Returns true	Pass
MN-5	testValidMovieName_WithNumbers	EP	Movie Name: "Oceans11"	Validate movie name with numbers	Returns true	No state change	Returns true	Pass
MN-6	testValidMovieName_WithApostrophe	EP	Movie Name: "Schindler's List"	Validate movie name with apostrophe	Returns true	No state change	Returns true	Pass
MN-7	testValidMovieName_WithHyphen	EP	Movie Name: "Spider-Man"	Validate movie name with hyphen	Returns true	No state change	Returns true	Pass
MN-8	testInvalidMovieName_Null	EP	Movie Name: null	Validate null movie name	Throws IllegalArgumentException	No state change; exception thrown	Throws IllegalArgumentException	Pass
MN-9	testInvalidMovi	EP	Movie Name:	Validate empty	Throws	No state	Throws	Pass

	eName_Empty		""	movie name	IllegalArgument Exception	change; exception thrown	IllegalArgumentException	
MN-10	testInvalidMovieName_OnlyWhitespace	EP	Movie Name: " "	Validate movie name with only whitespace	Throws IllegalArgument Exception	No state change; exception thrown	Throws IllegalArgumentException	Pass
MN-11	testInvalidMovieName_LowercaseStart	EP	Movie Name: "avatar"	Validate movie name with lowercase first letter	Throws IllegalArgument Exception	No state change; exception thrown	Throws IllegalArgumentException	Pass
MN-12	testInvalidMovieName_LowercaseSecondWord	EP	Movie Name: "The dark Knight"	Validate movie name with lowercase in second word	Throws IllegalArgument Exception	No state change; exception thrown	Throws IllegalArgumentException	Pass
MN-13	testInvalidMovieName_AllLowercase	EP	Movie Name: "the dark knight"	Validate movie name with all lowercase letters	Throws IllegalArgument Exception	No state change; exception thrown	Throws IllegalArgumentException	Pass
MN-14	testValidMovieName_AllUppercase	EP	Movie Name: "IT"	Validate movie name with all uppercase words	Returns true	No state change	Returns true	Pass
MN-15	testValidMovieName_SingleLetterWords	BVA	Movie Name: "A I"	Validate movie name with single letter words	Returns true	No state change	Returns true	Pass

7.3 test cases(Movie Id tests)

