Team:

Avneendra Kanva

Michael Pauly

Title: Media Preference Tracking

Project Summary:

This is a web based platform that will help users track movies that they have experienced. Based on the movies they rate and ignore, taking into account the ratings of other users, the system will provide recommendations on movies each user might enjoy.

Note: Text in red indicate stretch functionality that might not be fully represented in all diagrams.

Project Requirements:

Business Requirements		
ID	**Requirements**	**Priority**
	NONE	

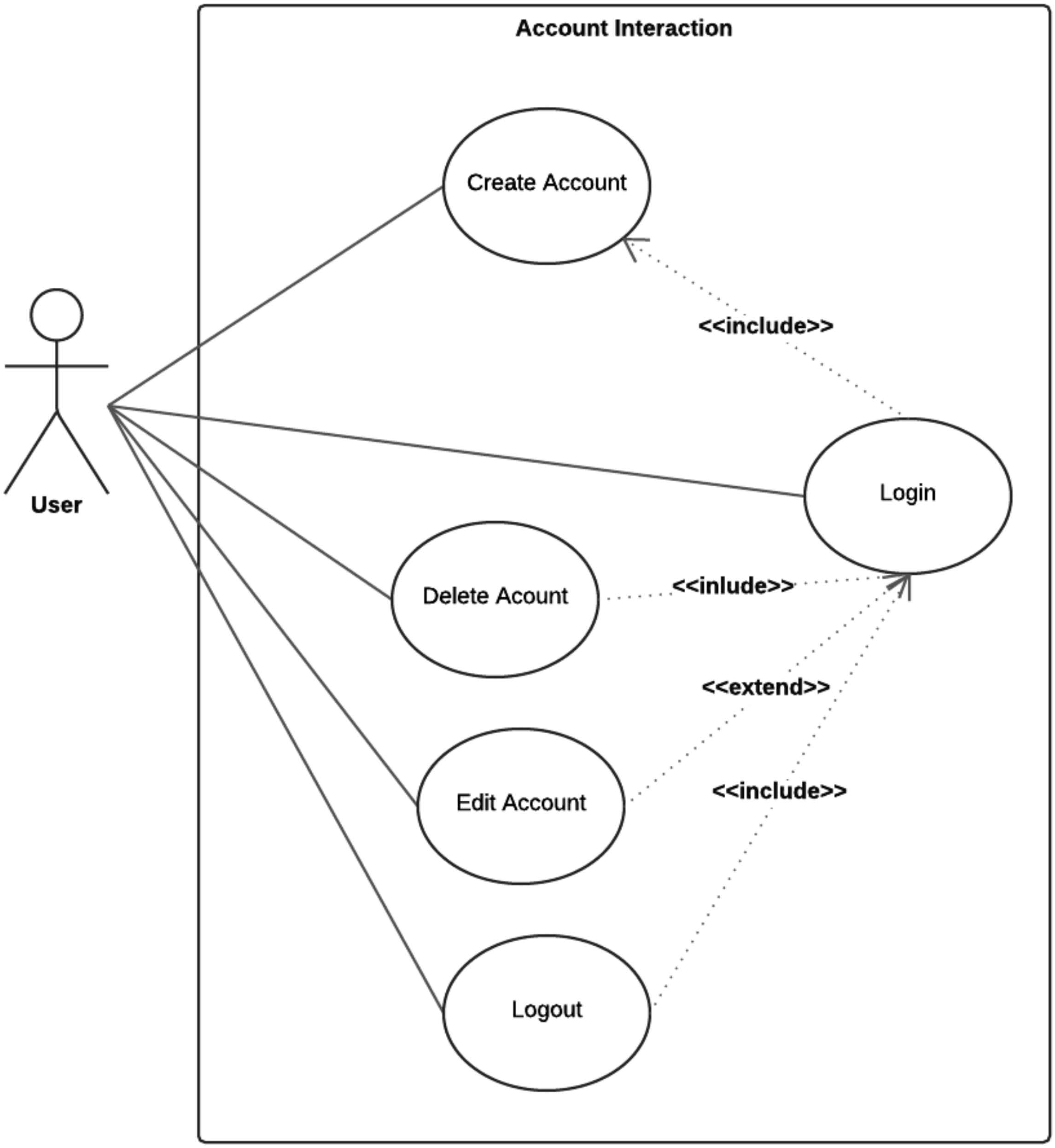
User Requirements		
ID	**Requirements**	**Priority**
UR-01	User can create new accounts that requires username, password, first name, last name, and email	HIGH
UR-02	User can log in after creating account.	HIGH
UR-03	User can modify account.	HIGH
UR-04	User can delete account.	HIGH
UR-05	User can log out.	HIGH
UR-06	User can toggle between recommendations/search/my list mode	HIGH
UR-07	User can edit preferences of movies	HIGH
UR-08	User able to search for movies with search feature	LOW
UR-09	User can toggle between ratings/ ignore/ wishlist while inside "my list" mode	MEDIUM
UR-10	Users can add comments to rated movies	LOW
UR-11	Commented ratings can be viewed when this movie is being suggested	LOW
UR-12	User can navigate within search results	LOW
UR-13	User can sort movies	LOW
UR-14	Users can navigate to IMDB page of movie	LOW

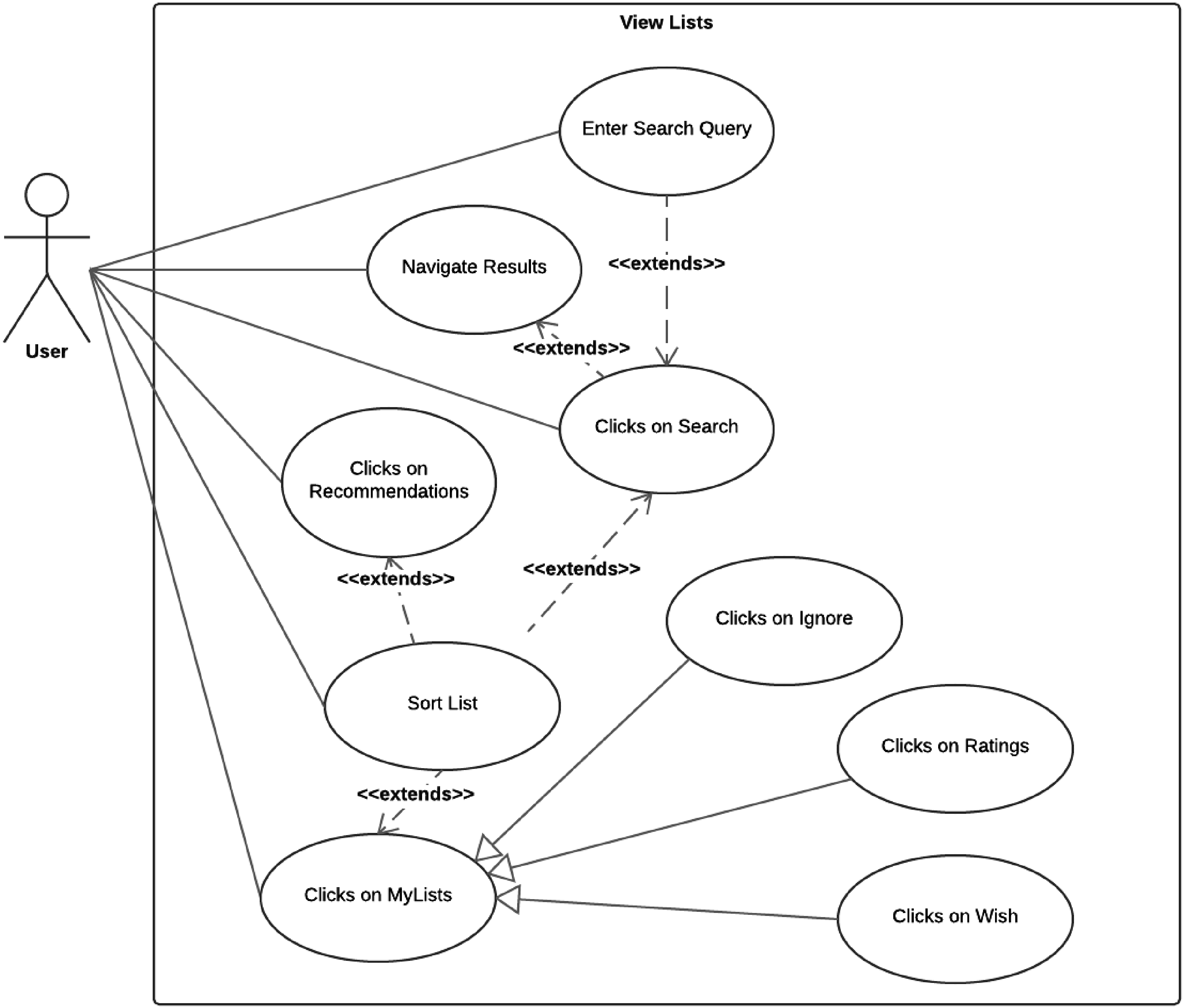
Functional Requirements		
ID	**Requirements**	**Priority**
FR-01	When user makes account, system verifies duplicate account does not exist	HIGH
FR-02	When user makes account, system verifies an acceptable password has been entered	HIGH
FR-03	System will remove a movie from your wishlist once you rate it	MEDIUM
FR-04	system will suggest 3-4 results while user is searching (autocomplete)	MEDIUM
FR-05	system will provide recommendations (top 20 - based on other users ratings)	HIGH
FR-06	System will provide URL to IMDB? of movie	LOW
FR-07	System will remove a movie from your ignore list once you rate it	LOW
FR-08	System will track previous/next set of search results to reduce errors	LOW
FR-09	System will not recommend movies from users ignore list	MEDIUM
FR-10	System will give a higher priority to wishlist movies when recommending	LOW

