

First Row Movies

Design Document
For Software Engineering
Version 2.0

Revision History

Date	Version	Description	Author
10/17/2012	0.1	Draft	Mitchell Verter
10/18/2012	0.3	Draft	Mitchell Verter
10/19/2012	1.0	1.0 Release	Mitchell Verter
11/09/2012	2.0	2.0 Release	Mitchell Verter, Jorge Yau

First Row Movies Founders

Name	Role
Mitchell Verter	Project Manager, coder
Jorge Yau	Chief Technology Officer, coder
Shamrat Basnet	Front-end Programmer, coder
Lovepreet Singh	Back-end Programmer, coder

Table of Contents

- 1. Introduction
 - 1.1.Overall Pictture
- 2. Overall Description
 - 2.1. Use Cases
 - 2.1.1. Use-Case Diagram
 - 2.1.2. Use-Case Descriptions
 - 2.2.Collaboration Diagrams
 - 2.3.ECS System State Diagram
- 3. ER Diagrams
 - 3.1.ER Class Diagram
 - 3.2.ER Database Diagram
 - 3.2.1. ER Class Diagrams
 - 3.2.2. ER Database Diagram
- 4. Detailed Pseudo-Code
 - 4.1.User Classes
 - 4.2.Item Classes
 - 4.3.Control Modules
 - 4.4.GUI Modules
- 5. System Screens
 - 5.1.User-Interface Screenshots

9

Design Document

1. Introduction

1.1 Overall Picture

First Row Movies (FRM) promises to be the premiere entertainment hub on the internet for all lovers of movies. FRM entices the guest into becoming a full fledged user by allowing her to search through our vast collection, browse individual titles, read movie descriptions, and expose her to the critical insights and ratings of our First Row Movies Member Community. Once she becomes a registered user, she will then be able to participate with others and submit her own commentary and ratings. She will also be able purchase movies and watch them online!

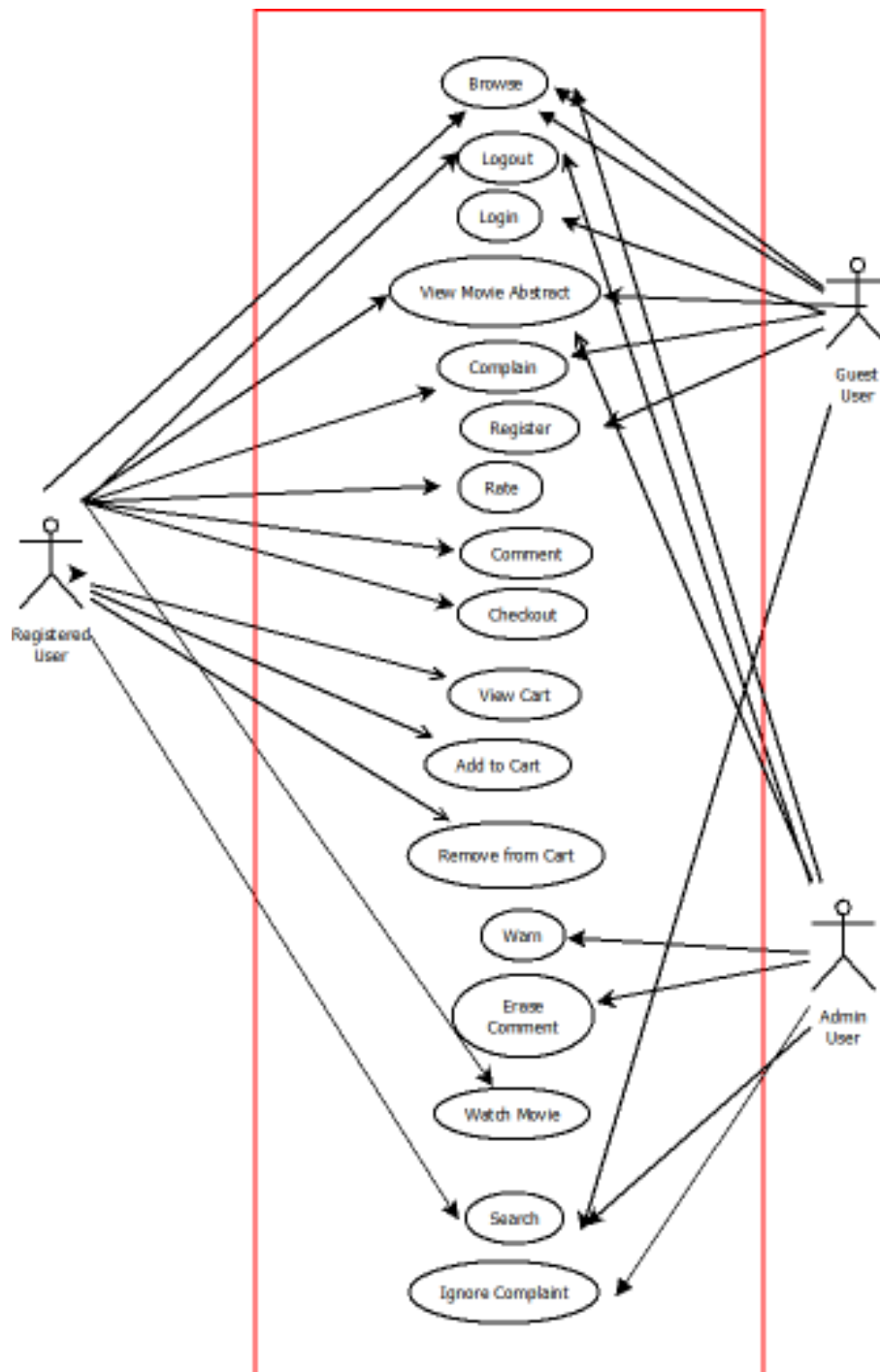
Much of the administration of FRM is already handled by the system itself: the SPAM-detection engine detects when a user is a spammer and automatically eliminates them from the system. Thus, the Administrator is responsible only for regulating the flow of conversation on the system, removing comments that have been flagged as inappropriate.

Our software is a web application and therefore it requires the appropriate software on the client end and the server end. On the client end, our software will run on any runs on any contemporary web browser that supports standard HTML, CSS 2.0, JavaScript, and PHP 5.0. It should run well on any browser available for any modern desktop / laptop computer (Internet Explorer, Firefox, Chrome, Safari, Opera, etc) or mobile device. The server requires a PHP5.0 processing daemon and a MySQL database. Any LAMP/WAMP setup should suffice, as will any other server (IIS, etc) with PHP and mySQL modules installed.

This document is meant to show the features of First Row Movies. Not only will this document help and guide developers but it will also serve as a legal document for the prospective client. This Design Document explores in detail the structure and design of First Row Movies through diagrams and intricate pseudo-code of the classes, models, and views in our system.

2. Overall Description

2.1 Use Case Diagram



2.2 Use Case Descriptions

Search: Any Guest User or Registered User may search for movies using any combination of search vectors (id, title, stars, director, year, etc.)

Browse: Any Guest User or Registered User may browse through movies by category.

View Movie Information: Any Guest User or Registered User may read the abstract (summary, id, title, stars, director, year, etc) of a movie. The user will also see comments made by Registered Users.

Complain: Any Guest User or Registered User may complain about a comment she or he deems to be inappropriate. Flagged comments will sent to the Administrator for review.

Register : Any Guest User may register in order to become a Registered User.

Login: One must login to the system to be identified as a Registered User or an Administrator. Beofore one logs in, one is considered to be a Guest User.

Watch Movie: Any Registered User may buy a movie.

Rate: Any Registered User may rate a movie. Our system will automatically analyse the rating behavior of each Registered User in order to prevent spamming.

Comment: Any Registered User may comment on a movie.

Add To Cart: Any Registered User may add an item to her or his shopping cart.

View Cart: Any Registered User may view the items in her or his cart.

Remove From Cart: Any Registered User may remove items from her or his cart.

Checkout: Any Registered User may purchase the items in her or his cart.

Erase Comment: Any Administrator may erase a comment that has been flagged for deletion.

Ignore Complaint: Any Administrator may choose to ignore a comment being flagged for deletion.

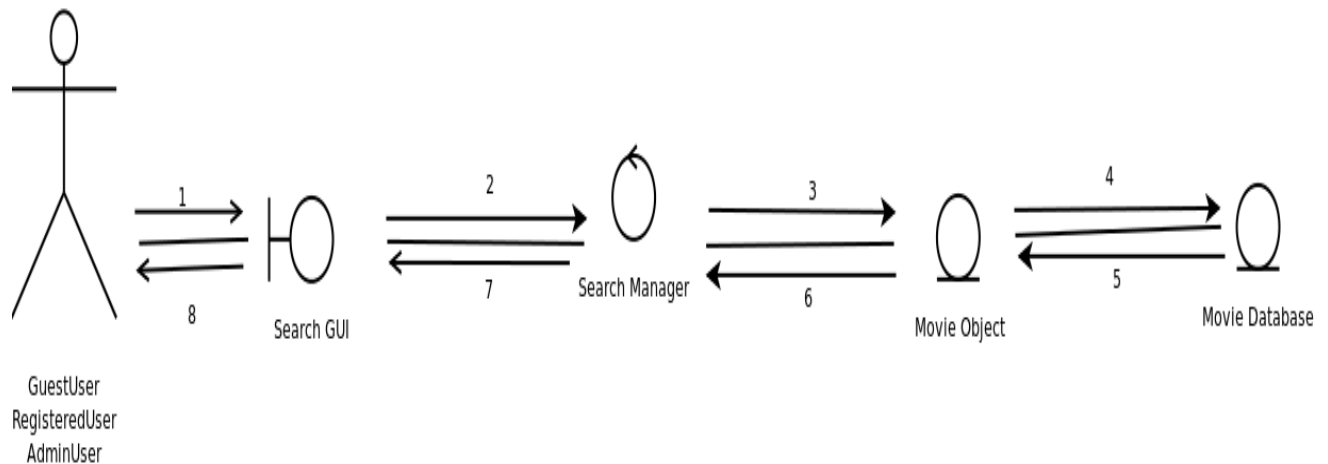
Warn: Any Administrator may warn a Registered User for writing inappropriate comments.

Logout: Any Registered User or Administrator may log out of the system.

