

1. Introduction

Movie Central is a subscription-based media service application which offers online streaming of movies. This service has two types of users, admins and customers. Admin has rights to manage movies and to view customer or movie stats. A registered customer can watch movies and subscribe himself for watching paid movies. This application is a Restful web application with Spring Boot framework. The User Interface is developed with React JS and Redux while, database is hosted on Amazon RDS SQL server. The application is hosted on URL : <http://ec2-34-209-69-136.us-west-2.compute.amazonaws.com>

2. Bonus Feature Implemented

1. Movie Recommendation: Movie recommendations are showed under the tab ‘Recommended for you’. The recommendations are based on the movies customer has watched. On the basis of particular director’s or actor’s movie customer has watched, other movies directed by the same director or actor are showed as recommendations to customer.
2. Implemented elastic search for full text search.

3. High Level and Component Level Design

High level design for the application is:

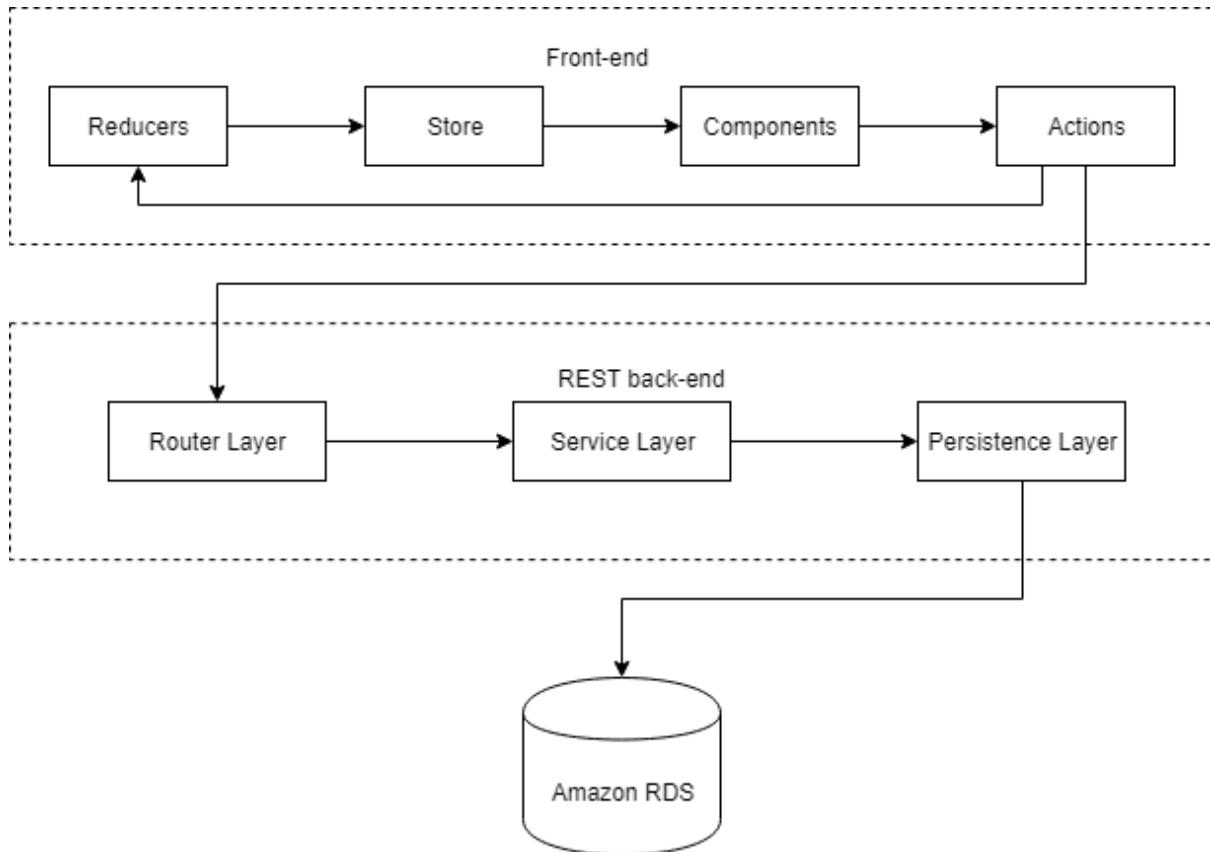


Fig 3.1: High Level Design

Component level design is:

Fig 3.2: Component Level Design

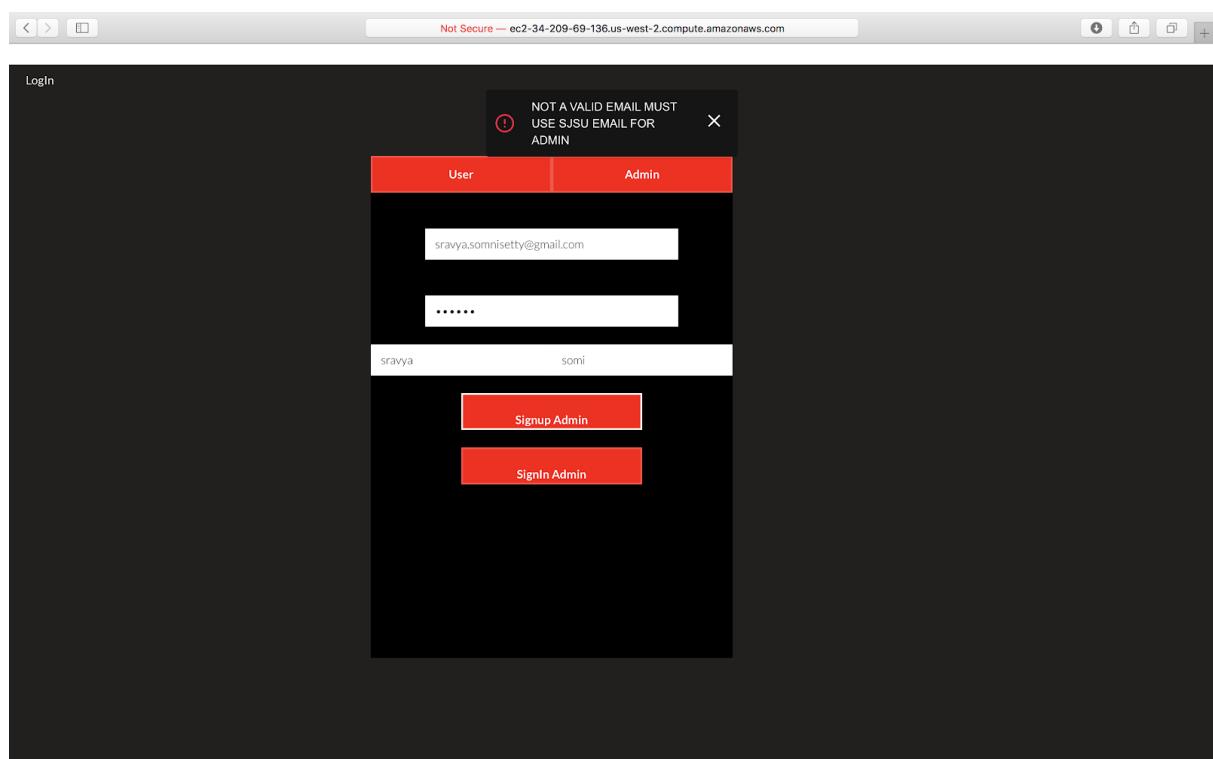
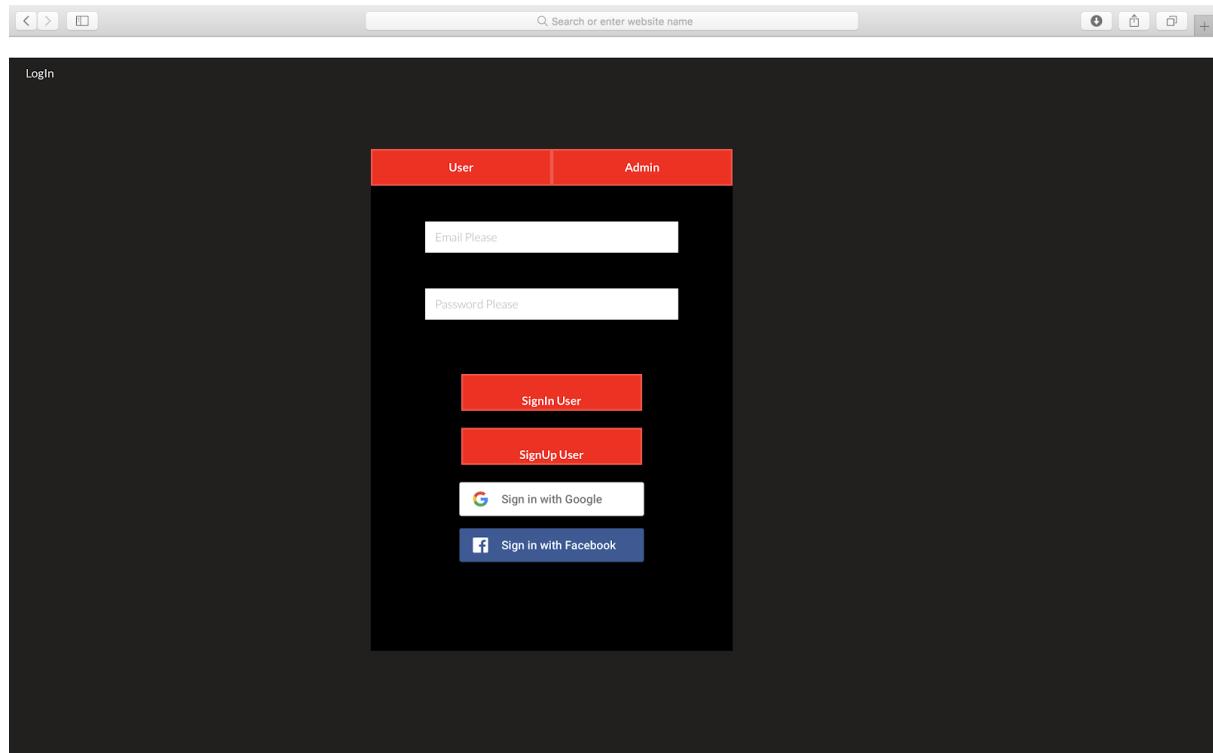
4. Technology Choices