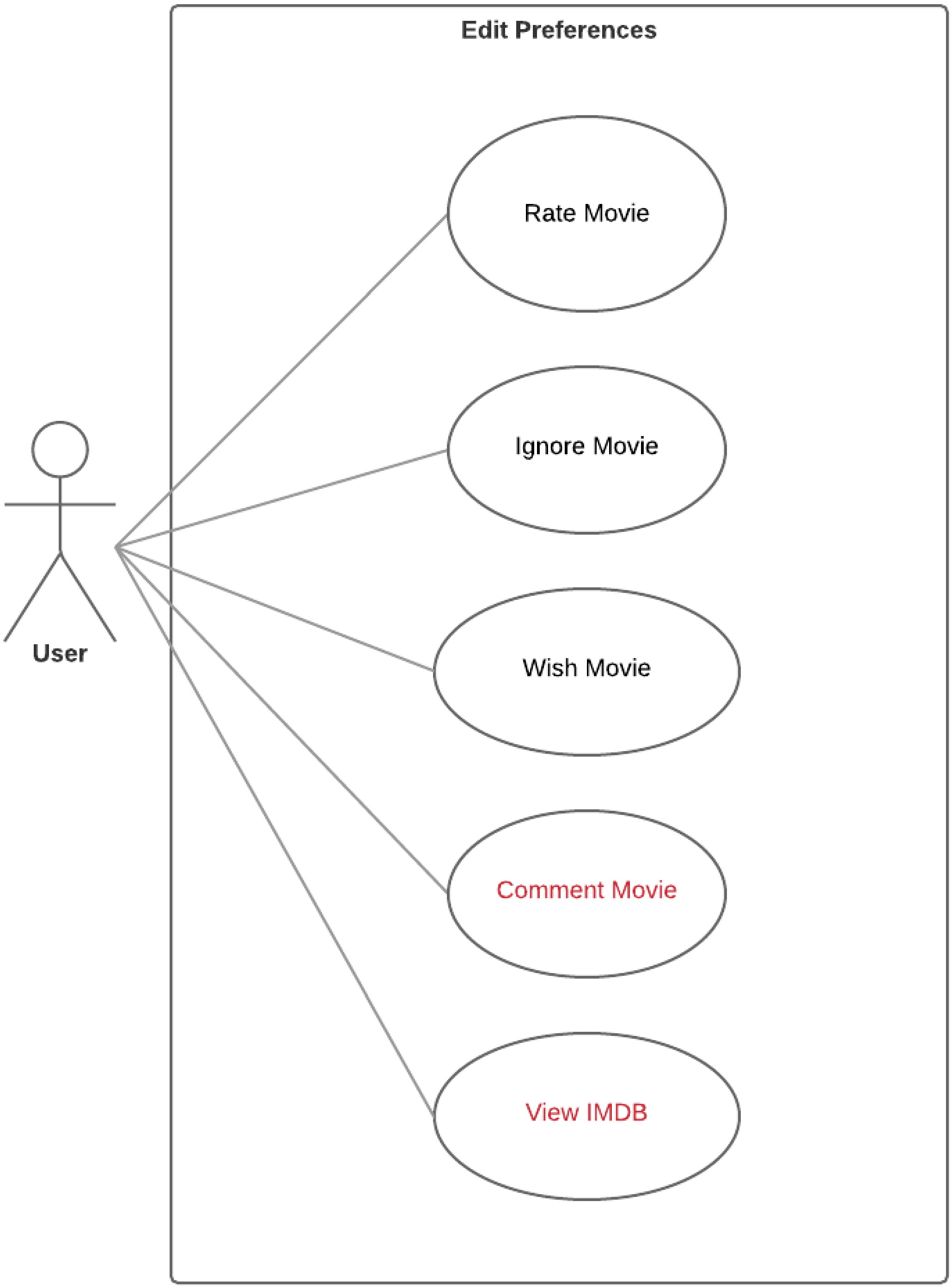
Non-Functional Requirements		
ID	**Requirements**	**Priority**
NFR-01	all users can log in	HIGH
NFR-02	system maintains all users information	HIGH
NFR-03	database is secure	HIGH
NFR-04	System will restrict number of failed log in attempts	LOW
NFR-05	System will auto generate an email to verify user email upon account creation	LOW
NFR-06	System will auto generate a "reset password" email	LOW
NFR-07	Users can only be logged in from one location	LOW
NFR-08	System will recognize when it is at beginning/end of list and prevent use of non-relevant buttons (e.g. previous/next)	LOW

Use Cases:

Actors: User




View Lists
User
Enter Search Query
Navigate Results
<<extends>>
<<extends>>
Clicks on Search
Clicks on Recommendations
<<extends>>
<<extends>>
Clicks on Ignore
Sort List
Clicks on Ratings
<<extends>>
Clicks on MyLists
Clicks on Wish


Edit Preferences
User
Rate Movie
Ignore Movie
Wish Movie
Comment Movie
View IMDB

Use Case Documents:

Use Case ID:	UR-01	
Use case name:	Create Account	
Description	User can create an account with a valid username, email address, and password.	
Actors:	User	
Pre-Condition:	User is on splash of website.	
Post-Condition:	The user’s name, username, email, and password are stored in the database.	
Frequency of Use:	As per user need.	
Flow of events:	Actor Action	System Action
	1. User enters the required field information; name, username, email, & password.	
	2. User clicks create account button	System takes user to their recommendations page
Exceptional Flow of Events	2. a. User omitted required information	System prompts user to enter missing information
	2. b. User enters duplicate id, or email	System informs user of duplicate conflict and asks user to change duplicated information
	2. c. User’s password does not meet the requirement	System prompts user to change password

Use Case ID:	UR-02	
Use case name:	Log in	
Description	User can log into their account.	
Actors:	User	
Pre-condition:	User has created an account	
Post-Condition:	User is logged in and can navigate to the various pages	
Frequency of Use:	As per user need.	
Flow of events:	Actor Action	System Action
	1. User enters username and password.	
	2. User clicks login.	User is taken to personal home page, User logged in flag is set in database
Exceptional Flow of events:	2.a. User does not fill in all required fields	System prompts user for missing information (username/password)
	2.b. Username & password combination do not match	System notifies user of incorrect password.

Use Case ID:	UR-03	
Use case name:	Edit Account	
Description	User can change their profile information.	
Actors:	User	
Pre-Condition:	User is logged in	
Post-Condition:	System updates the modified information in its database	
Frequency of Use:	As per user need.	
Flow of events:	Actor Action	System Action
	1. User clicks on edit profile link.	System takes user to their profile page
	2. User changes desired information	
	3. User clicks save changes button	System updates the info in its database.
Exceptional Flow of Events	3. a. User omitted required information	System prompts user to enter missing information
	3. b. User enters duplicate id, or email	System informs user of duplicate conflict and asks user to change duplicated information
	3. c. User’s password does not meet the requirement	System prompts user to change password

Use Case ID:	UR-04	
Use case name:	Delete account.	
Description	User can delete their own account	
Actors:	User	
Pre-Condition:	User is logged in	
Post-Condition:	System deletes the account information from its database.	
Frequency of Use:	As per user need.	
Flow of events:	Actor Action	System Action
	1. User clicks on edit profile link.	System takes user to the new page containing info about the user.
	2. User clicks on option to delete account	System asks for confirmation if user wants to delete account.
	3. User clicks on yes	System removes the account info from database. And takes user to main splash page.

Use Case ID:	UR-05	
Use case name:	Logout	
Description	User can log out of their account	
Actors:	User	
Pre-Condition:	User is logged in to their account.	
Post-Condition:	User is logged out of their account	
Frequency of Use:	As per user need.	
Flow of events:	Actor Action	System Action
	1. User clicks on log out button	System clears the login flag in database and takes the user to the main splash page

Use Case ID:	UR-06	
Use case name:	View Lists	
Description	User can click the recommendations, search, or “my lists” button to change views	
Actors:	User	
Pre-Condition:	User is logged in.	
Post-Condition:	User can edit preferences of a movie	
Frequency of Use:	As per user need.	
Flow of events:	Actor Action	System Action
Event 1:	1. User clicks on Recommendation toggle button	System displays list of top 20 suggestions based on this user’s ratings list.
Event 2:	1. User clicks on Search toggle button	System displays default search page of entire movie database
Event 3:	1. User clicks on “My list” toggle button	System displays default ratings list result, and displays the various user specific list filter options

