CHOONZ

By Team 2

Agenda

- Concept
- Sprint Planning
- Technologies Used
- Continuous Integration
- Testing
- Demonstration
- Sprint Review
- Conclusion

Client requirements.

- A fully functional CRUD application for a song website
- Variety of listing needed. Such as playlists and albums
- Sleek card design required
- Test driven approach

Our approach to meet these requirements (BN)

-Backend: Java, Maven, SpringBoot, MySQL.

-Frontend: HTML, JS, CSS, Bootstrap.

-Testing: Junit, Selinum, Jmeter, SonarQube

Sprint Planning (BN)

- We planned our project to be conducted in two Sprint
- The first sprint focused on meeting the MVP specification
- The second sprint aimed at adding extra features that will make the overall product better

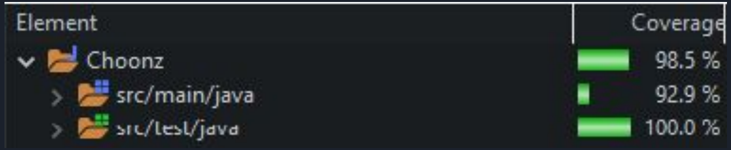
Risks involved (SG)

- Poor team communication - 6
- Illness - 8
- Inability getting application fully working - 10
- Poor planning - 12
- Poor time allocation - 8
- Task management issues - 5

Testing

Back-End Testing (MB)

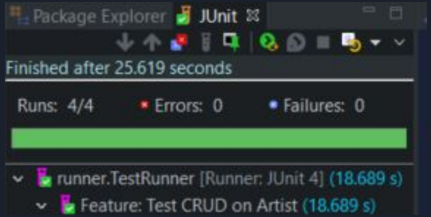
- White Box Testing as had access to the source code
- Used mix of Unit Testing and Integration Testing
- Achieved final src/main test coverage of 92.9%.



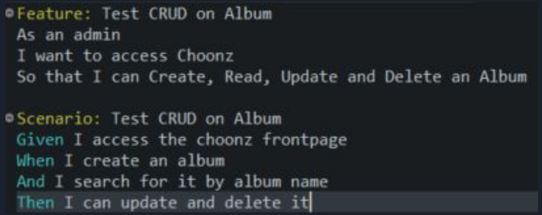
Element	Coverage
Choonz	98.5 %
src/main/java	92.9 %
src/test/java	100.0 %

Front-End Testing (SG)

- Utilised automated testing through Selenium
- Tests were written to examine the CRUD functionality of all entities
- Utilised Behaviour-Driven Development softwares (Cucumber and Gherkin)

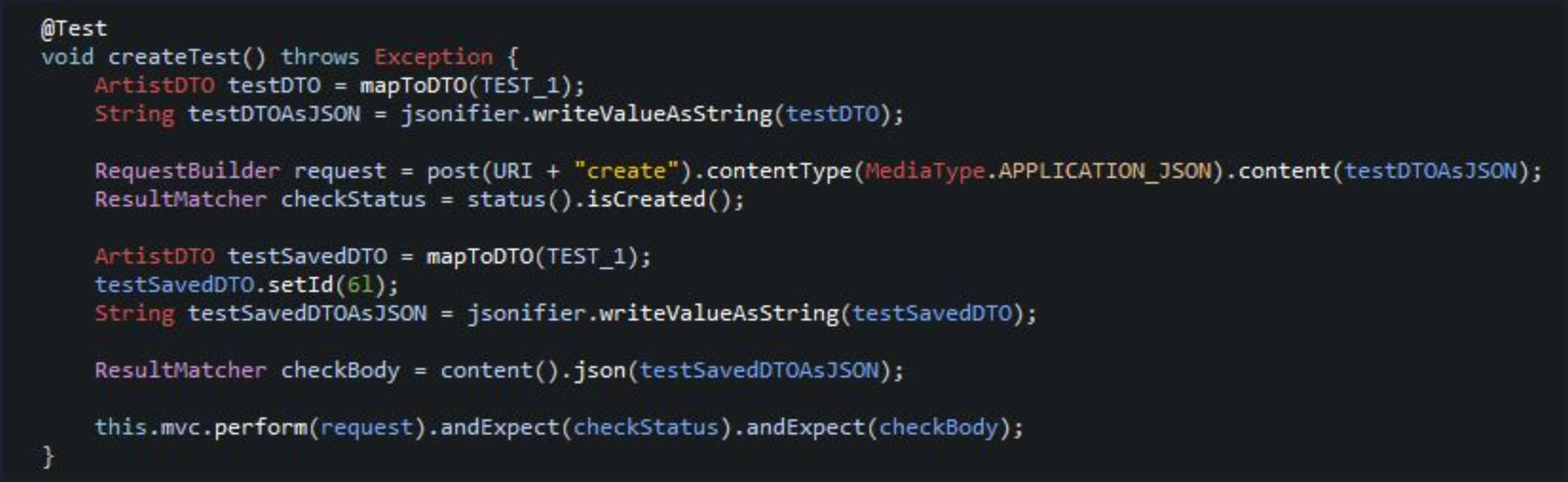


```
Finished after 25.619 seconds
Runs: 4/4    Errors: 0    Failures: 0
runner.TestRunner [Runner: JUnit 4] (18.689 s)
  Feature: Test CRUD on Artist (18.689 s)
```



