U·
ST

TESTING
ON
MUZIX APPLICATION

Submitted By

163595 - Amal Ram M

U84536 -Anandh Suseela

163645 - C V Ramesh Kumar Reddy

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We are extremely glad to present our project as a part of our training. We take this opportunity to express our sincere thanks to those who helped us in bringing out the report of my project.

I am deeply grateful to Stack Route , for their help and suggestions throughout the project.

Express heartiest thanks to **Mrs. Koel Chowdhury**, Project Scrum Master for her encouragement and valuable suggestion. Finally, we express our thanks to all of our friends who gave us their valuable suggestion for successful completion of this project.

Amal Ram M

Anandh Suseela

C V Ramesh Kumar Reddy

1. SYSTEM SPECIFICATION

1.1.HARDWARE REQUIREMENTS:

Processor : Intel i3 or above

RAM : 2GB or Above

Hard Disk: PC with 20 GB

1.2.SOFTWARE REQUIREMENTS :-

Operating System Server: Linux.

DataBase Server: Oracle MySQL.

Client: Google Chrome, Version-88.

Tools: Spring Tool Suite-4, Visual Studio Code, PostMan, MySql WorkBench

1.3.TOOLS / ENVIRONMENT USED:

1.3.1.MySql WorkBench :

MySQL is an open-source relational database management system (RDBMS). "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

1.3.2.Spring Tool Suite -4

Spring Tool Suite (STS) is an Eclipse-based IDE which is dedicated for developing Spring-based projects. It is actively developed and maintained by the SpringSource community. STS facilitates and simplifies Spring-based applications development by providing robust project templates such as Spring Batch, Spring Integration, Spring Persistence (Hibernate + JPA), Spring MVC, etc. In addition, with Maven integration, STS releases developers from manually managing Spring jar files in their projects.

1.3.3.Visual Studio Code:

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python and C++.It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component used in Azure DevOps.

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. It supports a number of programming languages and a set of features that differs per language. It can be extended via extensions,available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.

2. PROJECT AIMS AND OBJECTIVES:

2.1.The Aims and Objectives Of The Given Project:

- Account Registration for the User in the Muzix App.
- Login facility for the Users.
- Users are allowed to search the Music
- Users are allowed to create the PlayList & add the Music to PlayList.
- Users were allowed to remove the Music from PlayList.
- Users are allowed to Add the Music to the BookMark.
- Users were allowed to remove the BookMark.
- Users were allowed to browse the suggested Music.
- Logout functionality for users.

2.2.Testing Aims and Objectives:

- Testing the register functionality.
- Testing the login functionality.
- Testing the search functionality.
- Testing the add to PlayList & removing Music from PlayList functionality
- Testing the add to BookMark & removing Music from BookMark .
- Unit Testing for AccountManager and MuzixManager with Mockito+J-unit and Rest-Assured+TestNg in the backend.
- Angular front end has to be tested using Protractor

2.3.Overall Description of Testing Techniques:

For the testing purpose there are two modules provided:

- Account Manager module.
- Muzix Manager module.

In the modules mentioned above we are testing in both the frontend and backend section using following technologies: -

2.4.Backend Testing Frameworks:

2.4.1.Mockito+J-unit:

Mockito is a popular mock framework which can be used in conjunction with JUnit. Mockito allows you to create and configure mock objects. Using Mockito greatly simplifies the development of tests for classes with external dependencies.

If we use Mockito in tests we can typically:

- Mock away external dependencies and insert the mocks into the code under test
- Execute the code under test
- Validate that the code executed correctly
- We can expect the output with the help of assertions.
- The major benefit is that without the help of databases we can directly inject the mocked values to the class we intended to inject.

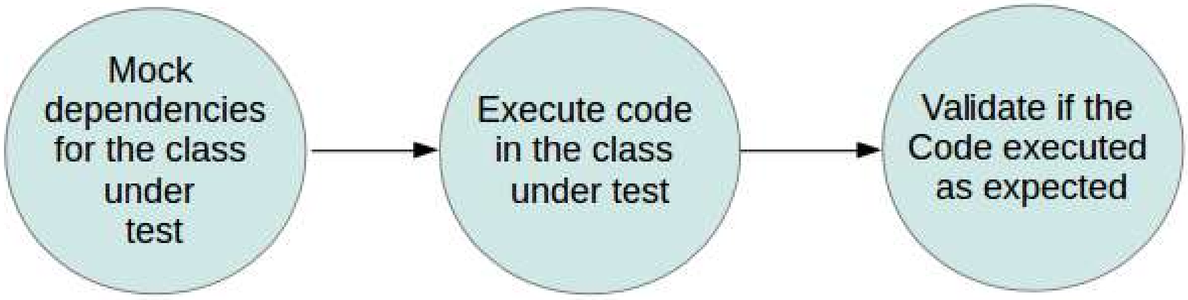

Mock dependencies for the class under test
Execute code in the class under test
Validate if the Code executed as expected

Fig:1.1 Mockito Flow Diagram

JUnit is a Java test framework and an open source project . JUnit 5 (also known as Jupiter) is the latest major release of JUnit. It consists of a number of discrete components:

- JUnit Platform - foundation layer which enables different testing frameworks to be launched on the JVM
- Junit Jupiter - is the JUnit 5 test framework which is launched by the JUnit Platform

- JUnit Vintage - legacy Test Engine which runs older tests.

As the usage of JUnit 5 requires multiple libraries to be present, we typically use it with a build system like Maven or Gradle. JUnit 5 needs at least Java 8 to run.

2.4.2.Rest-Assured+TestNg:

Rest assured is java library for testing Restful Web services. It can be used to test XML & JSON based web services. It supports GET, POST, PUT, PATCH, DELETE, OPTIONS and HEAD requests and can be used to validate and verify the response of these requests. Also it can be integrated with testing frameworks like JUnit, TestNG etc.

TestNG is a testing framework developed in the lines of JUnit and NUnit, however it introduces some new functionalities that make it more powerful and easier to use. TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc.

2.5.Frontend Testing Frameworks:

2.5.1.Protractor-Jasmine:

Protractor is an open source end-to-end testing framework for Angular and AngularJS applications. It was built by Google on the top of WebDriver. It also serves as a replacement for the existing AngularJS E2E testing framework called “Angular Scenario Runner”.

It also works as a solution integrator that combines powerful technologies such as NodeJS, Selenium, Jasmine, WebDriver, Cucumber, Mocha etc. Along with testing of AngularJS applications, it also writes automated regression tests for normal web applications. It allows us to test our application just like a real user because it runs the test using an actual browser.

Jasmine is an open-source testing framework for JavaScript. It aims to run on any JavaScript-enabled platform, to not intrude on the application nor the IDE, and to have easy-to-read syntax. It is heavily influenced by other unit testing frameworks, such as ScrewUnit, JSSpec, JSpec, and RSpec.

2.6.Added Development Functionalities in the Given Project :

- update user details.
- view user details.

Then verify above functionalities with Postman

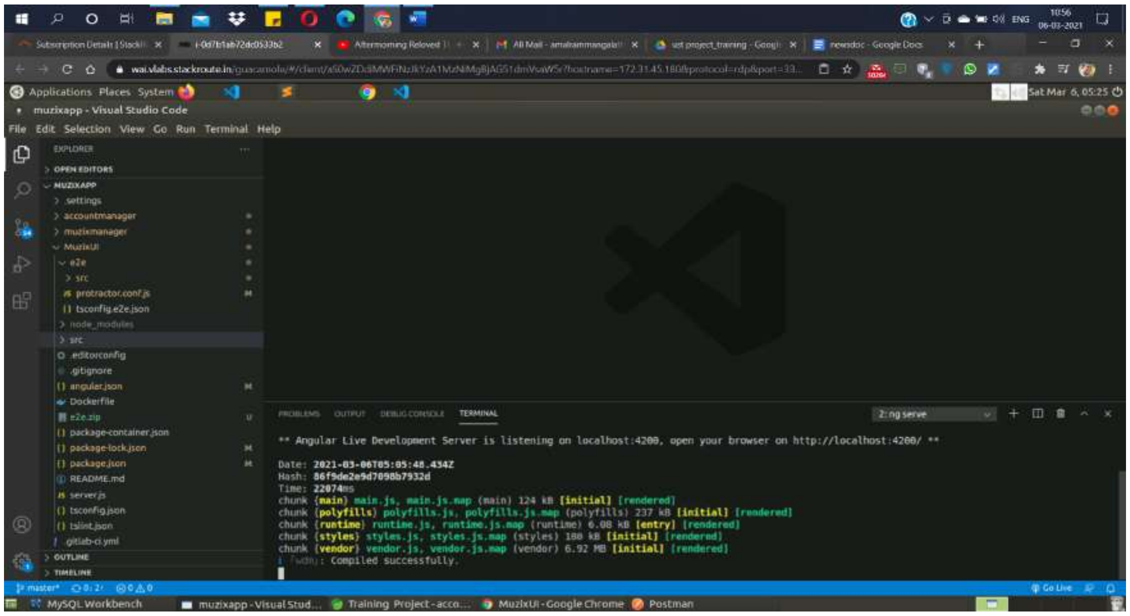
2.6.1 Postman

Postman is an interactive and automatic tool for verifying the APIs of your project. Postman is a Google Chrome app for interacting with HTTP APIs. It presents you with a friendly GUI for constructing requests and reading responses. It works on the backend, and makes sure that each API is working as intended.

In Postman, we create a request, and Postman looks at the response to make sure it has the element we want in it. As it is an automation tool, it drastically improves testing time and quality of the project. It helps in the early detection of bugs that might sprout at later stages and cause more damage to the system.Postman is the way to streamline the process of API testing. All APIs that we create and deploy first rigorously go through Postman so that any major or show stopper bugs are identified on time and fewer bugs leak through to later stages.

3. PROJECT HIERARCHY

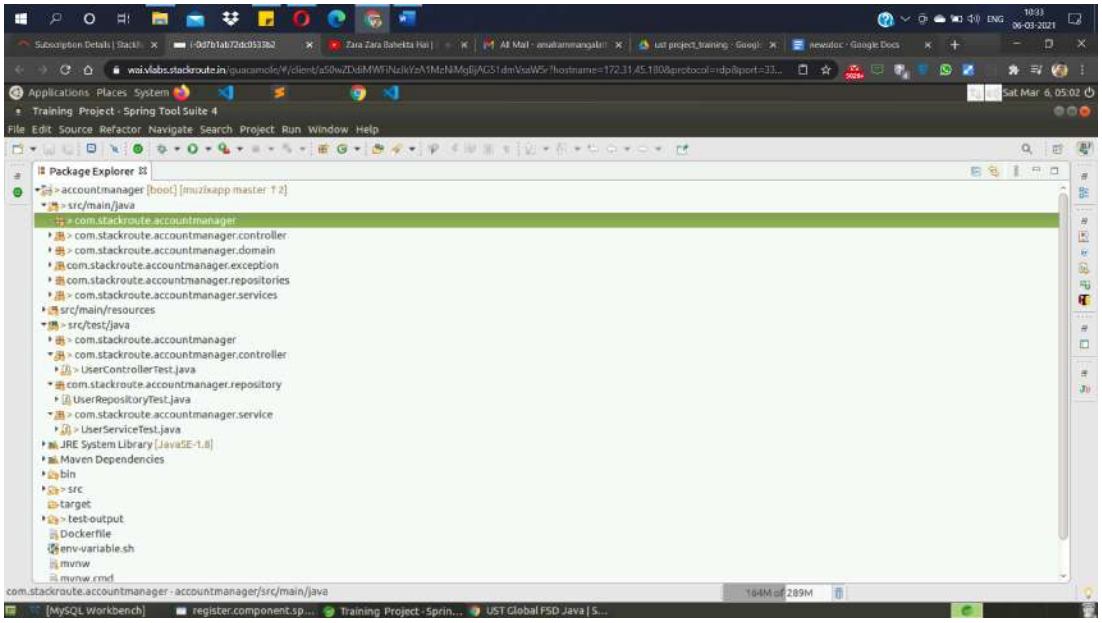
The MuzixUI should be run first for running the whole project. Then the Account Manager and Muzix Manager have to run by Spring Boot App. After successful running , users can be able Register, Login, Search track, add/delete music to playList, add/remove music to bookmark, surf to suggestions and LogOut. The MuzixApp Project hierarchy is shown below Figures which includes both testing and developing part of the project.


muzixapp - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: ng serve
master*
MySQL Workbench
muzixapp - Visual Stud...
Training Project - acco...
MuzixUI - Google Chrome
Postman

```
> OPEN EDITORS
v MUZIXAPP
  > .settings
  > accountmanager
  > muzixmanager
  v MuzixUI
    v e2e
      > src
      protractor.conf.js
      tsconfig.e2e.json
    > node_modules
    > src
    .editorconfig
    .gitignore
    angular.json
    Dockerfile
    e2e.zip
    package-container.json
    package-lock.json
    package.json
    README.md
    server.js
    tsconfig.json
    tslint.json
   .gitlab-ci.yml
> OUTLINE
> TIMELINE
```

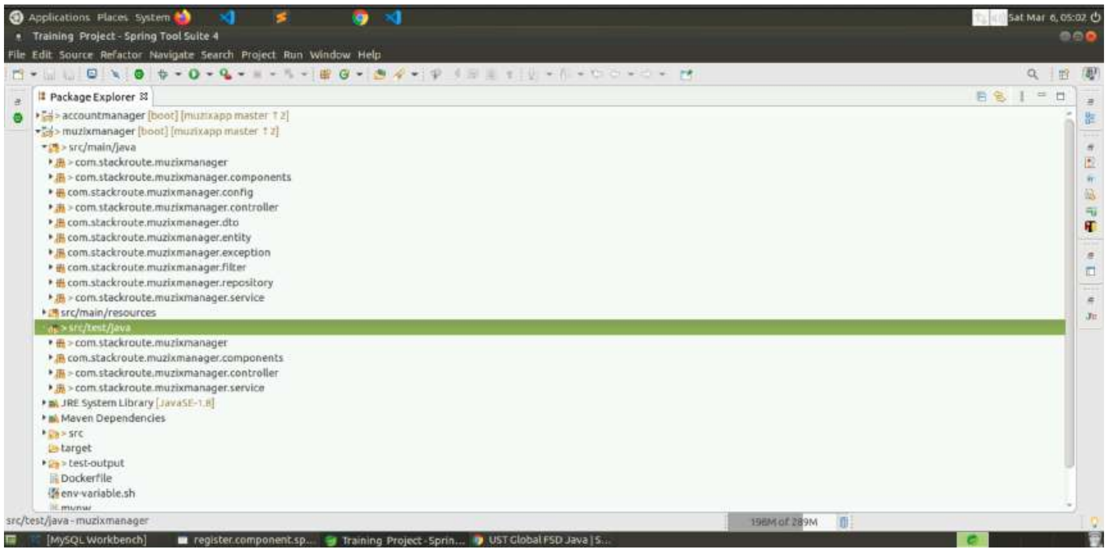
```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

Date: 2021-03-06T05:05:48.434Z
Hash: 86f9de2e9d7098b7932d
Time: 22074ms
chunk {main} main.js, main.js.map (main) 124 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 237 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.08 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 180 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 6.92 MB [initial] [rendered]
ℹ ｢wdm｣: Compiled successfully.
```

Fig. 3.1: MuzixApp


Training Project - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer

```
> accountmanager [boot] [muzixapp master ↑2]
  > src/main/java
    > com.stackroute.accountmanager
    > com.stackroute.accountmanager.controller
    > com.stackroute.accountmanager.domain
    com.stackroute.accountmanager.exception
    com.stackroute.accountmanager.repositories
    > com.stackroute.accountmanager.services
  src/main/resources
  > src/test/java
    > com.stackroute.accountmanager
    > com.stackroute.accountmanager.controller
      > UserControllerTest.java
    com.stackroute.accountmanager.repository
      UserRepositoryTest.java
    > com.stackroute.accountmanager.service
      > UserServiceTest.java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  bin
  > src
  target
  > test-output
  Dockerfile
  env-variable.sh
  mvnw
  mvnw.cmd
com.stackroute.accountmanager - accountmanager/src/main/java
```

Fig: 3.2 AccountManager


Training Project - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer

```
> accountmanager [boot] [muzixapp master ↑2]
> muzixmanager [boot] [muzixapp master ↑2]
  > src/main/java
    > com.stackroute.muzixmanager
    > com.stackroute.muzixmanager.components
    com.stackroute.muzixmanager.config
    > com.stackroute.muzixmanager.controller
    com.stackroute.muzixmanager.dto
    com.stackroute.muzixmanager.entity
    com.stackroute.muzixmanager.exception
    com.stackroute.muzixmanager.filter
    com.stackroute.muzixmanager.repository
    > com.stackroute.muzixmanager.service
  src/main/resources
  > src/test/java
    > com.stackroute.muzixmanager
    com.stackroute.muzixmanager.components
    > com.stackroute.muzixmanager.controller
    > com.stackroute.muzixmanager.service
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  > src
  target
  > test-output
  Dockerfile
  env-variable.sh
  mvnw
src/test/java - muzixmanager
```

Fig: 3.3 MuzixManager.