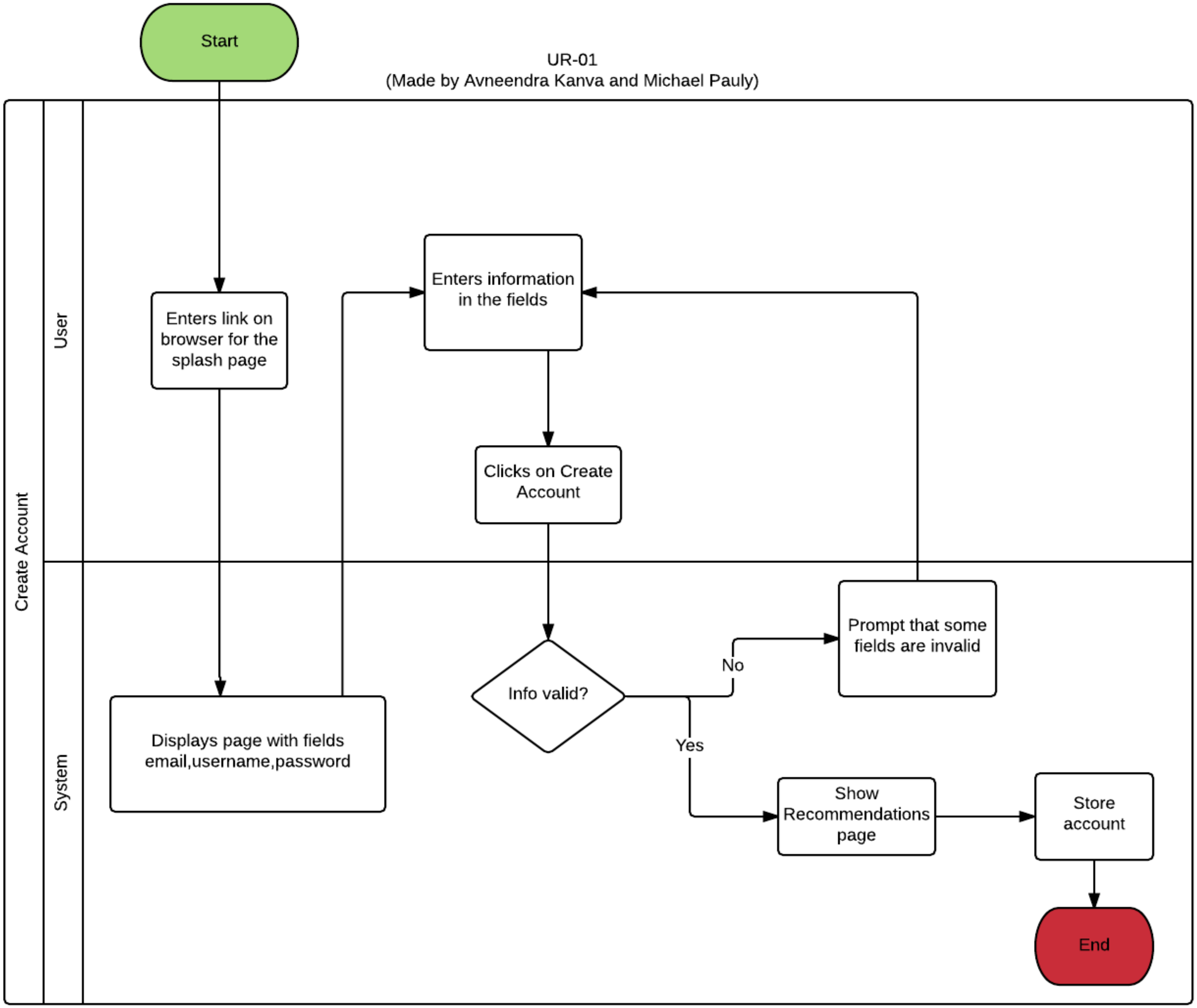
Use Case ID:	UR-07	
Use case name:	Edit preferences	
Description	Edit personal ratings/ignore/wish list of any displayed movie	
Actors:	User	
Pre-Condition:	User is logged in and system is showing any movie title information	
Post-Condition:	System updates the changes made by user in the database.	
Frequency of Use:	As per user need.	
Flow of events:	Actor Action	System Action
Event 1:	1. User toggles ignore option	System inverts previous setting, displays result
	2. User clicks save button	System updates database
Event 2:	1. User toggles wish list option	System inverts previous setting displays result
	2. User clicks save button	System updates database
Event 3:	1. User selects a rating from dropdown list	System displays user rating
	2. User clicks save button	System updates database

Use Case ID:	UR-08	
Use case name:	Enter search query	
Description	User can search for the movie name and update or add it to any list	
Actors:	User	
Pre-Condition:	User is logged in and is toggled to search mode	
Post-Condition:	A set number of results is displayed based on the query	
Frequency of Use:	As per user need.	
Flow of events:	Actor Action	System Action
	1. User types a movie name in the search bar.	System provides 3-4 suggestions of movie names as user types. (autocomplete)
Alternative Scenario:	2. a. User clicks on any of these 3-4 movie names.	System displays exact match
	2. b. User continues typing and clicks on search button (same as hitting enter)	Search results are displayed with default top 20 nearest matches showing
	2. c. User clears search results	System displays default search page

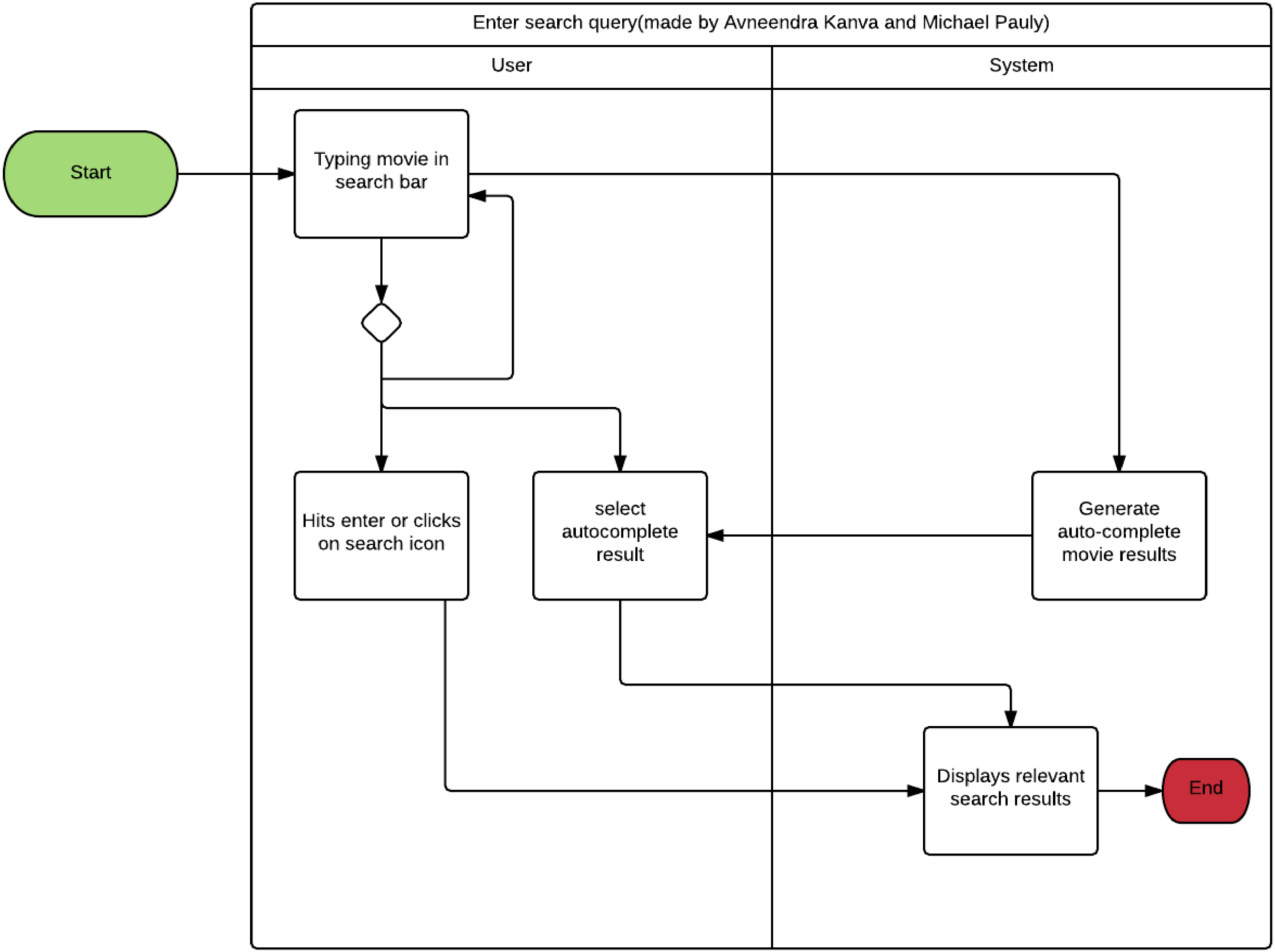
Use Case ID:	UR-09	
Use case name:	Navigate MyLists	
Description	User can view their personal ratings, ignore, and wish lists	
Actors:	User	
Pre-Condition:	User is logged in and toggled to MyLists	
Post-Condition:	System shows entire list of users edited movies	
Frequency of Use:	As per user need.	
Flow of events:	Actor Action	System Action
Alternative flow of events:	1. a. User clicks on Ignore	System displays entire list of user’s ignored movies
	1. b. User clicks on Wish	System displays entire list of user’s wishlist movies
	1. c. User clicks on Ratings	System displays entire list of user’s rated movies

Use Case ID:	UR-12	
Use case name:	Navigate Search results	
Description	User can display more results beyond the first 20 movies	
Actors:	User	
Pre-Condition:	User is logged in and on search page	
Post-Condition:	System shows 20 movies	
Frequency of Use:	As per user need.	
Flow of events:	Actor Action	System Action
	1. User clicks on view next set button	System returns results of next 20 movies based on sort criteria
Alternative flow of events	2. User clicks on view previous set button	System returns results of previous 20 movies based on sort criteria