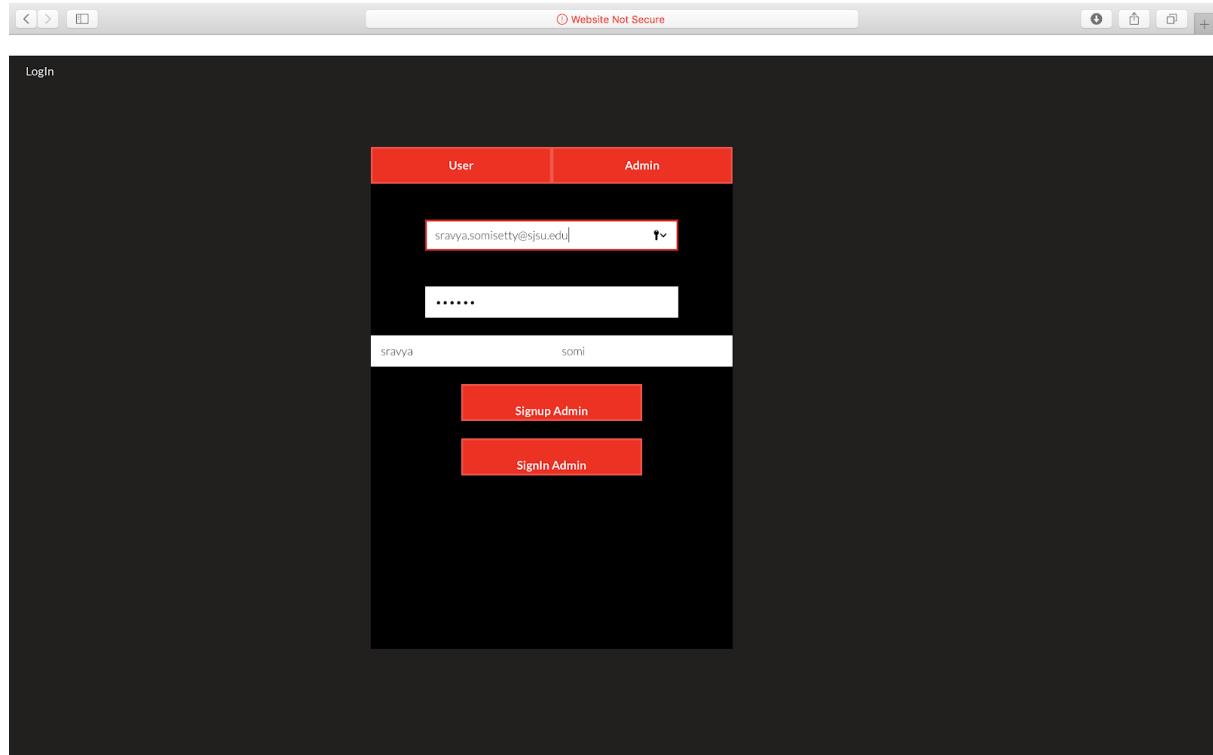
1. Front End: (HTML, CSS, Java script, Bootstrap, React JS)
 - a. React JS for building User Interface.
 - b. Redux for Application State Management.
 - c. Firebase for OAuth using Google and Facebook for the user.
2. Back End:(Java 10, Spring Restful API)
 - a. Java Spring Boot and backend data is accessed as Restful Web Service.
 - b. Maven as a dependency management tool.
 - c. Log4j is used for logging the information throughout the flow
 - d. Spring AOP for writing aspects in the backend.
 - e. Spring Security is used for authorization and authentication
 - f. Spring Data ElasticSearch is used for ElasticSearch implementation in the project.
 - g. Spring Data JPA as ORM for Persisting data to MySQL.
 - h. Junit, mockito and power mockito for unit testing
3. Data Base
 - a. Amazon cloud RDS MySQL.
 - b. Amazon cloud Elastic Search for ElasticSearch implementation.
4. Cloud Service
 - a. Amazon Cloud EC2 is used for deploying frontend and backend modules.
 - b. Amazon Cloud Elastic Search for Elastic cloud.

5. Description of Features with Screenshots

1. admin login/signup:

Following is the page for Admin login and login is restricted for admins with sjsu IDs.





2. Following is the screen for my movies page.

Not Secure — ec2-34-209-69-136.us-west-2.compute.amazonaws.com

Home My Movies User activity Report Movie activity report Financial report LogOut Search for a movie Search For New Movie

Filter by Availability Filter by Genere Filter for a movie with Title...