4. BACKEND TESTING

Here we test the different functionalities of the project. This testing is done in order to make sure that all the given functionalities are behaving and responding as per the standards and the client requirements.

For the testing purpose, the technologies used are :

- Mockito+J-Unit testing- For back-end testing.
- Rest-Assured+TestNg- For back-end testing.
- Protractor+Jasmine framework- For front-end testing.

As per the hierarchy , the testing is done for the backend initially that consists of two modules.

- Account Manager module.
- Muzix Manager module.

From this first Back-end testing is performed on the Account Manager module.

<u>4.1Account Manager Module:</u>

Account Manager module consists testing portions for :

4.1.1.User Controller Test
4.1.2.User Service Test

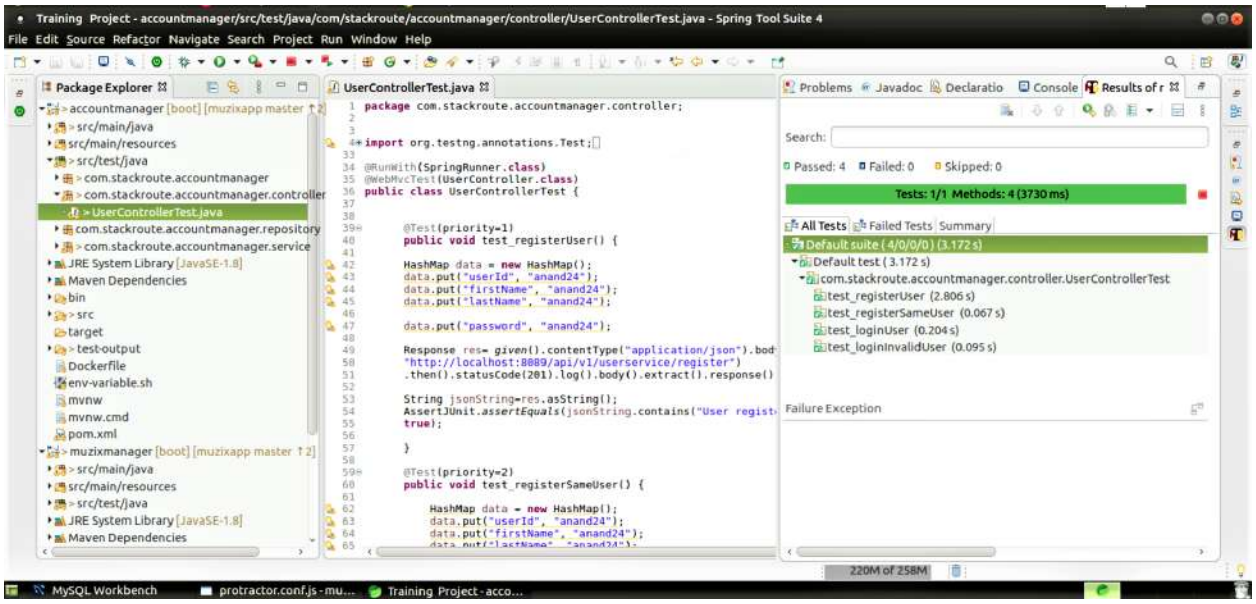
4.1.1.USER CONTROLLER TEST :-

The User Controller part is tested using the Rest-Assured+TestNg framework. The functionalities assigned in the controller part are:

- Register user(Saving the user)
- Logging in the user.

For all the above mentioned functionalities the positive and the negative test cases are being performed. Total test cases runned and the results are also recorded.

The User Controller is tested by using the RestAssured+TestNG Test framework. Tested by Positive and negative test cases. The Expected results are recorded

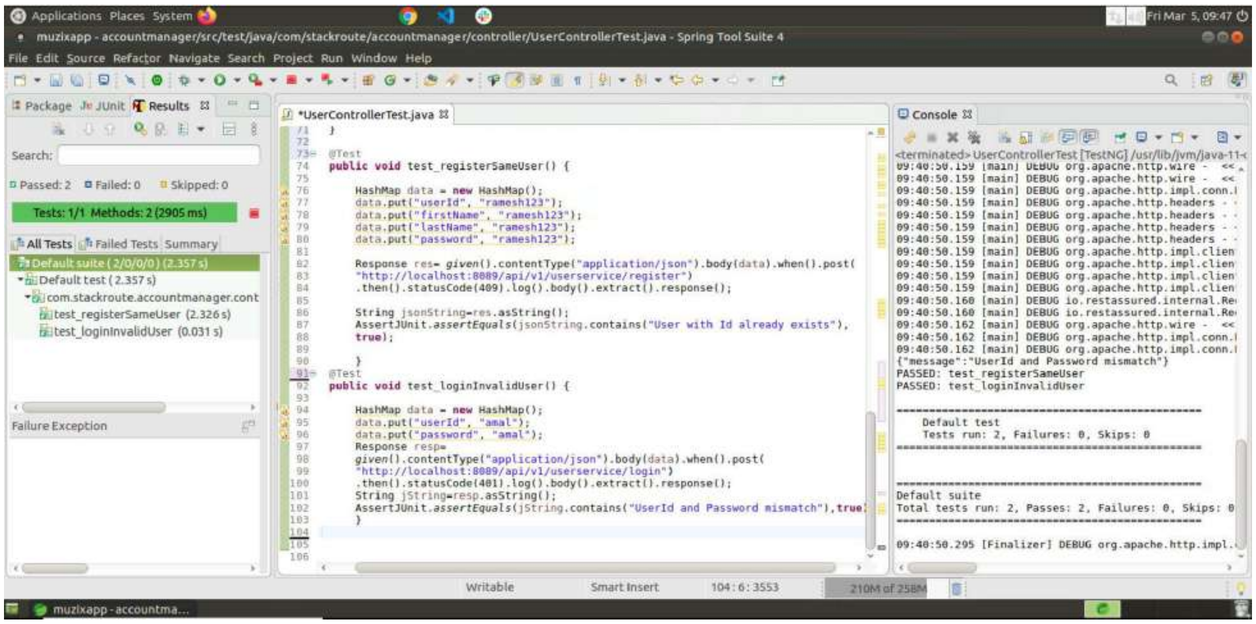


Training Project - accountmanager/src/test/java/com/stackroute/accountmanager/controller/UserControllerTest.java - Spring Tool Suite 4

```java
package com.stackroute.accountmanager.controller;


import org.testng.annotations.Test;

@RunWith(SpringRunner.class)
@WebMvcTest(UserController.class)
public class UserControllerTest {


        @Test(priority=1)
        public void test_registerUser() {

        HashMap data = new HashMap();
        data.put("userId", "anand24");
        data.put("firstName", "anand24");
        data.put("lastName", "anand24");

        data.put("password", "anand24");

        Response res= given().contentType("application/json").bod
        "http://localhost:8089/api/v1/userservice/register")
        .then().statusCode(201).log().body().extract().response()

        String jsonString=res.asString();
        AssertJUnit.assertEquals(jsonString.contains("User regist
        true);

        }

        @Test(priority=2)
        public void test_registerSameUser() {

            HashMap data = new HashMap();
            data.put("userId", "anand24");
            data.put("firstName", "anand24");
```

```
Search:
Passed: 4   Failed: 0   Skipped: 0
Tests: 1/1  Methods: 4 (3730 ms)

All Tests | Failed Tests | Summary
Default suite ( 4/0/0/0 ) (3.172 s)
  Default test ( 3.172 s)
    com.stackroute.accountmanager.controller.UserControllerTest
      test_registerUser  (2.806 s)
      test_registerSameUser  (0.067 s)
      test_loginUser  (0.204 s)
      test_loginInvalidUser  (0.095 s)

Failure Exception
```

Fig: 4.1 Code Snippet of Test Cases of User Controller



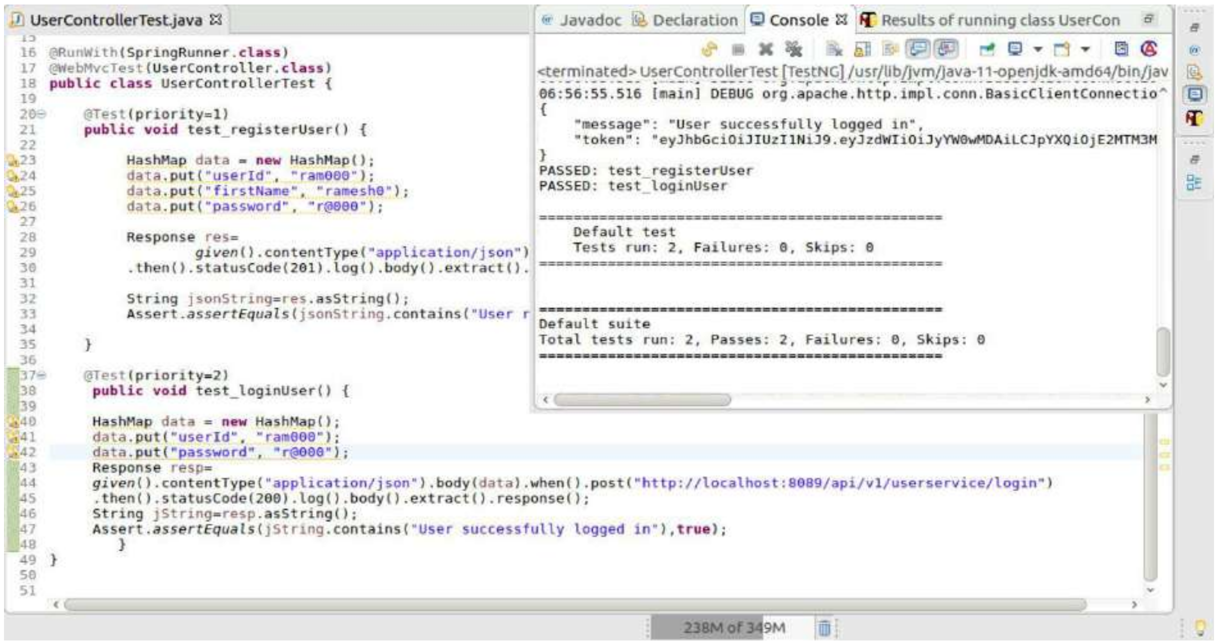
muzixapp - accountmanager/src/test/java/com/stackroute/accountmanager/controller/UserControllerTest.java - Spring Tool Suite 4

```java
    }

    @Test
    public void test_registerSameUser() {

        HashMap data = new HashMap();
        data.put("userId", "ramesh123");
        data.put("firstName", "ramesh123");
        data.put("lastName", "ramesh123");
        data.put("password", "ramesh123");

        Response res= given().contentType("application/json").body(data).when().post(
        "http://localhost:8089/api/v1/userservice/register")
        .then().statusCode(409).log().body().extract().response();

        String jsonString=res.asString();
        AssertJUnit.assertEquals(jsonString.contains("User with Id already exists"),
        true);

        }
    @Test
    public void test_loginInvalidUser() {

        HashMap data = new HashMap();
        data.put("userId", "amal");
        data.put("password", "amal");
        Response resp=
        given().contentType("application/json").body(data).when().post(
        "http://localhost:8089/api/v1/userservice/login")
        .then().statusCode(401).log().body().extract().response();
        String jString=resp.asString();
        AssertJUnit.assertEquals(jString.contains("UserId and Password mismatch"),true
        }
```

```
Search:
Passed: 2   Failed: 0   Skipped: 0
Tests: 1/1  Methods: 2 (2905 ms)

All Tests | Failed Tests | Summary
Default suite ( 2/0/0/0 ) (2.357 s)
  Default test ( 2.357 s)
    com.stackroute.accountmanager.cont
      test_registerSameUser  (2.326 s)
      test_loginInvalidUser  (0.031 s)

Failure Exception
```

```
<terminated> UserControllerTest [TestNG] /usr/lib/jvm/java-11-
09:40:50.159 [main] DEBUG org.apache.http.wire -  <<
09:40:50.159 [main] DEBUG org.apache.http.wire -  <<
09:40:50.159 [main] DEBUG org.apache.http.impl.conn.
09:40:50.159 [main] DEBUG org.apache.http.headers -
09:40:50.159 [main] DEBUG org.apache.http.headers -
09:40:50.159 [main] DEBUG org.apache.http.headers -
09:40:50.159 [main] DEBUG org.apache.http.headers -
09:40:50.159 [main] DEBUG org.apache.http.impl.clien
09:40:50.159 [main] DEBUG org.apache.http.impl.clien
09:40:50.159 [main] DEBUG org.apache.http.impl.clien
09:40:50.159 [main] DEBUG org.apache.http.impl.clien
09:40:50.160 [main] DEBUG io.restassured.internal.Re
09:40:50.160 [main] DEBUG io.restassured.internal.Re
09:40:50.162 [main] DEBUG org.apache.http.wire -  <<
09:40:50.162 [main] DEBUG org.apache.http.impl.conn.
09:40:50.162 [main] DEBUG org.apache.http.impl.conn.
{"message":"UserId and Password mismatch"}
PASSED: test_registerSameUser
PASSED: test_loginInvalidUser

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 2, Passes: 2, Failures: 0, Skips: 0
===============================================

09:40:50.295 [Finalizer] DEBUG org.apache.http.impl.
```

Fig:4.2 Code Snippet of negative Test cases for User Controller test.



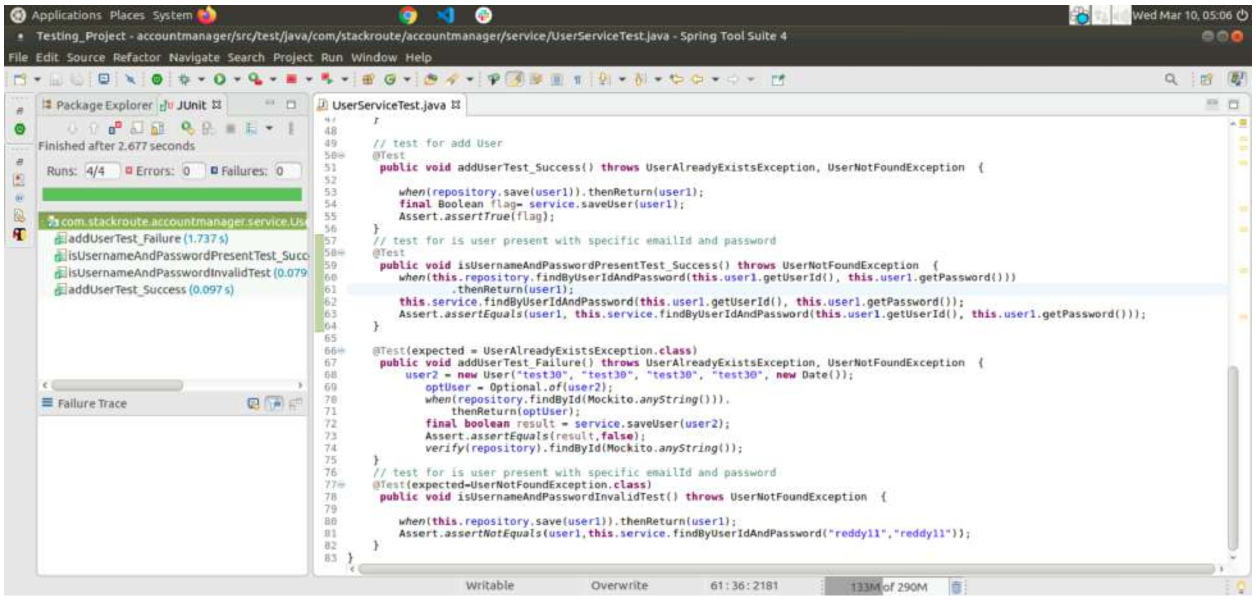
```java
@RunWith(SpringRunner.class)
@WebMvcTest(UserController.class)
public class UserControllerTest {

    @Test(priority=1)
    public void test_registerUser() {

         HashMap data = new HashMap();
         data.put("userId", "ram000");
         data.put("firstName", "ramesh0");
         data.put("password", "r@000");

         Response res=
                 given().contentType("application/json")
         .then().statusCode(201).log().body().extract().

         String jsonString=res.asString();
         Assert.assertEquals(jsonString.contains("User r

    }

    @Test(priority=2)
     public void test_loginUser() {

     HashMap data = new HashMap();
     data.put("userId", "ram000");
     data.put("password", "r@000");
     Response resp=
     given().contentType("application/json").body(data).when().post("http://localhost:8089/api/v1/userservice/login")
     .then().statusCode(200).log().body().extract().response();
     String jString=resp.asString();
     Assert.assertEquals(jString.contains("User successfully logged in"),true);
        }
}
```

```
<terminated> UserControllerTest [TestNG] /usr/lib/jvm/java-11-openjdk-amd64/bin/jav
06:56:55.516 [main] DEBUG org.apache.http.impl.conn.BasicClientConnectio
{
    "message": "User successfully logged in",
    "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJyYW0wMDAiLCJpYXQiOjE2MTM3M
}
PASSED: test_registerUser
PASSED: test_loginUser

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 2, Passes: 2, Failures: 0, Skips: 0
===============================================
```

Fig:4.3Account Manager User Controller Test using RestAssured-TestNG tool.