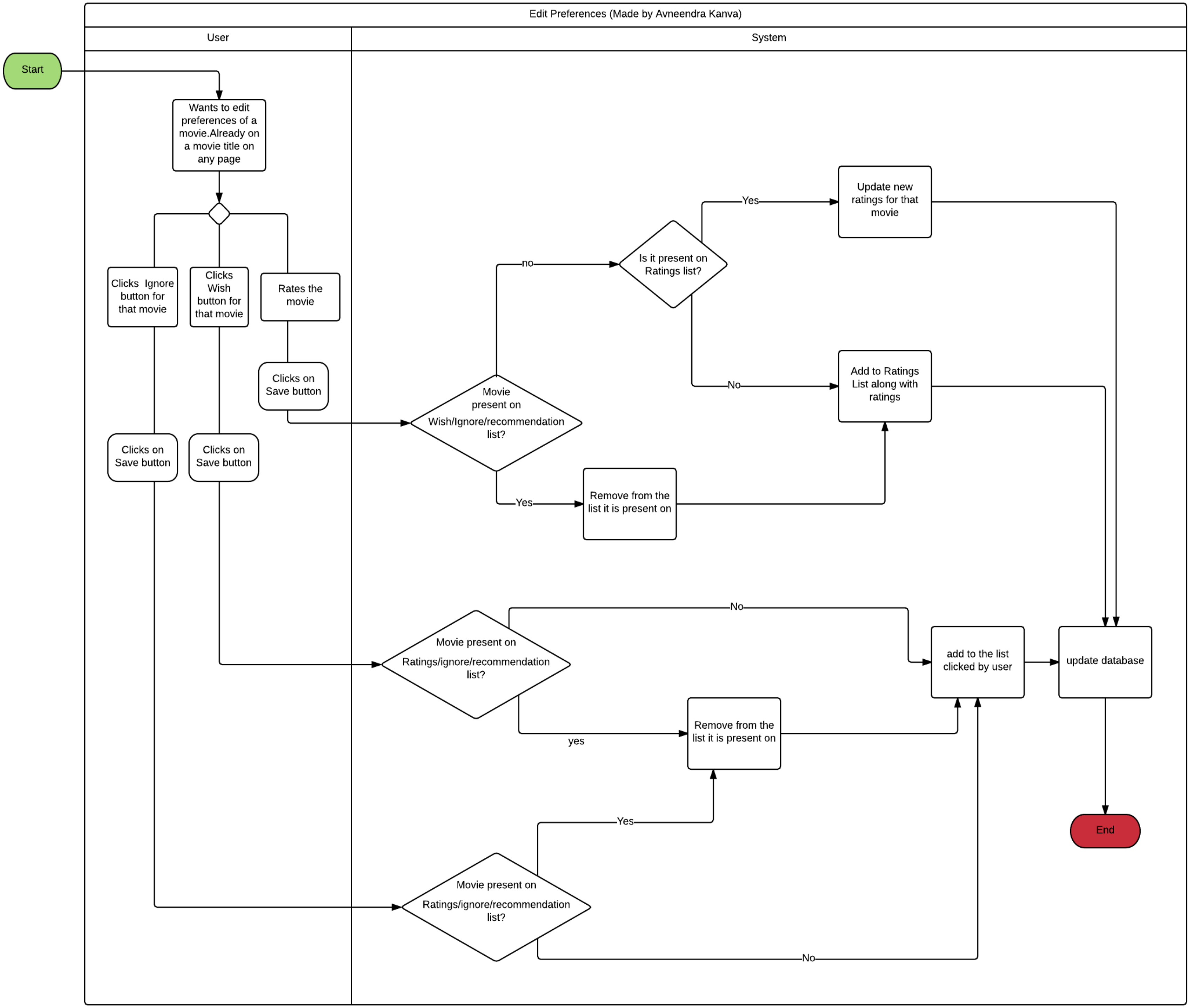
Activity Diagrams:


Start
UR-01
(Made by Avneendra Kanva and Michael Pauly)
Create Account
User
System
Enters link on browser for the splash page
Enters information in the fields
Clicks on Create Account
Displays page with fields email,username,password
Info valid?
No
Yes
Prompt that some fields are invalid
Show Recommendations page
Store account
End

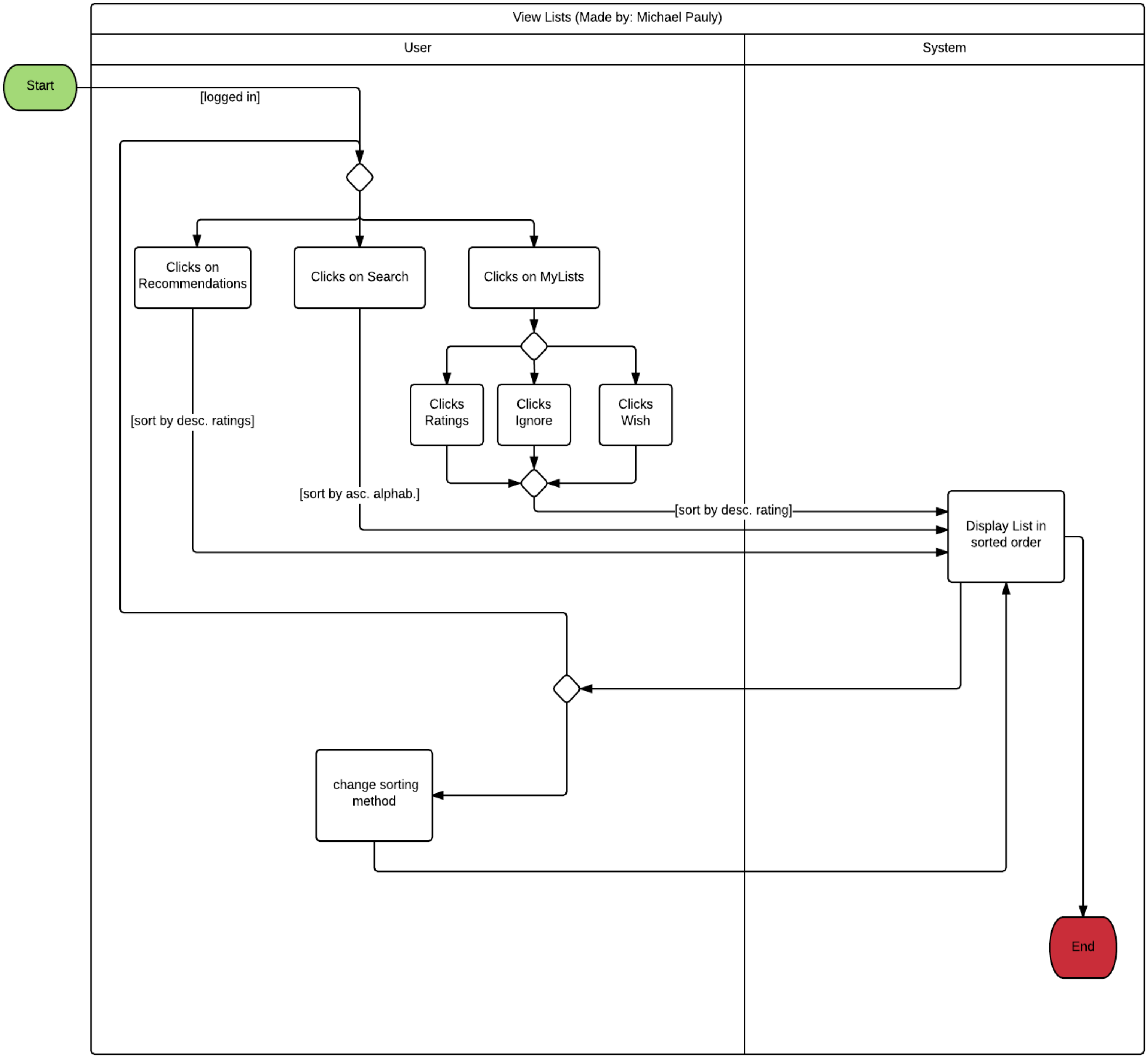
UR-08


Enter search query(made by Avneendra Kanva and Michael Pauly)
User
System
Start
Typing movie in search bar
Hits enter or clicks on search icon
select autocomplete result
Generate auto-complete movie results
Displays relevant search results
End

UR-07


Edit Preferences (Made by Avneendra Kanva)
User
System
Start
Wants to edit preferences of a movie.Already on a movie title on any page
Clicks Ignore button for that movie
Clicks Wish button for that movie
Rates the movie
Clicks on Save button
Clicks on Save button
Clicks on Save button
Movie present on Wish/Ignore/recommendation list?
no
Is it present on Ratings list?
Yes
Update new ratings for that movie
No
Add to Ratings List along with ratings
Yes
Remove from the list it is present on
Movie present on Ratings/ignore/recommendation list?
No
add to the list clicked by user
update database
yes
Remove from the list it is present on
Yes
Movie present on Ratings/ignore/recommendation list?
No
End

UR-06 and UR-09



Data Storage:

All data will be stored in a MySQL database. This will be housed on the same server as the system itself.

Classes:

DatabaseConnection - class to bridge between system and database.

List – classes that will temporarily store data until user saves changes.

The following tables will store the information in the database.

UserInfo

uID (int)	**fname (string)**	**lname (string)**	**email (string)**	**username (string)**	**password (string)**
1					
:					

Movies

mID (string)	**title (string)**
movID01	
:	

ActiveUsers

uID (int)	**active (boolean)**	**lastlogin (date)**
1		
:		