Wonder Boys
Grady is a 50-ish English professor who hasn't had a thing published in years -- not since he wrote his award winning "Great American Novel" 7 years a...
Released Date : 2000-02-22
Ratings : ★7

The Amityville Horror
George Lutz and his wife Kathleen, move into their Long Island dream house with their children only for their lives to be turned into a hellish nightmare...
Released Date : 1979-07-26
Ratings : ★6.3

Wonder Woman
It's one hundred and fifty seconds of pure fantasy.
Released Date : 2013-09-29
Ratings : ★6

Wonder Woman
The video opens with a barrage of explosive imagery along with an audio track of a siren taken from the 1970s TV show Wonder Woman. The following scen...
Released Date : 1978-11-29
Ratings : ★5.8

Edit Movie Delete Movie Edit Movie Delete Movie Edit Movie Delete Movie Edit Movie Delete Movie

Not Secure — ec2-34-209-69-136.us-west-2.compute.amazonaws.com

Home My Movies User activity Report Movie activity report Financial report LogOut gone with the wind Search For New Movie

Gone with the Wind
The spoiled daughter of a well-to-do plantation owner is forced to use every means at her disposal to claw her way out of poverty, following Maj. Gen...
Released Date : 1939-12-15
Ratings : ★7.9

Red Green Blue Gone With the Wind
Red Green Blue Gone with the Wind is a phosphorescent deconstruction of David O. Selznick's Technicolor classic Gone with the Wind (1939). Through the...
Released Date : 2001-09-05
Ratings : ★0

Gone with the Wind: Restoring a Legend
DVD featurette on the restoration of Gone With The Wind (1939).
Released Date : 2004-11-09
Ratings : ★0

Lynyrd Skynyrd - Gone With The Wind
This music documentary tracks the meteoric rise and tragic fall of Lynyrd Skynyrd as well as the bands' tragic downfall.
Released Date : 2015-10-16
Ratings : ★0

Add Movie Add Movie Add Movie Add Movie

3. Following is the screen for adding and editing movie details for admin:

Not Secure — ec2-34-209-69-136.us-west-2.compute.amazonaws.com

Home My Movies User activity Report Movie activity report Financial report LogOut gone with the wind Search For New Movie

The screenshot shows a modal dialog for editing a movie record. The title is 'GONE WITH THE WIND'. The form fields are as follows:

Title	Lynyrd Skynyrd - Gone With The Wind
Director	
ImageUrl	https://image.tmdb.org/t/p/w300/w75KJjL0oKWQw8viipfxDiKi4LC.jpg
Studio	
Actors	
Price \$	0
Availability	Free
MovieUrl	

Buttons at the bottom: Close (red), Add This Movie (green).

Side panels show ratings: Ratings: ★7.9 (left) and Ratings: ★0 (right). Buttons: Add Movie.

Not Secure — ec2-34-209-69-136.us-west-2.compute.amazonaws.com

Home My Movies User activity Report Movie activity report Financial report LogOut gone with the wind Search For New Movie

The screenshot shows a modal dialog for editing a movie record. The title is 'GONE WITH THE WIND'. The form fields are as follows:

Title	Lynyrd Skynyrd - Gone With The Wind
Director	Tom O'Dell
ImageUrl	https://image.tmdb.org/t/p/w300/w75KJjL0oKWQw8viipfxDiKi4LC.jpg
Studio	Annapurna
Actors	Ronnie Van Zant, Lynyrd Skynyrd
Price \$	5\$
Availability	Paid
MovieUrl	https://www.youtube.com/embed/FbMvdXLDRWQ

Buttons at the bottom: Close (red), Add This Movie (green).

Side panels show ratings: Ratings: ★7.9 (left) and Ratings: ★0 (right). Buttons: Add Movie.

4. Following are the screens financial report screen for Admin

A screenshot of a web browser window titled "Not Secure — ec2-34-209-69-136.us-west-2.compute.amazonaws.com". The page displays a user management interface with a dark background. At the top, there are navigation links: Home, My Movies, User activity Report, Movie activity report, Financial report, LogOut, and a search bar labeled "Search For New Movie". Below these, there are three tabs: PayPerview Users, All Unique Users (which is selected and highlighted in red), and All Subscribed Users. A table lists user information with columns: Username, Firstname, Lastname, and Email. The data shows three users: admin (Firstname: Ravi, Email: ravi@gmail.com), user2@gmail.com (Email: user2@gmail.com), and User3@gmail.com (Firstname: user3, Lastname: user3, Email: user3@gmail.com).

Username	Firstname	Lastname	Email
admin	Ravi		ravi@gmail.com
user2@gmail.com			
User3@gmail.com	user3	user3	user3@gmail.com

A screenshot of a web browser window titled "Not Secure — ec2-34-209-69-136.us-west-2.compute.amazonaws.com". The page displays a user management interface with a dark background. At the top, there are navigation links: Home, My Movies, User activity Report, Movie activity report, Financial report, LogOut, and a search bar labeled "Search For New Movie". Below these, there are three tabs: PayPerview Users, All Unique Users (which is selected and highlighted in red), and All Subscribed Users. A table lists user information with columns: Username, Firstname, Lastname, and Email. The data shows eight users: 1ra4vi3@gmail.com (Firstname: Ravi, Lastname: Teja, Email: 1ra4vi3@gmail.com), admin (Firstname: Ravi), namrata.kasar@sjtu.edu (Firstname: Namrata, Lastname: Kasar), sravya.somisetty@sjtu.edu (Firstname: sravya, Lastname: somi, Email: sravya.somisetty@sjtu.edu), user1@gmail.com (Email: user1@gmail.com), user2@gmail.com (Email: user2@gmail.com), user3@gmail.com (Firstname: user3, Lastname: user3, Email: user3@gmail.com), and user4@gmail.com (Firstname: User4, Lastname: user4, Email: user4@gmail.com).

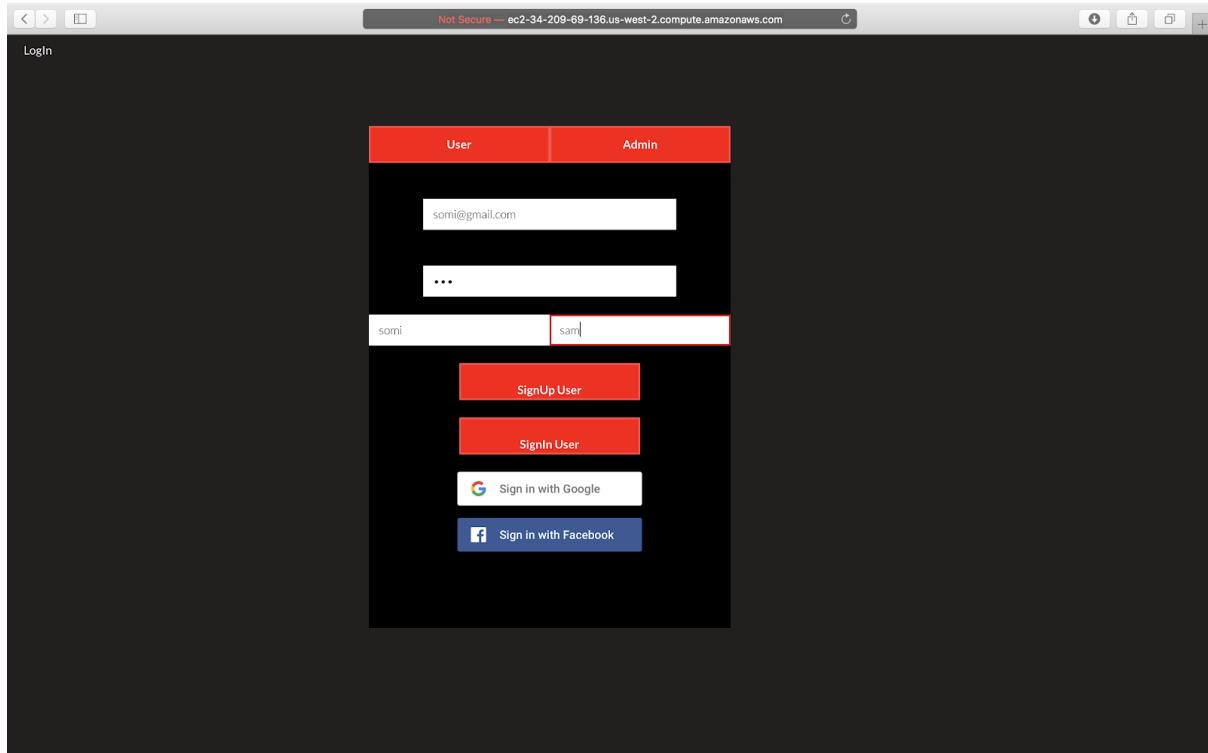
Username	Firstname	Lastname	Email
1ra4vi3@gmail.com	Ravi	Teja	1ra4vi3@gmail.com
admin	Ravi		
namrata.kasar@sjtu.edu	Namrata	Kasar	
sravya.somisetty@sjtu.edu	sravya	somi	sravya.somisetty@sjtu.edu
user1@gmail.com			flower
user2@gmail.com			
user3@gmail.com	user3	user3	user3@gmail.com
user4@gmail.com	User4	user4	user4@gmail.com