3. Specific Requirements

3.1 Collaboration Diagrams

3.1.1 Shared Collaboration Diagrams

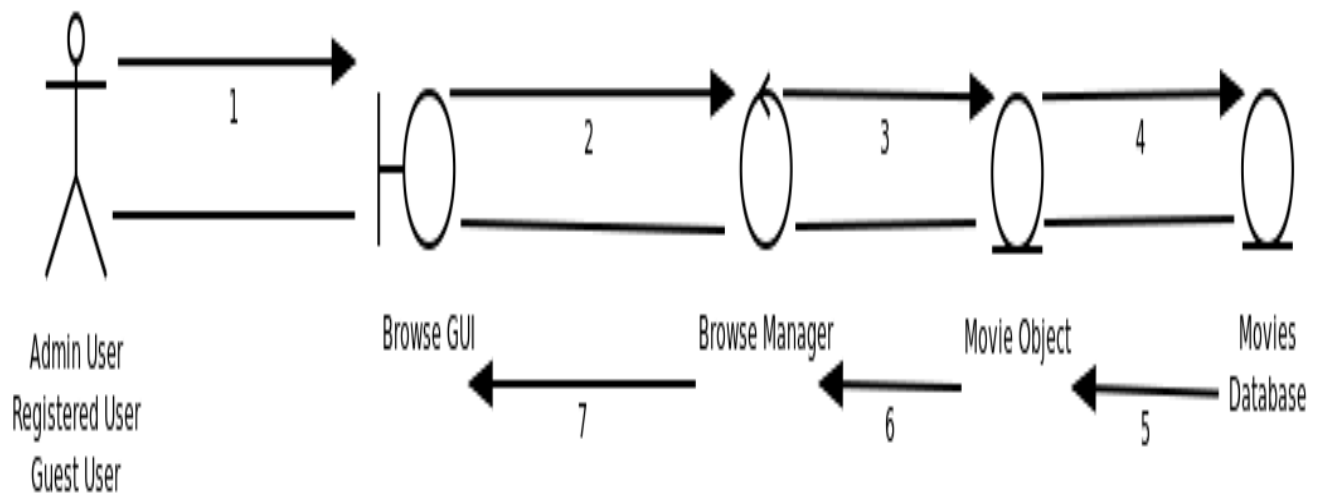


1. GuestUser, RegisteredUser, or AdminUser click on Search GUI.
2. Search GUI transfers the information to search manager.
3. Search manager passes the information to movie object.
4. Movie object verifies the search by looking into the database.
5. Movie Database passes the verified information to user object.
6. User object processes the information to search manager.
7. The information is published in Search GUI.
8. GuestUser/RegisteredUser/Visitor gets the result for searched information.

Exception

5. Search result not found
6. Empty database information sent to search manager
8. GUI displays "No Results Found"

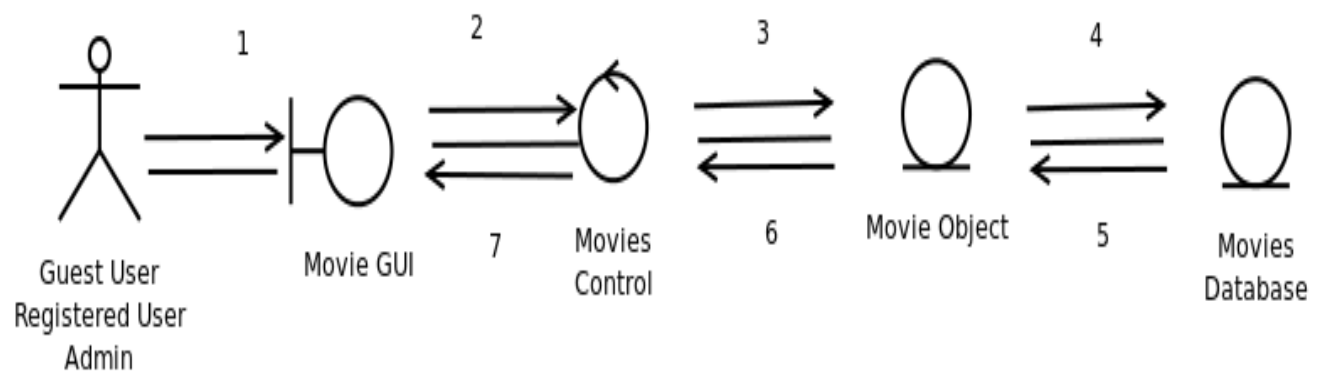
Figure A1: Search



Browse Collaboration Diagram

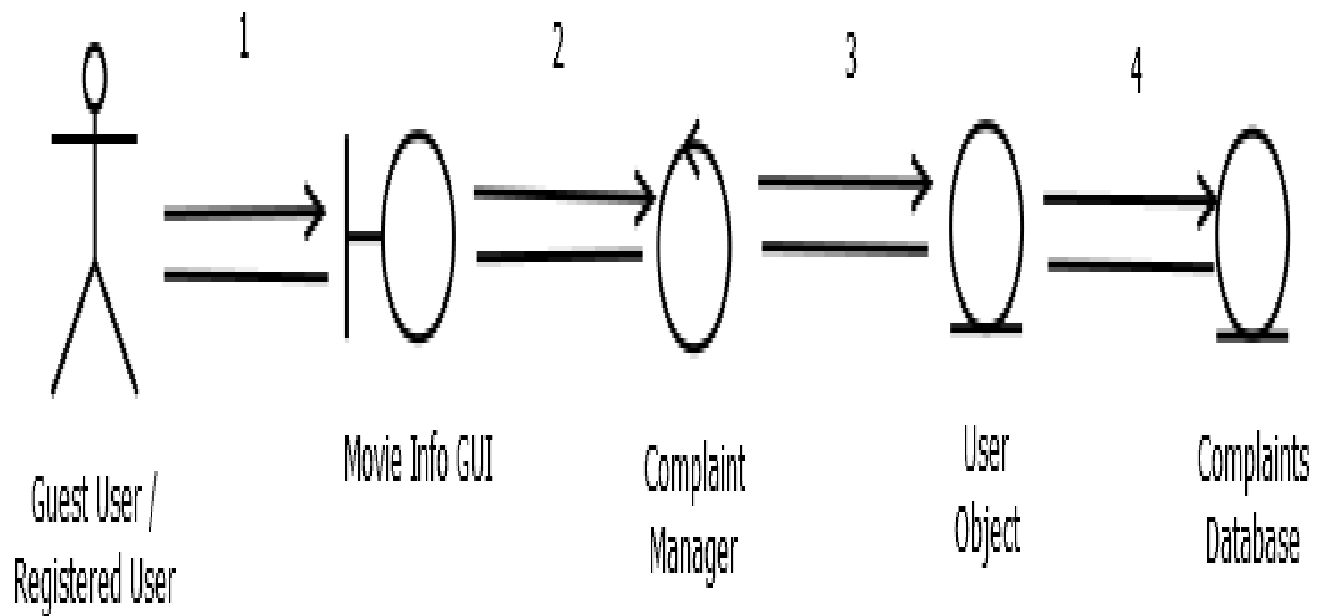
1. Admin User, Registered User, or Guest User visits the Browse
2. User information passed to Browse manager
3. Pass info to movie object
4. Browse object queries movies database
5. Return list of movies that matches browse selection
6. Pass result to Browse manager
7. Admin User, Registered User, or Guest User see the updated Browse GUI with list of Movies

Figure A2: Browse



1. GU/RU/AU clicks on the Movies GUI
2. Request is sent to the Movies control module for movie=movieID
3. A movie object is generated
4. Request is sent to the Movies database containing movieID
5. MovieObject is initialized with
(Name, Year, Genre, Director, Stars, RunningTime, Summary)
4. Information about the Movie is returned to Control
5. Abstract of Movie
(Name, Year, Genre, Director, Stars, RunningTime, Summary)
displayed on GUI

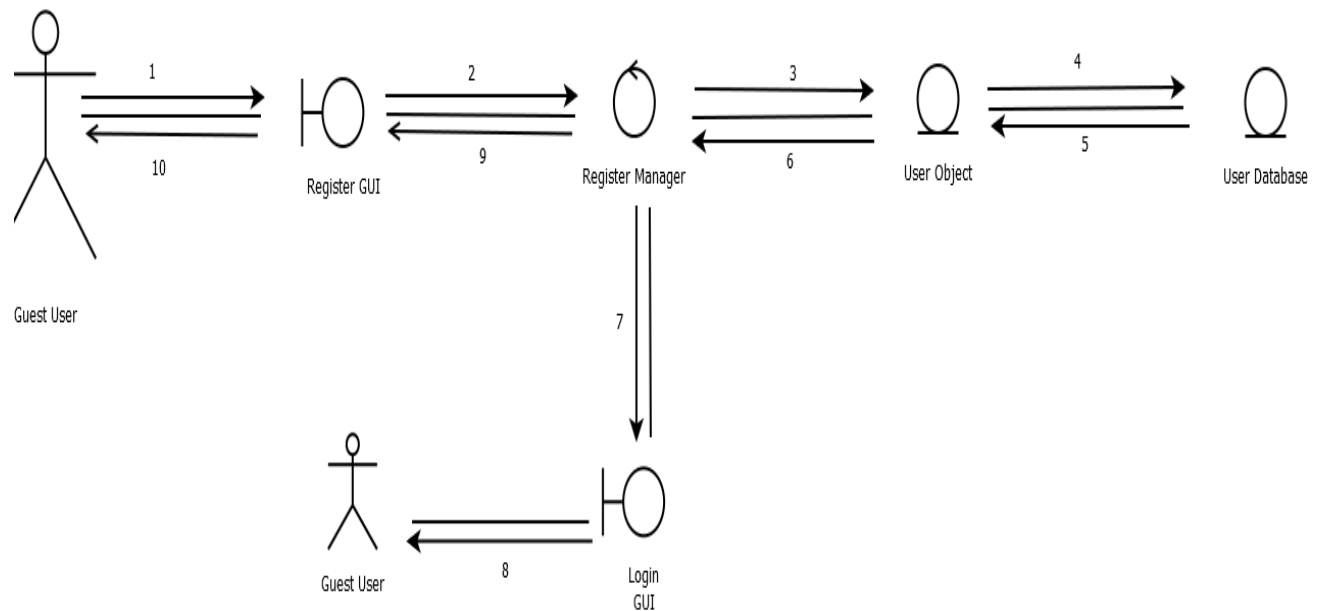
Figure A3: View Movie Info



1. GU or RU clicks 'send complain' on a comment when viewing comments on the movie page
2. The complaint data (userID and comment) is sent to be processed in the Complaint Manager
3. The Complaint Manager passes complaint information to the User Object
4. The Complaint is inserted into the Complaints Database