Ratings

uID (int)	**mID (string)**	**rating (int)**

Wish

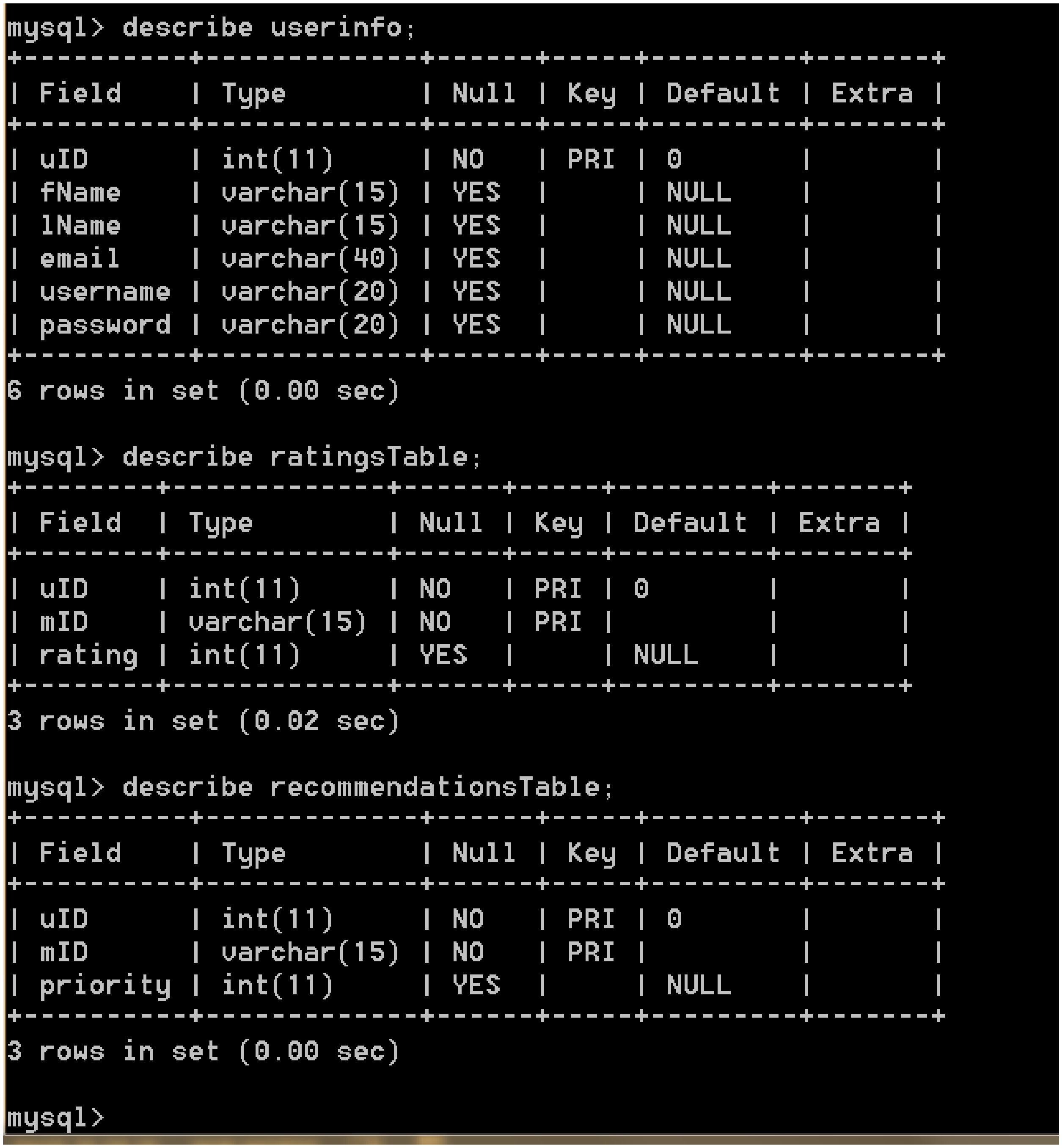
uID (int)	**mID (string)**	**state (boolean)**

Ignore

uID (int)	**mID (string)**	**state (boolean)**

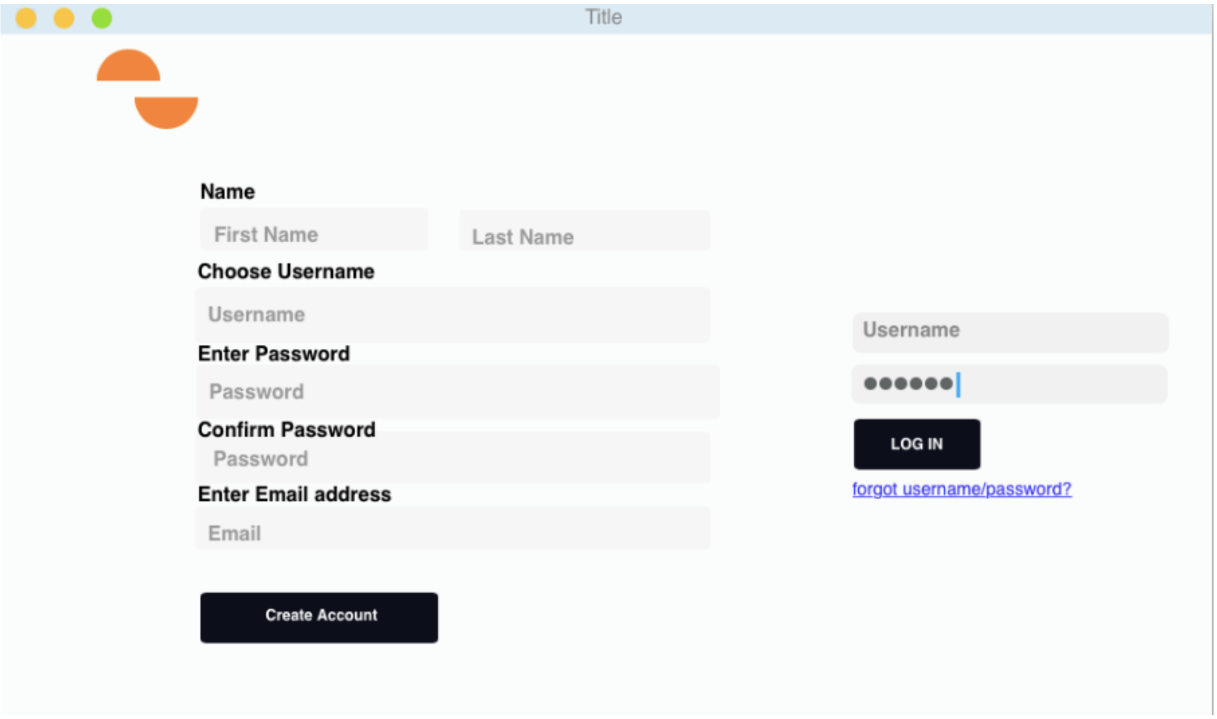
Recommendations

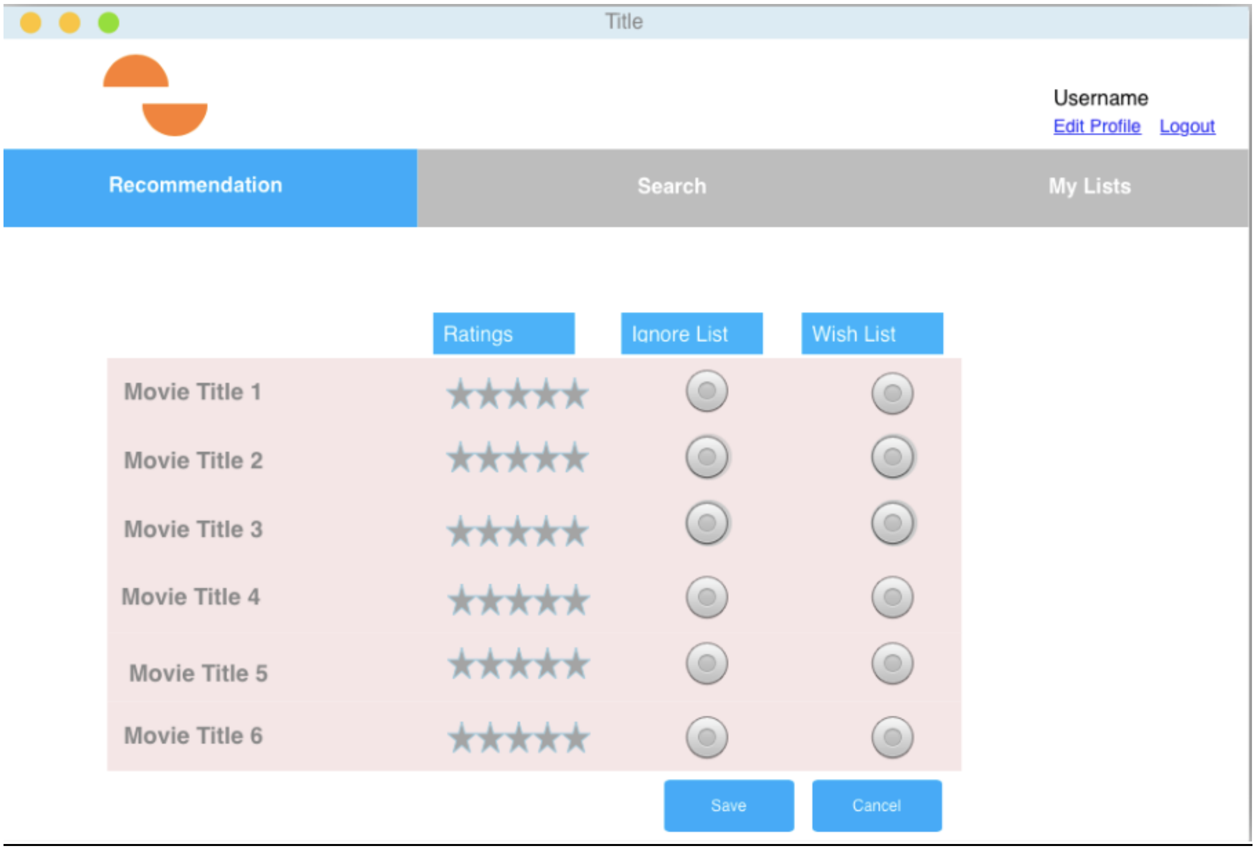
uID (int)	**mID (string)**	**priority (int)**