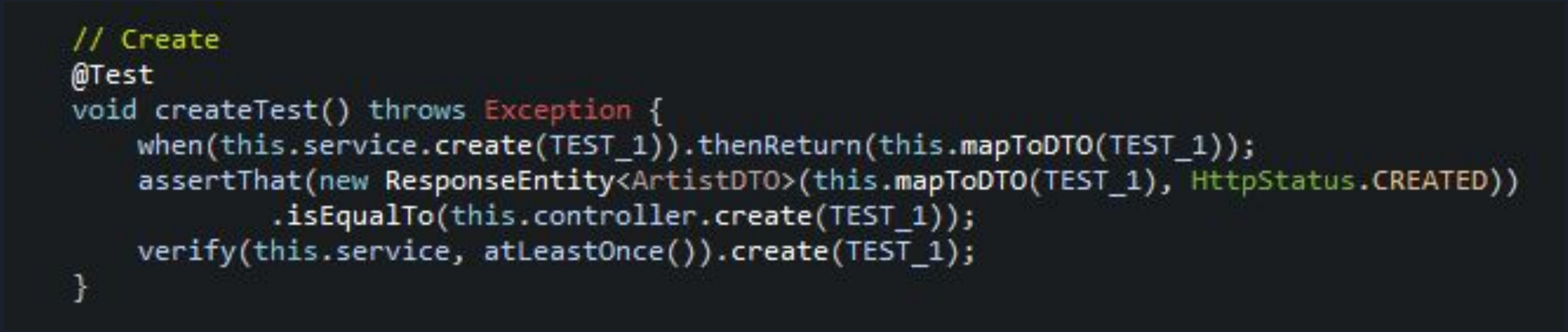
```
Feature: Test CRUD on Album
 As an admin
 I want to access Choonz
 So that I can Create, Read, Update and Delete an Album

Scenario: Test CRUD on Album
 Given I access the choonz frontpage
 When I create an album
 And I search for it by album name
 Then I can update and delete it
```

Automated Testing (MB)



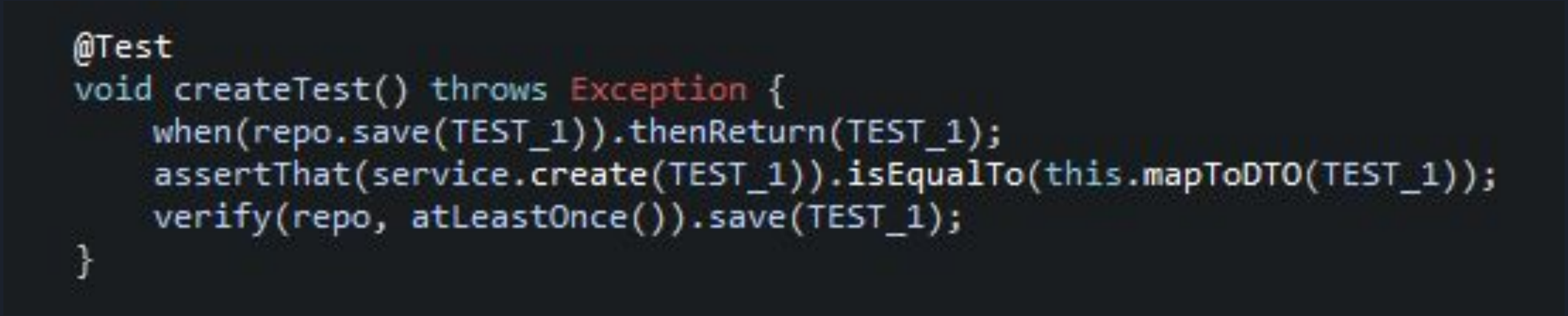
```java
@Test
void createTest() throws Exception {
    ArtistDTO testDTO = mapToDTO(TEST_1);
    String testDTOAsJSON = jsonifier.writeValueAsString(testDTO);

    RequestBuilder request = post(URI + "create").contentType(MediaType.APPLICATION_JSON).content(testDTOAsJSON);
    ResultMatcher checkStatus = status().isCreated();

    ArtistDTO testSavedDTO = mapToDTO(TEST_1);
    testSavedDTO.setId(6l);
    String testSavedDTOAsJSON = jsonifier.writeValueAsString(testSavedDTO);

    ResultMatcher checkBody = content().json(testSavedDTOAsJSON);

    this.mvc.perform(request).andExpect(checkStatus).andExpect(checkBody);
}
```