Figure A4: Complain

3.1.2 Guest User Collaboration Diagrams

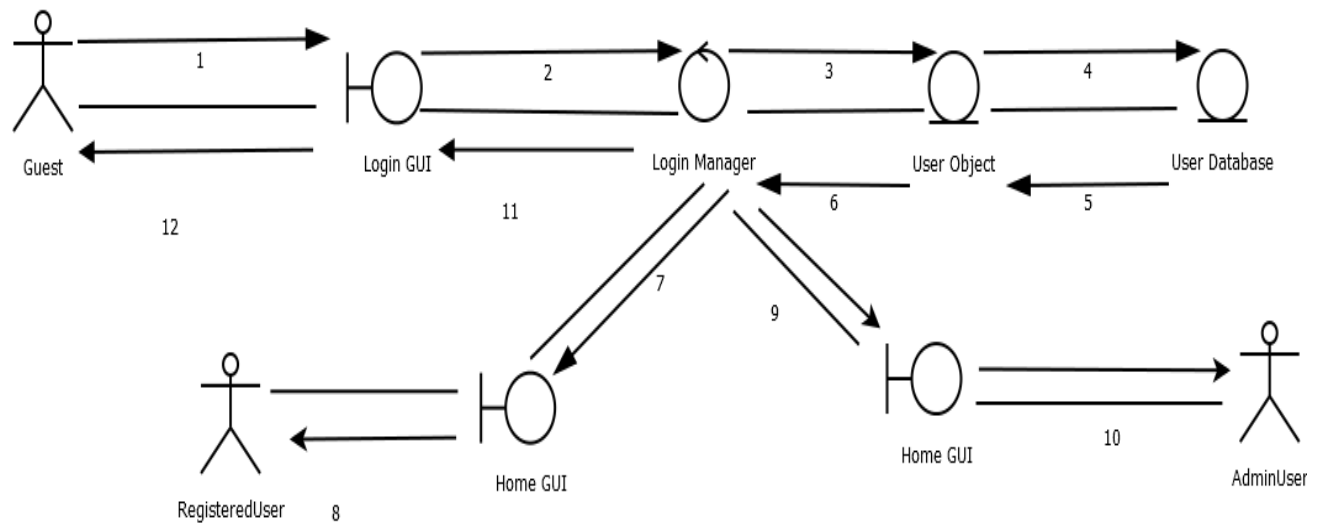


1. Guest User clicks on 'Register GUI' to be registered as a user .
2. The information is then passed to the register object.
3. This information is passed to the user object and generates password.
4. The information is stored in the user database.
5. User database send the information to user object.
6. The correct information passed from the user object is then passed to register object.
7. This opens login GUI and welcomes the new user.
8. This turns a Guest User be a Registered User.

Exceptional Case:

- 1.The visitor enter the information and clicks 'Submit'.
- 2.The information is passed to Register Object.
9. The processed information is wrong so is returned to Register GUI.
10. Guest User is specified the information provided is wrong.

B1: Register



1. Guest User fills in account information on the login page
2. User info passed to login manager
3. Pass login info to user object
4. Login object queries user database with login info
5. User database passes back query results to user object
6. Login manager receives results
7. Login manager shows Home GUI
8. System recognizes end user as Customer
9. Login manager shows main admin page
10. System recognizes end user as Admin

Exception 1

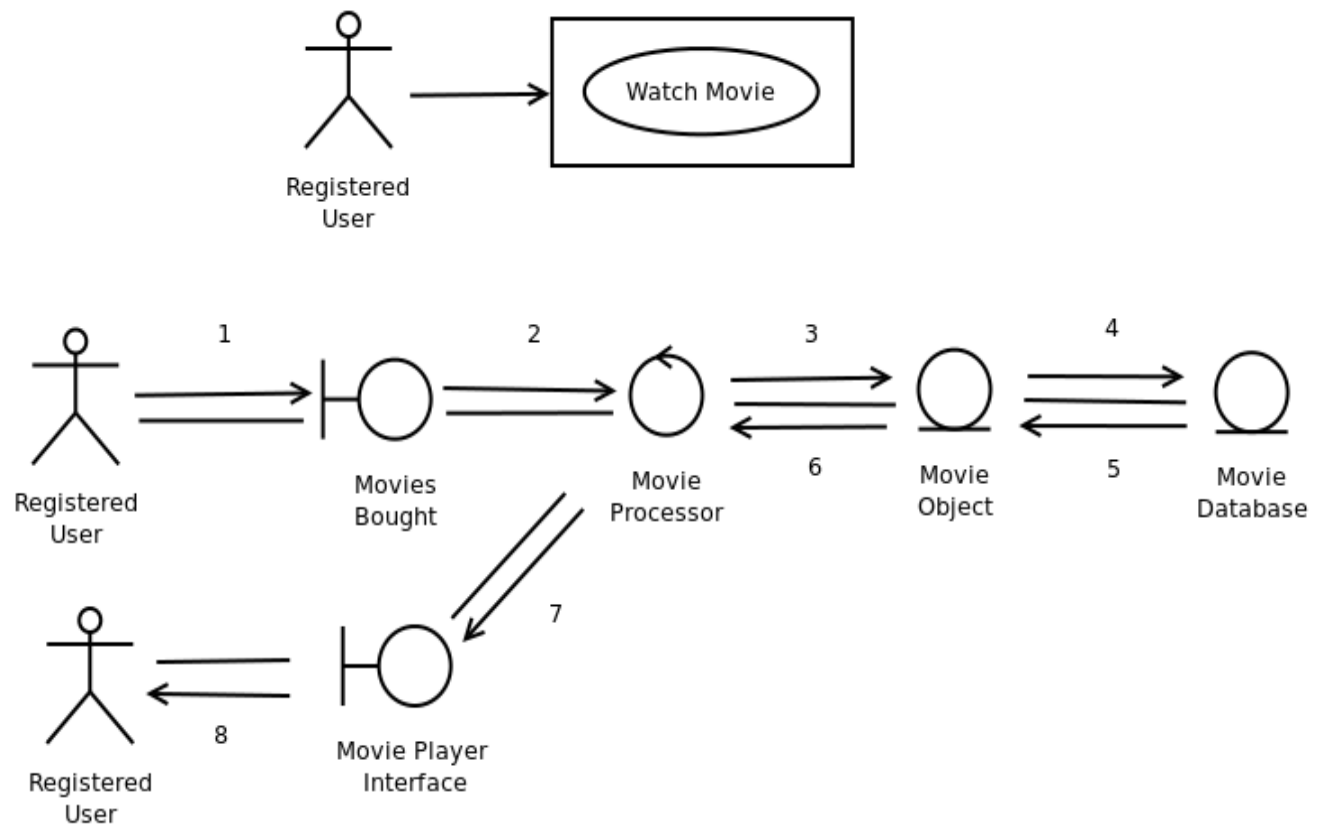
5. DB error sent to Login Object
6. Failure sent to Login Manager
11. Login manager presents Failure message on GUI
12. Guest receives error message

Exception 2

7. First time login user present with choose interest page

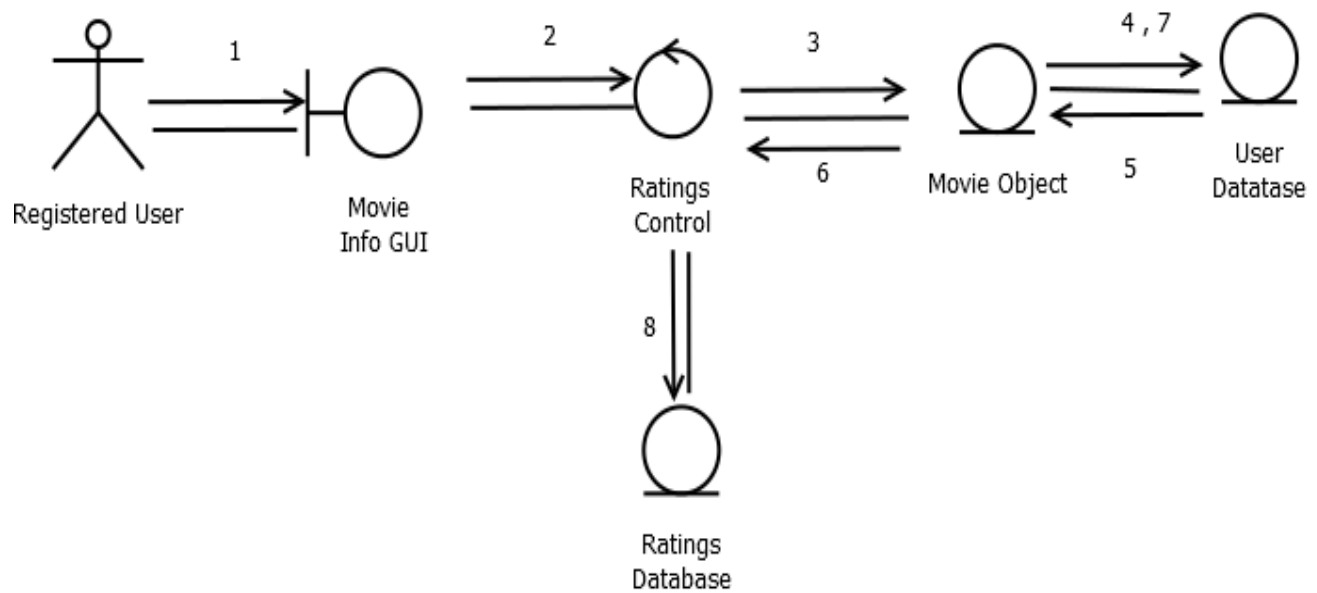
Figure B2: Login

3.1.3 Regular User Collaboration Diagrams



1. Registered User visits 'movies bought' interface
2. RU clicks on a movie to watch
3. Movie processor creates instance of movie object
4. Movie object retrieves data on specific movie
5. Movie data is sent back to movie object
6. Movie data is sent back to movie processor
7. Movie Processor redirects data and user's location to movie player GUI
8. Movie Player Interface waits for RU to click play

C1: Watch Movie



1. Registered User clicks on movie rating interface on Movie Info GUI
2. Rating Interface submits UserID, MovieID, and Rating to Rating Control
3. Rating Control passes UserID, MovieID, and Rating to movie
4. Control checks User DB to check User's ratings permissions
5. User DB returns RU's permissions
6. Ratings Control sends success or failure message back to Ratings GUI
7. Ratings Control updates Registered User Database to record Rating Behavior
8. Ratings Control updates Rating Database with new rating

Exceptions:

1. RU presses submit button without selecting a rating
2. RU does not have permissions to rate movie

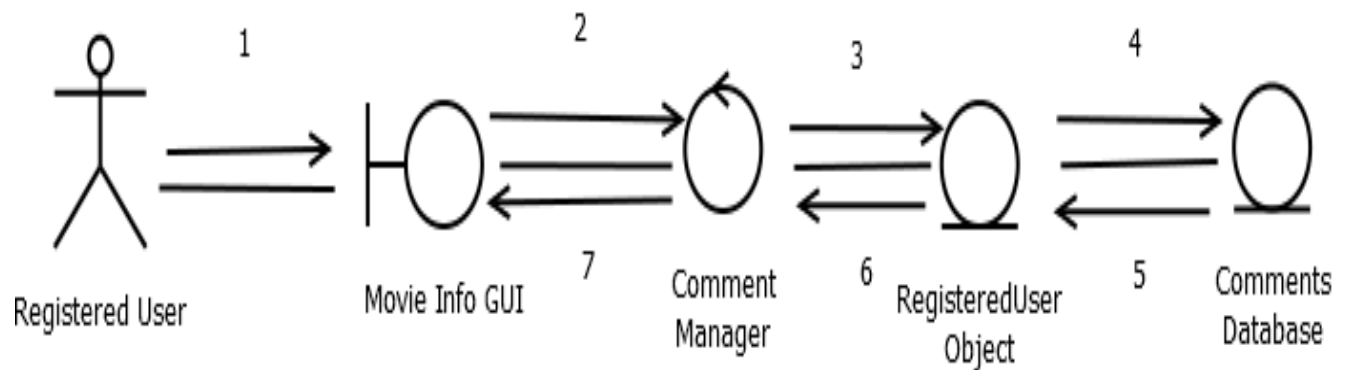
Rating Control checks User DB to check User's ratings permissions