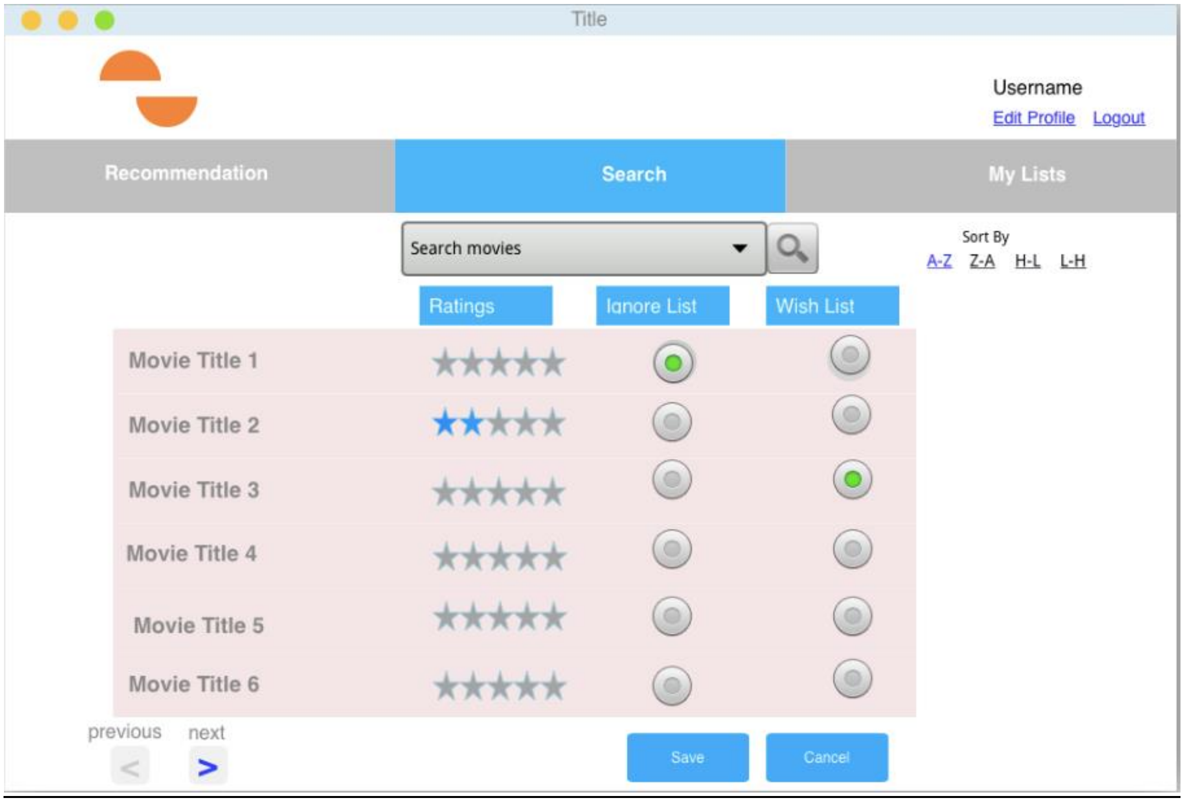
```
mysql> describe userinfo;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| uID      | int(11)     | NO   | PRI | 0       |       |
| fName    | varchar(15) | YES  |     | NULL    |       |
| lName    | varchar(15) | YES  |     | NULL    |       |
| email    | varchar(40) | YES  |     | NULL    |       |
| username | varchar(20) | YES  |     | NULL    |       |
| password | varchar(20) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> describe ratingsTable;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| uID    | int(11)     | NO   | PRI | 0       |       |
| mID    | varchar(15) | NO   | PRI |         |       |
| rating | int(11)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)

mysql> describe recommendationsTable;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| uID      | int(11)     | NO   | PRI | 0       |       |
| mID      | varchar(15) | NO   | PRI |         |       |
| priority | int(11)     | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

<u>UI Mockups:</u>





	Ratings	Ignore List	Wish List
Movie Title 1	★★★★★	○	○
Movie Title 2	★★★★★	○	○
Movie Title 3	★★★★★	○	○
Movie Title 4	★★★★★	○	○
Movie Title 5	★★★★★	○	○
Movie Title 6	★★★★★	○	○



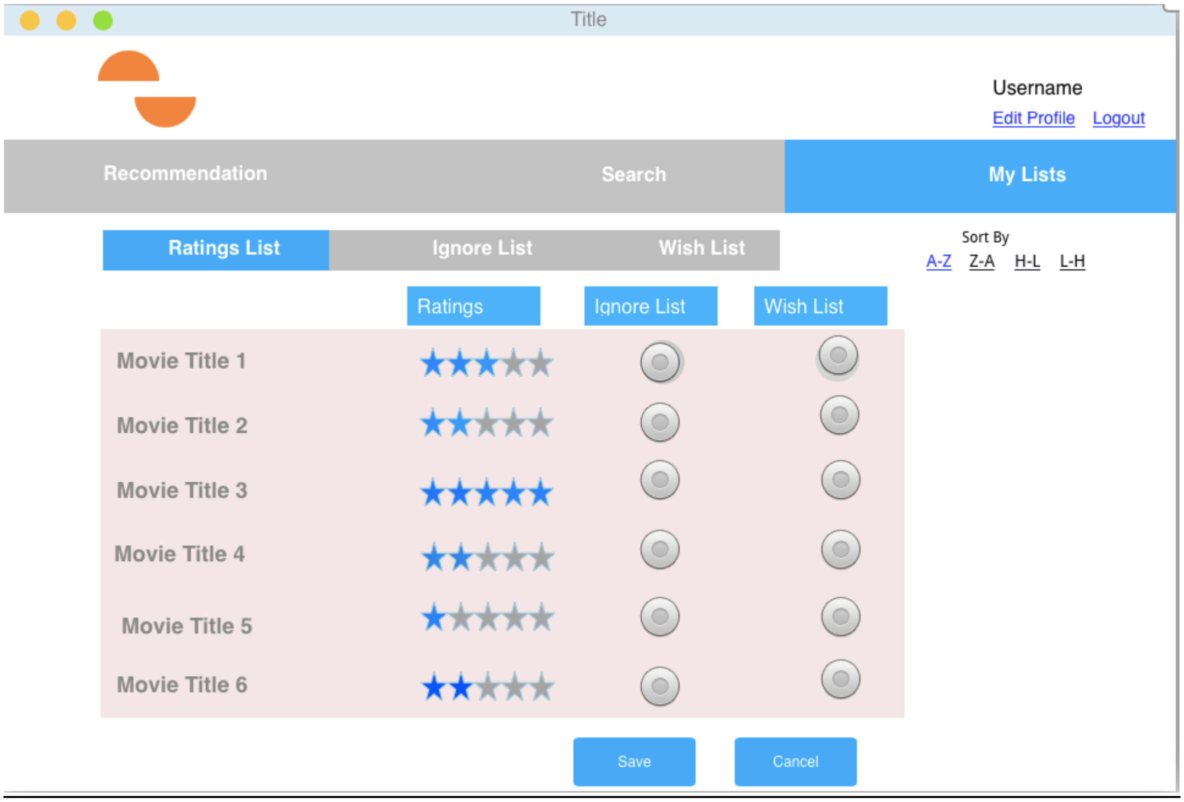
Title

Username

Edit Profile Logout

Sort By
A-Z Z-A H-L L-H

	Ratings	Ignore List	Wish List
Movie Title 1	☆☆☆☆☆	●	○
Movie Title 2	★★☆☆☆	○	○
Movie Title 3	☆☆☆☆☆	○	●
Movie Title 4	☆☆☆☆☆	○	○
Movie Title 5	☆☆☆☆☆	○	○
Movie Title 6	☆☆☆☆☆	○	○



Title

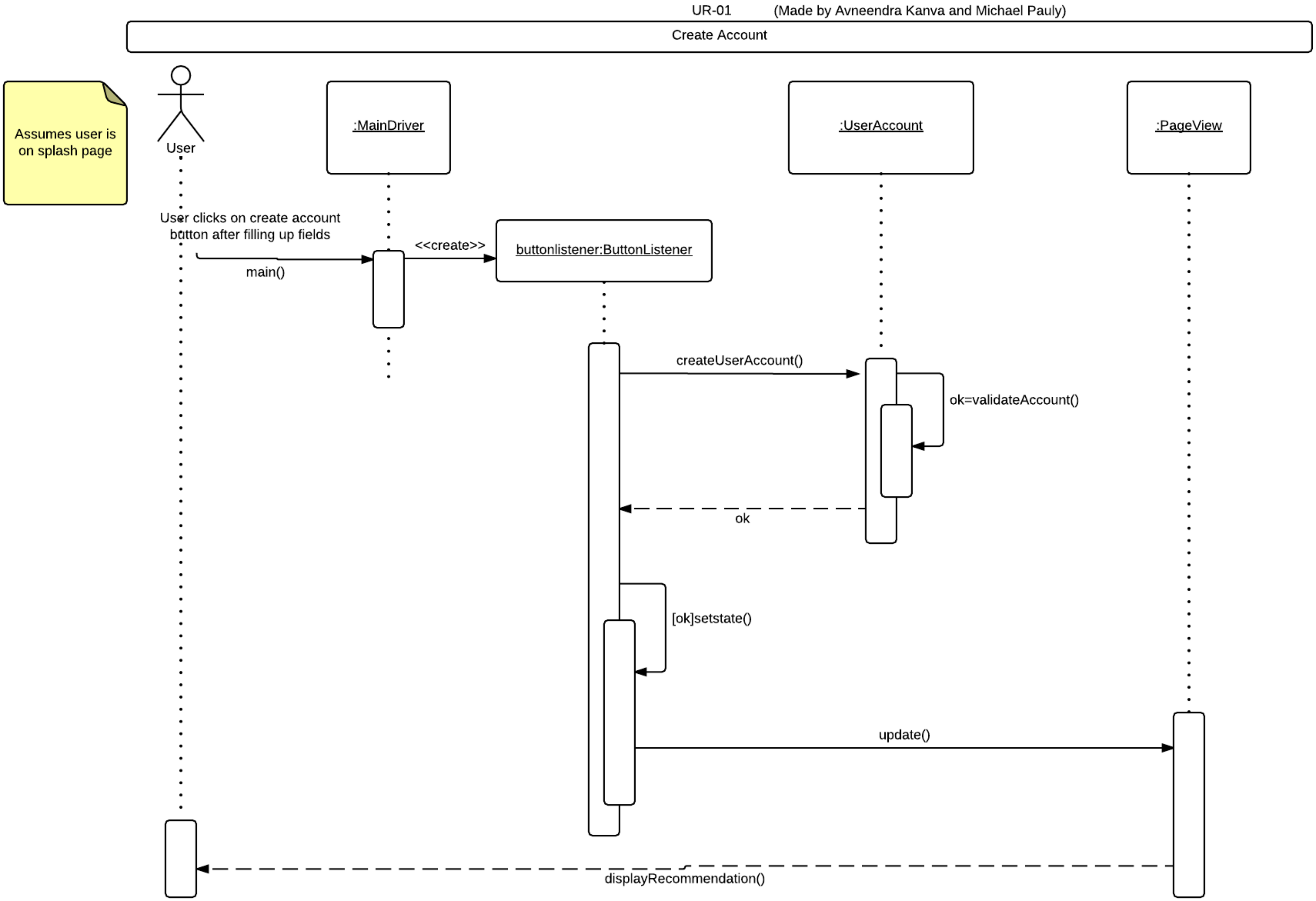
Username

Edit Profile Logout

Sort By
A-Z Z-A H-L L-H

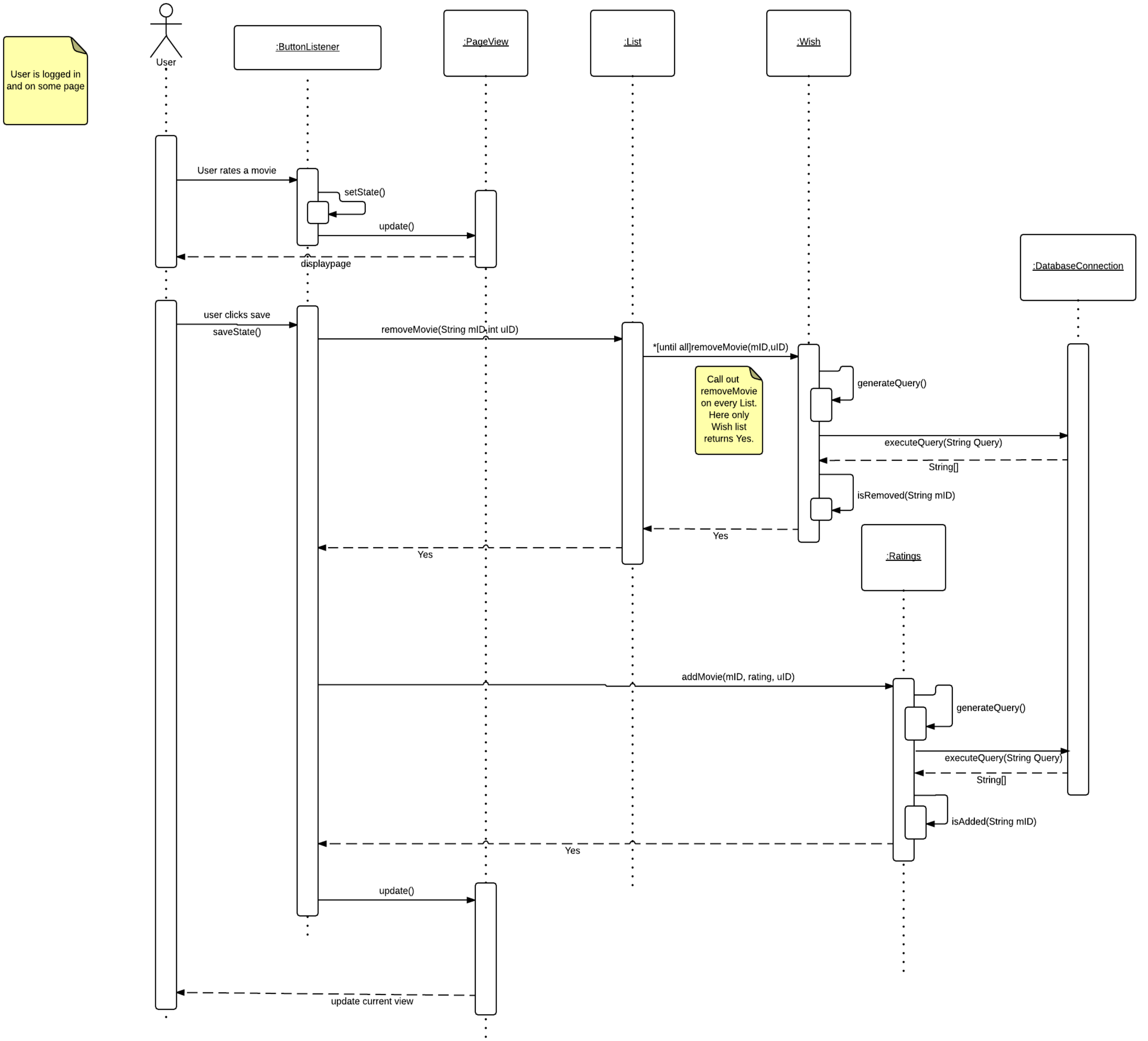
	Ratings	Ignore List	Wish List
Movie Title 1	★★★☆☆	○	○
Movie Title 2	★★☆☆☆	○	○
Movie Title 3	★★★★★	○	○
Movie Title 4	★★☆☆☆	○	○
