# FlixBook: A Movie and Tv Tracker from the Future

**Detailed Design**

**<COP 4331C, Fall, 2014>**

---

| Version | Date | Who | Comment |
|---|---|---|---|
| **v0.0** | **10/18/2014** | **Lakshmidhar C.** | **Began Work on Document** |
| **v0.1** | **10/19/2014** | **Lakshmidhar C.** | **Added Design Issues** |
| **v0.2** | **10/23/2014** | **Michael W.** | **Added Data Flow and State Machine Diagrams.** |
| **v0.3** | **10/23/2014** | **Roman L.** | **Added Class Diagram.** |

Team Name:

Group 15

Team Members(click on name for website):


Roman Larinov - rlarionov@knights.ucf.edu

Ramses Mederos - ramsesmederos@knights.ucf.edu

Lakshmidhar Chigurupati - lakshmidharc@knights.ucf.edu

Michael Wahlberg - wahlberg2012@knights.ucf.edu

Michael Pittman - m.pittman@knights.ucf.edu

Benjamin Kirksey -


---

**Table of Contents**

**Design Issues**:

During our planning phase, we decided that we would use a CSS Framework called Bootstrap for creating a website design that is easy to use and has a very unique appeal to the user. Some of the issues related with using this framework is that it limits our control over the CSS of the website as we're required to use specific class names. This is an issue because it means that we could encounter compatibility issues when using our own custom css due to using the same class names. A solution to this problem is to use the framework to create the website first and then add custom CSS afterwards so that we are aware of what class names are already being used.

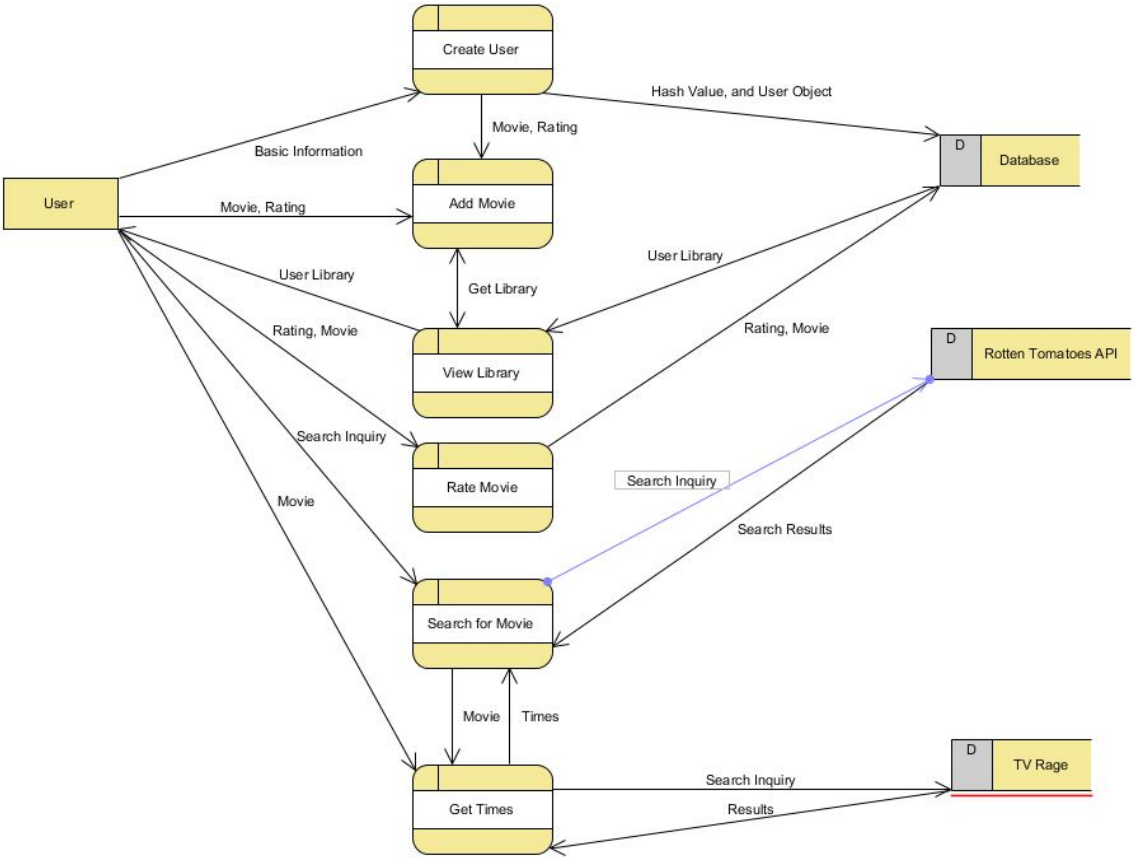
We decided to integrate Bootstrap into our website design because it allows us focus more on the backend functionality so that the business logic is as efficient and as bug-free as possible. The use of Bootstrap also allows us to ensure a pleasant user experience because Bootstrap is a widely used CSS framework by many large companies.