4. User DB returns RU's permissions
5. Ratings Control sends success or failure message back to Ratings GUI
6. Ratings Control updates Registered User Database to record Rating Behavior
7. Ratings Control updates Movie Database with new rating

Exceptions:

1. RU presses submit button without selecting a rating
2. RU does not have permissions to rate movie

Figure C2: Rate



1. RU visits the comment interface of a movie page
2. RU submits comment to be processed by the comment manager
3. The Comment Manager accesses RegisterUser object to processes the comment with a timestamp
4. The data is inserted to Comments Database
5. The comment information is sent back to the RegisteredUser object
6. The comment information is passed back to the comment manager
7. The Comment Manager updates the Comment Interface with new comment(s)

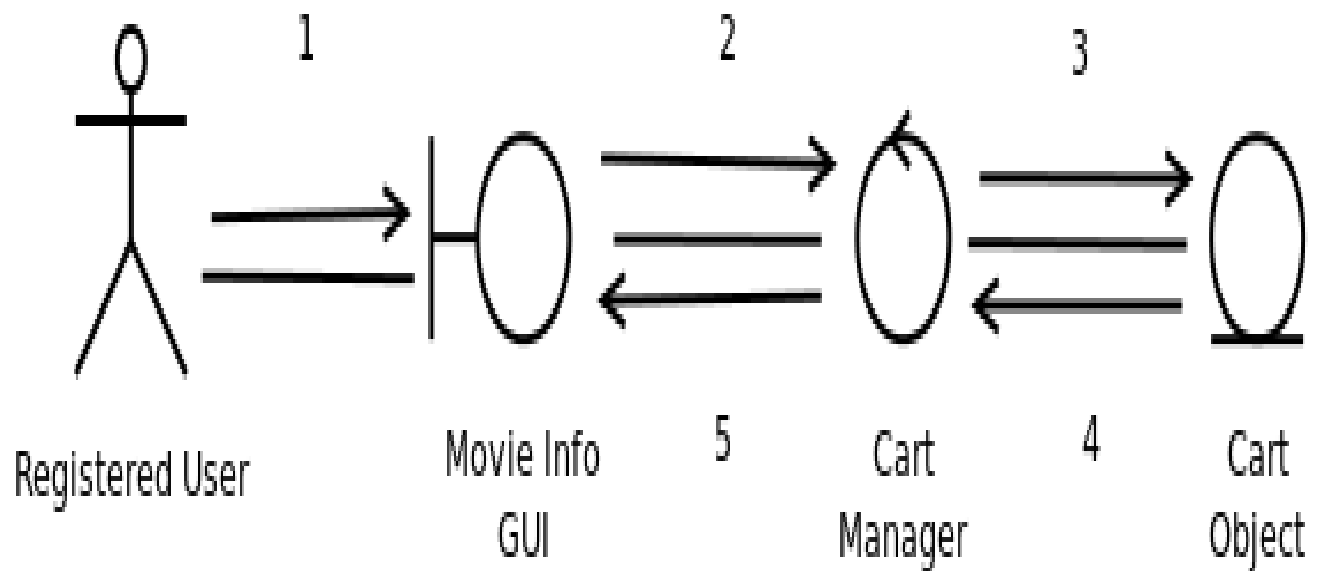
Exception 1:

2. Blank comment is submitted
7. The comment manger displays an error message to the GUI

Exception 2:

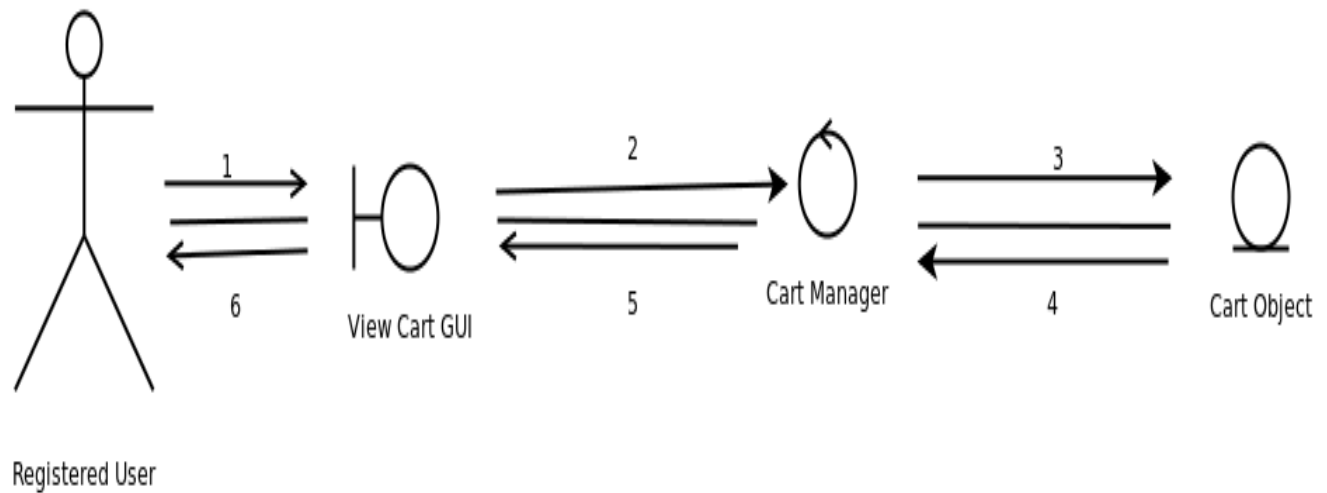
2. A comment that exceeds the character limit is submitted
7. The comment manager displays an error message to the GUI

Figure C3: Comment



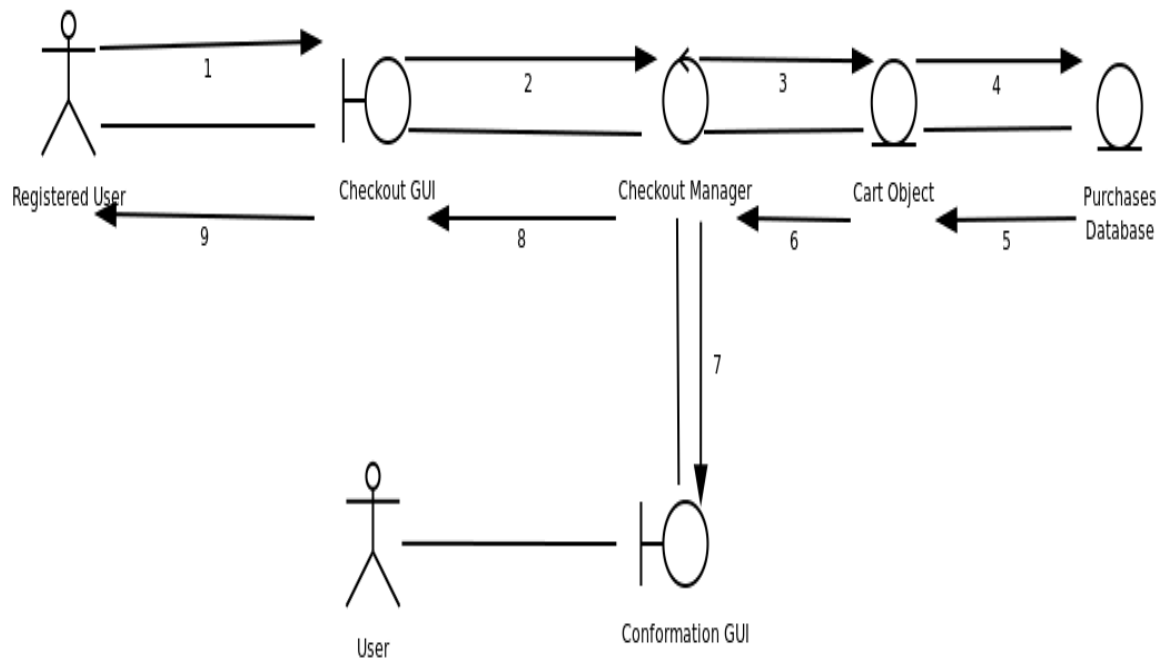
1. RU visits movie info page
2. RU clicks 'add to cart'
3. Cart manager adds the movie data to the cart object
4. Cart object sends back updated contents of cart
5. Updated cart information is sent back to cart GUI

Figure C5: Add to Cart



1. Registered User clicks on View Cart GUI.
2. GUI processes the information to the cart manager.
3. Cart manager then passes it to cart object.
- 4.. Cart object passes the information to cart manager.
5. Cart Manager recives the items and displays it to the View Cart GUI.
6. The items available in the cart is presented in the GUI.