PayPerview Users			
All Unique Users All Subscribed Users			
Username	Firstname	Lastname	Email
user2@gmail.com			
user3@gmail.com	user3	user3	

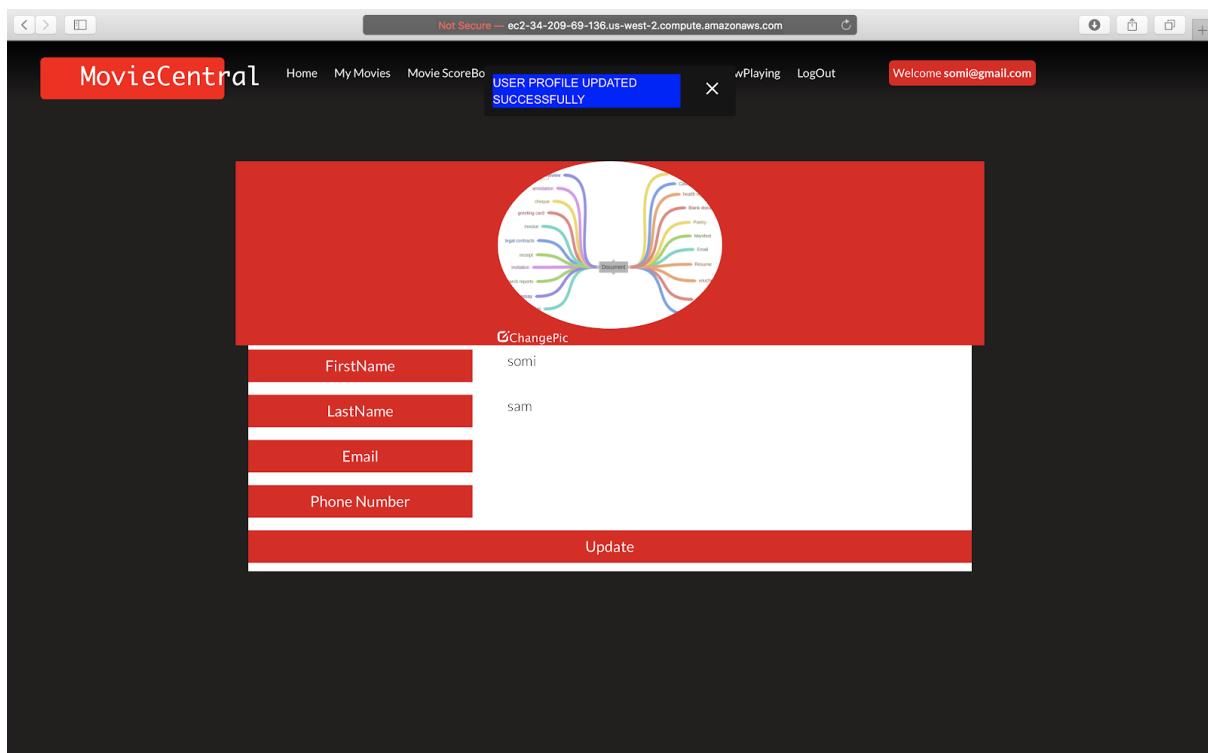
5. Following is the screen for user activity report for admin.

User activity Report			
Browse Customers User Play History Top 10 users			
Username	Firstname	Lastname	Email
1ra4vi3@gmail.com	Ravi	Teja	1ra4vi3@gmail.com
admin	Ravi		
namrata.kasar@sjtu.edu	Namrata	Kasar	
srawya.somisetty@sjtu.edu	srawya	somi	srawya.somisetty@sjtu.edu
user1@gmail.com			flower
user2@gmail.com			
user3@gmail.com	user3	user3	
user4@gmail.com	User4	user4	

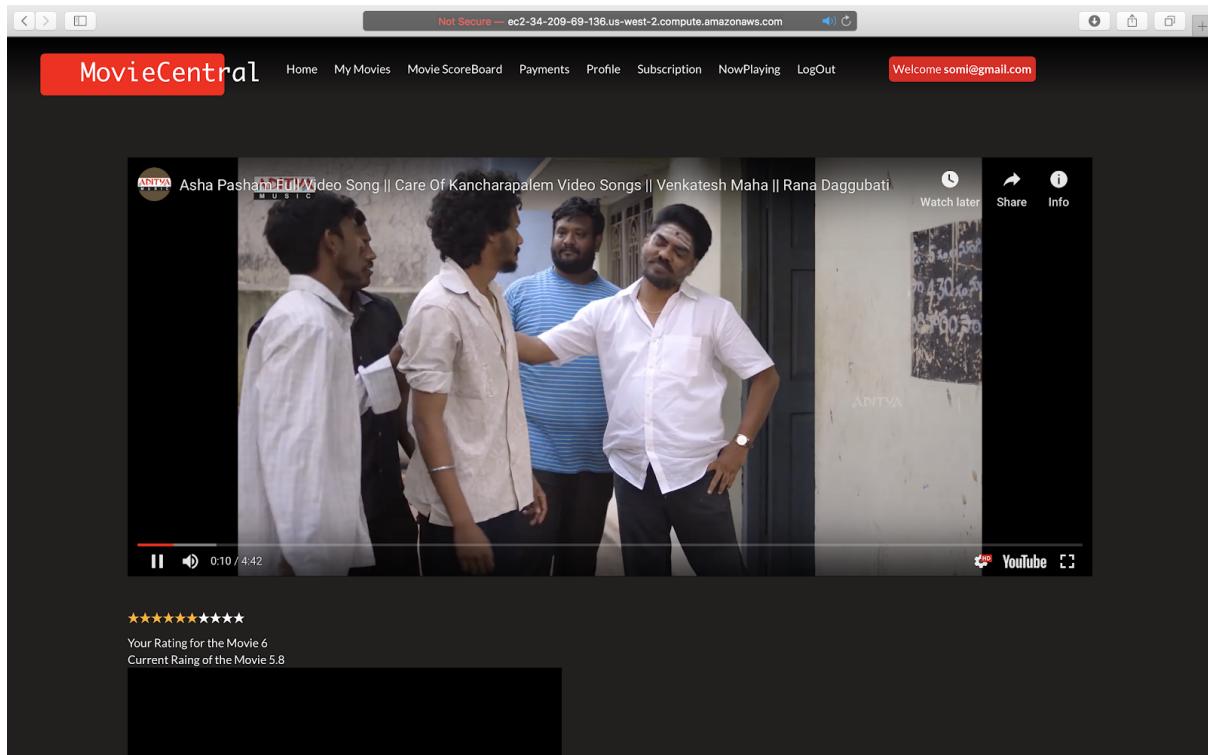
6. Following is the screen for user login.