4.1.2.USER SERVICE TEST :-

The User Service is tested by using the Mockito+JUnit Test framework.



```
Testing_Project - accountmanager/src/test/java/com/stackroute/accountmanager/service/UserServiceTest.java - Spring Tool Suite 4

Finished after 2.677 seconds
Runs: 4/4    Errors: 0    Failures: 0

com.stackroute.accountmanager.service.Us
  addUserTest_Failure (1.737 s)
  isUsernameAndPasswordPresentTest_Succ
  isUsernameAndPasswordInvalidTest (0.079
  addUserTest_Success (0.097 s)
```

```java
    // test for add User
    @Test
     public void addUserTest_Success() throws UserAlreadyExistsException, UserNotFoundException  {

        when(repository.save(user1)).thenReturn(user1);
        final Boolean flag= service.saveUser(user1);
        Assert.assertTrue(flag);
    }
    // test for is user present with specific emailId and password
    @Test
     public void isUsernameAndPasswordPresentTest_Success() throws UserNotFoundException  {
        when(this.repository.findByUserIdAndPassword(this.user1.getUserId(), this.user1.getPassword()))
                .thenReturn(user1);
        this.service.findByUserIdAndPassword(this.user1.getUserId(), this.user1.getPassword());
        Assert.assertEquals(user1, this.service.findByUserIdAndPassword(this.user1.getUserId(), this.user1.getPassword()));
    }

    @Test(expected = UserAlreadyExistsException.class)
     public void addUserTest_Failure() throws UserAlreadyExistsException, UserNotFoundException  {
          user2 = new User("test30", "test30", "test30", "test30", new Date());
            optUser = Optional.of(user2);
            when(repository.findById(Mockito.anyString())).
                thenReturn(optUser);
            final boolean result = service.saveUser(user2);
            Assert.assertEquals(result,false);
            verify(repository).findById(Mockito.anyString());
    }
    // test for is user present with specific emailId and password
    @Test(expected=UserNotFoundException.class)
     public void isUsernameAndPasswordInvalidTest() throws UserNotFoundException  {

        when(this.repository.save(user1)).thenReturn(user1);
        Assert.assertNotEquals(user1,this.service.findByUserIdAndPassword("reddy11","reddy11"));
    }
}
```

Fig 4.4: Code snippet of User Service Test.

4.2 Muzix Manager Module:

Muzix Manager module consists testing portions for :

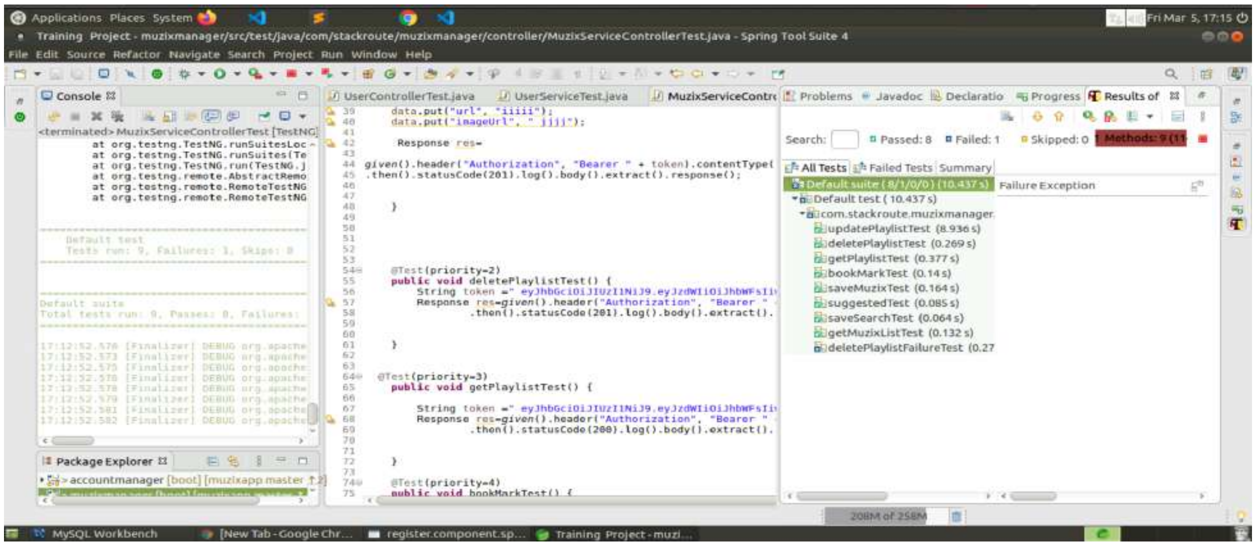
4.2.1.Muzix Controller Test
4.2.2.Bookmark Service Test
4.2.3.Muzix Service Test
4.2.4.PlayList Service Test
4.2.5.CreateHistory Service Test
4.2.6.User Service Test

4.2.1.MUZIX CONTROLLER TEST :-

The User Controller part is tested using the Rest-Assured+TestNg framework. The functionalities assigned in the controller part are:

- Register user(Saving the user)
- Logging in the user.

For all the above mentioned functionalities the positive are being performed. Total test cases runned and the results are also recorded.



Applications Places System
Fri Mar 5, 17:15
Training Project - muzixmanager/src/test/java/com/stackroute/muzixmanager/controller/MuzixServiceControllerTest.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Console
UserControllerTest.java
UserServiceTest.java
MuzixServiceContr
Problems
Javadoc
Declaratio
Progress
Results of
Package Explorer
MySQL Workbench
[New Tab - Google Chr...
register.component.sp...
Training Project - muzi...
208M of 258M

```
<terminated> MuzixServiceControllerTest [TestNG]
        at org.testng.TestNG.runSuitesLoc
        at org.testng.TestNG.runSuites(Te
        at org.testng.TestNG.run(TestNG.j
        at org.testng.remote.AbstractRemo
        at org.testng.remote.RemoteTestNG
        at org.testng.remote.RemoteTestNG

===============================================
    Default test
    Tests run: 9, Failures: 1, Skips: 0
===============================================

===============================================
Default suite
Total tests run: 9, Passes: 8, Failures:
===============================================

17:12:52.570 [Finalizer] DEBUG org.apache
17:12:52.573 [Finalizer] DEBUG org.apache
17:12:52.575 [Finalizer] DEBUG org.apache
17:12:52.578 [Finalizer] DEBUG org.apache
17:12:52.578 [Finalizer] DEBUG org.apache
17:12:52.579 [Finalizer] DEBUG org.apache
17:12:52.581 [Finalizer] DEBUG org.apache
17:12:52.582 [Finalizer] DEBUG org.apache
```

```
accountmanager [boot] [muzixapp master ↑2]
```

```java
        data.put("url", "iiiii");
        data.put("imageUrl", " jjjj");

         Response res=
given().header("Authorization", "Bearer " + token).contentType(
.then().statusCode(201).log().body().extract().response();


        }





        @Test(priority=2)
        public void deletePlaylistTest() {
            String token =" eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbWFsIi
            Response res=given().header("Authorization", "Bearer " 
                     .then().statusCode(201).log().body().extract().


        }


     @Test(priority=3)
        public void getPlaylistTest() {

            String token =" eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbWFsIi
            Response res=given().header("Authorization", "Bearer " 
                     .then().statusCode(200).log().body().extract().


        }

        @Test(priority=4)
        public void bookMarkTest() {
```

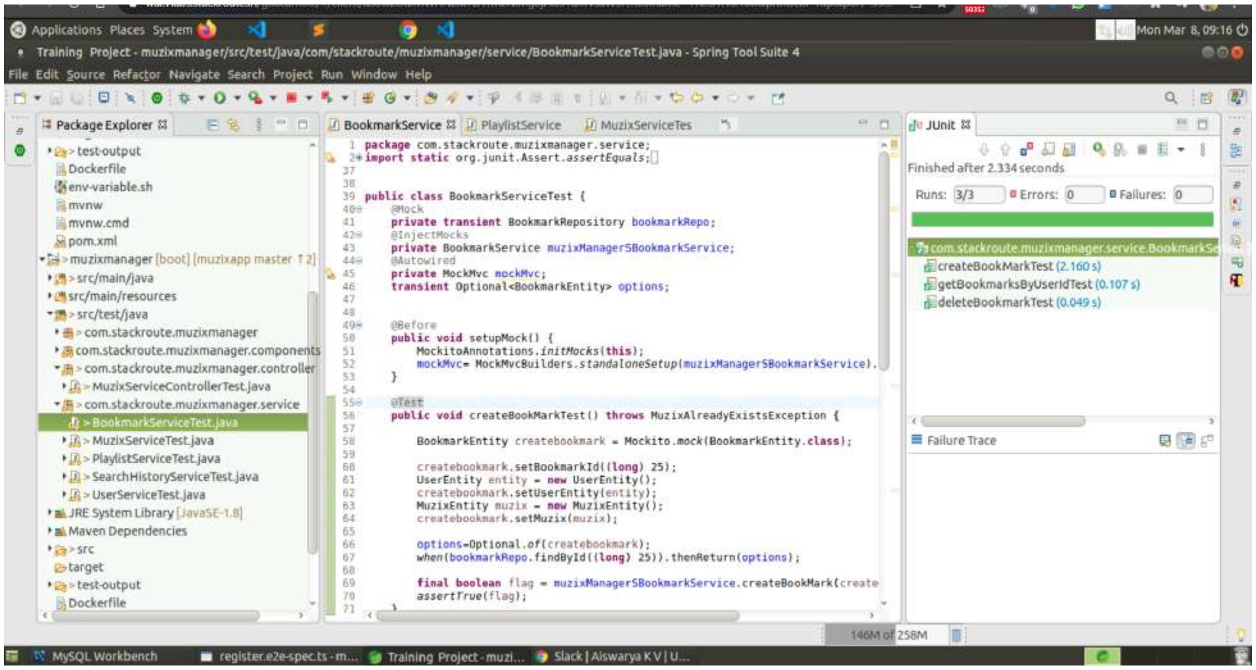
```
Search:    Passed: 8    Failed: 1    Skipped: 0    Methods: 9 (11
All Tests | Failed Tests | Summary
Default suite ( 8/1/0/0 ) (10.437 s)        Failure Exception
  Default test ( 10.437 s)
    com.stackroute.muzixmanager.
      updatePlaylistTest (8.936 s)
      deletePlaylistTest (0.269 s)
      getPlaylistTest (0.377 s)
      bookMarkTest (0.14 s)
      saveMuzixTest (0.164 s)
      suggestedTest (0.085 s)
      saveSearchTest (0.064 s)
      getMuzixListTest (0.132 s)
      deletePlaylistFailureTest (0.27
```

Fig 4.4 MuzixControllerTest

4.2.2.BOOKMARK SERVICE TEST:-

The Bookmark Service Test is tested by using the Mockito+JUnit Test framework.The functionalities assigned are:

- Create bookmark
- Delete bookmark.
- GetBookmark by userId.


Applications Places System
Mon Mar 8, 09:16
Training Project - muzixmanager/src/test/java/com/stackroute/muzixmanager/service/BookmarkServiceTest.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
test-output
Dockerfile
env-variable.sh
mvnw
mvnw.cmd
pom.xml
muzixmanager [boot] [muzixapp master ↑2]
src/main/java
src/main/resources
src/test/java
com.stackroute.muzixmanager
com.stackroute.muzixmanager.components
com.stackroute.muzixmanager.controller
MuzixServiceControllerTest.java
com.stackroute.muzixmanager.service
BookmarkServiceTest.java
MuzixServiceTest.java
PlaylistServiceTest.java
SearchHistoryServiceTest.java
UserServiceTest.java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
test-output
Dockerfile
BookmarkService
PlaylistService
MuzixServiceTes
JUnit
Failure Trace
146M of 258M
MySQL Workbench
register.e2e-spec.ts - m...
Training Project - muzi...
Slack | Aiswarya K V | U...

```java
package com.stackroute.muzixmanager.service;
import static org.junit.Assert.assertEquals;


public class BookmarkServiceTest {
    @Mock
    private transient BookmarkRepository bookmarkRepo;
    @InjectMocks
    private BookmarkService muzixManagerSBookmarkService;
    @Autowired
    private MockMvc mockMvc;
    transient Optional<BookmarkEntity> options;


    @Before
    public void setupMock() {
        MockitoAnnotations.initMocks(this);
        mockMvc= MockMvcBuilders.standaloneSetup(muzixManagerSBookmarkService).
    }

    @Test
    public void createBookMarkTest() throws MuzixAlreadyExistsException {

        BookmarkEntity createbookmark = Mockito.mock(BookmarkEntity.class);

        createbookmark.setBookmarkId((long) 25);
        UserEntity entity = new UserEntity();
        createbookmark.setUserEntity(entity);
        MuzixEntity muzix = new MuzixEntity();
        createbookmark.setMuzix(muzix);

        options=Optional.of(createbookmark);
        when(bookmarkRepo.findById((long) 25)).thenReturn(options);

        final boolean flag = muzixManagerSBookmarkService.createBookMark(create
        assertTrue(flag);
    }
```

```
Finished after 2.334 seconds
Runs: 3/3   Errors: 0   Failures: 0
com.stackroute.muzixmanager.service.BookmarkSe
  createBookMarkTest (2.160 s)
  getBookmarksByUserIdTest (0.107 s)
  deleteBookmarkTest (0.049 s)
```

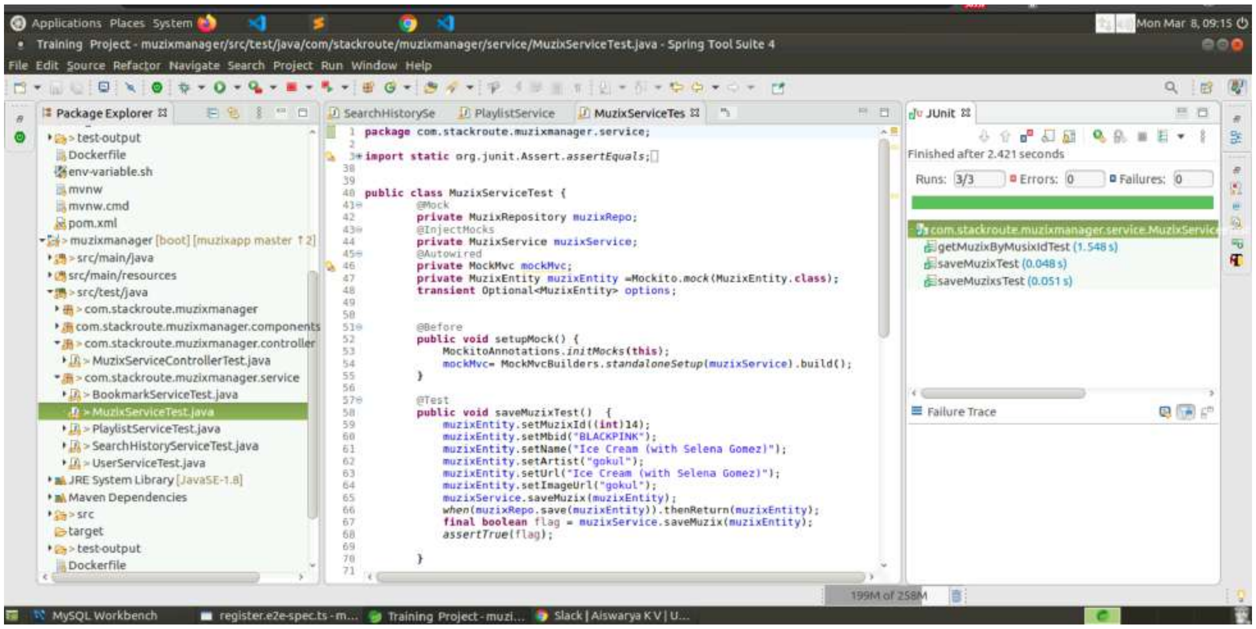
Fig 4.5 BookmarkService Test

For all the above mentioned functionalities the positive are being performed. Total test cases runned and the results are also recorded.