Figure C6: View Cart



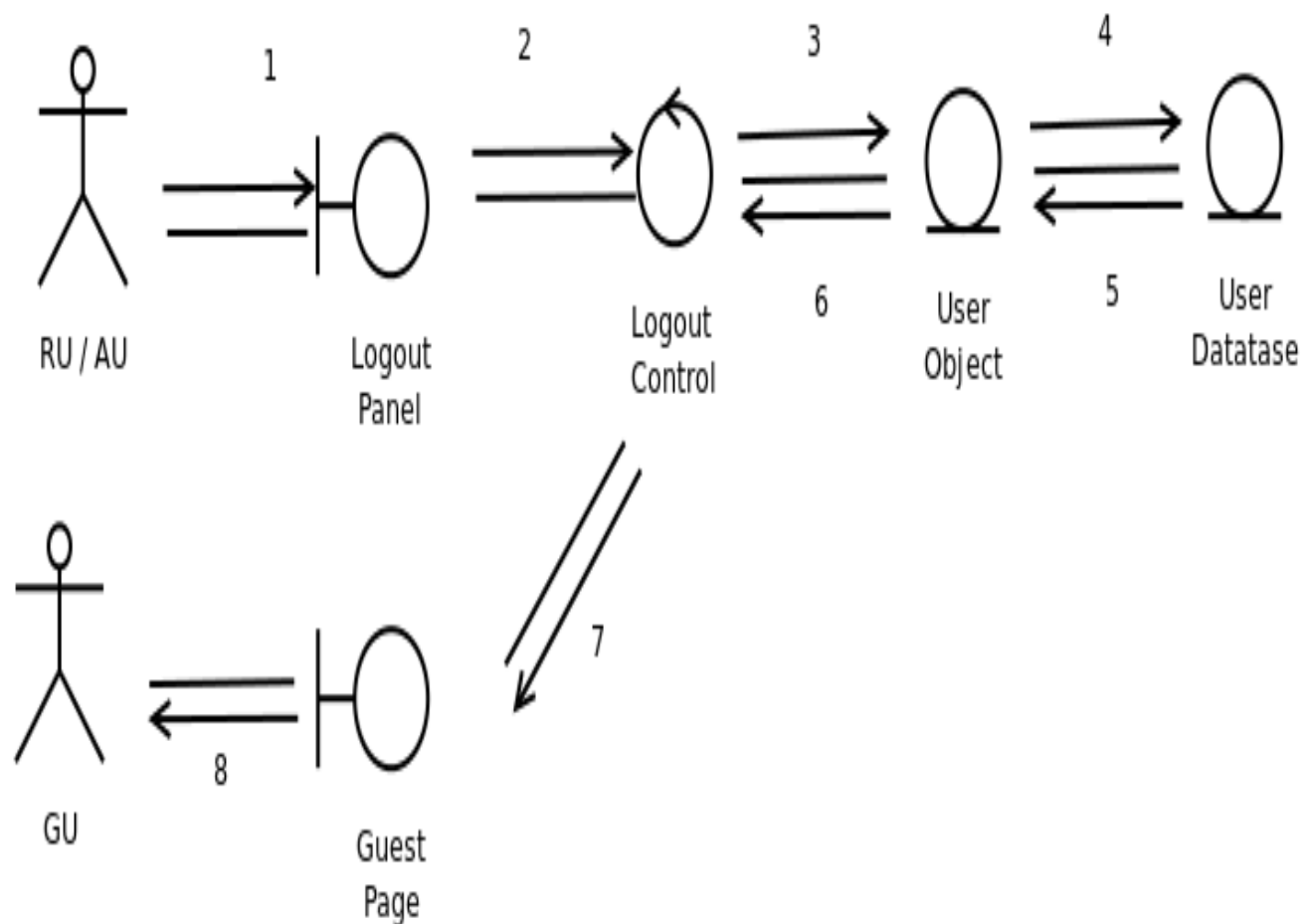
Checkout Collaboration Diagram

1. Registered User visits Checkout page
2. RU and Item info passed to Checkout manager
3. Pass RU and Item info to Cart Object
4. Cart object adds purchase data to purchases database
5. Item information is added and sent to cart object
6. Purchase status and information is sent to to checkout manager
7. RU goes Confirmation GUI for purchase

Exception

5. Movie is already bought
6. Don't checkout item
8. Displays error message "Already purchased movie"
9. RU presented with error message

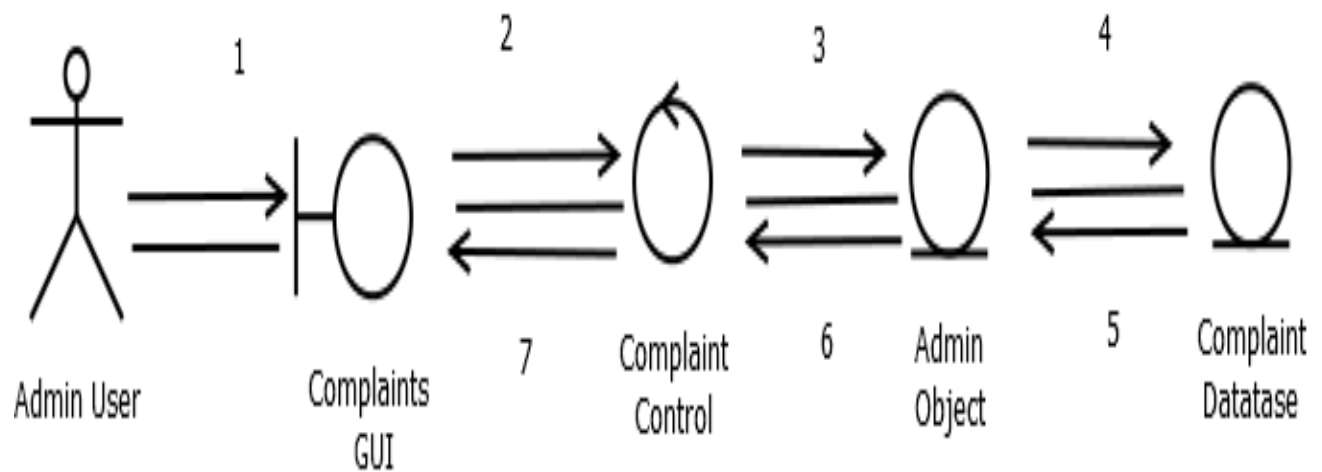
Figure C7: Checkout



1. Registered User/Admin User clicks on the Logout button
2. Logout Control processes request, verifies user's desire to logout
3. Controller passes logout to user database, ends user's session, completes any pending transactions
4. Database returns results to Logout Control
5. Logout Control produces Guest page stating "You have now been logged out"
- 6 End User is now considered to be a Guest

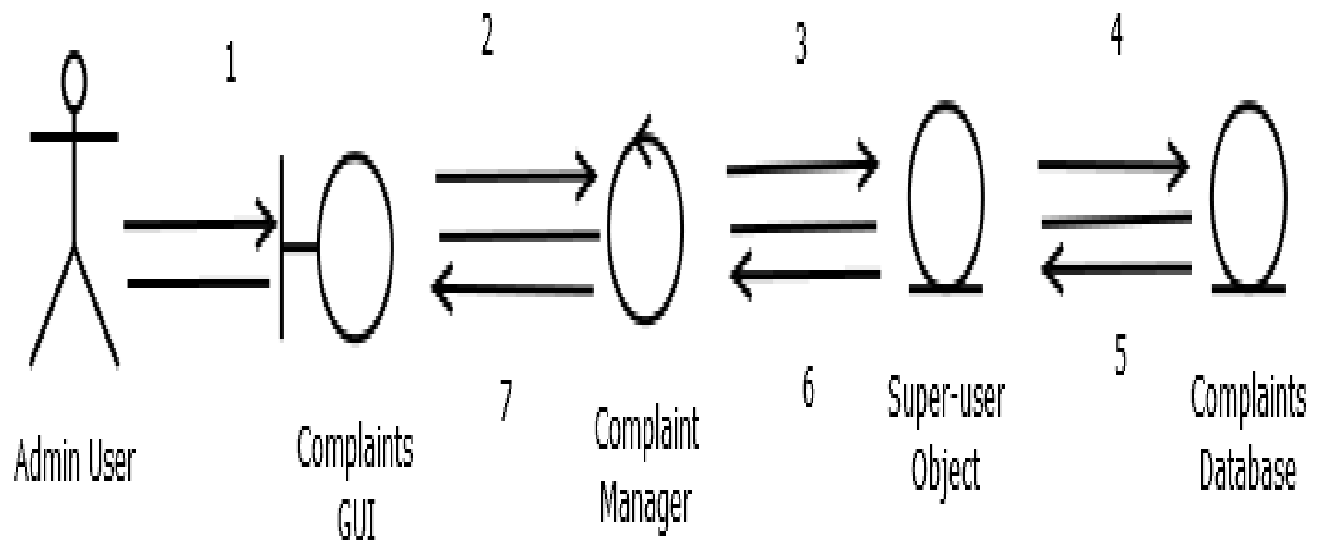
Figure C8: Logout

3.1.4 Admin User Collaboration Diagrams



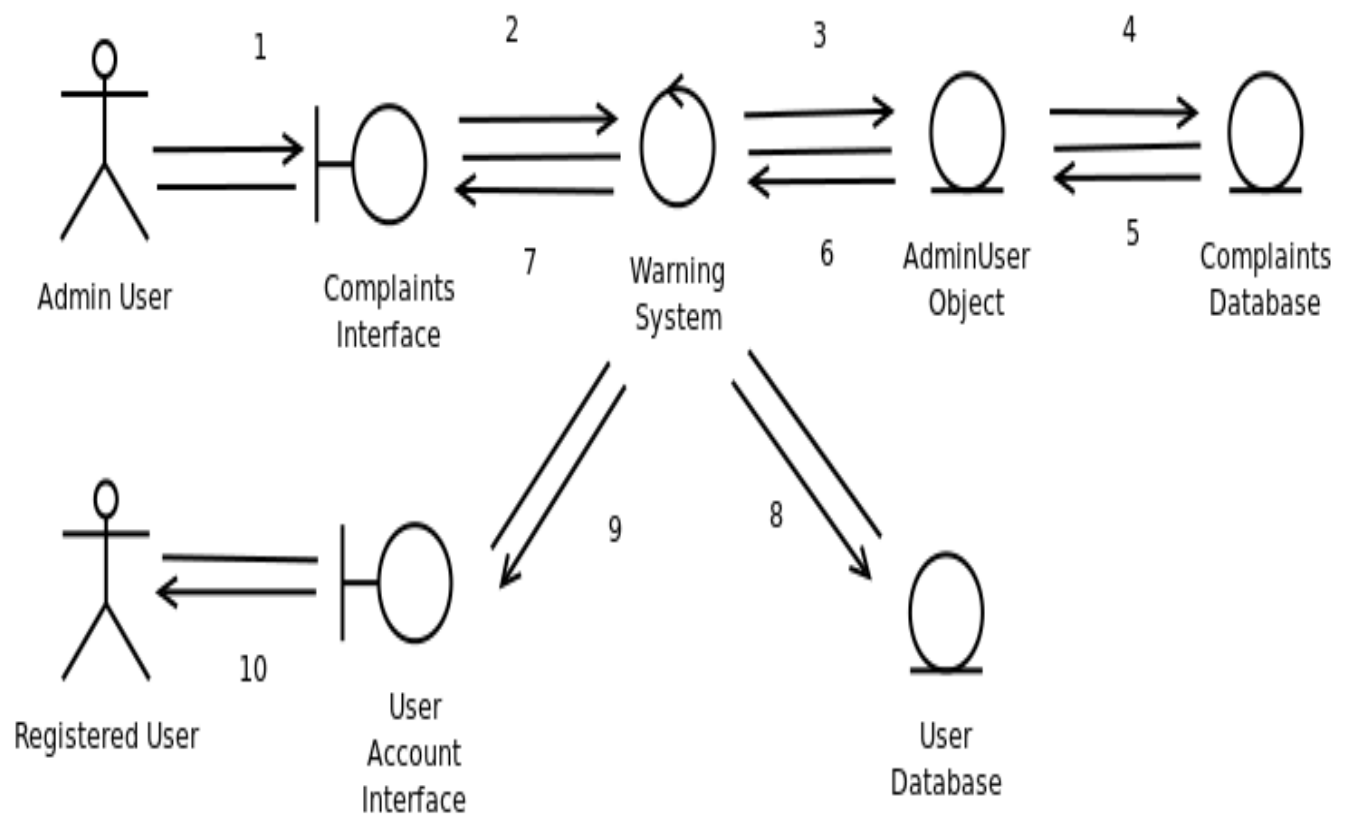
1. Admin User selects comments for deletion from Complaints GUI
2. GUI sends deletion request to Complaints Control
3. Complaints to be deleted are sent to Database
4. Complaints are sent to complaint database
5. Database returns remaining Flagged Comments from Database
6. The remaining Complaints are sent to complaint control
7. The remaining Complaints are displayed on the GUI