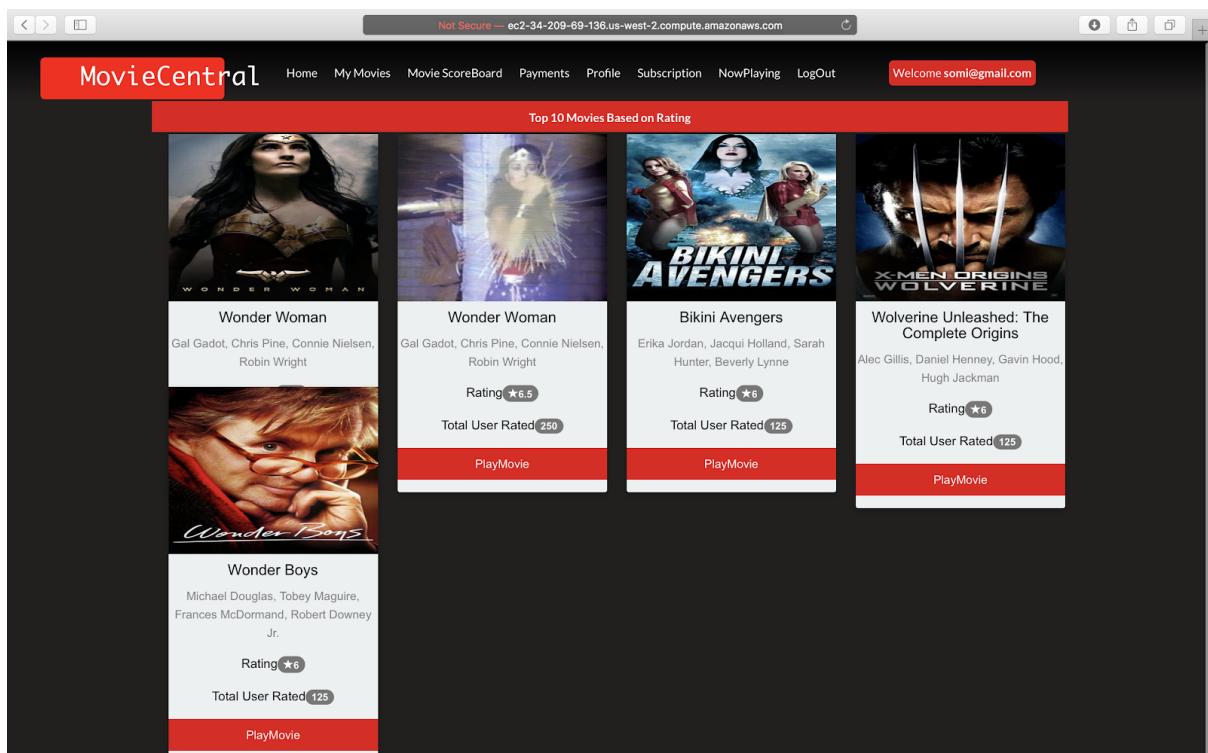
7. Following is the screen for user profile where he can edit the details.



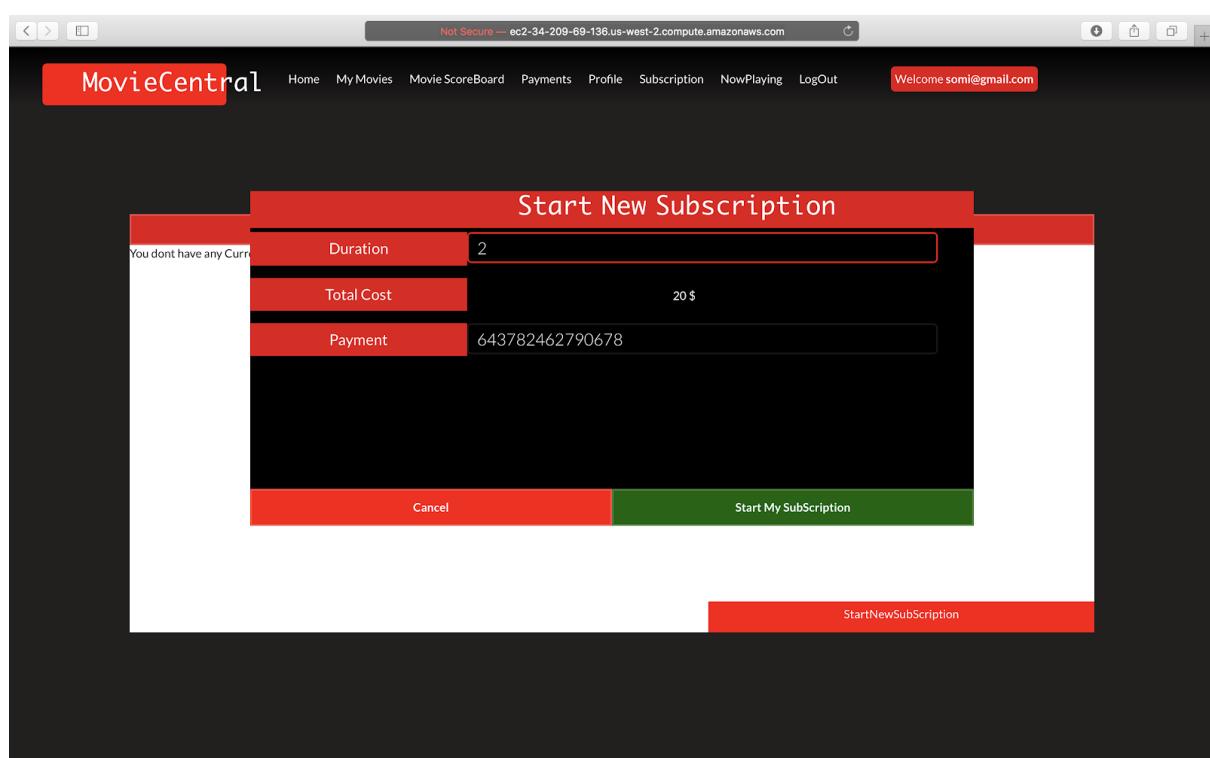
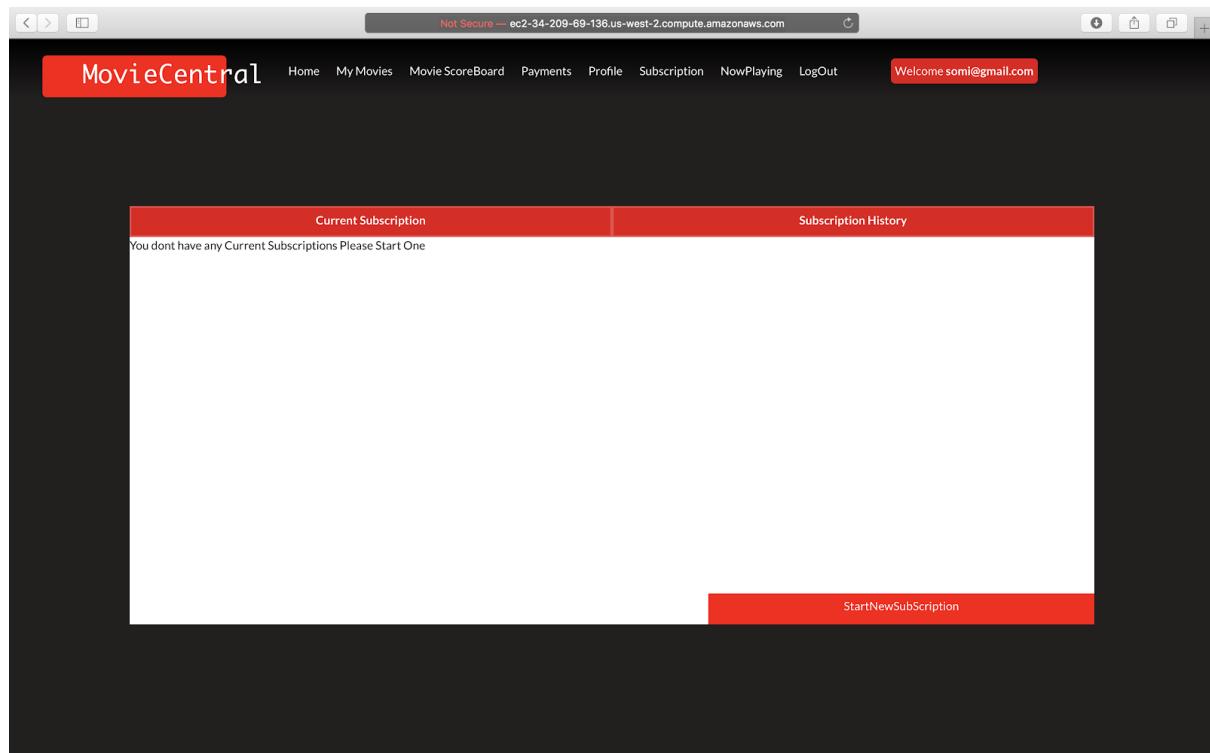
8. Following is the screen where user can watch the movie.