We will also be using Express.js, a web application framework. We will be using this because it allows us to interact with our node.js backend in a very intuitive manner. This also allows us to make the usage of the website a seamless process and therefore allow us to provide an excellent user experience. This is crucial because it allows for us retain as many users as possible, which in turn allows us to maintain the product for an extended time period. A potential risk with using this framework is that it might have compatibility issues with some remote parts of our backend. This could potentially be a major risk as it would mean that the product might not function correctly even though the business logic is correct. A possible solution to this problem would to be test all the basic functionality of the product upon edition of new frontend code.

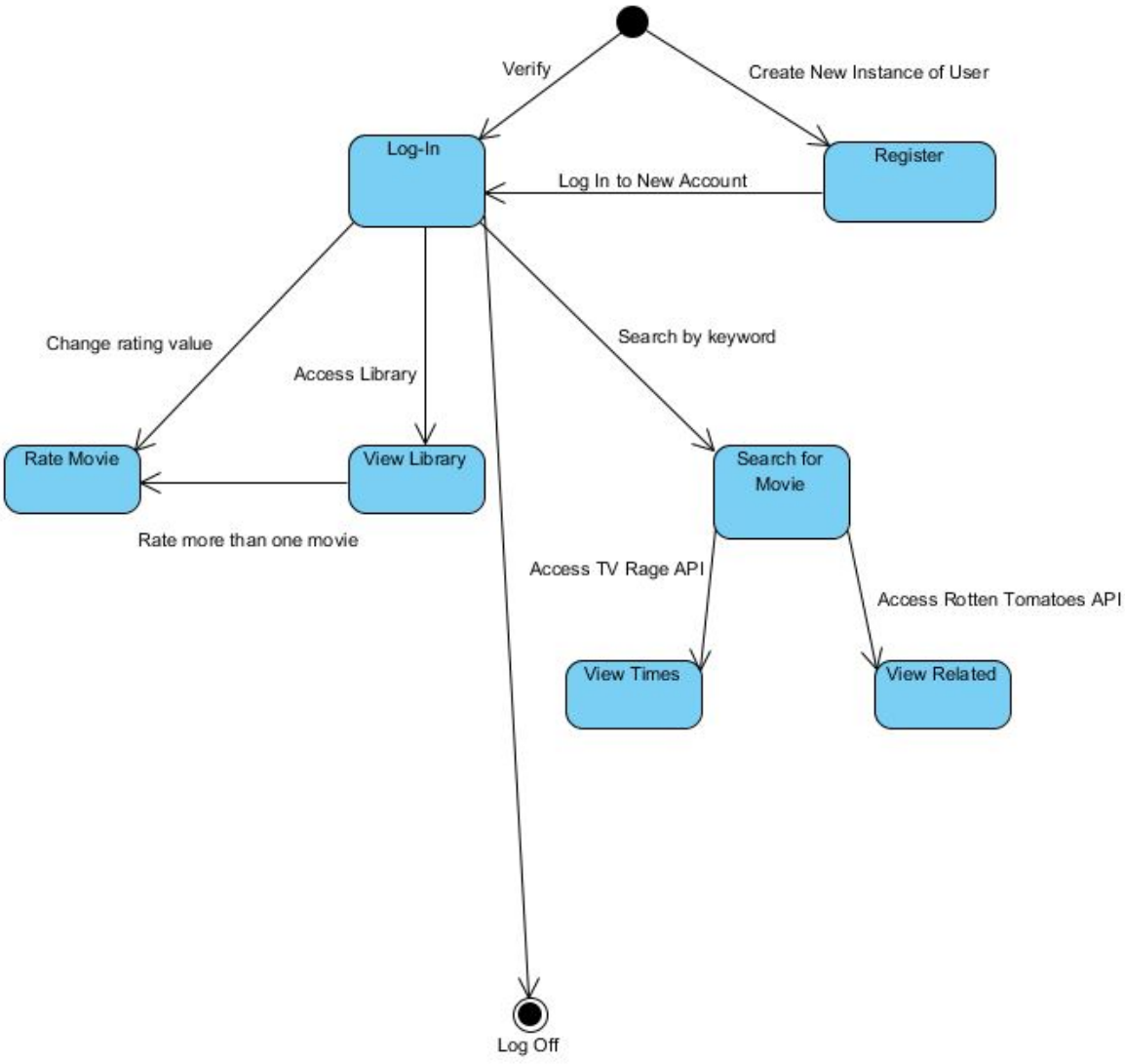
To summarize, the whole of the frontend is structured to allow for an excellent user experience as well as an intuitive user interface. These were the primary concerns when coming up with the design plans. As previously stated, the user experience and intuitive user interface were primary concerns because it is the first step towards attracting and retaining users. Based on this and our research about various user interfaces, we came to the conclusion that the best way to achieve our goals would be to use the CSS Framework, Bootstrap, and the Web Application Framework, Express.js.