D1: Erase Comment



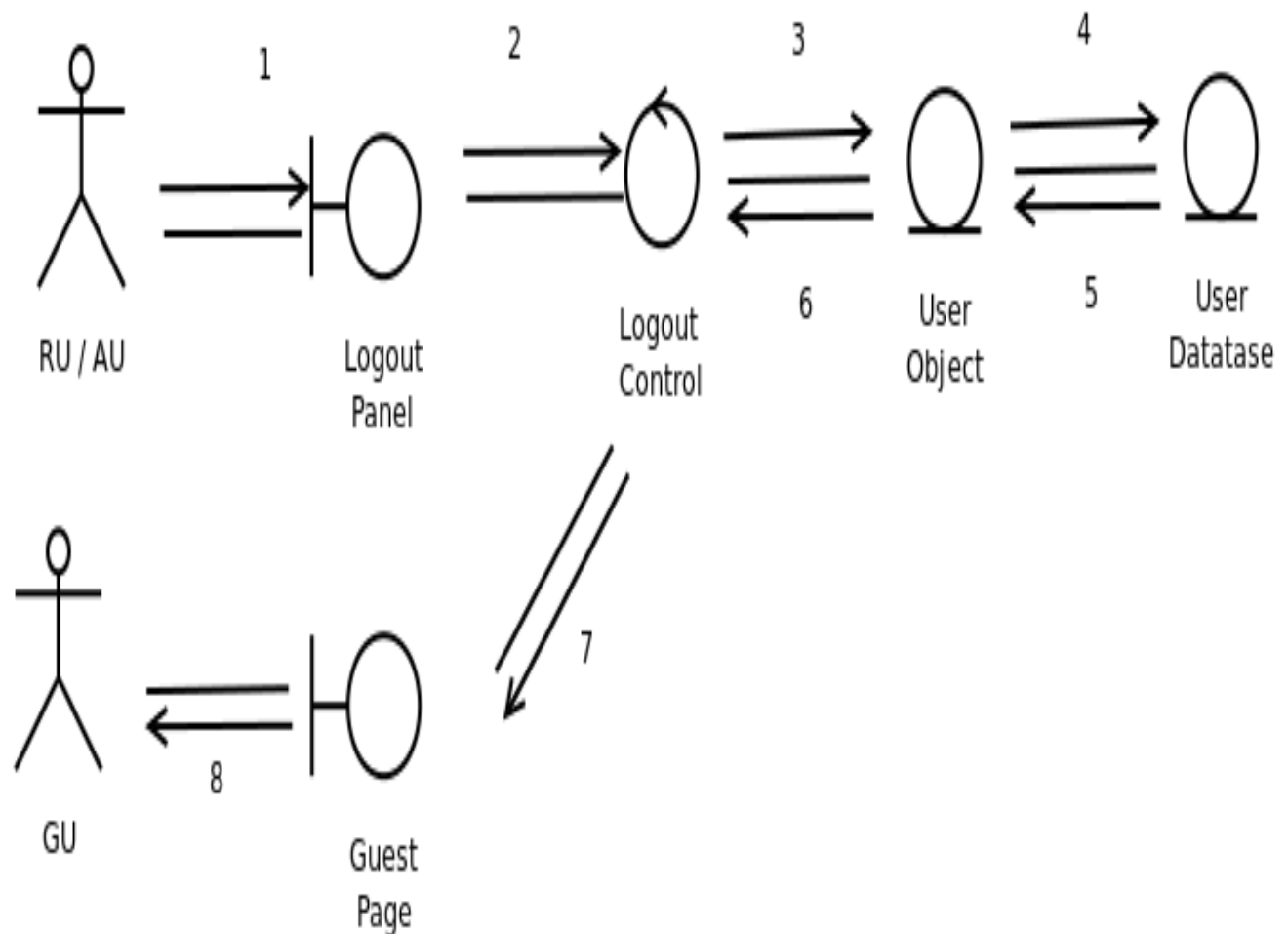
1. Admin User views the Complaints GUI to see recent complaints
2. AU clicks 'ignore complaint'
3. The Complaint manager sends complaint information to super-user object
4. The complaint data is sent to the Complaints Database to remove the complaint
5. The complaints database returns a list of recent complaints
6. The complaints data is sent back to Complaint Manage
7. The Complaints manger refreshes list of complaints

Figure D2: Ignore Complaint



1. Admin-user views complaints interface to see recent complaints
2. Admin-user clicks 'send warning' to user
3. Warning system accesses super-user object's 'warn user' method
4. The AU object sends an update to the complaints database
5. The complaint data (username, date, comment) is sent to the AU object
6. The complaint information is sent back to warning system
7. The complaint is removed from 'recent complaints'
8. The warning system sends query to increase the user's 'warnings' counter by one
9. A warning message (with date and comment) is sent to username
10. The Registered User receives warning message when he/she views their account page

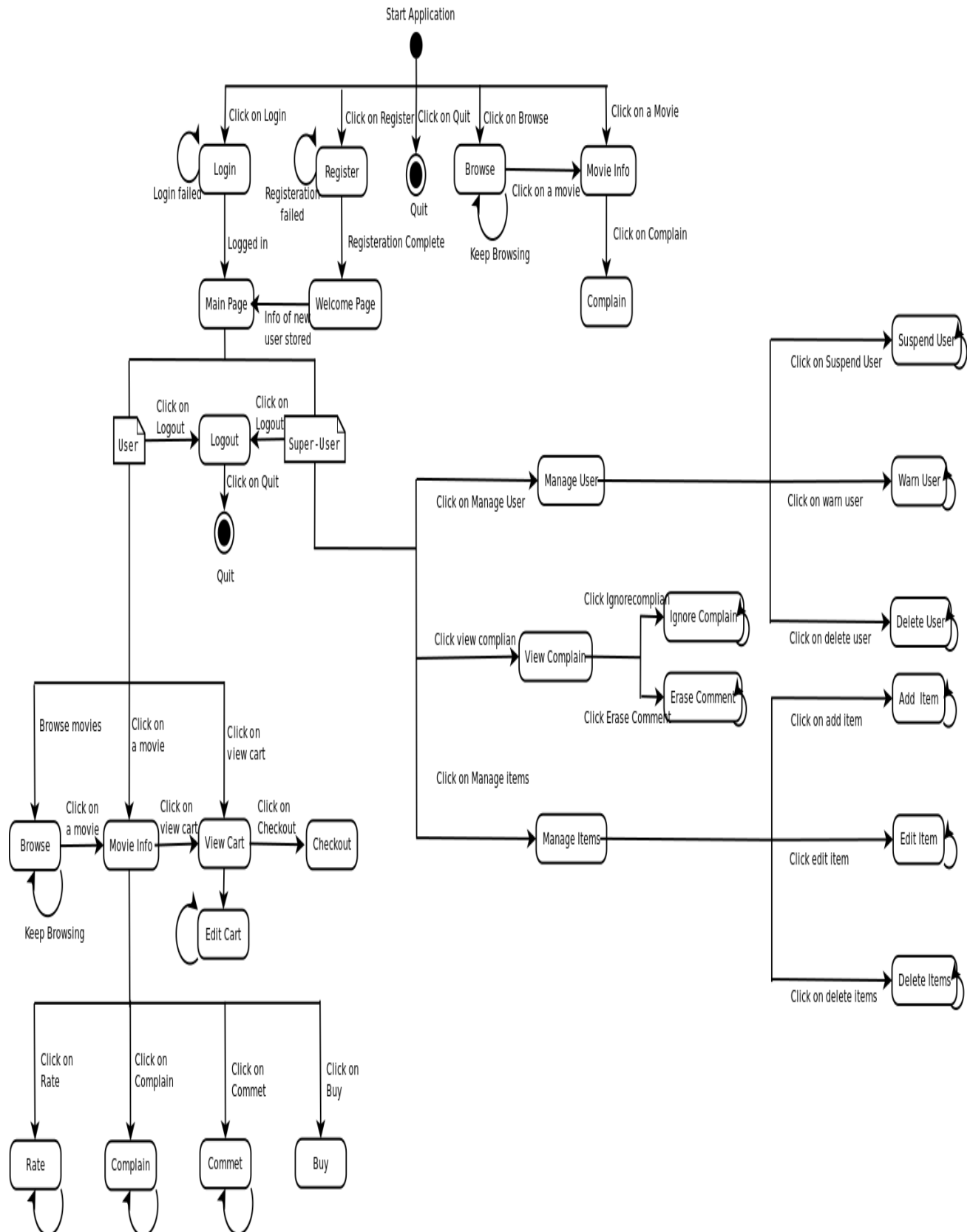
Figure D3: Warn User



1. Registered User/Admin User clicks on the Logout button
2. Logout Control processes request, verifies user's desire to logout
3. Controller passes logout to user database, ends user's session, completes any pending transactions
4. Database returns results to Logout Control
5. Logout Control produces Guest page stating "You have now been logged out"
- 6 End User is now considered to be a Guest

