

*******Lab2*******

Lab 2.1 **Create RDS instance with Default VPC *******

1. Create MySQL RDS instance in the default VPC.
2. Try to connect to the rds instance from the MySQL client which is install on your machine.
3. Create a database with name “CGDB” in the RDS instance created with default VPC.
4. Create following table “Movie_Angular” with following column
 - a. Mov_id- number
 - b. Mov_name –varchar
 - c. Mov_rat- number
 - d. Mov_gen – varchar
5. Insert some record in side it. And test.
6. Create EC2 instance in default VPC.
7. Connect to EC2 instance from your machine.
8. And connect to your RDS instance created from EC2 instance.

Lab 2.2 *****

1. Create a Spring Boot Application in STS.
2. Write a Rest Controller to map following url to perform CRUD operations on the database created in RDS instance in a default VPC
 - <http://<<EC2instancePublicURL:PORTNO/create/movie>
 - <http://<<EC2instancePublicURL:PORTNO/movie>
 - <http://<<EC2instancePublicURL:PORTNO /movie/3>
 - <http://<<EC2instancePublicURL:PORTNO /movie/delete/3>
3. Create appropriate DAO layer in spring boot application with JPA integration to connect to the above RDS instance database.
4. Deploy jar file of spring BOOT application in AWS EC2 which is created using default VPC.
5. Run the Spring Boot Application in EC2 instance.
6. Test the above URL for EC2 instance from the postman.

*******Lab3*******

Lab 3.1 **Custom VPC with Private and Public Subnet *******

9. Create custom VPC in AWS.
10. Create 1 public and 1 private subnet in the custom VPC. And associate all subnets mask with the VPC.
11. Create MySQL RDS instance in the private subnet VPC.
12. Create EC2 instance in Custom VPC public subnet mask.
13. Connect to EC2 instance from your machine.
14. Install MySQL DB client in Ec2 instance.

15. And connect to your RDS instance created in private subnet of the custom VPC from EC2 instance.
16. Create a database with name "CGDB" in the RDS instance.
17. Create following table "Movie_Angular" with following column
 - a. Mov_id- number
 - b. Mov_name –varchar
 - c. Mov_rat- number
 - d. Mov_gen – varchar
18. Insert some record in side it. And test.

Lab 3.2 *****

7. Create a Spring Boot Application in STS.
8. Write a Rest Controller to map following url to perform CRUD operations on the database created in RDS instance in a custom VPC.
 - <http://<<EC2instancePublicURL:PORTNO/create/movie>
 - <http://<<EC2instancePublicURL:PORTNO/movie>
 - <http://<<EC2instancePublicURL:PORTNO /movie/3>
 - <http://<<EC2instancePublicURL:PORTNO /movie/delete/3>
9. Create appropriate DAO layer in spring boot application with JPA integration to connect to the above RDS instance database.
10. Deploy jar file of spring BOOT application in AWS EC2 which is created using Custom VPC.
11. Run the Spring Boot Application in EC2 instance.
12. Test the above URL for EC2 instance from the postman.

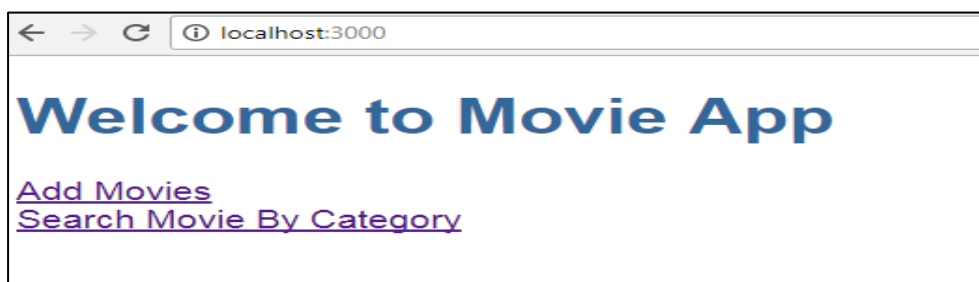
*****Lab 4*****

Lab 4.1 *****

Create a "Movie App" wherein movies could be maintained according to genres like Fiction, Drama and Satire. There is also a search option allowing to search for movies.