4.2.3.MUZIX SERVICE TEST:-

The Bookmark Service Test is tested by using the Mockito+JUnit Test framework.The functionalities assigned are:

- Save music
- Save music List.
- Get Music by musicId



```
package com.stackroute.muzixmanager.service;

import static org.junit.Assert.assertEquals;


public class MuzixServiceTest {
        @Mock
        private MuzixRepository muzixRepo;
        @InjectMocks
        private MuzixService muzixService;
        @Autowired
        private MockMvc mockMvc;
        private MuzixEntity muzixEntity =Mockito.mock(MuzixEntity.class);
        transient Optional<MuzixEntity> options;


        @Before
        public void setupMock() {
            MockitoAnnotations.initMocks(this);
            mockMvc= MockMvcBuilders.standaloneSetup(muzixService).build();
        }

        @Test
        public void saveMuzixTest()  {
            muzixEntity.setMuzixId((int)14);
            muzixEntity.setMbid("BLACKPINK");
            muzixEntity.setName("Ice Cream (with Selena Gomez)");
            muzixEntity.setArtist("gokul");
            muzixEntity.setUrl("Ice Cream (with Selena Gomez)");
            muzixEntity.setImageUrl("gokul");
            muzixService.saveMuzix(muzixEntity);
            when(muzixRepo.save(muzixEntity)).thenReturn(muzixEntity);
            final boolean flag = muzixService.saveMuzix(muzixEntity);
            assertTrue(flag);

        }
```

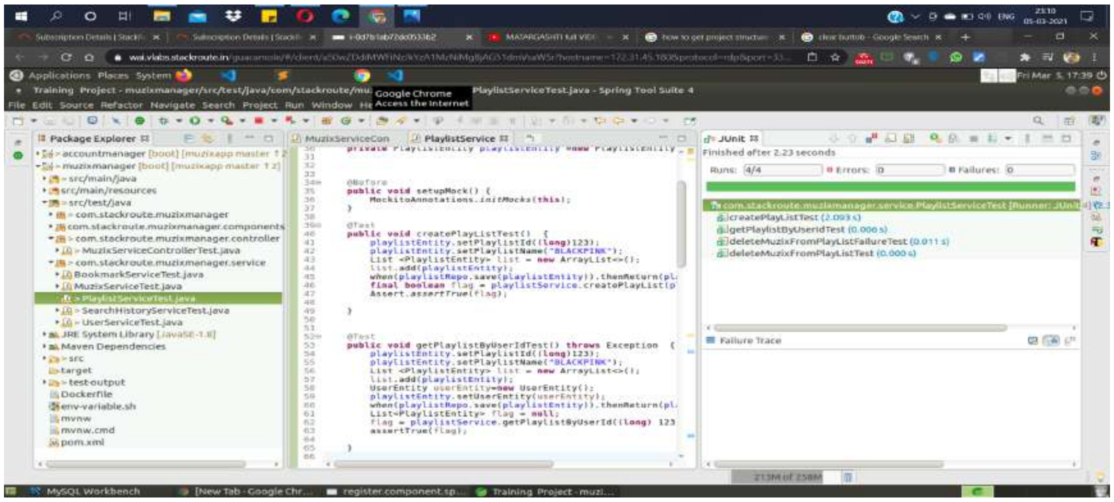
```
Finished after 2.421 seconds
Runs: 3/3    Errors: 0    Failures: 0

com.stackroute.muzixmanager.service.MuzixService
  getMuzixByMusixIdTest (1.548 s)
  saveMuzixTest (0.048 s)
  saveMuzixsTest (0.051 s)
```

Fig 4.6: MuzixServiceTest

4.2.4.PLAYLIST SERVICE TEST:-

The Bookmark Service Test is tested by using the Mockito+JUnit Test framework.The functionalities assigned are:

- create playlist
- get the playlist by user id .
- delete music from playlist.



```
    @Before
    public void setupMock() {
        MockitoAnnotations.initMocks(this);
    }

    @Test
    public void createPlayListTest()  {
        playlistEntity.setPlaylistId((long)123);
        playlistEntity.setPlaylistName("BLACKPINK");
        List <PlaylistEntity> list = new ArrayList<>();
        list.add(playlistEntity);
        when(playlistRepo.save(playlistEntity)).thenReturn(pl
        final boolean flag = playlistService.createPlayList(p
        Assert.assertTrue(flag);

    }


    @Test
    public void getPlaylistByUserIdTest() throws Exception  {
        playlistEntity.setPlaylistId((long)123);
        playlistEntity.setPlaylistName("BLACKPINK");
        List <PlaylistEntity> list = new ArrayList<>();
        list.add(playlistEntity);
        UserEntity userEntity=new UserEntity();
        playlistEntity.setUserEntity(userEntity);
        when(playlistRepo.save(playlistEntity)).thenReturn(pl
        List<PlaylistEntity> flag = null;
        flag = playlistService.getPlaylistByUserId((long) 123
        assertTrue(flag);

    }
```

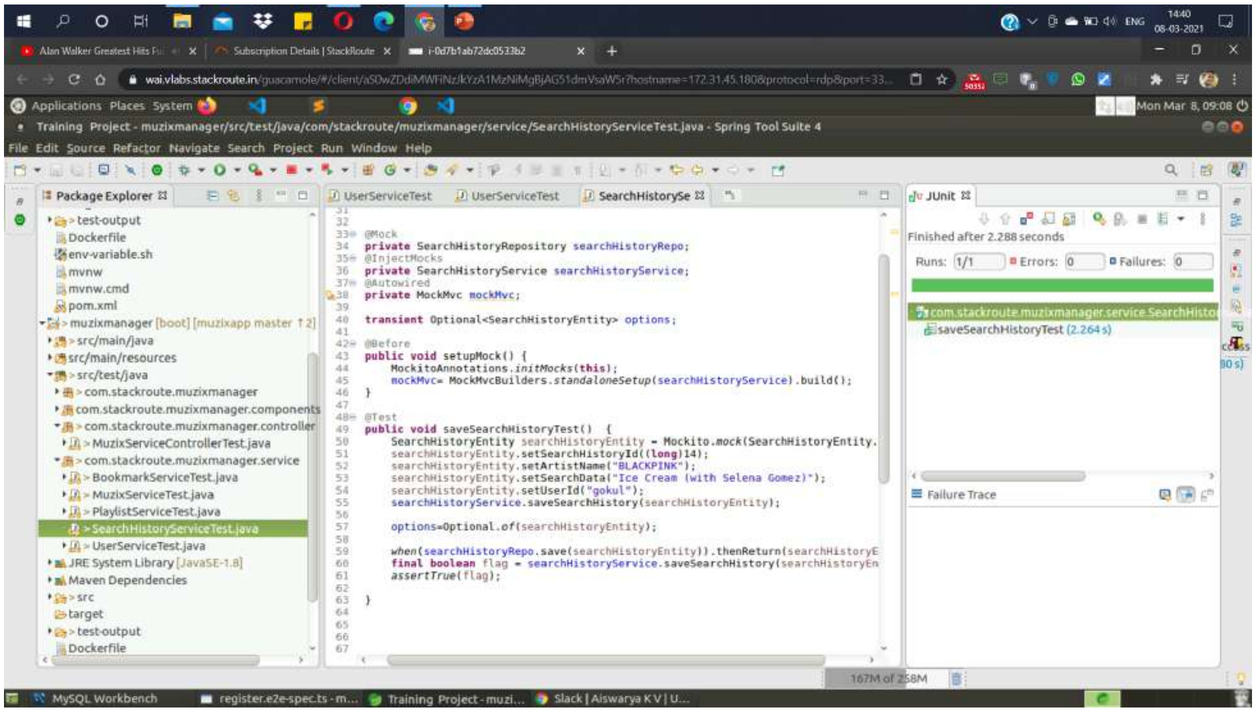
```
Finished after 2.23 seconds
Runs: 4/4    Errors: 0    Failures: 0

com.stackroute.muzixmanager.service.PlaylistServiceTest [Runner: JUnit 4]
  createPlayListTest (2.093 s)
  getPlaylistByUserIdTest (0.006 s)
  deleteMuzixFromPlayListFailureTest (0.011 s)
  deleteMuzixFromPlayListTest (0.000 s)
```

Fig 4.7 playlistServiceTest

4.2.5. SEARCH HISTORY SERVICE TEST:-

The Bookmark Service Test is tested by using the Mockito+JUnit Test framework.The functionalities assigned is:

- Save search History



```java
@Mock
private SearchHistoryRepository searchHistoryRepo;
@InjectMocks
private SearchHistoryService searchHistoryService;
@Autowired
private MockMvc mockMvc;

transient Optional<SearchHistoryEntity> options;

@Before
public void setupMock() {
    MockitoAnnotations.initMocks(this);
    mockMvc= MockMvcBuilders.standaloneSetup(searchHistoryService).build();
}

@Test
public void saveSearchHistoryTest()  {
    SearchHistoryEntity searchHistoryEntity = Mockito.mock(SearchHistoryEntity.
    searchHistoryEntity.setSearchHistoryId((long)14);
    searchHistoryEntity.setArtistName("BLACKPINK");
    searchHistoryEntity.setSearchData("Ice Cream (with Selena Gomez)");
    searchHistoryEntity.setUserId("gokul");
    searchHistoryService.saveSearchHistory(searchHistoryEntity);

    options=Optional.of(searchHistoryEntity);

    when(searchHistoryRepo.save(searchHistoryEntity)).thenReturn(searchHistoryE
    final boolean flag = searchHistoryService.saveSearchHistory(searchHistoryEn
    assertTrue(flag);

}
```

Finished after 2.288 seconds

Runs: 1/1 Errors: 0 Failures: 0

com.stackroute.muzixmanager.service.SearchHistor
saveSearchHistoryTest (2.264 s)

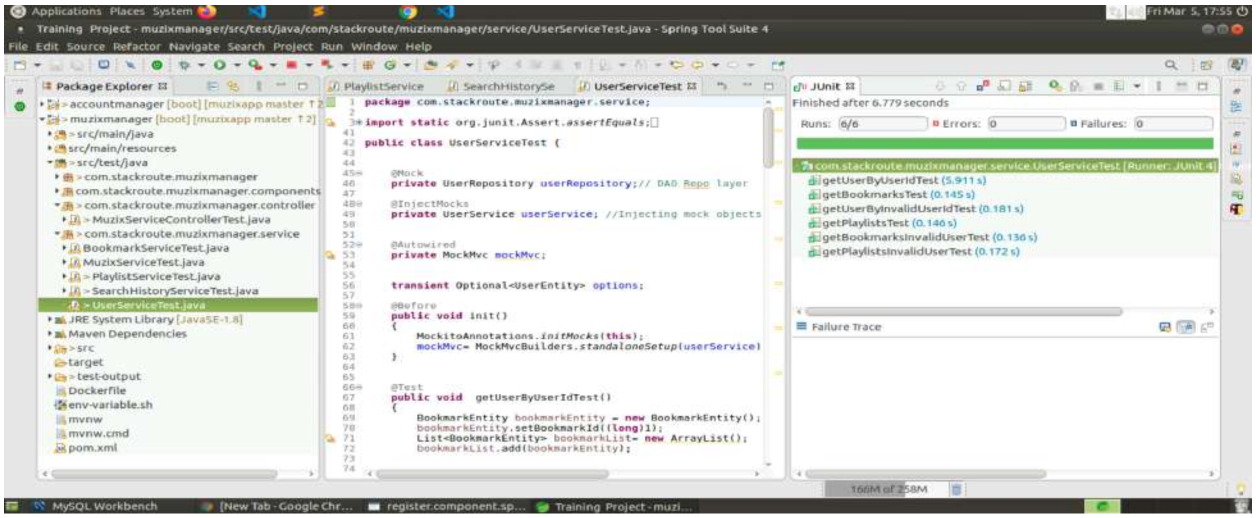
Fig 4.8: SearchHistoryTest

For the above mentioned functionality the positive test case is being performed.

4.2.5.PLAYLIST SERVICE TEST:-

The Bookmark Service Test is tested by using the Mockito+JUnit Test framework.The functionalities assigned are:

- get user by user id.
- get playlist
- getBookmark.



```
package com.stackroute.muzixmanager.service;

import static org.junit.Assert.assertEquals;

public class UserServiceTest {


    @Mock
    private UserRepository userRepository;// DAO Repo layer

    @InjectMocks
    private UserService userService; //Injecting mock objects


    @Autowired
    private MockMvc mockMvc;


    transient Optional<UserEntity> options;

    @Before
    public void init()
    {
        MockitoAnnotations.initMocks(this);
        mockMvc= MockMvcBuilders.standaloneSetup(userService)
    }


    @Test
    public void  getUserByUserIdTest()
    {
        BookmarkEntity bookmarkEntity = new BookmarkEntity();
        bookmarkEntity.setBookmarkId((long)1);
        List<BookmarkEntity> bookmarkList= new ArrayList();
        bookmarkList.add(bookmarkEntity);
```

```
Finished after 6.779 seconds
Runs: 6/6    Errors: 0    Failures: 0
com.stackroute.muzixmanager.service.UserServiceTest [Runner: JUnit 4]
    getUserByUserIdTest (5.911 s)
    getBookmarksTest (0.145 s)
    getUserByInvalidUserIdTest (0.181 s)
    getPlaylistsTest (0.146 s)
    getBookmarksInvalidUserTest (0.136 s)
    getPlaylistsInvalidUserTest (0.172 s)
```

Fig 4.9: UserServiceTest

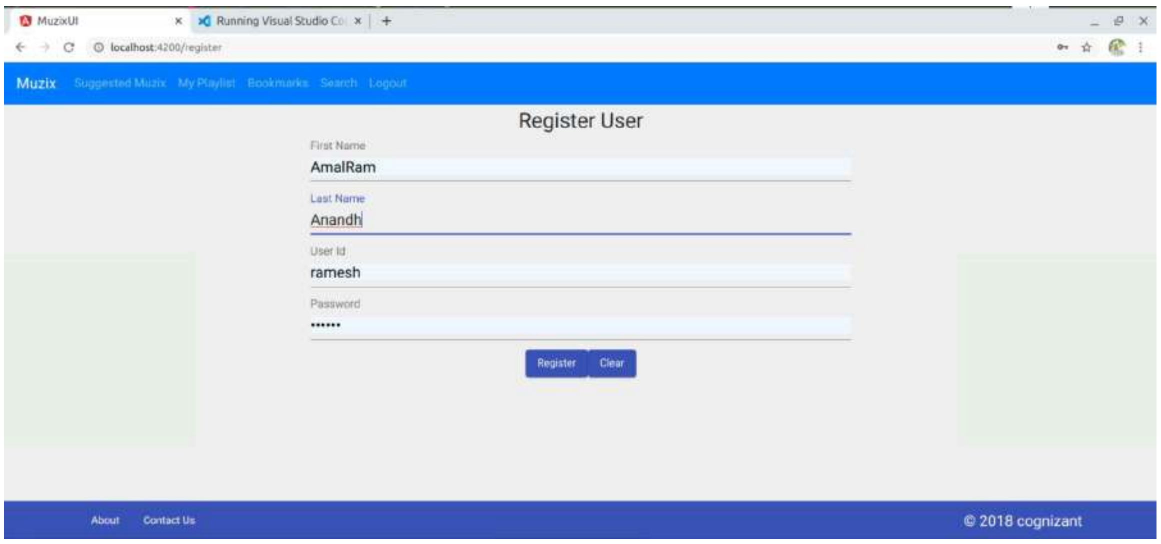
For all the above mentioned functionalities the positive are being performed. Total test cases runned and the results are also recorded.