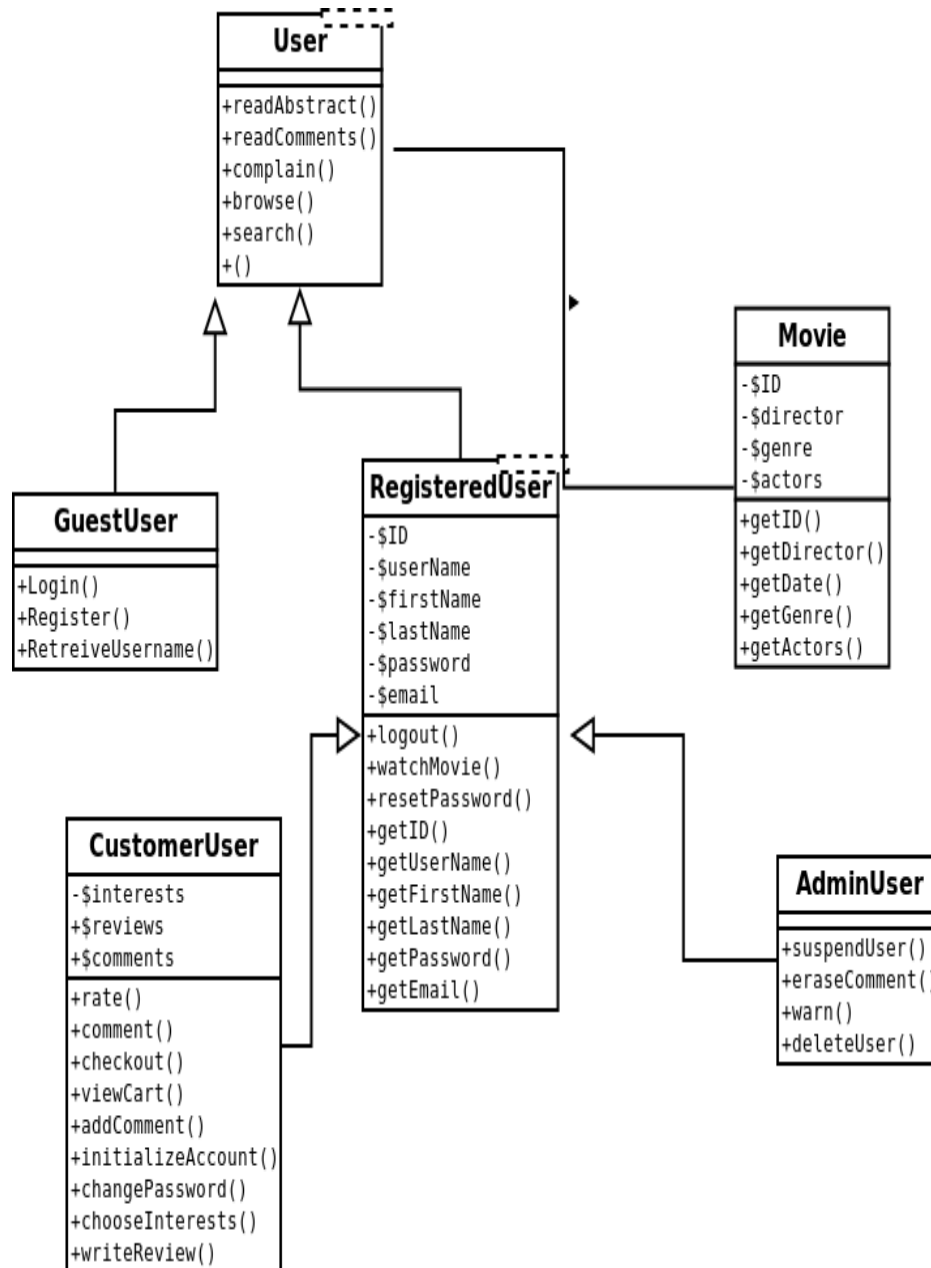
Figure D4: Logout

2.3 ESC System State Diagram

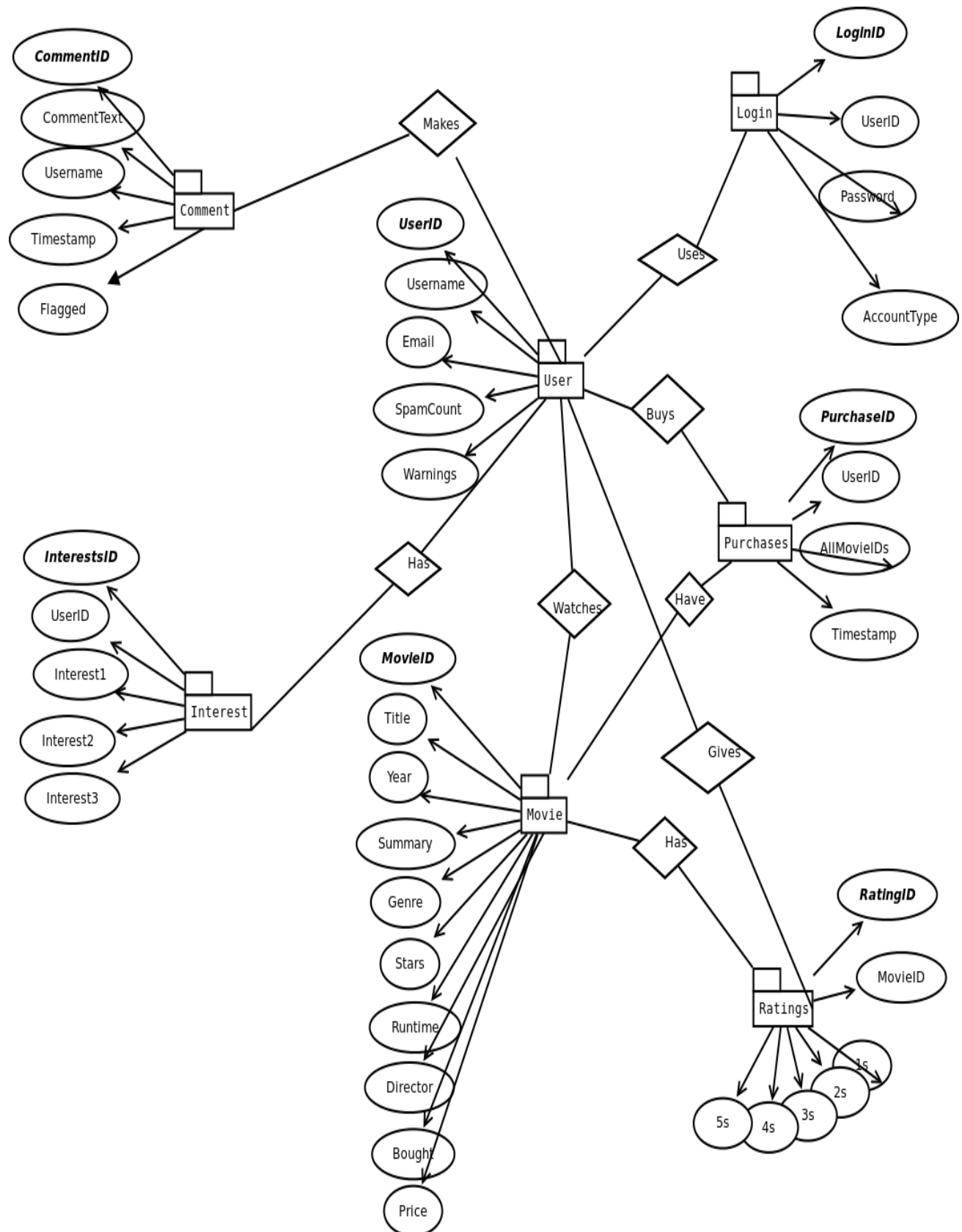


3. ER Diagrams

3.1 ER Class Diagram



3.1 ER Class Diagram



Guest User Class

(extends the User Class)

Attributes:

- ♣ private \$userID;
- ♣ private \$username;
- ♣ private \$accountType;
- ♣ private \$userEmail;

Functions:

- ♣ public function login(\$userID, \$userPassword)
// This allows a guest user to log in to the system and become a registered user or admin user
- ♣ public function register()
// This allows a guest user to become a registered user
- ♣ public function complainComment(\$commentID)
// This allows a guest user to send a complaint about a comment identified by commentID

RegisteredUser Class

(extends the User Class)

Attributes:

- ⤴ private \$userID;
- ⤴ private \$username;
- ⤴ private \$userEmail;

Functions:

- ⤴ public function getID()
// This returns the id of the registered user
- ⤴ public function getUsername()
// This returns the username of the registered user
- ⤴ public function getEmail()
// This returns the email address of the registered user
- ⤴ public function complainComment(\$commentID)
// This allows a registered user to send a complaint about a comment identified by commentID
- ⤴ public function login(\$userID, \$userPassword)
// This allows a registered user to send a complaint about a comment identified by commentID
- ⤴ public function rateMovie(\$userID, \$movieID, \$rating)
// This allows a registered user to give a rating to a movie.
- ⤴ public function submitComment(\$userID, \$movieID, \$commentText)
// This allows a registered user to post a comment to the movie page identified by movieID

Movie Class

Attributes:

- ♣ private \$movieID;
- ♣ private \$movieName;
- ♣ private \$movieYear;
- ♣ private \$movieSummary;
- ♣ private \$movieGenre;
- ♣ private \$movieDirector;
- ♣ private \$movieStars;
- ♣ private \$movieRuntime;
- ♣ private \$movieImageLocation;
- ♣ private \$movieLocation;
- ♣ private \$movieRating;
- ♣ private \$moviePrice;

Functions:

- ♣ function __construct(\$movieID)
// Constructor for the Movie class
- ♣ public function playMovie(\$userID, \$movieID)
// Verifies that user has purchased the movie, plays movie in movieLocation
- ♣ public function browse()
// Displays all information about movie
- ♣ public function getMovieId()
// Returns movie id
- ♣ public function getMovieName()
// Returns movie name
- ♣ public function getMovieYear()
// Returns movie year
- ♣ public function getMovieSummary()
// Returns movie summary
- ♣ public function getMovieGenre()
// Returns movie genre
- ♣ public function getMovieDirector()
// Returns movie director
- ♣ public function getMovieStars()
// Returns movie stars
- ♣ public function getMovieRuntime()
// Returns movie runtime
- ♣ public function getMovieImageLocation()
// Returns file location of movie poster
- ♣ public function getMovieLocation()
// Returns location of movie file

- ♣ public function watchMovie()
// plays the movie
- ♣ public function getPrice()
// Returns movie price

Movie Class

Attributes:

- ⤴ private \$movieID;
- ⤴ private \$movieName;
- ⤴ private \$movieYear;
- ⤴ private \$movieSummary;
- ⤴ private \$movieGenre;
- ⤴ private \$movieDirector;
- ⤴ private \$movieStars;
- ⤴ private \$movieRuntime;
- ⤴ private \$movieImageLocation;
- ⤴ private \$movieLocation;
- ⤴ private \$movieRating;
- ⤴ private \$moviePrice;

Functions:

- ⤴ function __construct(\$movieID)
// Constructor for the Movie class
- ⤴ public function playMovie(\$userID, \$movieID)
// Verifies that user has purchased the movie, plays movie in movieLocation
- ⤴ public function browse()
// Displays all information about movie
- ⤴ public function getMovieId()
// Returns movie id
- ⤴ public function getMovieName()
// Returns movie name
- ⤴ public function getMovieYear()
// Returns movie year
- ⤴ public function getMovieSummary()
// Returns movie summary
- ⤴ public function getMovieGenre()
// Returns movie genre
- ⤴ public function getMovieDirector()
// Returns movie director
- ⤴ public function getMovieStars()
// Returns movie stars
- ⤴ public function getMovieRuntime()
// Returns movie runtime
- ⤴ public function getMovieImageLocation()
// Returns file location of movie poster
- ⤴ public function getMovieLocation()
// Returns location of movie file

- ♣ public function watchMovie()
// plays the movie
- ♣ public function getPrice()
// Returns movie price

Cart Class

Attributes:

- ⤴ private \$totalPrice;
- ⤴ private \$checkoutCart = array();

Functions:

- ⤴ public function getAmountItems()
// Returns the length of \$checkoutCart
- ⤴ public int getCurrentTotal()
// Returns the current total price of items in the cart
- ⤴ public function removeFromCart(\$movie)
// Removes a movie from the checkout cart
- ⤴ public int checkout(\$userID, \$allMovieIDs)
// Executes purchase of all movies by the user.

