Caitlyn Chau
CS 151 – 2
Professor Karra
Feb. 22, 2020

Use Cases

Use Case 1: Logging in

Step	User's Action	System's Response
1	User selects Sign [I]n	
2		System prompts user for their username
3	User enters username	
4		System prompts user for their password
5	User enters password	
6		System checks that username exists
7		System checks the corresponding value (password) for the key (username)
8		System presents menu: `[R]eserve [V]iew [C]ancel [O]ut`

Variation # 1: Incorrect credentials

1.1. Start at Step 5
1.2. User entered an incorrect username or password
1.3. Repeat Step 5 with an error message

Use Case 2: Logging out

Step	User's Action	System's Response
1	User selects E[X]it	
2		System traverses through each user in `UserList` and traverses through each `MovieTicket` of that user
3		System outputs a line for each `MovieTicket` to `reservations.txt`

Use Case 3: Signing up

Step	User's Action	System's Response
1	User selects Sign [U]p	
2		System prompts user for a unique username
3	User enters username	
4		System validates username
5		System prompts user for a password
6	User enters password	
7		System adds new key/value pair to `UserMap`
8		System creates new `User` and adds it to `UserList`
9		System writes username and password to `users.txt`
10		System creates new `Transaction` and presents main menu:

		`[R]eserve [V]iew [C]ancel [O]ut`

Variation # 1: Username exists already

1.1. Start at step 3
1.2. User enters a username that already exists
1.3. System prints out "Error: That username already exists"
1.4. Repeat Step 3

Use Case 4: Reserving

Step	User's Action	System's Response
1	User carries out **Logging in**	
2	User selects [R]eserve	
3		System prompts user for a date between 12/23/2020 and 1/2/2021
4	User enters a date	
5		System checks date is valid format and within specified dates
6		System prompts user to select a movie time `(6:30 PM or 8:30 PM)`
7	User selects a time	
8		System checks date is valid format and one of two times
9		System prompts user to select a section: `[M]ain, [W]est, [E]ast, [S]outh`
10	User selects a section	
11		System checks input is a valid section
12		System prompts user to select a seat within section's occupancy
13	User selects a seat	
14		System checks that seat is within section occupancy and is not taken
15		System creates a new `Seat` object and sets it to occupied
16		System creates a new `MovieTicket` object and adds it to users's `ReservationList`
17		System returns to menu: `[R]eserve [V]iew [C]ancel [O]ut`

Variation # 1: Invalid date format

1.1. Start at Step 3
1.2. User enters a date not in the format mm/dd/yyyy
1.3. System prints out "Error: Invalid date format"
1.4. Repeat Step 3

Variation # 2: Wrong date

2.1. Start at Step 4
2.2. User enters a date that is not between 12/23/2020 and 1/2/2021
2.3. System prints out "Error: Showings for this movie are between 12/23/2020 and 1/2/2021"
2.4. Repeat Step 4

Variation # 3: Invalid time format
3.1. Start at Step 5
3.2 User enters a time not in the format hh:mm a
3.3. System prints out “Error: Invalid time format”
3.4. Repeat Step 5
Variation # 4: Wrong time
4.1. Start at Step 5
4.2. User enters a time that is not 6:30 PM or 8:30 PM
4.3. System prints out “Error: select a showtime: 6:30 PM or 8:30 PM”
4.4. Repeat Step 5
Variation # 5: Invalid section
5.1. Start at Step 7
5.2. User enters an invalid section letter
5.3. System prints out “Error: Invalid section letter”
5.4. Repeat Step 7
Variation # 6: Seat number out of range
6.1. Start at Step 9
6.2. User enters a seat number that is out of range for the chosen section
6.4. System prints out “Error: Invalid seat number”
6.5. Repeat Step 9
Variation # 7: Seat is taken
7.1. Start at Step 9
7.2. User enters a seat number that is occupied
7.3. System checks Cinema’s open seats
7.4. System prints out “Error: Seat is taken. Select another seat.”
7.5. Repeat Step 9

Use Case 5: Viewing reservations

Step	User’s Action	System’s Response
1	User carries out **Logging in**	
2	User selects [V]iew	
3		System prints out all of user’s `MovieTickets` from `ReservationList` sorted by date, then time
4		System returns to menu: `[R]eserve [V]iew [C]ancel [O]ut`

Use Case 6: Cancelling reservations

Step	User’s Action	System’s Response
1	User selects [C]ancel	
2		System prints out all of user’s reserved `MovieTickets` and prompts user to select which one to cancel
3	User selects which ticket number they want to cancel	

4		System removes `MovieTicket` from user's `ReservationList`
5		System returns to menu: `[R]eserve [V]iew [C]ancel [O]ut`

Variation # 1: Invalid input

1.1. Start at Step 2

1.2. User enters input that is not an integer

1.3. System prints out "Error: Invalid input"

1.4. Repeat Step 2

Variation # 2: Integer not within range of number of MovieTickets

2.1. Start at Step 2

2.2. User enters an integer that is less than 1 or greater than the number of MovieTickets

2.3. System prints out " Error: Invalid ticket"

2.4. Repeat Step 2

Use Case 7: Finish reservation

Step	User's Action	System's Response
1	User carries out **Reserving**	
2	User selects [O]ut	
3		System calculates total cost of `MovieTickets` taking discounts into consideration
4		System prints out receipt with confirmation number, reserved seats, and date(s) and time(s) of movie tickets
5		System displays main menu `Sign [U]p   Sign [I]n   E[X]it`