5 FRONT END TESTING

In this part, let's traverse through the front end pages and test it with using protractor.

5.1 USER REGISTER PAGE:-

Here the user should be able to register themself in the account. Register user contains four fields that are to be filled by the user.In the register user test case browser.get("http://localhost:4200/register") trigger chrome browser to get (http://localhost:4200/register) page. Then browser element will search page elements like Firstname, last name, userid and password and send corresponding values to database when we click on register button and expecting current url in browser be http://localhost:4200/login for successful Registration & http://localhost:4200/register for existing credentials and unsuccessful registration.



Register User

First Name
AmalRam

Last Name
Anandh

User Id
ramesh

Password
••••••

Fig 5.1 User Registration

5.2 USER LOGIN PAGE:-

Here the user should be able to login with valid user id and password.If a user is able to navigate to the search page.Otherwise,if it is invalid credentials then it will remain on the same page.

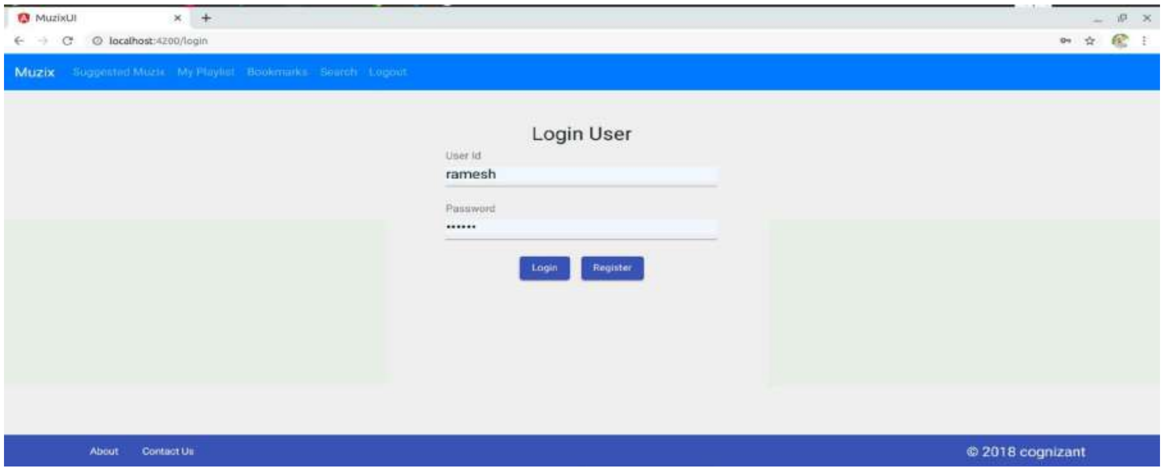


Fig 5.2: User Login Page.

5.3 SEARCH PAGE:-

After login , the user is navigated to the search page.

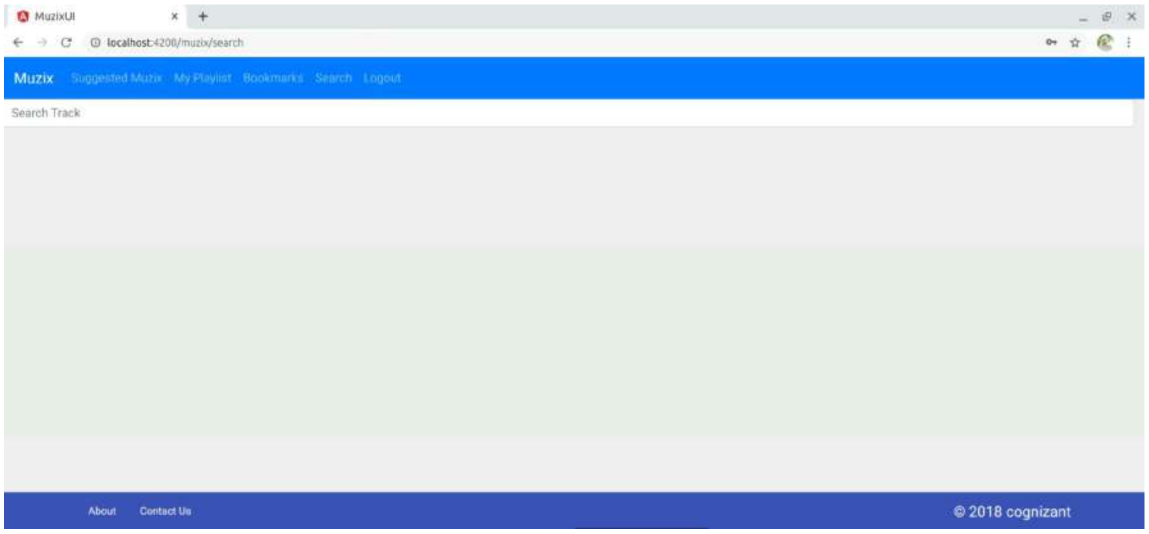
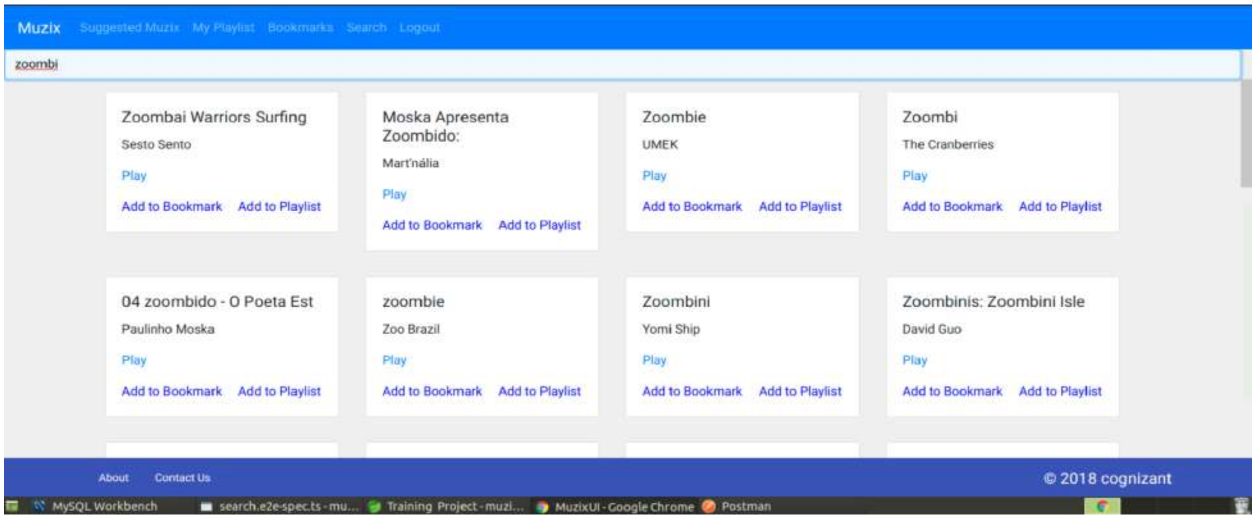


fig 5.3: Search Page

Then the user can search music in the search track and enter the textbox.



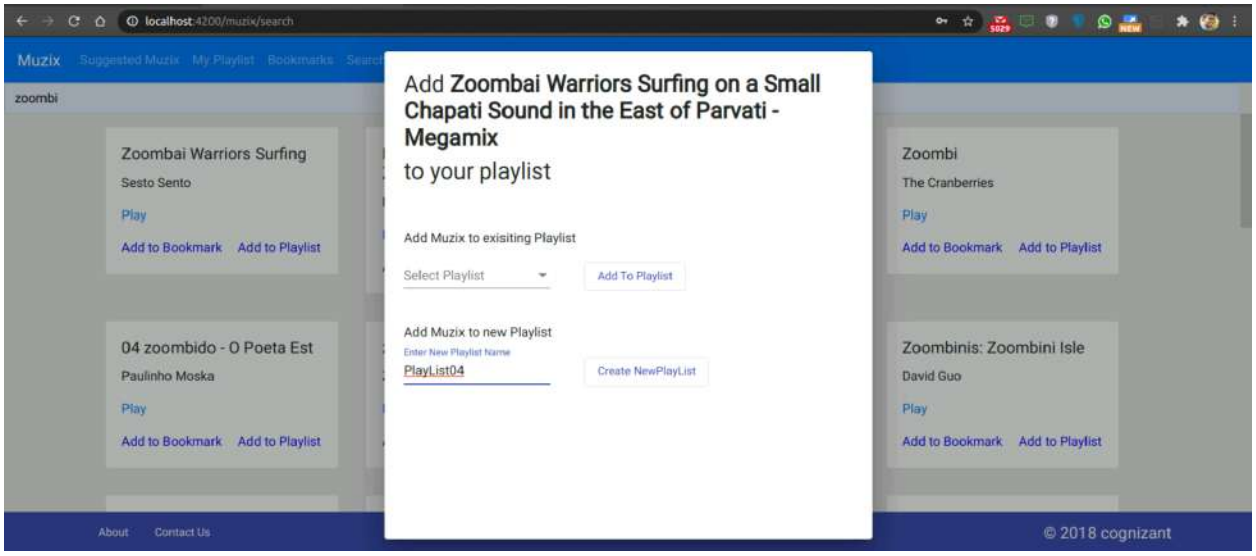
zoombi

Zoombai Warriors Surfing Sesto Sento Play Add to Bookmark Add to Playlist	Moska Apresenta Zoombido: Mart'nália Play Add to Bookmark Add to Playlist	Zoombie UMEK Play Add to Bookmark Add to Playlist	Zoombi The Cranberries Play Add to Bookmark Add to Playlist
04 zoombido - O Poeta Est Paulinho Moska Play Add to Bookmark Add to Playlist	zoombie Zoo Brazil Play Add to Bookmark Add to Playlist	Zoombini Yomi Ship Play Add to Bookmark Add to Playlist	Zoombinis: Zoombini Isle David Guo Play Add to Bookmark Add to Playlist

Fig 5.4 After searching

5.4 ADDING PLAYLIST :-

The user is creating a new playlist.



zoombi

Add **Zoombai Warriors Surfing on a Small Chapati Sound in the East of Parvati - Megamix** to your playlist

Add Muzix to exisiting Playlist

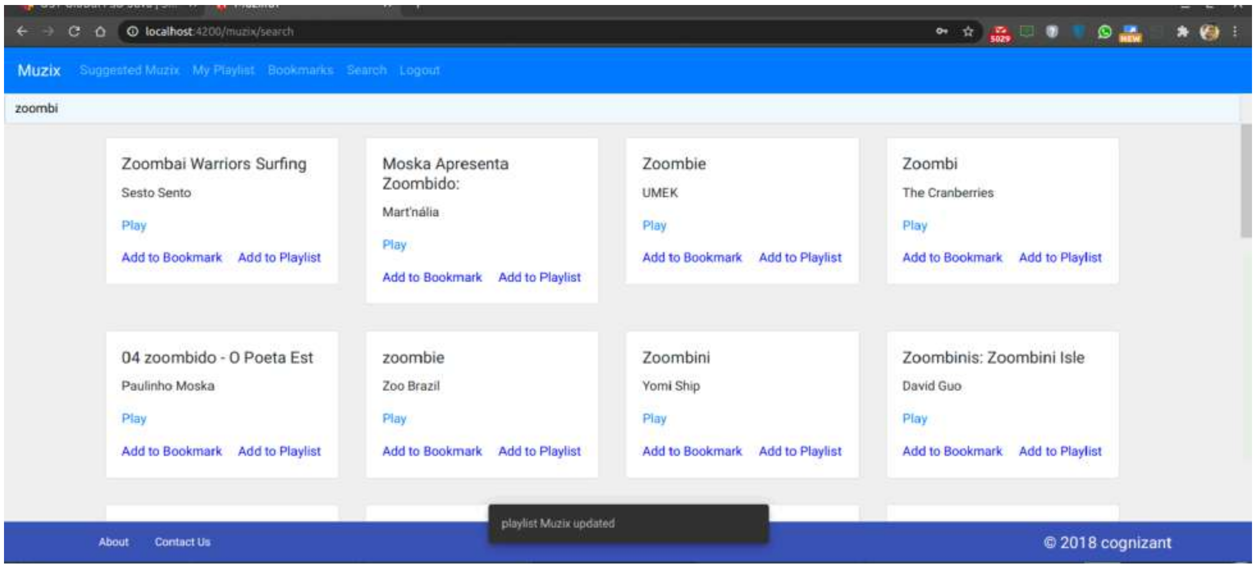
Select Playlist | Add To Playlist

Add Muzix to new Playlist

Enter New Playlist Name: PlayList04 | Create NewPlayList

Fig 5.5 Creating a new playList

The user is available to see the snackbar showing an updated playlist message.



playlist Muzix updated

Fig 5.6 playlist added message using snackbar.

Now the user is searching for music for adding to an existing playlist

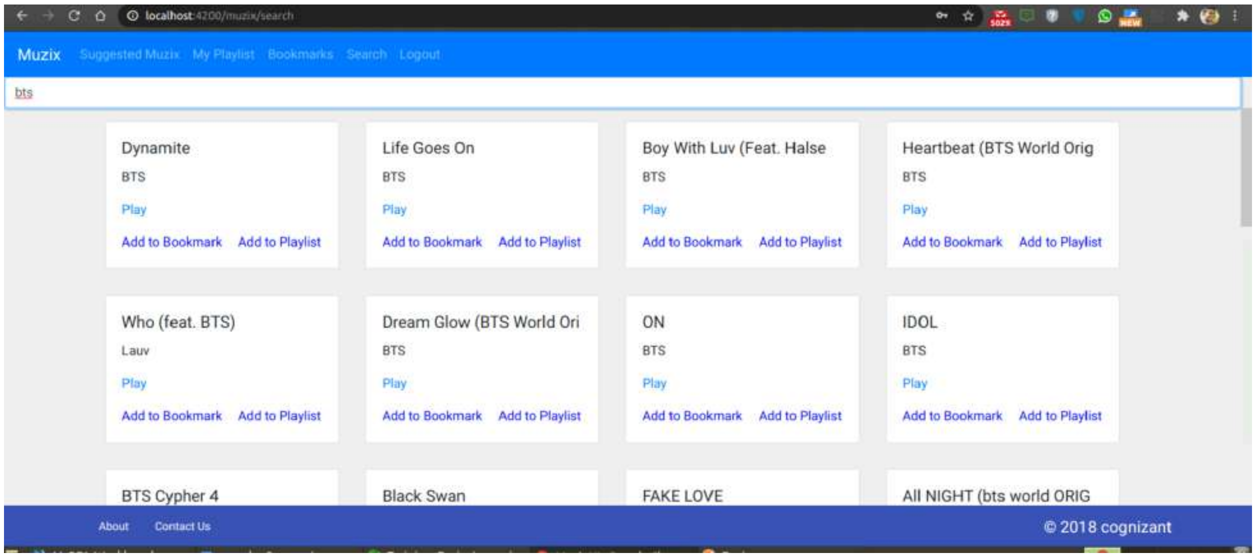


Fig 5.7 Search music.

Then the user is adding music to an existing music.