Comment Class

Attributes:

```
private $commentID;  
private $commentText;  
private $userName;  
private $timestamp;
```

Functions:

```
⤴ function __construct($commentID, $commentText, $userName, $timestamp)  
    // Constructor for the Comment class  
  
⤴ public int getCommentID()  
⤴  
    // Returns the ID of the comment  
  
⤴ public function getCommentText()  
    // Gets the text of the comment  
  
⤴ public function getUserName()  
    // Gets the name of the commenter  
  
⤴ public function getTimestamp()  
    // Gets the date of the commenter
```

Complaint Class

Attributes:

- ⤴ private \$complaintID;
- ⤴ private \$commentID;
- ⤴ private \$reason;

Functions:

- ⤴ function __construct(\$complaintID, \$commentID, \$reason)
// Constructor for the Complaint class
- ⤴ public function getComplaintID()
// Returns the complaint ID
- ⤴ public int getCommentID()
// Returns the ID of the comment being flagged
- ⤴ public function getReason()
// Gets the reason for the complaint

Cart Class

Attributes:

- ⤴ private \$totalPrice;
- ⤴ private \$checkoutCart = array();

Functions:

- ⤴ public function getAmountItems()
// Returns the length of \$checkoutCart
- ⤴ public int getCurrentTotal()
// Returns the current total price of items in the cart
- ⤴ public function removeFromCart(\$movie)
// Removes a movie from the checkout cart
- ⤴ public int getCurrentTotal(\$userID, \$allMovieIDs)
// Executes purchase of all movies by the user.

Rating Class

Attributes:

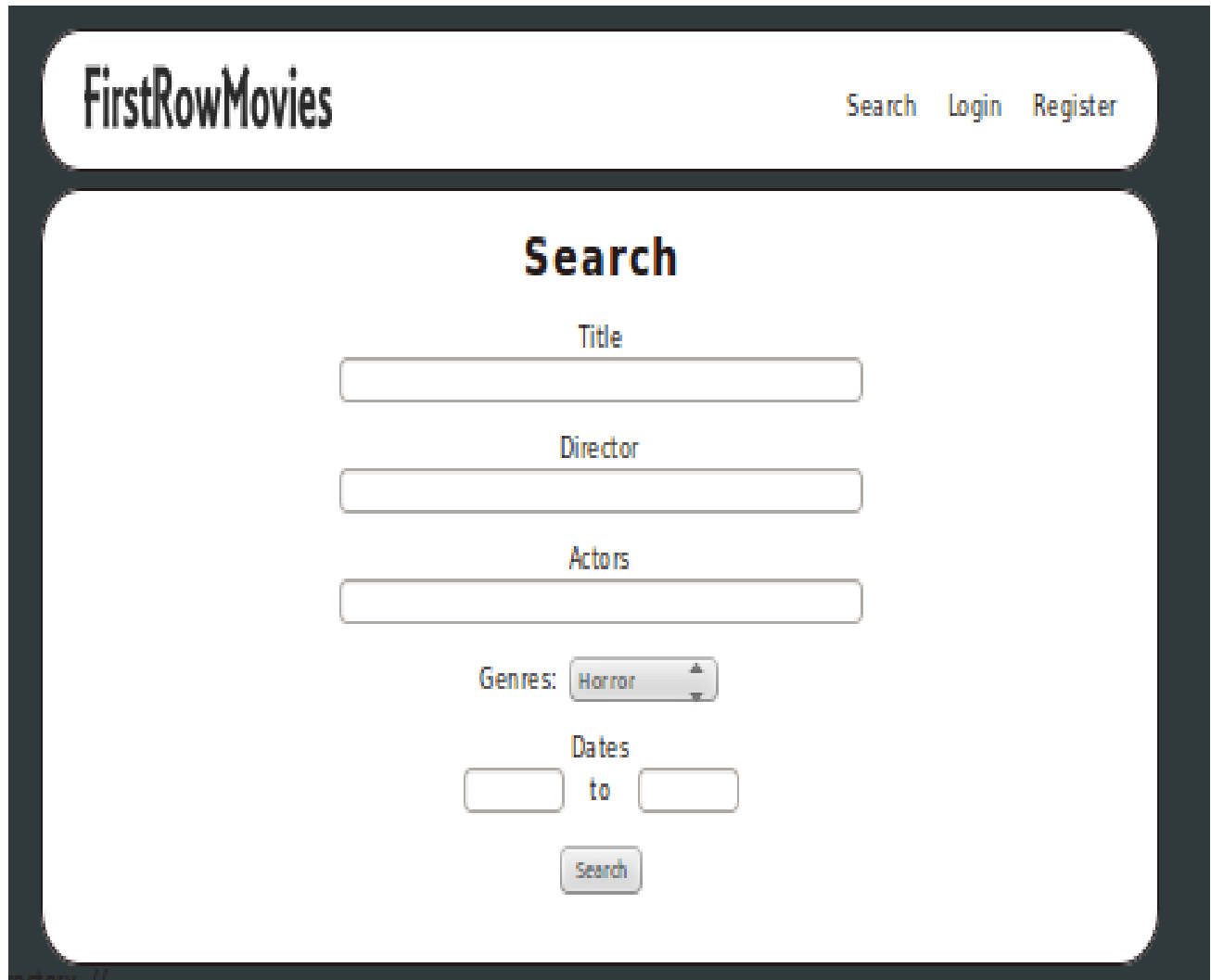
- ⤴ private \$movieID;
- ⤴ private \$average ;
- ⤴ private \$oneStars;
- ⤴ private \$twoStars;
- ⤴ private \$threeStars;
- ⤴ private \$fourStars;
- ⤴ private \$fiveStars;

Functions:

- ⤴ function __construct(\$movieID)
// Constructor for the Rating class
- ⤴ public function getAverage ()
// Returns the average rating
- ⤴ public int getOnes()
// Returns the number of one star ratings
- ⤴ public int getTwos()
// Returns the number of two star ratings
- ⤴ public int getThrees()
// Returns the number of three star ratings
- ⤴ public int getFours()
// Returns the number of four star ratings
- ⤴ public int getFives()
// Returns the number of five star ratings

5. System Screens

5.1 User-Interface Prototypes



The image shows a user-interface prototype for the 'FirstRowMovies' website. The interface is set against a dark grey background. At the top, there is a white header bar with the 'FirstRowMovies' logo on the left and three links: 'Search', 'Login', and 'Register' on the right. Below the header is a large white rounded rectangle containing the search interface. The word 'Search' is centered at the top of this rectangle in a large, bold, black font. Below it are three input fields, each with a label above it: 'Title', 'Director', and 'Actors'. The 'Actors' label is in a reddish-brown color. Below these fields is a 'Genres' section with a label and a dropdown menu currently showing 'Horror'. Underneath the genres is a 'Dates' section with two input fields separated by the word 'to'. At the bottom of the search area is a 'Search' button.

FirstRowMovies

Search Login Register

Search

Title

Director

Actors

Genres: Horror

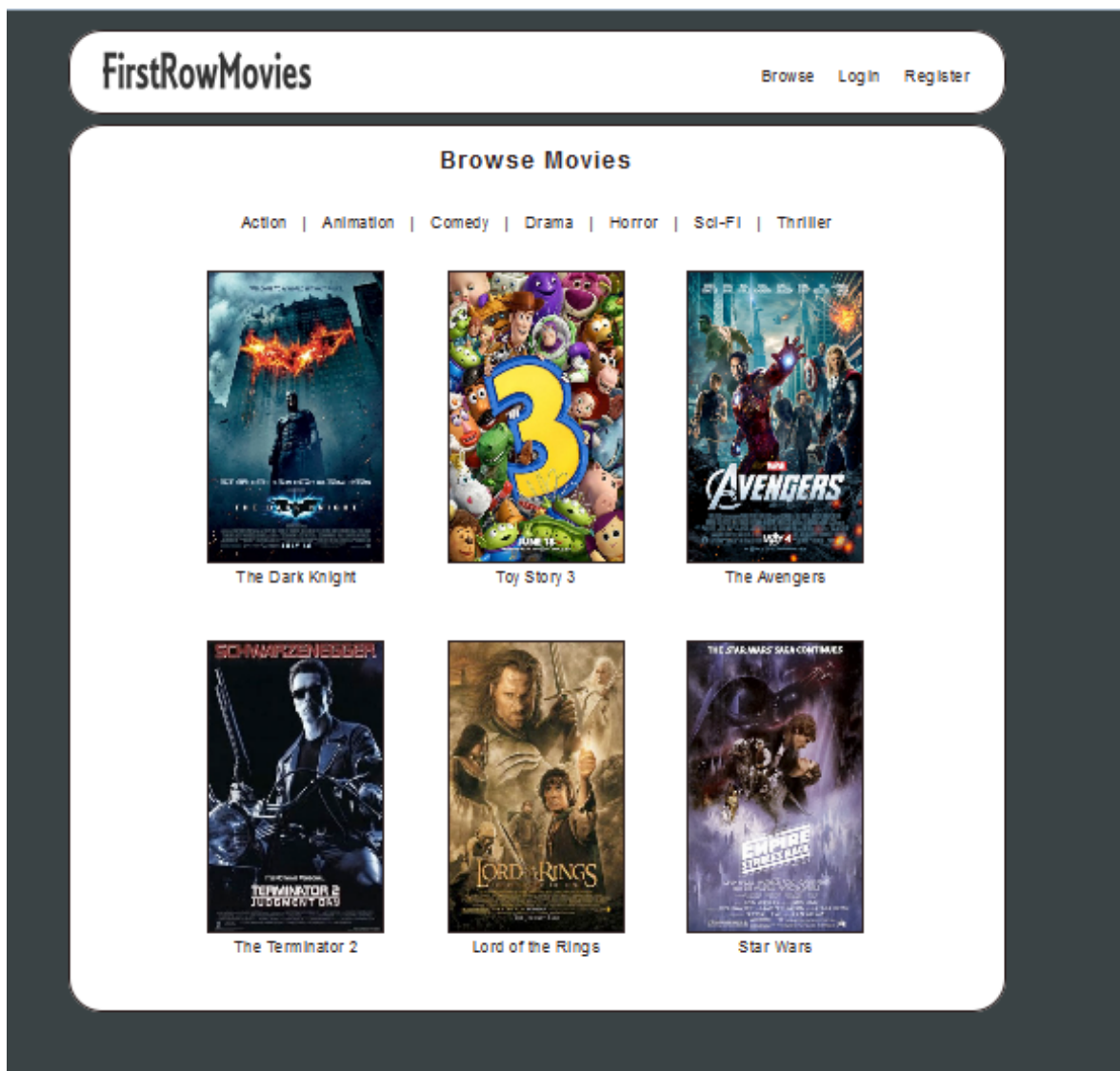
Dates

to

Search

a. Search:

Using this interface, users can search for movies according to many criteria.



a. Browse / Main Page:

This is the main page for Guest Users and Registered Users.

Using this interface, users can search for movies according to many criteria.