CONTRIBUTIONS:

1. SIDDHANT KUMAR(PES2201800129):RECOMMENDER SYSTEM(RECOMMENDING MOVIES BASED ON USER ATTRIBUTES)
2. SKANDAN KA(PES2201800064):PREDICTING THE MOST LIKED VIDEOS AND RECENTLY VIEWED
3. SAMEER PRASAD SUBHEDAR(PES2201800323): PREDICTING MOST PLAYED VIDEOS

Object Oriented Analysis and Design with Software Engineering Laboratory

Subject Code: UE18CS355

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
1.	Recommender System	Recommend Movies based on User Behaviour(Collaborative Filtering)	The movie to be recommended or should be present in the dataset	1.We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast and more preferable than pearson coefficient. 2.We first check if the movie name input is in the database and if it is we use our	Movie:" Memento "	Movies Similar to "Memento" should be displayed(recommended)	Movies Similar to "Memento " should be displayed(recommended)	PASS

				recommendation system to **find similar movies** and sort them based on their similarity distance and output only the **top 10** movies with their distances from the input movie.				
2.	Recommender System	Recommend Movies based on User Behaviour(Collaborative Filtering)	The movie to be recommended or should be present in the dataset	1.We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast and more preferable than pearson coefficient. 2.We first check if the movie name input is in the database and if it is we use our recommendation system to **find similar movies** and sort them based on their similarity distance and output only the **top 10** movies with their distances from the input movie.	Movie:"Toy Story"	Movies Similar to "Toy Story" should be displayed(recommended)	Movies Similar to "Toy Story" should be displayed(recommended)	PASS

3.	Recommender System	Recommend Movies based on User Behaviour(Collaborative Filtering)	The movie to be recommended or should be present in the dataset	1.We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast and more preferable than pearson coefficient. 2.We first check if the movie name input is in the database and if it is we use our recommendation system to **find similar movies** and sort them based on their similarity distance and output only the **top 10** movies with their distances from the input movie.	Movie:" Dusk Till Dawn"	Movies Similar to "Dusk Till Dawn" should be displayed(recommended)	Movies Similar to "Dusk Till Dawn" should be displayed(recommended)	PASS
4.	Recommender System	Recommend Movies based on User Behaviour(Collaborative Filtering)	The movie to be recommended or should be present in the dataset	1.We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast and more preferable than pearson coefficient.	Movie:" Wonder Woman"	Movies Similar to "Wonder Woman" should be displayed(recommended)	Movies Similar to "Wonder Woman" should be displayed(recommended)	PASS

				2.We first check if the movie name input is in the database and if it is we use our recommendation system to **find similar movies** and sort them based on their similarity distance and output only the **top 10** movies with their distances from the input movie.				
5.	Recomender System	Recommend Movies based on User Behaviour(Collaborative Filetring)	The movie to be recommended or should be present in the dataset	1.We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast and more preferable than pearson coefficient. 2.We first check if the movie name input is in the database and if it is we use our recommendation system to **find similar movies** and sort them based on their similarity distance and	Movie:"IP man 3"	Movies Similar to "IP man 3" should be displayed(recommended)	Movies Similar to "IP man 3" should be displayed(recommended)	PASS

				output only the **top 10** movies with their distances from the input movie.				
6.	Recommender System	Recommend Movies based on User Behaviour(Collaborative Filtering)	The movie to be recommended or should be present in the dataset	1.We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast and more preferable than pearson coefficient. 2.We first check if the movie name input is in the database and if it is we use our recommendation system to **find similar movies** and sort them based on their similarity distance and output only the **top 10** movies with their distances from the input movie.	Movie:" Usual Suspects"	Movies Similar to "Usual Suspects" should be displayed(recommended)	Movies Similar to "Usual Suspects" should be displayed(recommended)	PASS
7.	Recommender System	Recommend Movies based on User Behaviour(Collaborative Filtering)	The movie to be recommended or should be present in the dataset	1.We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast	Movie:"Bottle Rocket"	Movies Similar to "Bottle Rocket" should be displayed(recommended)	Movies Similar to "Bottle Rocket" should be displayed(recommended)	PASS

				and more preferable than pearson coefficient. 2.We first check if the movie name input is in the database and if it is we use our recommendation system to **find similar movies** and sort them based on their similarity distance and output only the **top 10** movies with their distances from the input movie.				
8.	Recommender System	Recommend Movies based on User Behaviour(Collaborative Filtering)	The movie to be recommended or should be present in the dataset	1.We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast and more preferable than pearson coefficient. 2.We first check if the movie name input is in the database and if it is we use our recommendation system to **find similar**	Movie:"Shazam"	Movies Similar to "Shazam" should be displayed(recommended)	Movies Similar to "Shazam" will not be displayed(recommended) as this movie isn't there in the dataset.	FAIL