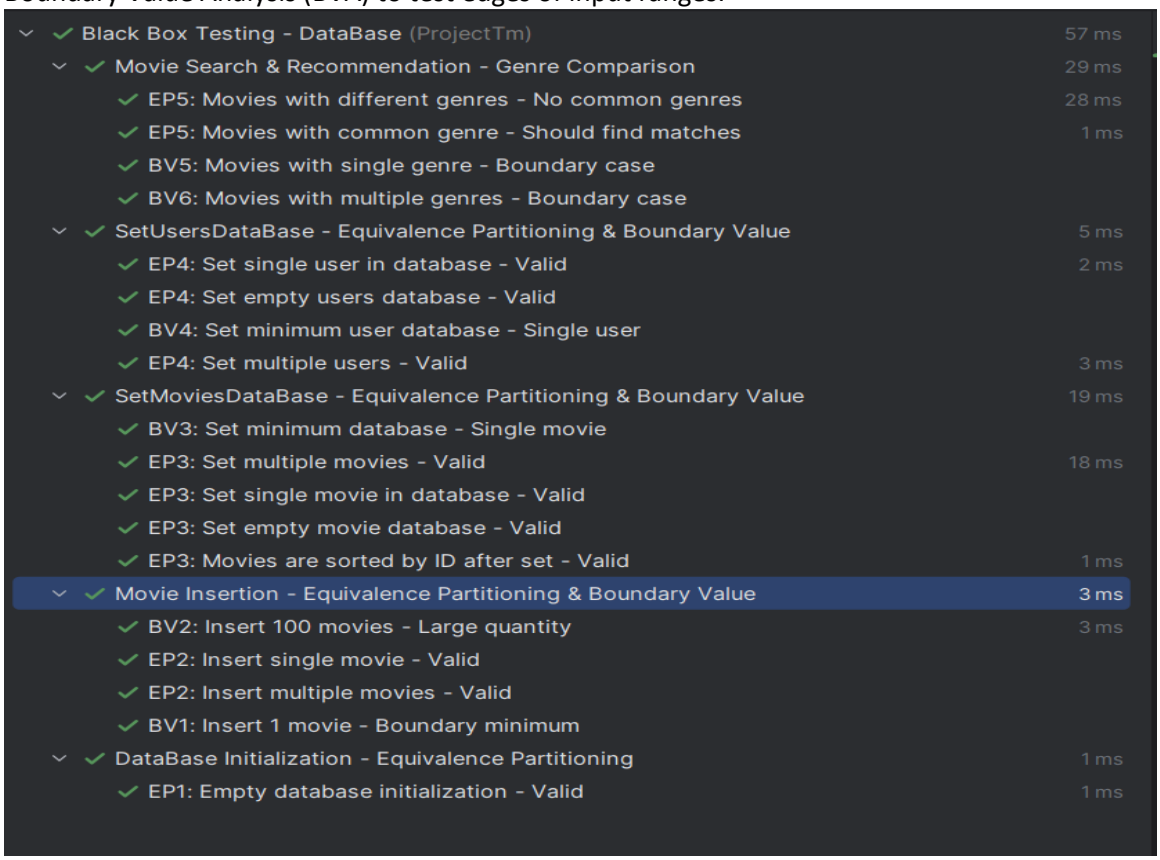
Test ID	Test Name	Technique	Test Data	Test Steps	Expected Result	Post-Conditions	Actual Result	Status
MID-1	testValidMovieId_CorrectFormat	EP	Movie ID: "TDK001", Movie Name: "The Dark Knight"	Validate movie ID with correct format	Returns true	1 movie in database	Returns true	Pass
MID-2	testValidMovieId_SingleLetterPrefix	EP	Movie ID: "A001", Movie Name: "Avatar"	Validate movie ID with single letter prefix	Returns true	1 movie in database	Returns true	Pass
MID-3	testValidMovieId_MultipleLetters	EP	Movie ID: "ITBI001", Movie Name: "Inception The Beginning Id"	Validate movie ID with multiple capital letters	Returns true	1 movie in database	Returns true	Pass
MID-4	testValidMovieId_MinimumNumeric	BVA	Movie ID: "TDK000", Movie Name: "The Dark Knight"	Validate movie ID with minimum numeric sequence (000)	Returns true	1 movie in database	Returns true	Pass
MID-5	testValidMovieId_MaximumNumeric	BVA	Movie ID: "TDK999", Movie Name: "The Dark Knight"	Validate movie ID with maximum numeric sequence (999)	Returns true	1 movie in database	Returns true	Pass
MID-6	testValidMovieId_LeadingZeros	BVA	Movie ID: "TDK001", Movie Name: "The Dark Knight"	Validate movie ID with leading zeros in numeric part	Returns true	1 movie in database	Returns true	Pass
MID-7	testValidMovieId_LongPrefix	EP	Movie ID: "TCOTNLTWTW001", Movie Name: "The Chronicles Of Narnia The Lion The Witch The Wardrobe"	Validate movie ID with very long prefix	Returns true	1 movie in database	Returns true	Pass
MID-8	testInvalidMovieId_Null	EP	Movie ID: null, Movie Name: "The Dark Knight"	Validate null movie ID	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-9	testInvalidMovieId_Empty	EP	Movie ID: "", Movie Name: "The Dark Knight"	Validate empty movie ID	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-10	testInvalidMovieId_WrongPrefix	EP	Movie ID: "XYZ001", Movie Name: "The Dark Knight"	Validate movie ID with wrong prefix letters	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-11	testInvalidMovieId_NoPrefix	EP	Movie ID: "001", Movie Name: "Avatar"	Validate movie ID with only numeric part, no prefix	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-12	testInvalidMovieId_PartialPrefix	EP	Movie ID: "TD001", Movie Name: "The Dark Knight"	Validate movie ID with partial prefix match	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-13	testInvalidMovieId_ExtraPrefix	EP	Movie ID: "TDKX001", Movie Name: "The Dark Knight"	Validate movie ID with extra letters in prefix	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-14	testInvalidMovieId_TooFewNumbers	BVA	Movie ID: "TDK01", Movie Name: "The Dark Knight"	Validate movie ID with less than 3 numeric digits	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass