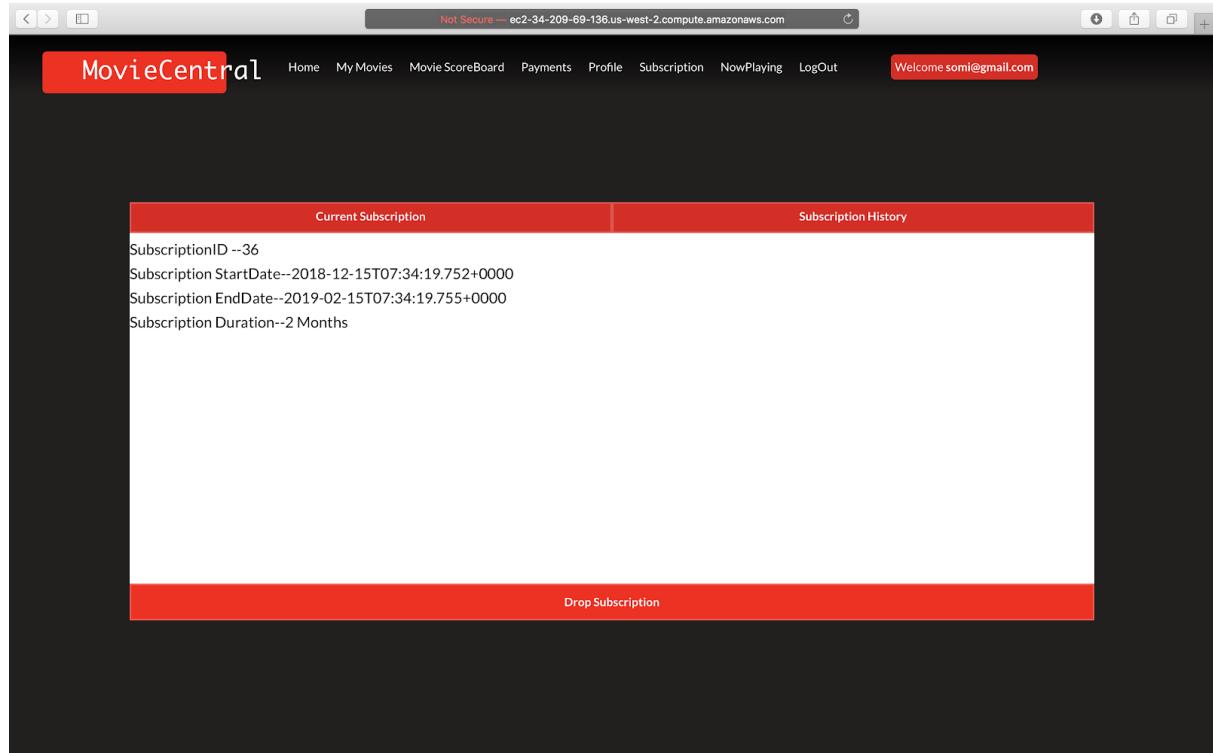
9. Following is the scoreboard for user where he can see top 10 movies.



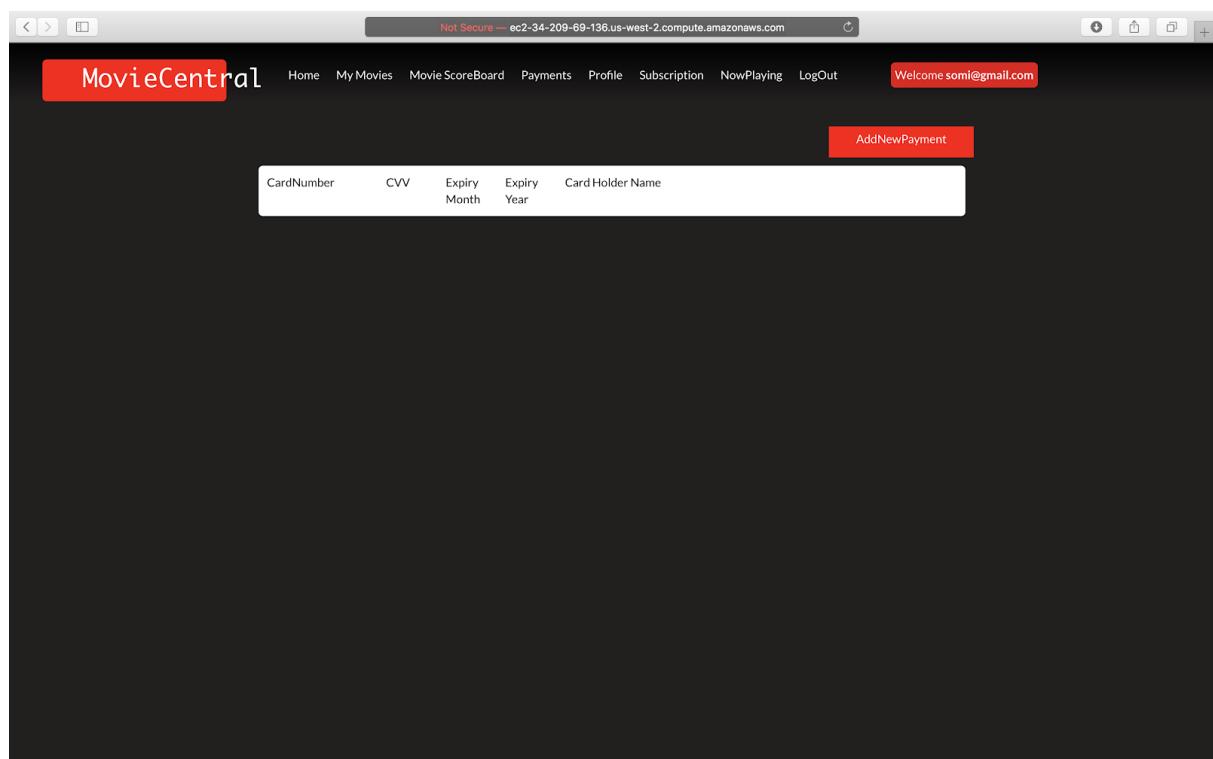
10. Following are the screens for subscription of user where he can subscribe himself.