Movie Title 5	★☆☆☆☆	○	○
Movie Title 6	★★☆☆☆	○	○

User Interactions:


UR-01
(Made by Avneendra Kanva and Michael Pauly)
Create Account
Assumes user is on splash page
User
:MainDriver
:UserAccount
:PageView
User clicks on create account button after filling up fields
<<create>>
buttonlistener:ButtonListener
main()
createUserAccount()
ok=validateAccount()
ok
[ok]setstate()
update()
displayRecommendation()

UR-07 (Made by Avneendra Kanva)

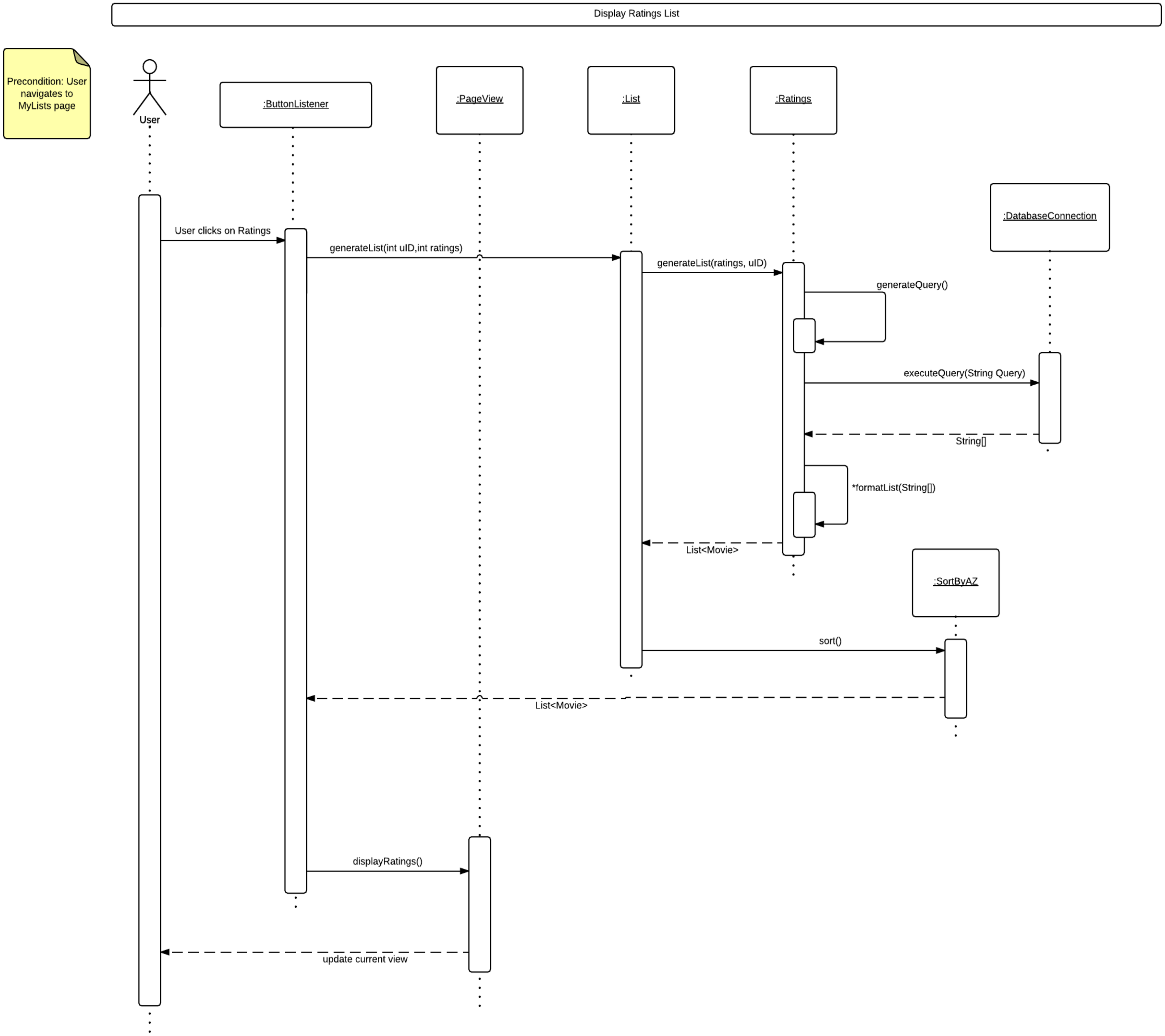
Rate a Movie that is on Wish List



User is logged in and on some page

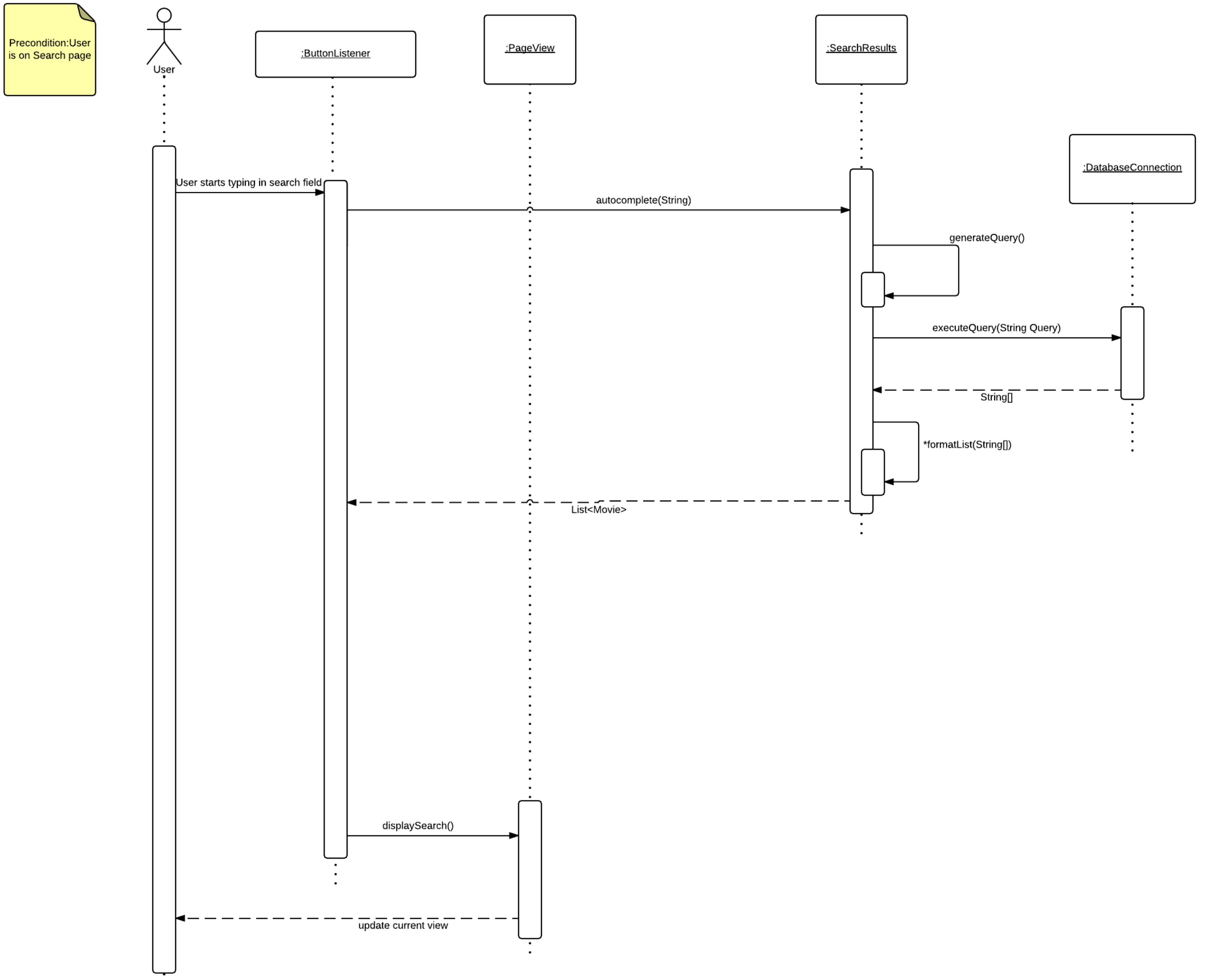
Call out removeMovie on every List. Here only Wish list returns Yes.

UR-09 (Made by Michael Pauly)

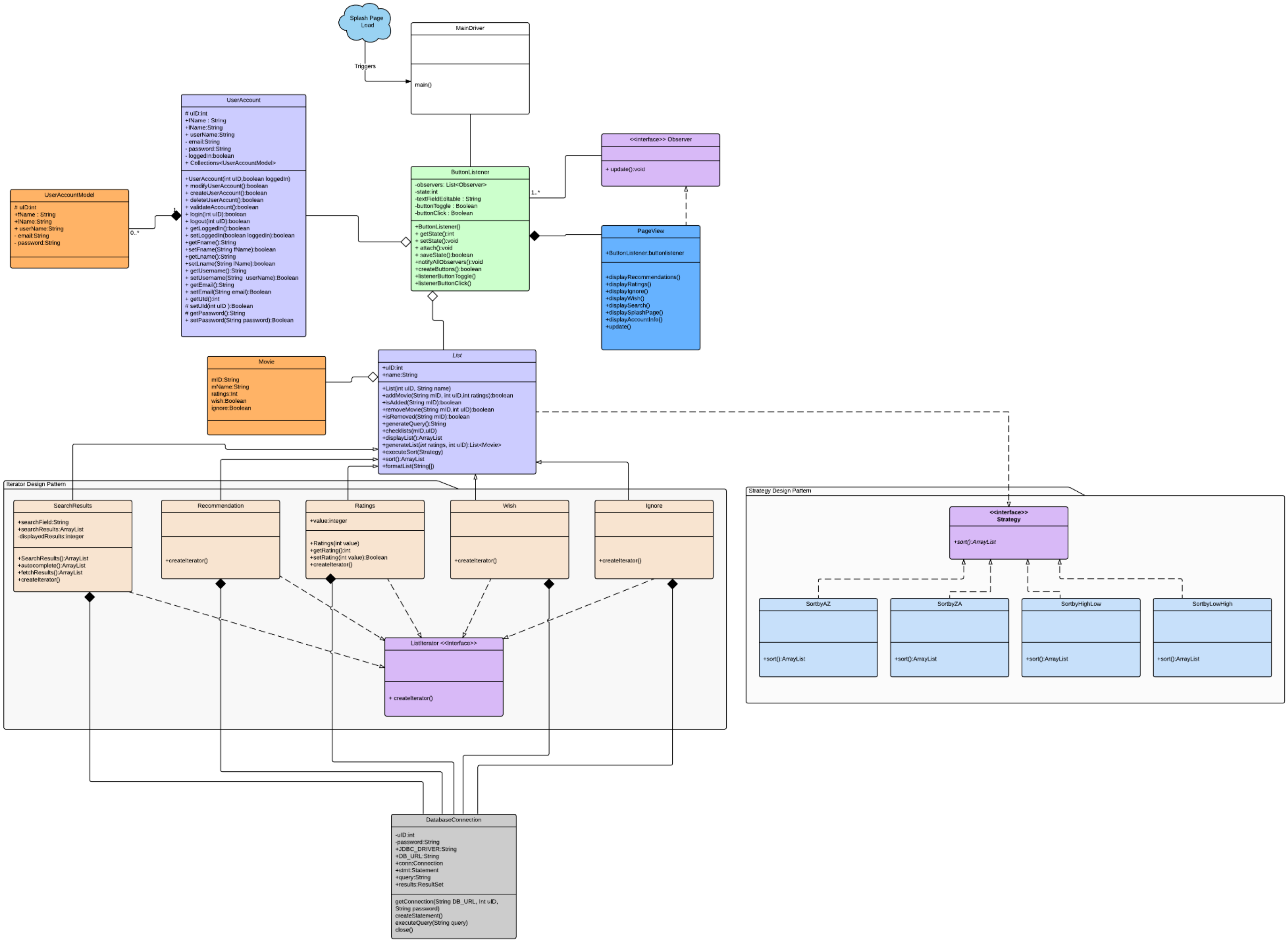


UR-08 (Made by Avneendra Kanva and Michael Pauly)

Search Query



Class Diagram:


Splash Page Load
Triggers
MainDriver
main()
UserAccountModel
# uID:int
+fName : String
+lName:String
+ userName:String
- email:String
- password:String
0..*
1
UserAccount
# uID:int
+fName : String
+lName:String
+ userName:String
- email:String
- password:String
- loggedIn:boolean
+ Collections<UserAccountModel>
+UserAccount(int uID,boolean loggedIn)
+ modifyUserAccount():boolean
+ createUserAccount():boolean
+ deleteUserAccunt():boolean
+ validateAccount():boolean
+ login(int uID):boolean
+ logout(int uID):boolean
+ getLoggedIn():boolean
+ setLoggedIn(boolean loggedIn):boolean
+getFname():String
+setFname(String fName):boolean
+getLname():String
+setLname(String lName):boolean
+ getUsername():String
+ setUsername(String userName):Boolean
+ getEmail():String
+ setEmail(String email):Boolean
+ getUId():int
# setUId(int uID):Boolean
# getPassword():String
+ setPassword(String password):Boolean
ButtonListener
-observers: List<Observer>
-state:int
-textFieldEditable : String