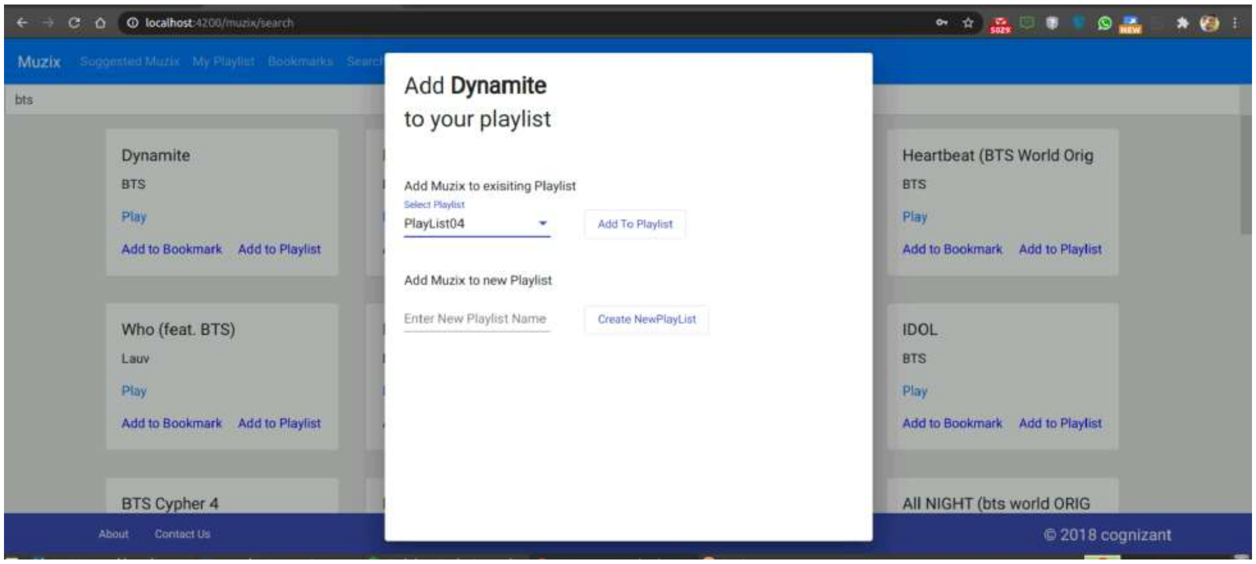
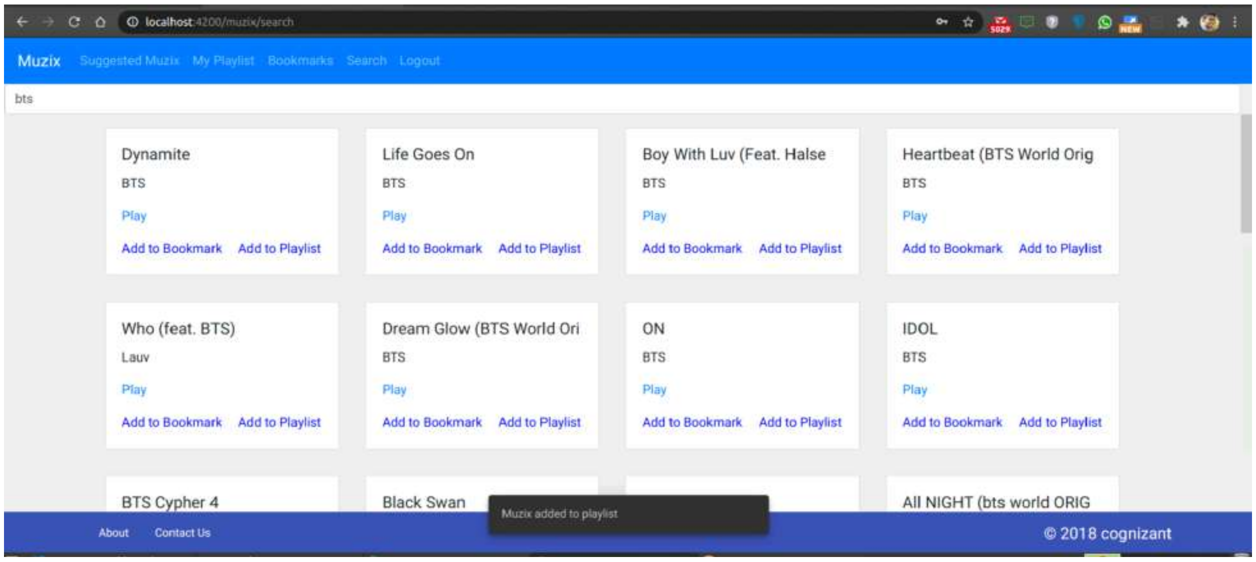


Fig 5.8: Adding music to Playlist

The browser will show the updated message in the snack bar

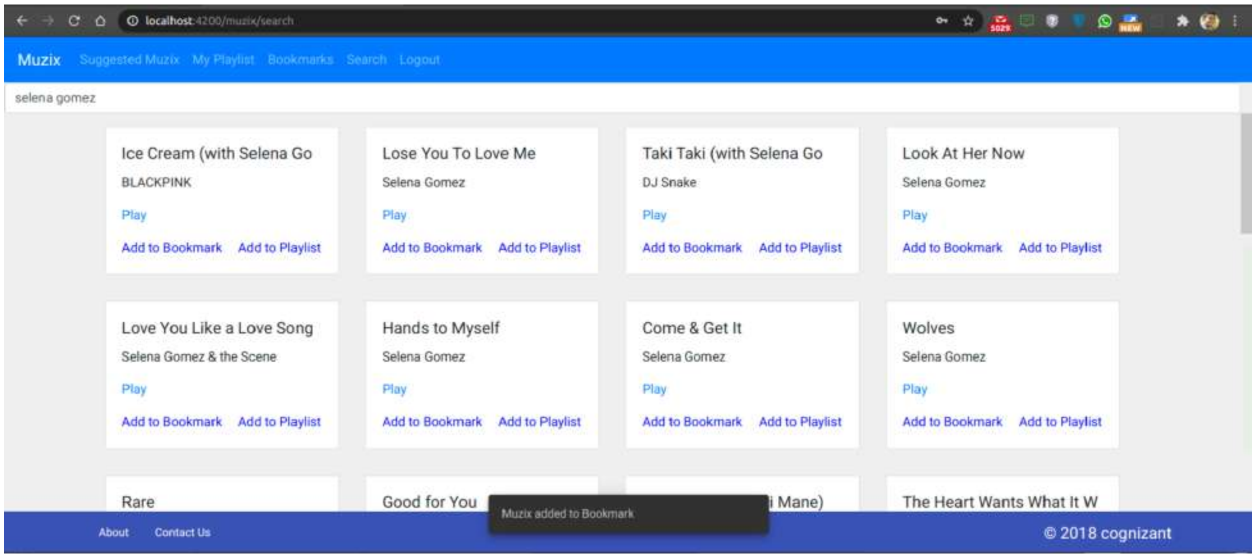


Muzix added to playlist

Fig 5.9 Snack bar message

5.5 ADD TO BOOKMARK :-

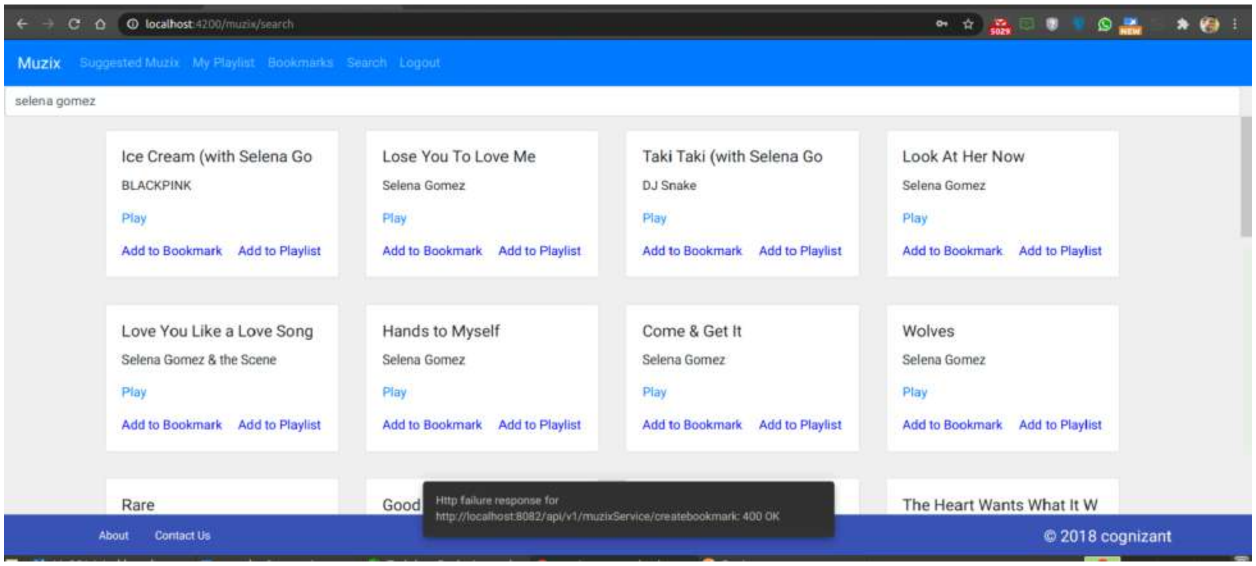
The user is adding music to the bookmark.



Muzix added to Bookmark

Fig: 5.9 :Add to bookmark

And its negative case is also performed, on adding already added music to bookmark, it is showing the error message in the snack bar .



Http failure response for http://localhost:8082/api/v1/muzixService/createbookmark: 400 OK

Fig 5.10 Add to bookmark with the same music.

5.6 REMOVE FROM PLAYLIST :-

The user’s playlist page and removing the first music from it.

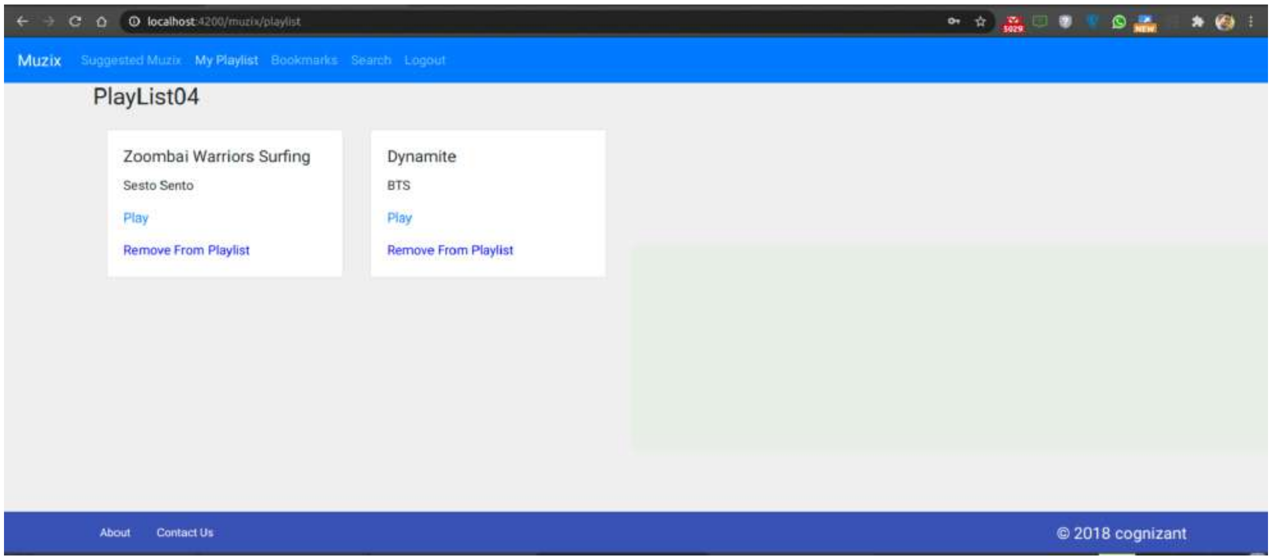
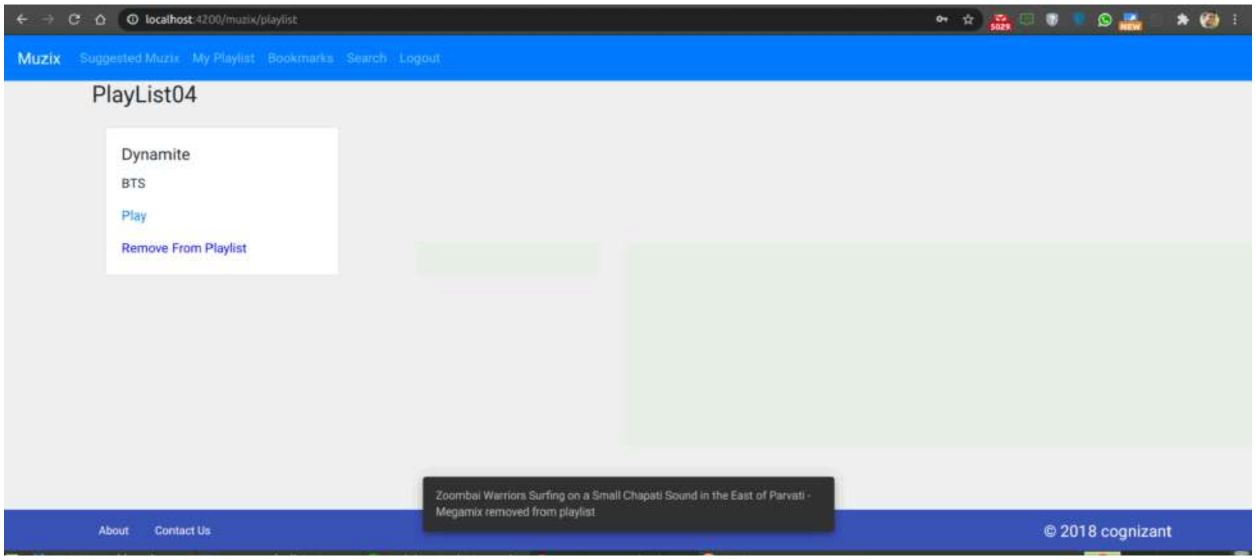


Fig 5.11 Playlist Page



Zoombai Warriors Surfing on a Small Chapati Sound in the East of Parvati - Megamix removed from playlist

Fig 5.12 Removed the first Music

5.7 REMOVE FROM BOOKMARK :-

Now the user is removing the music from bookmark

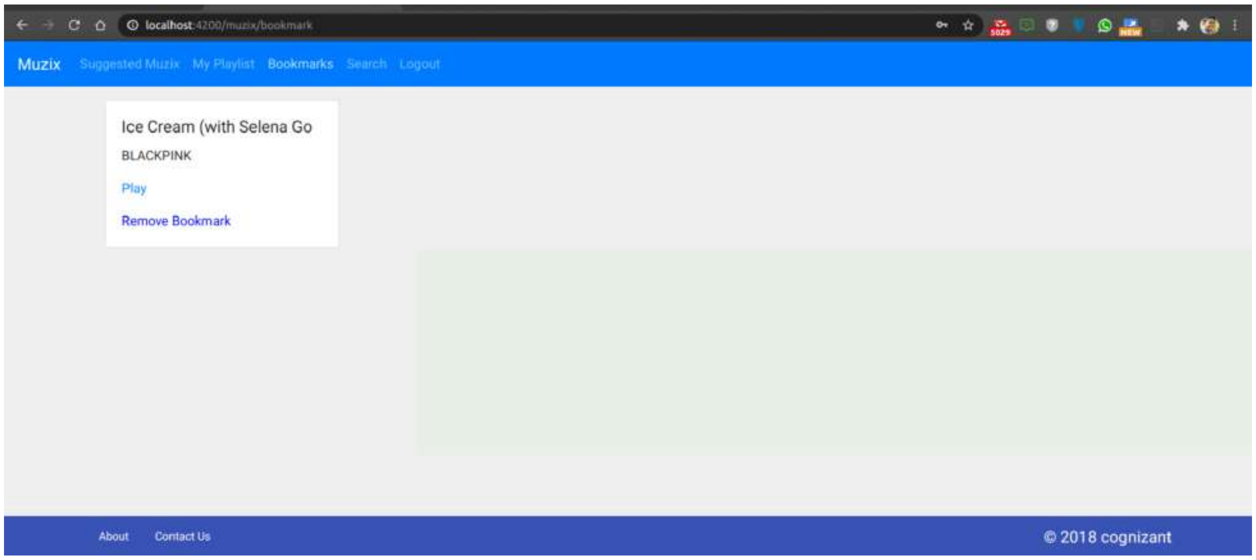
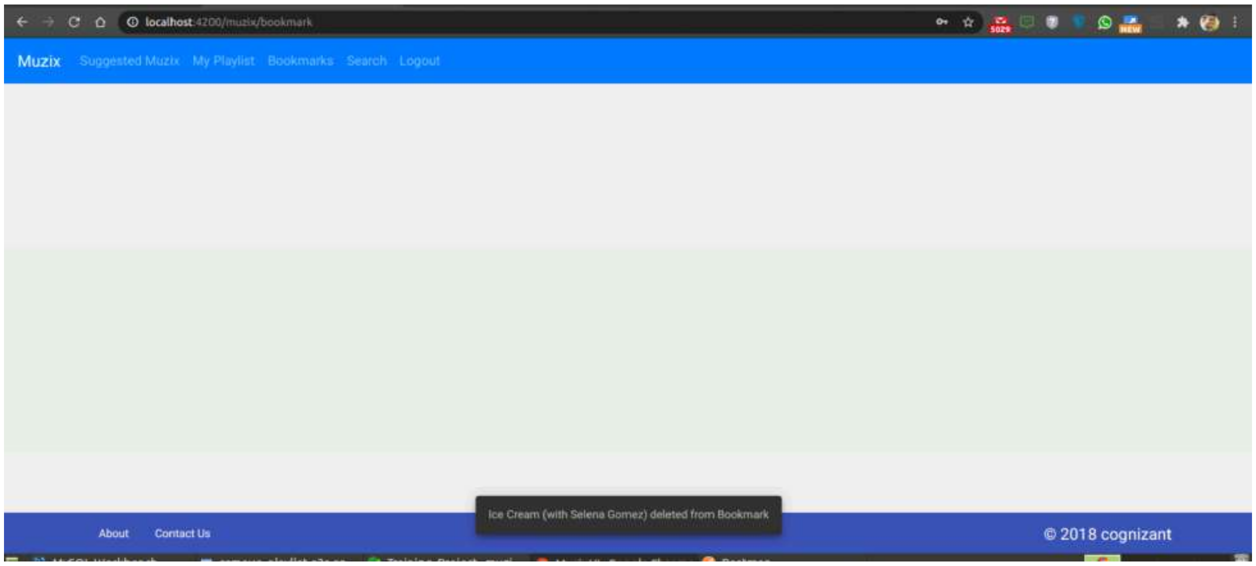