## Detailed Design Information

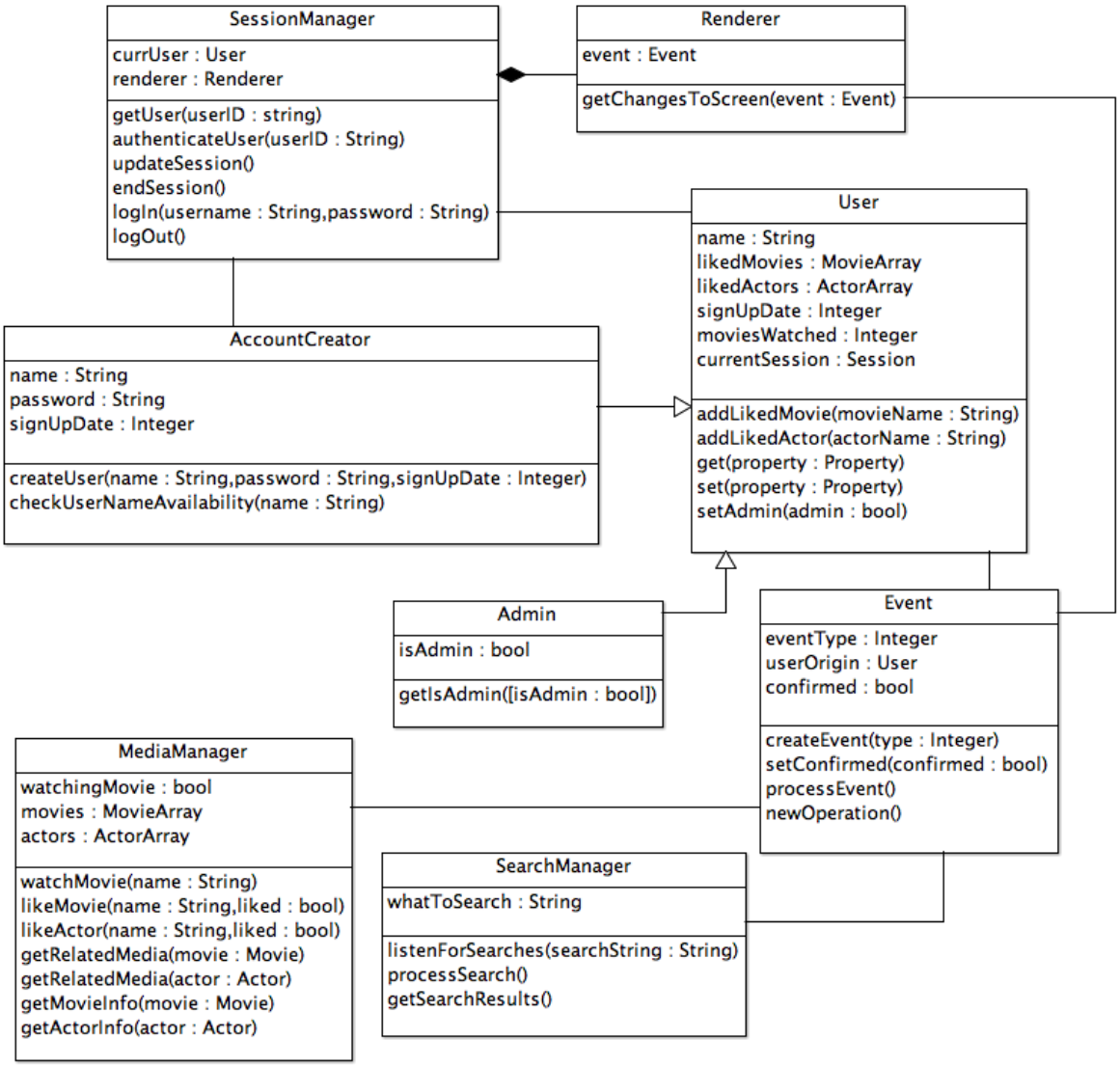
## Data Flow Diagram



**State Machine Diagram**



**Class Diagram**



**Trace of Requirements to Design**

| Requirement | Trace |
|---|---|
| Users must have the ability to create accounts and have the ability to login and logout easily. | When 'create account' button is clicked on the user interface a new 'User Account' entity is created in the the User Account Database. When the 'login' button is clicked on the user interface and authentication is verified the FlixBook server will access the User Account Database and send the specified information to the client. When the 'logout' button is clicked on the user interface the user will be brought to a generic screen for non-user interaction. |
| Users must have the ability to search movies based on title, actors, year, or genre. | The Rotten Tomatoes API, which our web application is using, allows for queries of movies based on movie title, movie actors, movie release date, and/or movie genre. |
| Users must have the ability to save movies to a library that can only be accessed through the user account on which it was created. | Each 'User Account' entity in the user account database stores a list of the movies that the user has saved into his/her library. The library (list of movies) is presented to the user through the user interface. |
| The web application must have the ability to recommend movies to the users. | A list of recommended movies is presented to the user on the user's homepage. The algorithm that picks movies to recommend is based off of the movies that the user has saved into his/her library. |
| The web application must have the ability to alert/remind users of when specified movies are coming on television or being released into theatres. | Once a day the FlixBook server will search the Rotten Tomatoes API for upcoming movies. If a movie is coming out in theatres in one week the server will send an email to all users who have signed up for an alert for that movie. |
| The interface must have a guest user home page which will allow the user to login or create an account. | The initial interface that every user is brought to when they enter the FlixBook URL into the browser is the guest user interface. This user interface has buttons for 'login' and 'create account'. The 'login' button will call the login() method on the server and the 'create account' button will call the createAccount() method on the server. |
