ASSIGNMENT QUESTION 13

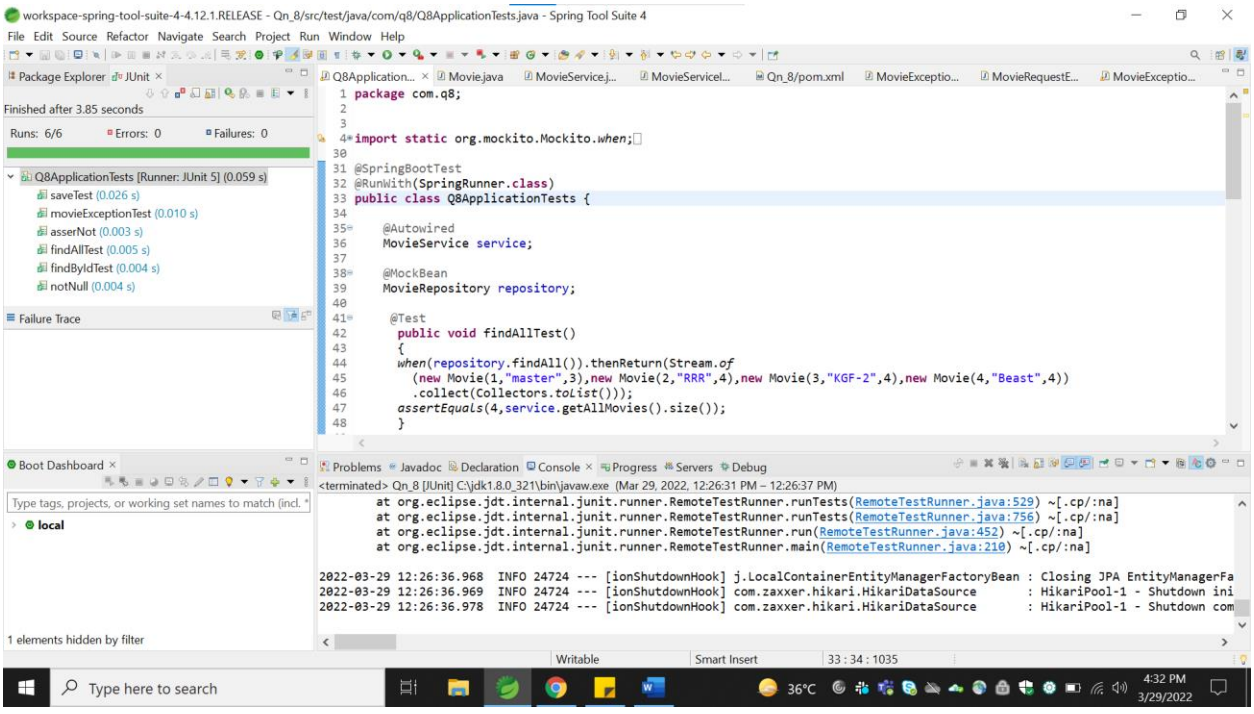
UNIT TEST CASES

```
package com.q8;


import static org.mockito.Mockito.when;

import java.util.Optional;
import java.util.stream.Collectors;
import java.util.stream.Stream;

import org.junit.Test;
import org.junit.runner.RunWith;

import static org.assertj.core.api.Assertions.fail;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotEquals;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertThrows;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.test.context.junit4.SpringRunner;
import static org.junit.Assert.fail;

import com.q8.exception.MovieException;
import com.q8.exception.MovieRequestException;
```

```
import com.q8.model.Movie;
import com.q8.repository.MovieRepository;
import com.q8.service.MovieService;

@SpringBootTest
@RunWith(SpringRunner.class)
public class Q8ApplicationTests {

        @Autowired
        MovieService service;

        @MockBean
        MovieRepository repository;

         @Test
          public void findAllTest()
          {
          when(repository.findAll()).thenReturn(Stream.of
            (new Movie(1,"master",3),new Movie(2,"RRR",4),new Movie(3,"KGF-2",4),new
Movie(4,"Beast",4))
            .collect(Collectors.toList()));
          assertEquals(4,service.getAllMovies().size());
          }

        @Test
        public void saveTest()
        {
         Movie movie =  new Movie(1,"Beast",4);
    when(repository.save(movie)).thenReturn(movie);
```

```
        assertEquals(movie,service.addMovie(movie));

    }

    @Test
    public void asserNot()
    {
     Movie movie =  new Movie(1,"Beast",4);
     Movie movieOne =  new Movie(2,"rrr",3);
when(repository.save(movie)).thenReturn(movie);
      assertNotEquals(movie,service.addMovie(movieOne));

    }


    @Test
    public void findByIdTest()
    {
            Movie movie = new Movie(1, "RRR",3);
                    when(repository.findById(1)).thenReturn(Optional.of(movie));
    }


     @Test
     public void notNull(){
       Movie movie = new Movie(1,"master",4);
         assertNotNull(movie);
      }
```

```
    @Test
    public void movieExceptionTest() {



    Movie one = new Movie(1,"rrr",3);
    Movie two = new Movie(2,"fff",4);
    service.addMovie(one);
    service.addMovie(two);
  MovieRequestException movieException = assertThrows(MovieRequestException.class,
    () -> service.getmovie(3));
  assertEquals("movie not found with this id ::", movieException.getMessage());
    }
}
```



1 RESULTS