Fig 5.12: Bookmark Page

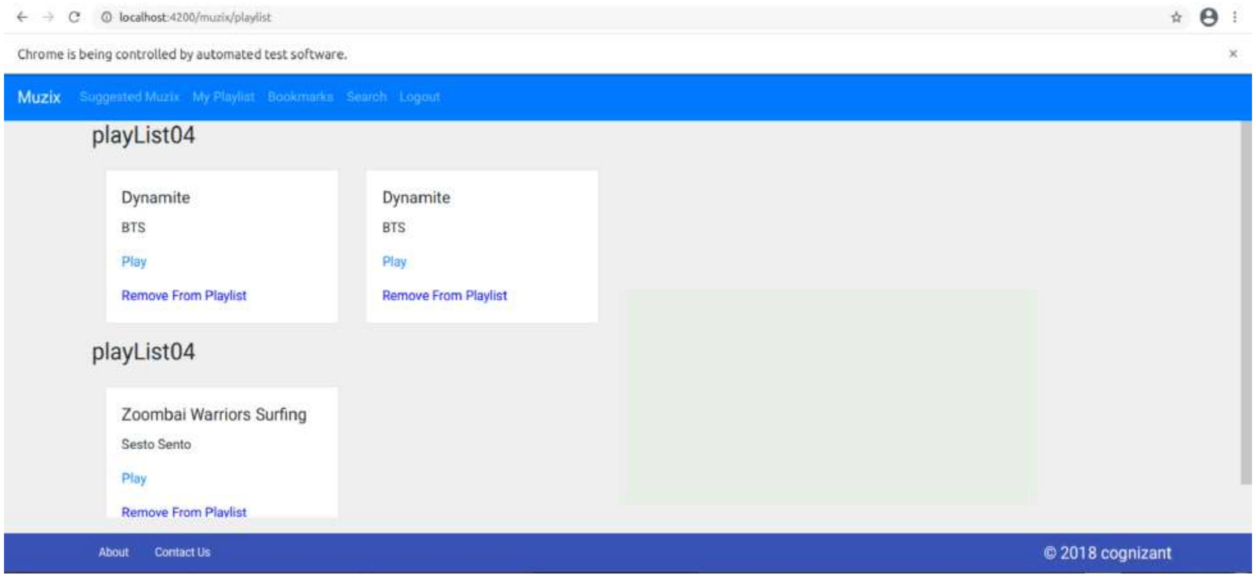


Ice Cream (with Selena Gomez) deleted from Bookmark

Fig 5.13: Removing music

5.8 ADDING SAME PLAYLIST NAME AND SAME MUSIC UNDER A PLAYLIST:-

In this part the user is creating a playlist with the same name and adding the same music under a playlist, it's getting created and added.



localhost:4200/muzix/playlist

Chrome is being controlled by automated test software.

Muzix Suggested Muzix My Playlist Bookmarks Search Logout

playList04

Dynamite
BTS
Play
Remove From Playlist

Dynamite
BTS
Play
Remove From Playlist

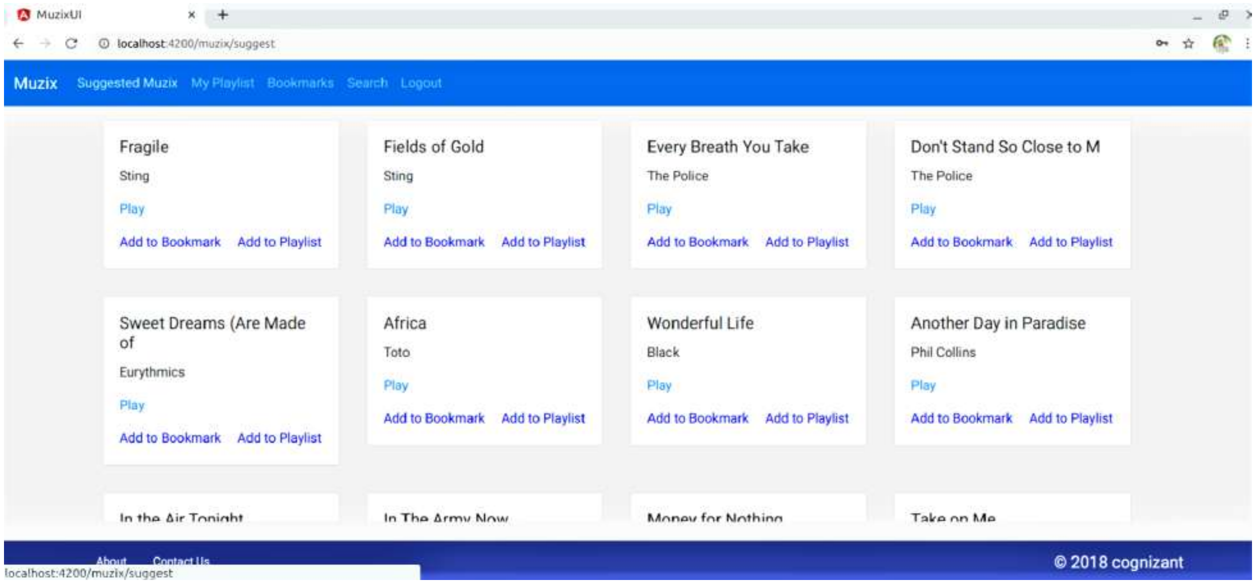
playList04

Zoombai Warriors Surfing
Sesto Sento
Play
Remove From Playlist

About Contact Us © 2018 cognizant

Fig 5.15: Adding Playlist and Music

5.9 SUGGESTION PAGE:-



MuzixUI

localhost:4200/muzix/suggest

Muzix Suggested Muzix My Playlist Bookmarks Search Logout

Fragile
Sting
Play
Add to Bookmark Add to Playlist

Fields of Gold
Sting
Play
Add to Bookmark Add to Playlist

Every Breath You Take
The Police
Play
Add to Bookmark Add to Playlist

Don't Stand So Close to M
The Police
Play
Add to Bookmark Add to Playlist

Sweet Dreams (Are Made of
Eurythmics
Play
Add to Bookmark Add to Playlist

Africa
Toto
Play
Add to Bookmark Add to Playlist

Wonderful Life
Black
Play
Add to Bookmark Add to Playlist

Another Day in Paradise
Phil Collins
Play
Add to Bookmark Add to Playlist

In the Air Tonight

In The Army Now

Money for Nothing

Take on Me

About Contact Us © 2018 cognizant

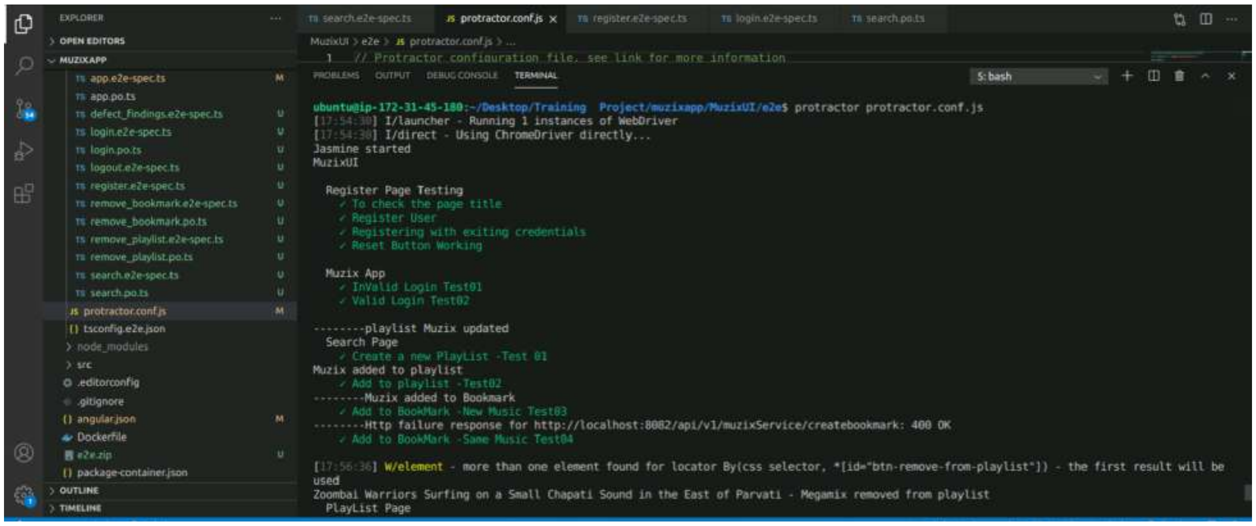
localhost:4200/muzix/suggest

Fig 5.16 : Suggestion Page

5.10 TESTING THE FRONTEND :-

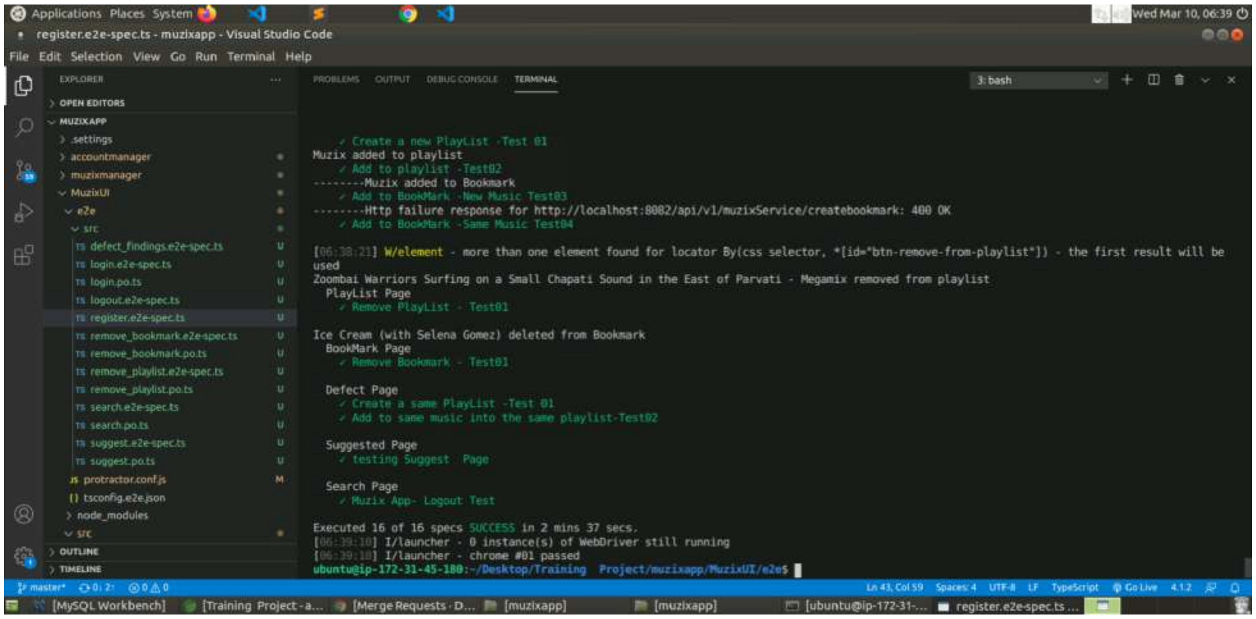
Using Protractor testing in the frontend It is testing the following conditions:-

- To Check Page Title
- Registering a user
- Registering with the same user
- Checking Reset Button
- Invalid Login
- Valid Login
- Create a new Playlist
- Add to Playlist
- Add to Bookmark
- Add same music to bookmark
- Remove from playlist.
- Remove from Bookmark
- Create a playlist with existing playlist name
- Add same music under a playlist
- Suggestion Page
- Logout



```
ubuntu@ip-172-31-45-180:~/Desktop/Training  Project/muzixapp/MuzixUI/e2e$ protractor protractor.conf.js
[17:54:30] I/launcher - Running 1 instances of WebDriver
[17:54:30] I/direct - Using ChromeDriver directly...
Jasmine started
MuzixUI

  Register Page Testing
    ✓ To check the page title
    ✓ Register User
    ✓ Registering with exiting credentials
    ✓ Reset Button Working

  Muzix App
    ✓ InValid Login Test01
    ✓ Valid Login Test02

--------playlist Muzix updated
  Search Page
    ✓ Create a new PlayList -Test 01
Muzix added to playlist
    ✓ Add to playlist -Test02
--------Muzix added to Bookmark
    ✓ Add to BookMark -New Music Test03
--------Http failure response for http://localhost:8082/api/v1/muzixService/createbookmark: 400 OK
    ✓ Add to BookMark -Same Music Test04

[17:56:36] W/element - more than one element found for locator By(css selector, *[id="btn-remove-from-playlist"]) - the first result will be
used
Zoombai Warriors Surfing on a Small Chapati Sound in the East of Parvati - Megamix removed from playlist
  PlayList Page
```

Fig 5.17 : Frontend Test using Protractor part 1

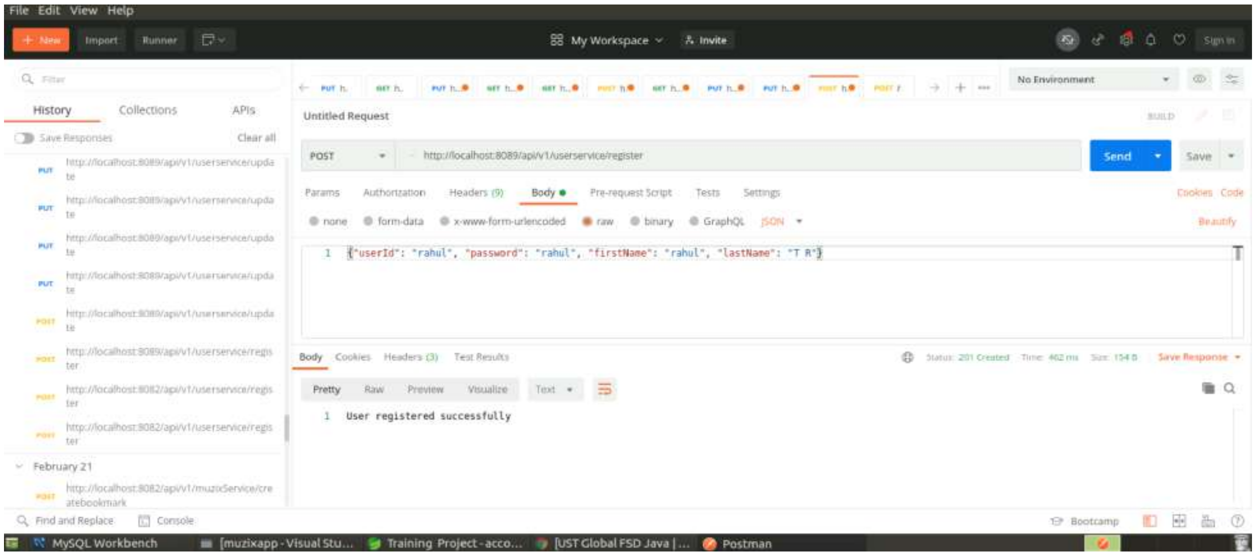


```
    ✓ Create a new PlayList -Test 01
Muzix added to playlist
    ✓ Add to playlist -Test02
--------Muzix added to Bookmark
    ✓ Add to BookMark -New Music Test03
--------Http failure response for http://localhost:8082/api/v1/muzixService/createbookmark: 400 OK
    ✓ Add to BookMark -Same Music Test04

[06:38:21] W/element - more than one element found for locator By(css selector, *[id="btn-remove-from-playlist"]) - the first result will be
used
Zoombai Warriors Surfing on a Small Chapati Sound in the East of Parvati - Megamix removed from playlist
  PlayList Page
    ✓ Remove PlayList - Test01

Ice Cream (with Selena Gomez) deleted from Bookmark
  BookMark Page
    ✓ Remove Bookmark - Test01

  Defect Page
    ✓ Create a same PlayList -Test 01
    ✓ Add to same music into the same playlist-Test02

  Suggested Page
    ✓ testing Suggest  Page

  Search Page
    ✓ Muzix App- Logout Test

Executed 16 of 16 specs SUCCESS in 2 mins 37 secs.
[06:39:10] I/launcher - 0 instance(s) of WebDriver still running
[06:39:10] I/launcher - chrome #01 passed
ubuntu@ip-172-31-45-180:~/Desktop/Training  Project/muzixapp/MuzixUI/e2e$
```

Fig 5.18 : Frontend Test using Protractor part 2

6. ADDING DEVELOPMENT FUNCTIONALITIES

In this, we developed the code for updating the user details and view user details for the backend. By adding controllers and service method functionalities.Then we inserted values and checked it using Postman.