MID-15	testInvalidMovieId_TooManyNumbers	BVA	Movie ID: "TDK0001", Movie Name: "The Dark Knight"	Validate movie ID with more than 3 numeric digits	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-16	testInvalidMovieId_NonNumericSuffix	EP	Movie ID: "TDKABC", Movie Name: "The Dark Knight"	Validate movie ID with non-numeric suffix	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-17	testInvalidMovieId_MixedNumericPart	EP	Movie ID: "TDK01A", Movie Name: "The Dark Knight"	Validate movie ID with mixed alphanumeric in numeric part	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-18	testInvalidMovieId_AddedNumericPart	EP	Movie ID: "TDK001A", Movie Name: "The Dark Knight"	Validate movie ID with added alphanumeric after numeric part	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-19	testInvalidMovieId_DuplicateNumeric	EP	Movie ID: "TDK001", Movie Name: "The Dark Knight" (duplicate)	Validate duplicate movie ID with same prefix and number	Throws IllegalArgumentException	1 movie in database (first entry only)	Throws IllegalArgumentException	Pass
MID-20	testInvalidMovieId_DuplicateNumericDifferentMovie	EP	First: "TDK001" / "The Dark Knight", Second: "A001" / "Avatar"	Validate movie ID with duplicate numeric part across different prefixes	Throws IllegalArgumentException	1 movie in database (first entry only)	Throws IllegalArgumentException	Pass
MID-21	testInvalidMovieId_DuplicateLetters	EP	First: "TDK001" / "The Dark Knight", Second: "TDK002" / "The Deep Knight"	Validate movie ID with duplicate prefix letters	Throws IllegalArgumentException	1 movie in database (first entry only)	Throws IllegalArgumentException	Pass
MID-22	testInvalidMovieId_LowercasePrefix	EP	Movie ID: "Tdk001", Movie Name: "The Dark Knight"	Validate movie ID with lowercase letters in prefix	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-23	testInvalidMovieId_SpecialCharacters	EP	Movie ID: "TDK@01", Movie Name: "The Dark Knight"	Validate movie ID with special characters	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-24	testInvalidMovieId_SpacesInId	EP	Movie ID: "TDK 001", Movie Name: "The Dark Knight"	Validate movie ID with spaces	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-25	testInvalidMovieId_NegativeNumbers	EP	Movie ID: "TDK-01", Movie Name: "The Dark Knight"	Validate movie ID with negative numbers	Throws IllegalArgumentException	Empty database	Throws IllegalArgumentException	Pass
MID-26	testValidMovieId_RepeatedDigits	BVA	Movie ID: "TDK111", Movie Name: "The Dark Knight"	Validate movie ID with all same digits in numeric part	Returns true	1 movie in database	Returns true	Pass
MID-27	testValidMovieId_SequentialNumbers	BVA	Movie ID: "TDK123", Movie Name: "The Dark Knight"	Validate movie ID with sequential numbers	Returns true	1 movie in database	Returns true	Pass

8. DataBase Class Testing

8.1 Overview

This document details the test cases designed to validate the functionality of the Database class. The strategy employs Equivalence Partitioning (EP) to cover valid/invalid classes of errors and Boundary Value Analysis (BVA) to test edges of input ranges.



8.2 Test Cases (Database Equivalence Partitioning)

Scen #	Scenario Description	Req #	Test Data	Test Conditions/ Steps	Expected Results/ Comments	Post-Conditions	Actual Results	Pass/ Fail
BB-1	Empty database initialization	EP1	new DataBase()	Create new DataBase instance	moviesDataBase and usersDataBase should be initialized as empty lists (size = 0)	Empty lists created	Lists initialized with size 0	Pass

Scen #	Scenario Description	Req #	Test Data	Test Conditions/ Steps	Expected Results/ Comments	Post-Conditions	Actual Results	Pass/ Fail
BB-2	Insert 1 movie (boundary minimum)	BV1	Movie(id="TDK001", name="The Dark Knight")	Insert single movie into empty database	Database size should be 1, movie ID matches inserted movie	1 movie in database	size=1, id="TDK001"	Pass
BB-3	Insert multiple movies	EP2	Movie1(id="TDK001"), Movie2(id="INC002")	Insert 2 movies into database	Database size should be 2	2 movies in database	size=2	Pass
BB-4	Insert 100 movies (large quantity)	BV2	100 Movies (id="M000" to "M099")	Insert 100 movies in a loop	Database size should be 100	100 movies in database	size=100	Pass
BB-5	Set empty movie database	EP3	Empty ArrayList	setMoviesDataBase(empty list)	Database size should be 0	Empty database	size=0	Pass
BB-6	Set minimum database (single movie)	BV3	ArrayList with 1 Movie(id="ZZZ999")	setMoviesDataBase(list with 1 movie)	Database size should be 1	1 movie in database	size=1	Pass
BB-7	Set multiple movies	EP3	ArrayList with 5 Movies (id="M000" to "M004")	setMoviesDataBase(list with 5 movies)	Database size should be 5	5 movies in database	size=5	Pass
BB-8	Verify sorting after set	EP3	Movies: id="CCC001", "AAA001", "BBB001"	setMoviesDataBase with unsorted movies	Movies sorted alphabetically by ID: AAA001, BBB001, CCC001	Sorted list	Order: AAA001, BBB001, CCC001	Pass
BB-9	Set empty users database	EP4	Empty ArrayList	setUsersDataBase(empty list)	Users database size should be 0	Empty users list	size=0	Pass