Use [Movies.java](#)

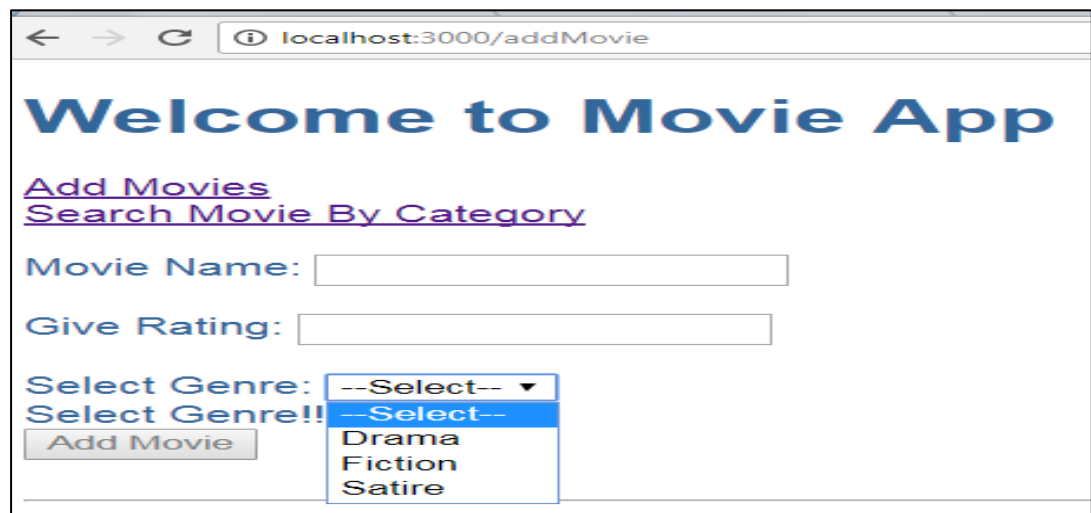
Refer below screen shot:



- a. When the user clicks on “Add Movies” -> Allows a User to add a movie.

The user can add movie(s) depending on Genre (drop-down list loaded with values [Drama, Fiction, Satire])

The user could add multiple movies. On Adding Movies will be added in Database with the help of Spring Web services (Restful). Use Spring restful with JPA concept Refer below screen shot:



localhost:3000/addMovie

Welcome to Movie App

[Add Movies](#)
[Search Movie By Category](#)

Movie Name:

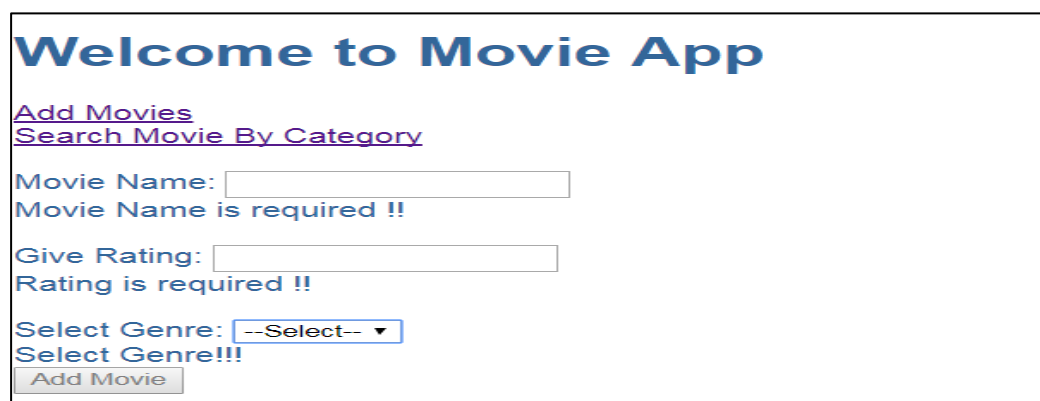
Give Rating:

Select Genre:

Select Genre!!