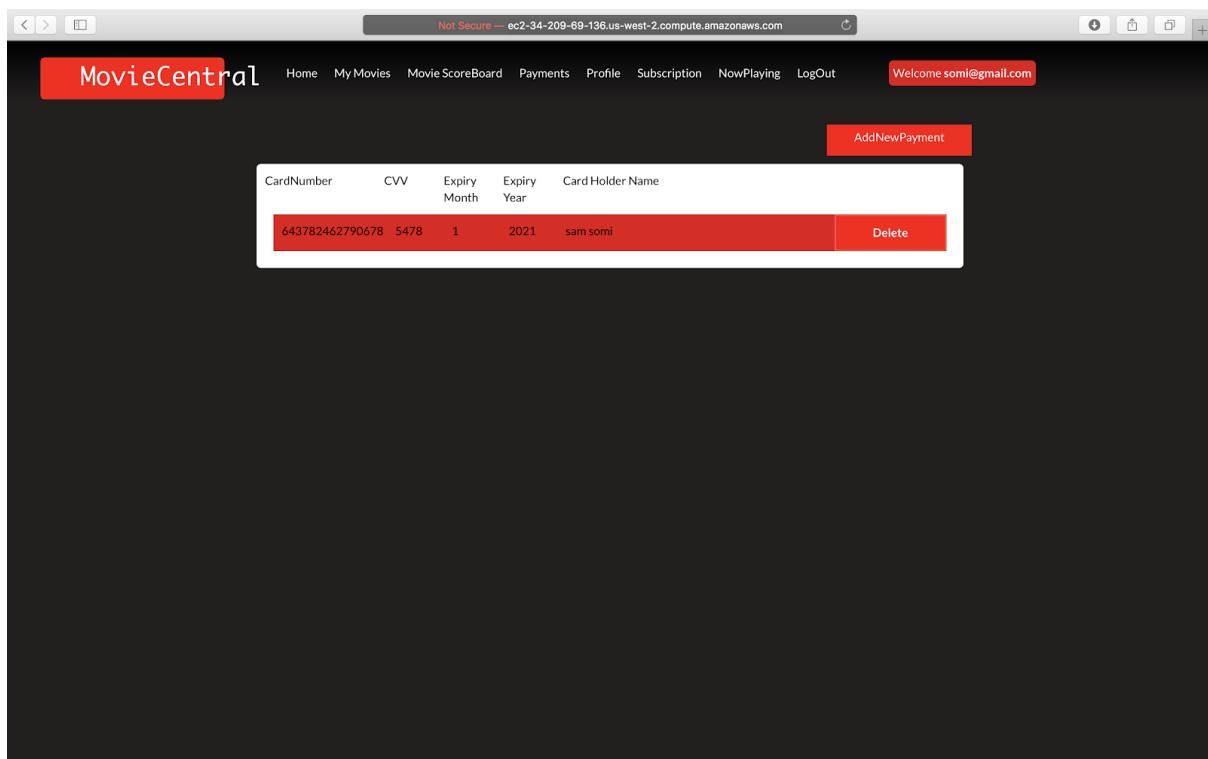
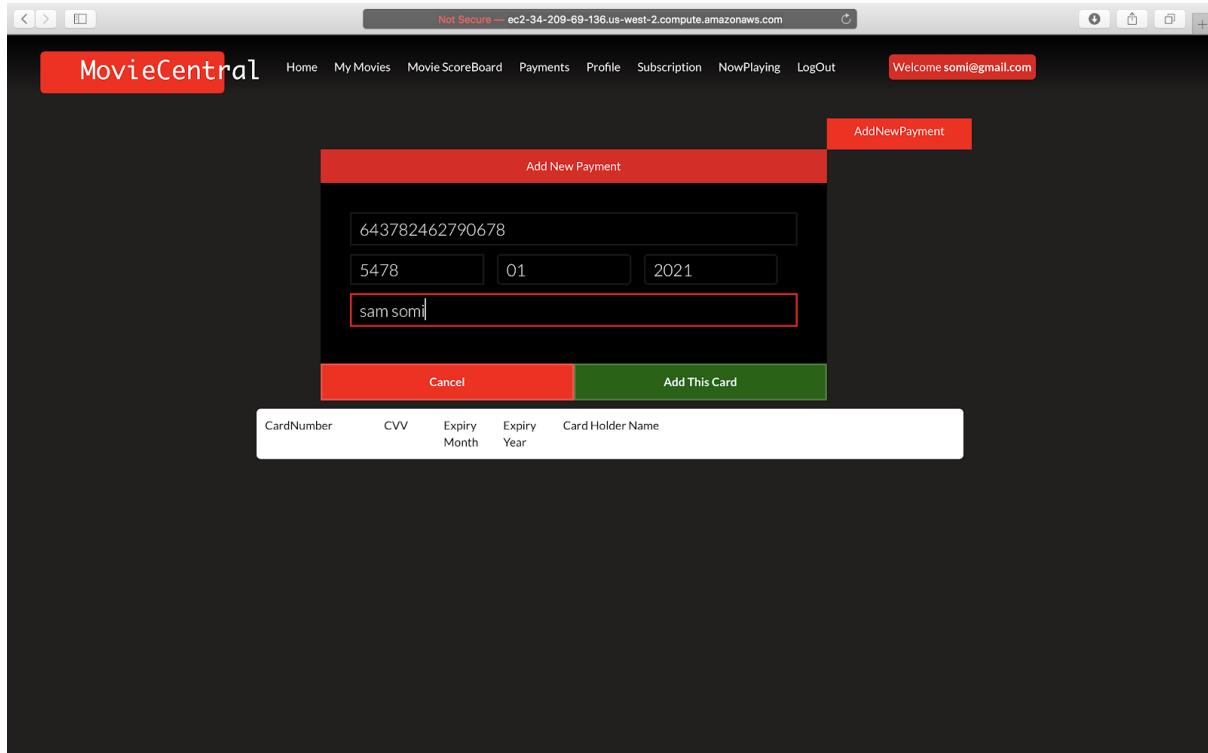


11. Following is the screen to see user's subscription details.



12. Payment screens for user.





6. Testing plan executed and results

We have used junit, mockito, power mock for unit testing the classes and we have used mockito for mocking the dependent classes.

Power mockito is used for mocking private methods in the classes. Example of the test case is given below:

```

public class UserControllerTest {

    @Test
    public void testuserVerification() {

        UserController usercontroller =new UserController();
        User user =new User();
        user.setUsername("User1@gmail.com");
        user.setPassword("user");
        User res=usercontroller.userSignin(user,null);
        assertEquals(res.isEnabled(), true);
    }
}

```

Expected response is checked on Postman for the particular REST api urls. Following screenshots show few of the responses we got for the REST APIs included in the project.

1. Get all the movies:

API url:

<http://ec2-34-209-69-136.us-west-2.compute.amazonaws.com:8080/movies/getallmovies>

The screenshot shows a Postman interface with the following details:

- Method: GET
- URL: http://ec2-34-209-69-136.us-west-2.compute.amazonaws.com:8080/movies/getallmovies
- Status: 200 OK
- Time: 232 ms
- Size: 9.83 KB
- Headers: Body, Cookies, Headers (3), Test Results
- JSON Response (Pretty):

```

1 [
2   {
3     "movieid": "11004",
4     "title": "Wonder Boys",
5     "genre": "Comedy, Drama",
6     "releasedate": "2000-02-22",
7     "studio": "Paramount Pictures",
8     "synopsis": "Grady is a 50-ish English professor who hasn't had a thing published in years -- not since he wrote his award win",
9     "image": "https://image.tmdb.org/t/p/w300//aaJFu5pK4YqBLKEkZzSBpjog68.jpg",
10    "movieurl": null,
11    "actors": "Michael Douglas, Tobey Maguire, Frances McDormand, Robert Downey Jr.,",
12    "director": "Curtis Hanson",
13    "country": "USA, Germany, UK, Japan",
14    "rating": "7",
15    "availability": "Free",
16    "price": "0",
17    "registereddate": null
18  },
19  {
20    "movieid": "11449",
21    "title": "The Amityville Horror",
22    "genre": "Drama, Horror, Mystery, Thriller",
23    "releasedate": "1979-07-26",
24    "studio": "MGM",
25    "synopsis": "Edgar Lutz and his wife Kathleen move into their long Island dream house with their children only for their life to be turned upside down by mysterious forces from the past. The Lutzes' new home is the site of the infamous Amityville Horror, which occurred in 1974. The house is haunted by the spirit of a woman who was killed there, and her spirit continues to haunt the house even after she died. The Lutzes' son, Michael, begins to have strange visions and hear voices, and his mother becomes increasingly paranoid and afraid. The Lutzes' neighbors, the Tappans, also begin to experience strange things, such as doors opening and closing on their own and objects moving around the house. The Lutzes' friends and family members also begin to notice odd things happening at the house, and they start to wonder if the house is really haunted or if it's just a coincidence. The Lutzes' son, Michael, begins to have strange visions and hear voices, and his mother becomes increasingly paranoid and afraid. The Lutzes' friends and family members also begin to notice odd things happening at the house, and they start to wonder if the house is really haunted or if it's just a coincidence."
  }
]

```

2. Get top 10 movies:

API url :

<http://ec2-34-209-69-136.us-west-2.compute.amazonaws.com:8080/movierating/gettopmovies>

Expected: Top 10 movies should be displayed.