Scen #	Scenario Description	Req #	Test Data	Test Conditions/ Steps	Expected Results/ Comments	Post-Conditions	Actual Results	Pass/ Fail
BB-10	Set minimum user database	BV4	ArrayList with 1 User(id="99 9999999", name="Test User")	setUsersDat aBase(list with 1 user)	Users database size should be 1	1 user in database	size=1	Pass
BB-11	Set multiple users	EP4	10 Users (id="100000 000" to "100000009 ")	setUsersDat aBase(list with 10 users)	Users database size should be 10	10 users in database	size=10	Pass
BB-12	Movies with common genre	EP5	Movie1(gen re="Action", "Crime"), Movie2(gen re="Action", "Thriller")	Add movies with overlapping genres	Both movies stored successfull y (size >= 2)	Movies with shared genre stored	size=2	Pass
BB-13	Movies with different genres	EP5	Movie1(gen re="Action") , Movie2(gen re="Romanc e")	Add movies with no overlapping genres	Both movies stored, no overlap	2 distinct movies stored	size=2	Pass
BB-14	Movies with single genre (boundary)	BV5	Movie1(gen re="Action") , Movie2(gen re="Action")	Add movies with exactly 1 genre each	Both movies stored with single genre	2 movies with single genre	size=2	Pass
BB-15	Movies with multiple genres (boundary)	BV6	Movie1(gen re="Action", "Sci-Fi","Adventu re"), Movie2(gen re="Advent ure","Fantas y")	Add movies with 3 and 2 genres respectively	Both movies stored with multiple genres	2 movies with multiple genres	size=2	Pass

8.3 Test Cases (Boundary Value Analysis)

8.3.1. Data Initialization

Scen #	Description	Test Conditions/Steps	Expected Results	Pass/Fail
BB-1	Empty database initialization	Create a new database instance	moviesDataBase and usersDataBase initialized as empty lists (size = 0).	Pass

8.3.2. Data Insertion (Boundaries)

a. Insert Movies

Scen #	Description	Test Conditions/Steps	Expected Results	Pass/Fail
BB-2	Insert 1 movie (min boundary)	Insert a single movie into database	Movie inserted successfully, database size = 1, movie ID matches (e.g., TDK001).	Pass
BB-3	Insert multiple movies	Insert 2 movies into database	Movie database size = 2.	Pass
BB-4	Insert 100 movies (large quantity)	Insert 100 movies using a loop	Movie database size = 100.	Pass

b. Insert Users

Scen #	Description	Test Conditions/Steps	Expected Results	Pass/Fail
BB-10	Set minimum user database	Add 1 user to list	User successfully added, database size = 1.	Pass
BB-11	Set multiple users	Add 10 users to list	Database size = 10.	Pass

8.3.3. Setting Database

a. Set Movie Database

Scen #	Description	Test Conditions/Steps	Expected Results	Pass/Fail
BB-5	Set empty movie database	Set movie database to []	Movie database size = 0.	Pass
BB-6	Set 1 movie	Set database to a list with 1 movie	Movie database size = 1.	Pass
BB-7	Set 5 movies	Set database to a list with 5 movies	Movie database size = 5.	Pass
BB-8	Verify sorting after set	Set database with unsorted movies	Movies sorted alphabetically by IDs (e.g., AAA001, BBB001, ...).	Pass

b. Set User Database

Scen #	Description	Test Conditions/Steps	Expected Results	Pass/Fail
BB-9	Set empty users database	Set users database to empty list ([])	User database size = 0.	Pass

8.3.4. Special Scenarios

This category handles specific data characteristics such as overlapping genres or single/multiple genres.

Scen #	Description	Test Conditions/Steps	Expected Results	Pass/Fail
BB-12	Movies with common genre	Insert movies with overlapping genres	Both movies stored successfully, database size = 2.	Pass
BB-13	Movies with different genres	Insert movies with distinct genres	Two distinct movies stored in the database, database size = 2.	Pass
BB-14	Movies with single genre (boundary)	Insert movies with exactly 1 genre each	Both movies with single genre stored successfully.	Pass
BB-15	Movies with multiple genres	Insert movies with 2-3 genres each	Both movies stored successfully with multiple genres.	Pass

Summary of Separated Parts:

Part	Number of Test Cases	Pass/Fail
Data Initialization	1	Pass
Data Insertion (Boundaries)	5	Pass
Setting Database	5	Pass
Special Scenarios	4	Pass