| Once a user logs in they should be brought to a registered user home page that will show them recommended movies based on the movies the user's library. | Once a user logs in to FlixBook, the information contained in the specified 'User Account' entity in the user account database is sent from the FlixBook server to the client. The client will then build a customized user interface which will present recommended movies to the user. |

| | |
|---|---|
| The registered user home page should also have a link to the user's library and account settings. | The registered user home page user interface includes both 'Library' and 'Account Settings' buttons. When the 'Library' button is clicked the client will request, from the server, the information needed to build the library user interface for the user. When the 'Account Settings' button is clicked the client will build a user interface which allows the user to update settings. When the user is done changing the setting he/she will click an 'Update Settings' button. When this button is clicked the client will send the server all of the new settings and the server will update the settings attributes in the user account database. |
| A screen for searching for movies must be available to both guest users and to registered users. The input for the search should be a string describing an actor, movie title, or movie genre. | The user interfaces for both guest-users and registered-users will have a 'Search Movie' input box. The string from this input box will be sent to the server. The server will use the string to search the Rotten Tomatoes API and will return the information to the client. |
| A button to return the user to the home page should be available at any time when navigating the web application. | All pages on the FlixBook web application have a 'FlixBook' logo button. When a user clicks this button the server sends the client the appropriate information to build the registered user's custom home page. If a guest user clicks the button the server sends the client the information to build the generic non-user home page. |