-buttonToggle : Boolean
-buttonClick : Boolean
+ButtonListener()
+ getState():int
+ setState():void
+ attach():void
+ saveState():boolean
+notifyAllObservers():void
+createButtons():boolean
+listenerButtonToggle()
+listenerButtonClick()
1..*
<<interface>> Observer
+ update():void
PageView
+ButtonListener:buttonlistener
+displayRecommendations()
+displayRatings()
+displayIgnore()
+displayWish()
+displaySearch()
+displaySplashPage()
+displayAccountInfo()
+update()
Movie
mID:String
mName:String
ratings:Int
wish:Boolean
ignore:Boolean
List
+uID:int
+name:String
+List(int uID, String name)
+addMovie(String mID, int uID,int ratings):boolean
+isAdded(String mID):boolean
+removeMovie(String mID,int uID):boolean
+isRemoved(String mID):boolean
+generateQuery():String
+checklists(mID,uID)
+displayList():ArrayList
+generateList(int ratings, int uID):List<Movie>
+executeSort(Strategy)
+sort():ArrayList
+formatList(String[])
Iterator Design Pattern
SearchResults
+searchField:String
+searchResults:ArrayList
-displayedResults:integer
+SearchResults():ArrayList
+autocomplete():ArrayList
+fetchResults():ArrayList
+createIterator()
Recommendation
+createIterator()
Ratings
+value:integer
+Ratings(int value)
+getRating():int
+setRating(int value):Boolean
+createIterator()
Wish
+createIterator()
Ignore
+createIterator()
ListIterator <<Interface>>
+ createIterator()
Strategy Design Pattern
<<interface>>
Strategy
+sort():ArrayList
SortbyAZ
+sort():ArrayList
SortbyZA
+sort():ArrayList
SortbyHighLow
+sort():ArrayList
SortbyLowHigh
+sort():ArrayList
DatabaseConnection
-uID:int
-password:String
+JDBC_DRIVER:String
+DB_URL:String
+conn:Connection
+stmt:Statement
+query:String
+results:ResultSet
getConnection(String DB_URL, Int uID,
String password)
createStatement()
executeQuery(String query)
close()