				movies and sort them based on their similarity distance and output only the **top 10** movies with their distances from the input movie.				
9.	Recommender System	Recommend Movies based on User Behaviour(Collaborative Filtering)	The movie to be recommended or should be present in the dataset	1.We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast and more preferable than pearson coefficient. 2.We first check if the movie name input is in the database and if it is we use our recommendation system to **find similar movies** and sort them based on their similarity distance and output only the **top 10** movies with their distances from the input movie.	Movie:" Kung Fu Panda"	Movies Similar to "Kung Fu Panda" should be displayed(recommended)	Movies Similar to "Kung Fu Panda" will not be displayed(recommended) as this movie isn't there.	FAIL

10.	Recommender System	Recommend Movies based on User Behaviour(Collaborative Filtering)	The movie to be recommended or should be present in the dataset	1.We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast and more preferable than pearson coefficient. 2.We first check if the movie name input is in the database and if it is we use our recommendation system to **find similar movies** and sort them based on their similarity distance and output only the **top 10** movies with their distances from the input movie.	Movie:" Avengers End Game"	Movies Similar to "Avengers End Game" should be displayed(recommended)	Movies Similar to "Avengers EndGame" will not be displayed(recommended) as this movie is not in the dataset.	FAIL

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
1.	Recently Viewed	Displays most recently viewed	Movie should be in the dataset used	1.Import the Dataset. 2.Sort based on Timestamp. 3.Display the required	data.head(1)	Toy Story	Toy Story	PASS

				Data				
2.	Recently Viewed	Displays 3 most recently viewed	Movie should be in the dataset used	1.Import the Dataset. 2.Sort based on Timestamp. 3.Display the required Data	data.head(3)	Toy Story Grumpier Old Men Usual Suspects	Toy Story Grumpier Old Men Usual Suspects	PASS
3.	Recently Viewed	Displays 5 most recently viewed	Movie should be in the dataset used	1.Import the Dataset. 2.Sort based on Timestamp. 3.Display the required Data	data.head(5)	Toy Story Grumpier Old Men Usual Suspects Dusk Till Dawn Bottle Rocket	Toy Story Grumpier Old Men Usual Suspects Dusk Till Dawn Bottle Rocket	PASS

4.	Recently Viewed	Displays 4 most recently viewed	Movie should be in the dataset used	1.Import the Dataset. 2.Sort based on Timestamp. 3.Display the required Data	data.head(4)	Toy Story Grumpier Old Men Usual Suspects Dusk Till Dawn	Toy Story Grumpier Old Men Usual Suspects Dusk Till Dawn	PASS
5.	Recently Viewed	Displays 2 most recently viewed	Movie should be in the dataset used	1.Import the Dataset. 2.Sort based on Timestamp. 3.Display the required Data	data.head(2)	Toy Story Grumpier Old Men	Toy Story Grumpier Old Men	PASS

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
1.	Liked	Displays	Movie	Import the	data.drop_	Bill Hicks:	Bill Hicks:	PASS

		most liked movies	displayed should be from the dataset	dataset, drop the duplicates following which we sort the movies based on rating	duplicates(subset='movieId').sort_values('rating', ascending = False).head(1)	Revelations (1993)	Revelations (1993)	
2.	Liked	Displays most liked movies	Movie displayed should be from the dataset	Import the dataset, drop the duplicates following which we sort the movies based on rating	data.drop_duplicates(subset='movieId').sort_values('rating', ascending = False).head(2)	Bill Hicks: Revelations (1993) Rain (2001)	Bill Hicks: Revelations (1993) Rain (2001)	PASS
3.	Liked	Displays most liked movies	Movie displayed should be from the dataset	Import the dataset, drop the duplicates following which we sort the movies based on rating	data.drop_duplicates(subset='movieId').sort_values('rating', ascending = False).head(3)	Bill Hicks: Revelations (1993) Rain (2001) George Carlin: Life Is Worth Losing (2005)	Bill Hicks: Revelations (1993) Rain (2001) George Carlin: Life Is Worth Losing (2005)	PASS
4.	Liked	Displays most liked movies	Movie displayed should be from the dataset	Import the dataset, drop the duplicates following which we sort the movies based on rating	data.drop_duplicates(subset='movieId').sort_values('rating', ascending = False).head(4)	Bill Hicks: Revelations (1993) Rain (2001) George Carlin: Life Is Worth Losing	Bill Hicks: Revelations (1993)Your Juice in the Hood Rain (2001) George Carlin: Life Is Worth Losing (2005) What Men Talk	PASS

						(2005) What Men Talk About (2010)	About (2010)	