POST http://localhost:8089/api/v1/userservice/register

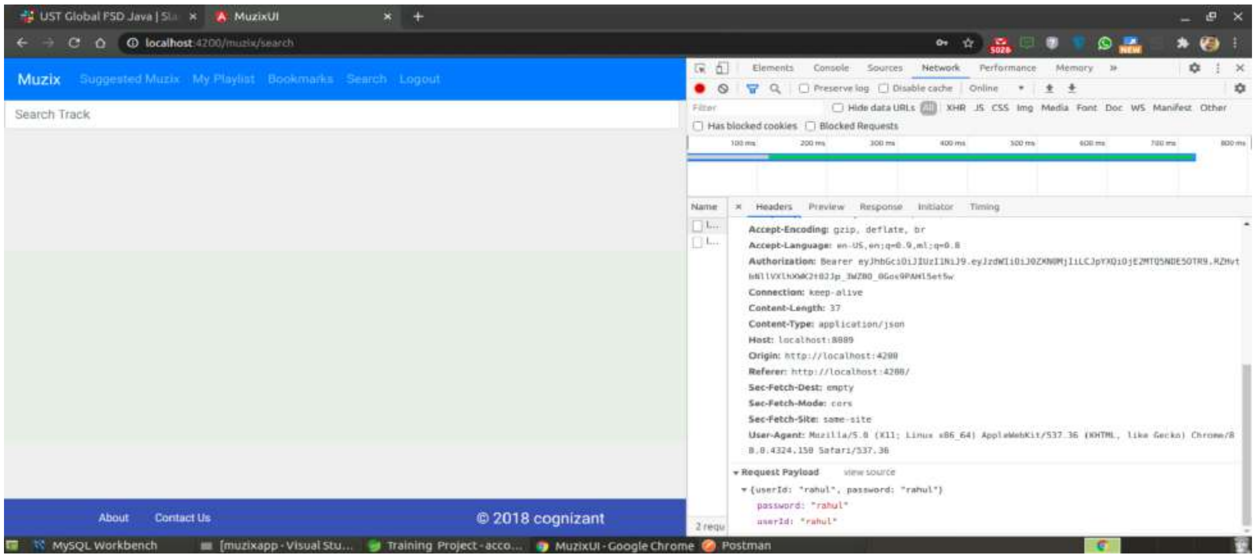
```
{"userId": "rahul", "password": "rahul", "firstName": "rahul", "lastName": "T R"}
```

Status: 201 Created

```
User registered successfully
```

Fig 6.1: Registering an user

After login we will get the token , that can be used to update and view the user details.

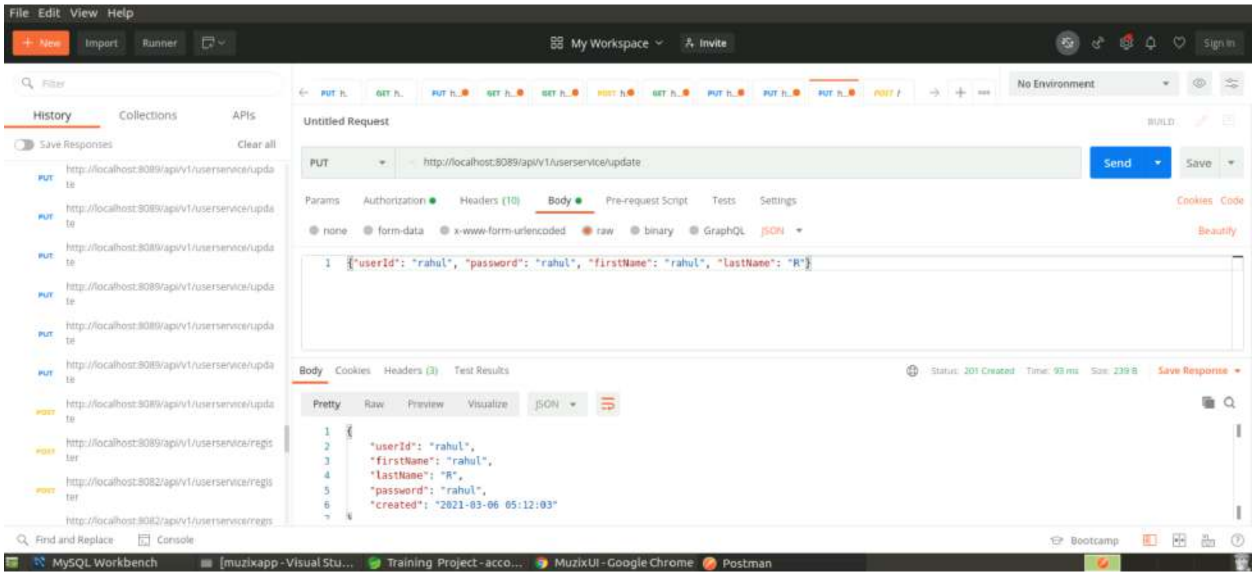


localhost:4200/muzix/search

```
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ml;q=0.8
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0ZXN0MjIiLCJpYXQiOjE2MTQ5NDE5OTR9.RZHvt
bNllVXlhXWK2t0JJp_3WZ80_0Gos9PAHl5et5w
Connection: keep-alive
Content-Length: 37
Content-Type: application/json
Host: localhost:8089
Origin: http://localhost:4200
Referer: http://localhost:4200/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/8
8.0.4324.150 Safari/537.36
▾ Request Payload
▾ {userId: "rahul", password: "rahul"}
    password: "rahul"
    userId: "rahul"
```

Fig 6.2 : Identifying token

Updating the user’s last name using postman by passing token in the authorization bearer part and showing its details in the response



Untitled Request

PUT http://localhost:8089/api/v1/userservice/update

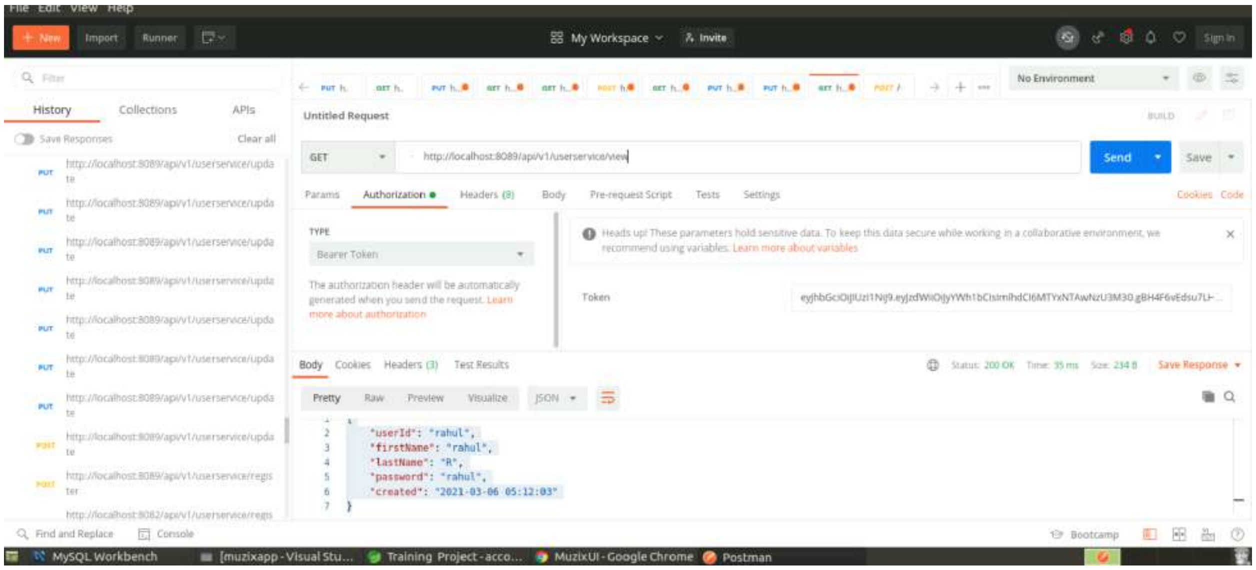
```
{"userId": "rahul", "password": "rahul", "firstName": "rahul", "lastName": "R"}
```

Status: 201 Created Time: 93 ms Size: 239 B

```
{
    "userId": "rahul",
    "firstName": "rahul",
    "lastName": "R",
    "password": "rahul",
    "created": "2021-03-06 05:12:03"
```

Fig 6.3 Update user details

Viewing the user details using postman by token passing



Untitled Request

GET http://localhost:8089/api/v1/userservice/view

Authorization TYPE: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

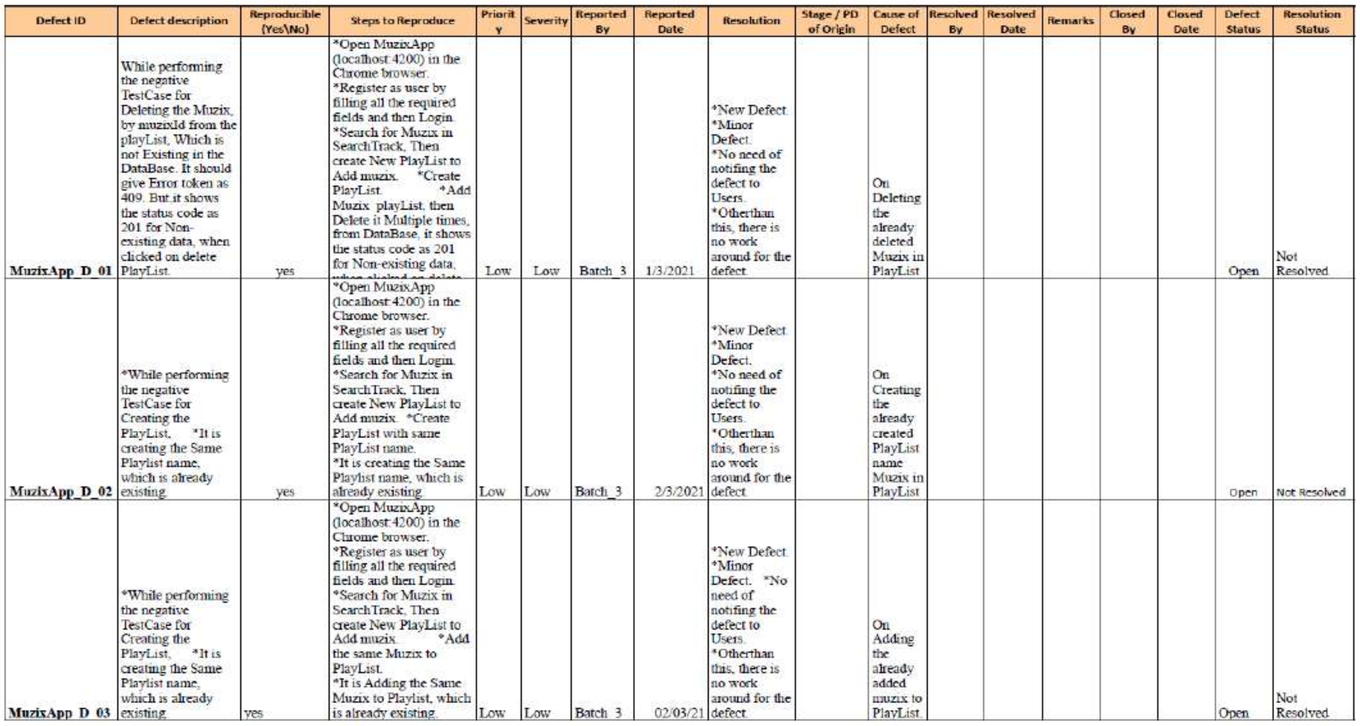
Token: eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJyYWh1bCIsImlhdCI6MTYxNTAwNzU3M30.gBH4F6vEdsu7Lf-...

Status: 200 OK Time: 35 ms Size: 234 B

```
{
    "userId": "rahul",
    "firstName": "rahul",
    "lastName": "R",
    "password": "rahul",
    "created": "2021-03-06 05:12:03"
}
```

Fig 6.4: View user details

7. DEFECT REPORT



Defect ID	Defect description	Reproducible (Yes\No)	Steps to Reproduce	Priority	Severity	Reported By	Reported Date	Resolution	Stage / PD of Origin	Cause of Defect	Resolved By	Resolved Date	Remarks	Closed By	Closed Date	Defect Status	Resolution Status
MuzixApp_D_01	While performing the negative TestCase for Deleting the Muzix, by muzixId from the playList, Which is not Existing in the DataBase. It should give Error token as 409. But it shows the status code as 201 for Non-existing data, when clicked on delete PlayList.	yes	*Open MuzixApp (localhost:4200) in the Chrome browser. *Register as user by filling all the required fields and then Login. *Search for Muzix in SearchTrack, Then create New PlayList to Add muzix. *Create PlayList. *Add Muzix playList, then Delete it Multiple times, from DataBase, it shows the status code as 201 for Non-existing data,	Low	Low	Batch_3	1/3/2021	*New Defect. *Minor Defect. *No need of notifing the defect to Users. *Otherthan this, there is no work around for the defect.		On Deleting the already deleted Muzix in PlayList						Open	Not Resolved
MuzixApp_D_02	*While performing the negative TestCase for Creating the PlayList, *It is creating the Same Playlist name, which is already existing.	yes	*Open MuzixApp (localhost:4200) in the Chrome browser. *Register as user by filling all the required fields and then Login. *Search for Muzix in SearchTrack, Then create New PlayList to Add muzix. *Create PlayList with same PlayList name. *It is creating the Same Playlist name, which is already existing	Low	Low	Batch_3	2/3/2021	*New Defect. *Minor Defect. *No need of notifing the defect to Users. *Otherthan this, there is no work around for the defect.		On Creating the already created PlayList name Muzix in PlayList						Open	Not Resolved
MuzixApp_D_03	*While performing the negative TestCase for Creating the PlayList, *It is creating the Same Playlist name, which is already existing.	yes	*Open MuzixApp (localhost:4200) in the Chrome browser. *Register as user by filling all the required fields and then Login. *Search for Muzix in SearchTrack, Then create New PlayList to Add muzix. *Add the same Muzix to PlayList. *It is Adding the Same Muzix to Playlist, which is already existing.	Low	Low	Batch_3	02/03/21	*New Defect. *Minor Defect. *No need of notifing the defect to Users. *Otherthan this, there is no work around for the defect.		On Adding the already added muzix to PlayList.						Open	Not Resolved

Fig 6.5: Defect Report Template.

8. CONCLUSION

The MuzixApp Project gives us a brief knowledge about testing using Rest-Assured+TestNg and Mockito+J-Unit testing for backend Testing as well as protractor in the front end.We also gone with the development part for updating and viewing the user details.

9. BIBLIOGRAPHY

- The Dependencies in the .Xml file can be added from the https://mvnrepository.com/.
- https://www.techiediaries.com/angular/jasmine-unit-testing-angular-9-tutorial/
- https://github.com/adarshgunnithan/news-app/tree/master/Server/src
- https://trailheadtechnology.com/ui-automation-testing-of-angular-apps-using-protractor-jasmine/
- https://www.tutorialspoint.com/protractor/protractor_elements_api.htm
- https://www.tutorialspoint.com/mockito/index.htm
- https://memorynotfound.com/unit-test-spring-mvc-rest-service-junit-mockito/
-