Drama
Fiction
Satire

Refer below screens for validations:



Welcome to Movie App

[Add Movies](#)
[Search Movie By Category](#)

Movie Name:

Movie Name is required !!

Give Rating:

Rating is required !!

Select Genre:

Select Genre!!!

← → ↻ localhost:3000/addMovie

Welcome to Movie App

[Add Movies](#)
[Search Movie By Category](#)

Movie Name:
Movie Name should be only alphanumeric

Give Rating:
Rating needs to be a number: eg: 2.5. Rating is on Scale 1-5

Select Genre:
Select Genre!!!

- b. When the user clicks on “Search Movie By Category” -> Allows user to search movie(s) by Category. [Depending on Genre (Drama,Fiction,Satire)]

There could be more than one movies depending on same Genre. Refer below screen shots:

← → ↻ localhost:3000/searchMovie

Welcome to Movie App

[Add Movies](#)
[Search Movie By Category](#)

Search Movie

Thus if Searching Movie on ‘Drama’ displays below Result:

← → ↻ localhost:3000/searchMovie

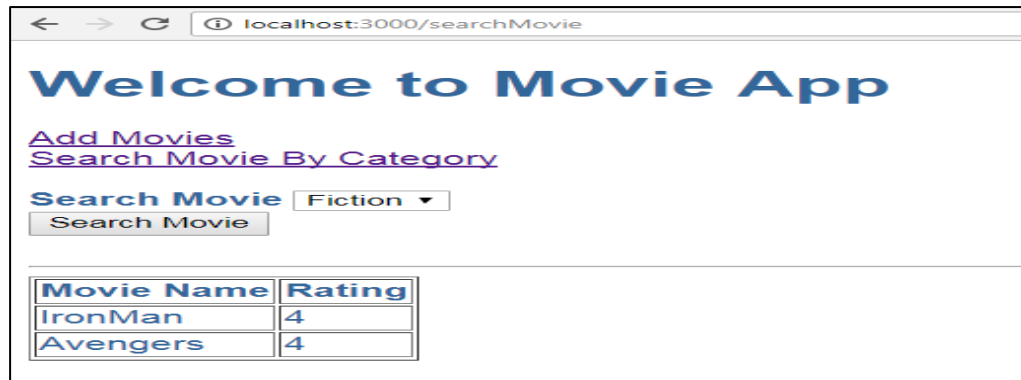
Welcome to Movie App

[Add Movies](#)
[Search Movie By Category](#)

Search Movie

Movie Name	Rating
HomeAlone	4
HorrorSeries	4

Thus if Searching Movie on ‘Fiction’ displays below Result:



Movie Name	Rating
IronMan	4
Avengers	4

Note:

- In above assignment movies are added via “Add Movies” and Searched via “Search Movie By Category”. **These are different components.**
- Make use of Angular 6 Forms.
- Make use of services.
- Make use of Router Module. Which should be created as a separate module and included in the main App Module.
- Use Spring restful & JPA concept to add movies & search movies.
- All adding & searching movies done through database

Note: Sample Code for the above requirement is already developed as a part of Angular 6 and Spring Boot assignments.

- There are two applications in the above code.
- One is front end angular 6 application running on 4200 port no created as a part of Angular6 Lab assignments.
- And another one is Spring Boot Rest application created as a part of Spring Lab assignments
- Create a Mysql Database instance using AWS RDS service as a part of JPA.
- Use mysql data base of AWS RDS to create a table structure in “Spring Boot REST”
- Upload the Spring Boot Movie App jar file in S3 bucket.
- Copy the Spring Boot Movie jar file from “**AWS S3 bucket**” to “AWS EC2 instance”
- Run your Spring Boot Rest Movie App jar file from EC2 instance.
- In your local machine create front end angular 6 application and call the Spring BOOT Rest URL which are running in AWS EC2 instance from Angular Service. And run the application.

Note: make use of AWS default VPN and default security groups in AWS for completing above lab assignment.