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result

1.	Most Played	Will display the most played videos	The videos played will be from the dataset	Import the tags dataset, make a list of tuples of the number of occurrences of a particular movie. Get the movie name from the movies dataset and map it to the number of occurrences of movieId.	for a in y:
					try:
					print('The movie',movies['title'][a[1]],' has been viewed',a[0],'times')
					except:
					
					continue
						The movie Virtuosity (1995) has been viewed 181 times	The movie Virtuosity (1995) has been viewed 181 times	PASS

2.	Most Played	Will display the most played videos	The videos played will be from the dataset	Import the tags dataset, make a list of tuples of the number of occurrences of a particular movie. Get the movie name from the movies dataset and map it to the number of occurrences of movieId.	for a in y:
					try:
					print('The movie',movies['title'][a[1]],' has been viewed',a[0],'times')
					except:
					
					continue
						The movie Billy Elliot (2000) has been viewed 54 times	The movie Billy Elliot (2000) has been viewed 54 times	PASS

3.	Most Played	Will display the most played videos	The videos played will be from the dataset	Import the tags dataset, make a list of tuples of the number of occurrences	for a in y:
					try:
					print('The movie',movies['titl
						The movie Grand Day Out with	The movie Grand Day Out with Wallace and	PASS

				of a particular movie. Get the movie name from the movies dataset and map it to the number of occurrences of movieId.	e'][a[1]],' has been viewed',a[0],'times') except: continue	Wallace and Gromit, A (1989) has been viewed 41 times	Gromit, A (1989) has been viewed 41 times	
4.	Most Played	Will display the most played videos	The videos played will be from the dataset	Import the tags dataset, make a list of tuples of the number of occurrences of a particular movie. Get the movie name from the movies dataset and map it to the number of occurrences of movieId.	for a in y: try: print('The movie',movies['title'][a[1]],' has been viewed',a[0],'times') except: continue	The movie Underneath (1995) has been viewed 35 times	The movie Underneath (1995) has been viewed 35 times	PASS
5.	Most Played	Will display the most played videos	The videos played will be from the dataset	Import the tags dataset, make a list of tuples of the number of occurrences of a particular movie. Get the movie name from the movies dataset and map it to the number of occurrences of movieId.	for a in y: try: print('The movie',movies['title'][a[1]],' has been viewed',a[0],'times') except: continue	The movie TiMER (2009) has been viewed 34 times	The movie TiMER (2009) has been viewed 34 times	PASS
6.	Most Played	Will display the most played videos	The videos played will be from the dataset	Import the tags dataset, make a list of tuples of the number of occurrences of a particular movie. Get the movie name from the movies dataset and map it to the number of	for a in y: try: print('The movie',movies['title'][a[1]],' has been viewed',a[0],'times') except: continue	The movie Tales from the Darkside: The Movie (1990) has been viewed 32 times	The movie Tales from the Darkside: The Movie (1990) has been viewed 32 times	PASS

				occurrences of movieId.				

7.	Most Played	Will display the most played videos	The videos played will be from the dataset	Import the tags dataset, make a list of tuples of the number of occurrences of a particular movie. Get the movie name from the movies dataset and map it to the number of occurrences of movieId.	for a in y:
					try:
					print('The movie',movies['title'][a[1]],' has been viewed',a[0],'times')
					except:
					
					continue
						The movie Quiz Show (1994) has been viewed 26 times	The movie Quiz Show (1994) has been viewed 26 times	PASS

8.	Most Played	Will display the most played videos	The videos played will be from the dataset	Import the tags dataset, make a list of tuples of the number of occurrences of a particular movie. Get the movie name from the movies dataset and map it to the number of occurrences of movieId.	for a in y:
					try:
					print('The movie',movies['title'][a[1]],' has been viewed',a[0],'times')
					except:
					
					continue
						The movie Tenet (2021) has been viewed 78 times	Key error	FAIL

9.	Most Played	Will display the most played videos	The videos played will be from the dataset	Import the tags dataset, make a list of tuples of the number of occurrences of a particular movie. Get the movie name from the movies dataset and map it to the number of occurrences of movieId.	for a in y:
					try:
					print('The movie',movies['title'][a[1]],' has been viewed',a[0],'times')
					except:
					
					continue
						The movie End Game (2019) has been viewed 104 times	Key error	FAIL

10.	Most Played	Will display the most played videos	The videos played will be from the dataset	Import the tags dataset, make a list of tuples of the number of occurrences of a particular	for a in y:
					try:
					print('The movie',movies['title'][a[1]],'
						The movie John Wick (2014) has been	Key error	FAIL

				movie. Get the movie name from the movies dataset and map it to the number of occurrences of movieId.	has been viewed',a [0],'times') except: continue	viewed 78 times		