Actual: List of movies with details (upto 10) is displayed as follows:

```

GET http://ec2-34-209-69-136.us-west-2.compute.amazonaws.com:8080/movierating/gettopmovies
[{"movieid": "241224", "title": "Wonder Woman", "genre": "Action, Adventure, Fantasy, Sci-Fi, War", "releasedate": "2013-09-29", "studio": "Warner Bros. Pictures", "synopsis": "It's one hundred and fifty seconds of pure fantasy.", "image": "https://image.tmdb.org/t/p/w300/6xrVlKEGBTJCPPhTubfAdvknrh5D.jpg", "movieurl": "https://www.youtube.com/embed/dJPEF7vwb6Q", "actors": "Gal Gadot, Chris Pine, Connie Nielsen, Robin Wright", "director": "Patty Jenkins", "country": "USA, China, Hong Kong", "rating": "6", "availability": "Subscription", "price": "10", "registereddate": null}, {"movieid": "306824"}]

```

3. Get all payment details for user “User3@gmail.com”

API url :

<http://ec2-34-209-69-136.us-west-2.compute.amazonaws.com:8080/payments/getpayments?username=User3@gmail.com>

Expected : All payment details (card details) of user should be displayed.

Actual : List of all payments of that user as per the database is seen as follows:

```

GET http://ec2-34-209-69-136.us-west-2.compute.amazonaws.com:8080/payments/getpayments?username=Us...
[{"id": 12, "cardnumber": "435647864325675", "cvv": "6543", "cardholder": "User#", "expirymonth": 1, "expiryyear": 2018, "username": "User3@gmail.com"}, {"id": 13, "cardnumber": "234567895432675", "cvv": "4453", "cardholder": "User3", "expirymonth": 1, "expiryyear": 2018, "username": "User3@gmail.com"}]

```

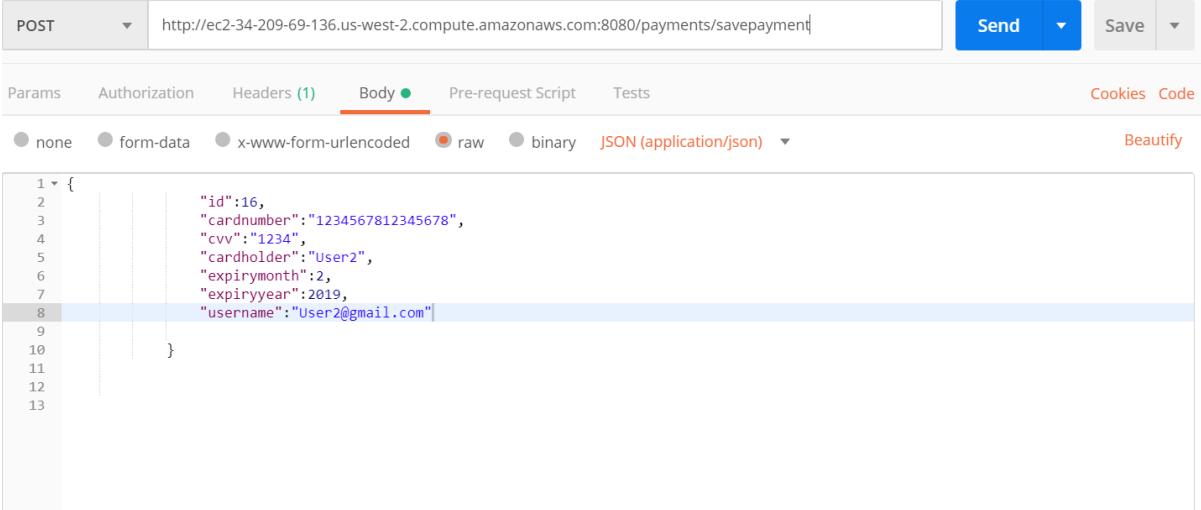
4. Saving payment details for user “User1@gmail.com”.

API url:

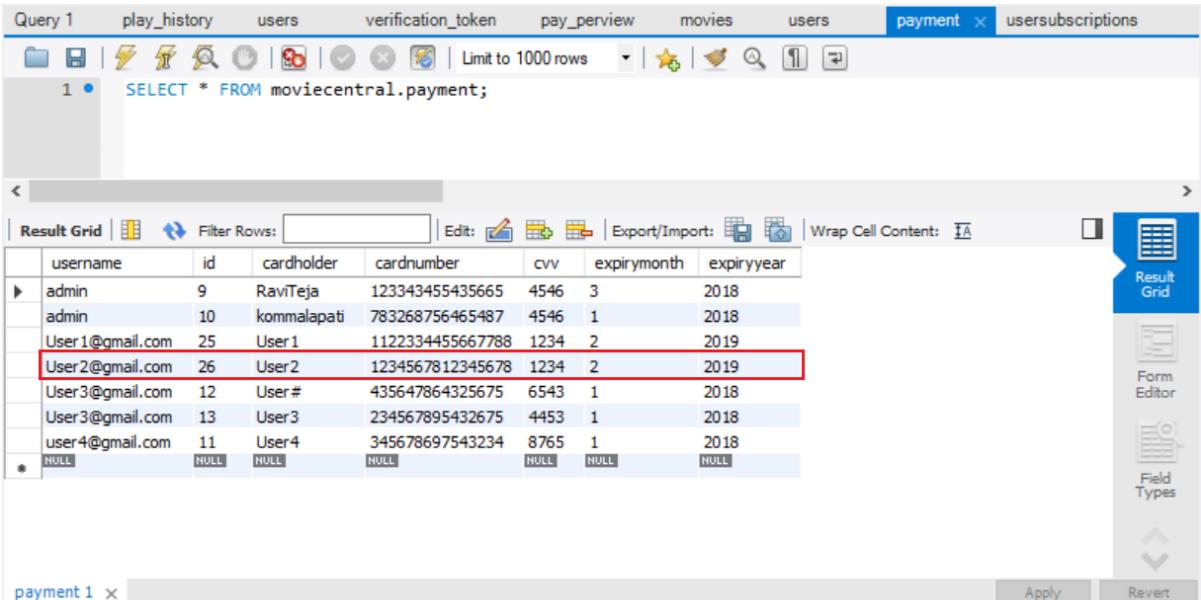
<http://ec2-34-209-69-136.us-west-2.compute.amazonaws.com:8080/payments/savepayment>

Expected: Payment details should be saved and payment object should be displayed.

Actual: Payment details are saved and payment object is displayed as follows:



```
POST http://ec2-34-209-69-136.us-west-2.compute.amazonaws.com:8080/payments/savepayment| Send Save  
Params Authorization Headers (1) Body ● Pre-request Script Tests Cookies Code  
none form-data x-www-form-urlencoded raw binary JSON (application/json) Beautify  
1 {  
2   "id":16,  
3   "cardnumber":"1234567812345678",  
4   "cvv":"1234",  
5   "cardholder":"User2",  
6   "expirymonth":2,  
7   "expiryyear":2019,  
8   "username":"User2@gmail.com"  
9 }  
10  
11  
12  
13
```



	username	id	cardholder	cardnumber	cvv	expirymonth	expiryyear
▶	admin	9	RaviTeja	123343455435665	4546	3	2018
▶	admin	10	kommalapati	783268756465487	4546	1	2018
▶	User1@gmail.com	25	User1	1122334455667788	1234	2	2019
▶	User2@gmail.com	26	User2	1234567812345678	1234	2	2019
▶	User3@gmail.com	12	User#	435647864325675	6543	1	2018
▶	User3@gmail.com	13	User3	234567895432675	4453	1	2018
▶	user4@gmail.com	11	User4	345678697543234	8765	1	2018
*		NULL	NULL	NULL	NULL	NULL	NULL

7. Lessons learned and possible Future Work

- We learned about Aspect Oriented Programming and how it can be implemented to overcome cross cutting concerns like Logging and Exception Handling.
- We have learnt about Spring Security, how it can be implemented to protect the API by using authentication and Authorization.
- We have implemented hibernate JPA for database management where we learnt about ORM mapping and table inheritance concepts and how to use the named queries.
- We learnt about implementation of CORS for project level not for method level.

- We have used annotation throughout the project to make efficient use of declarative programming.
- We have implemented elastic search for recommendations and making large scale data queries(full text search).
- As future scope, we can add TV shows and we can stream live videos as well.