Automated Testing (MB)



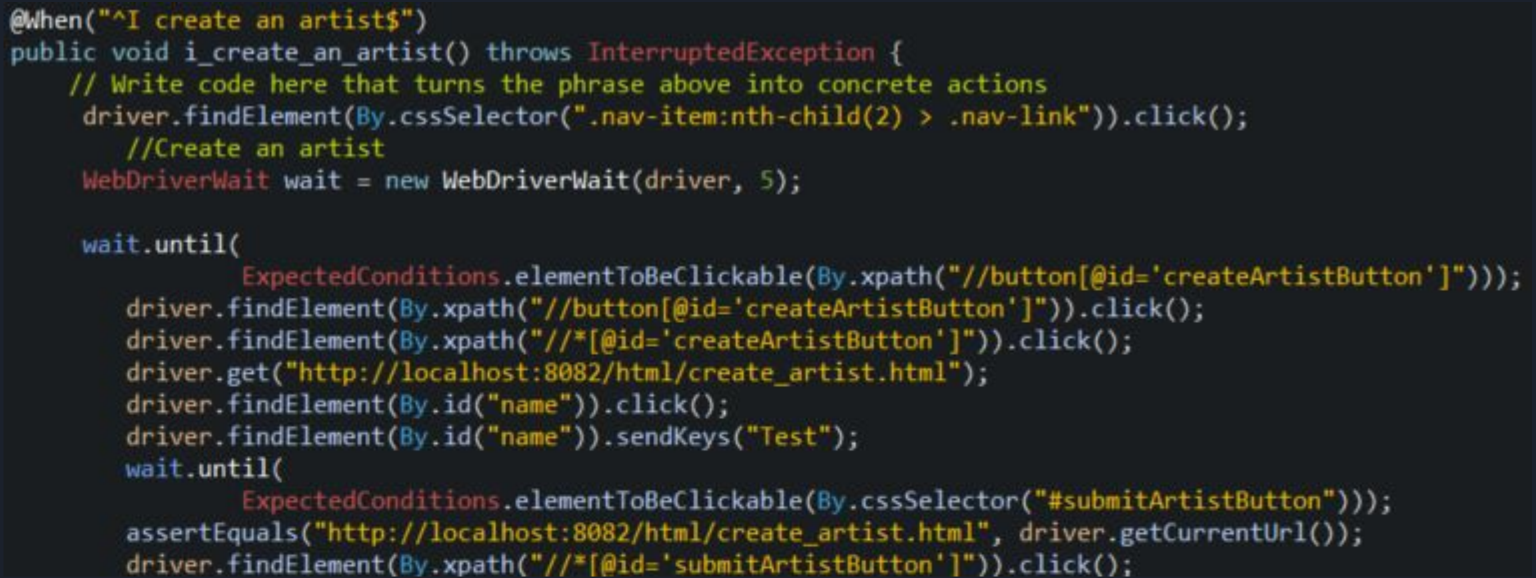
```
// Create
@Test
void createTest() throws Exception {
    when(this.service.create(TEST_1)).thenReturn(this.mapToDTO(TEST_1));
    assertThat(new ResponseEntity<ArtistDTO>(this.mapToDTO(TEST_1), HttpStatus.CREATED))
            .isEqualTo(this.controller.create(TEST_1));
    verify(this.service, atLeastOnce()).create(TEST_1);
}
```

Automated Testing (MB)



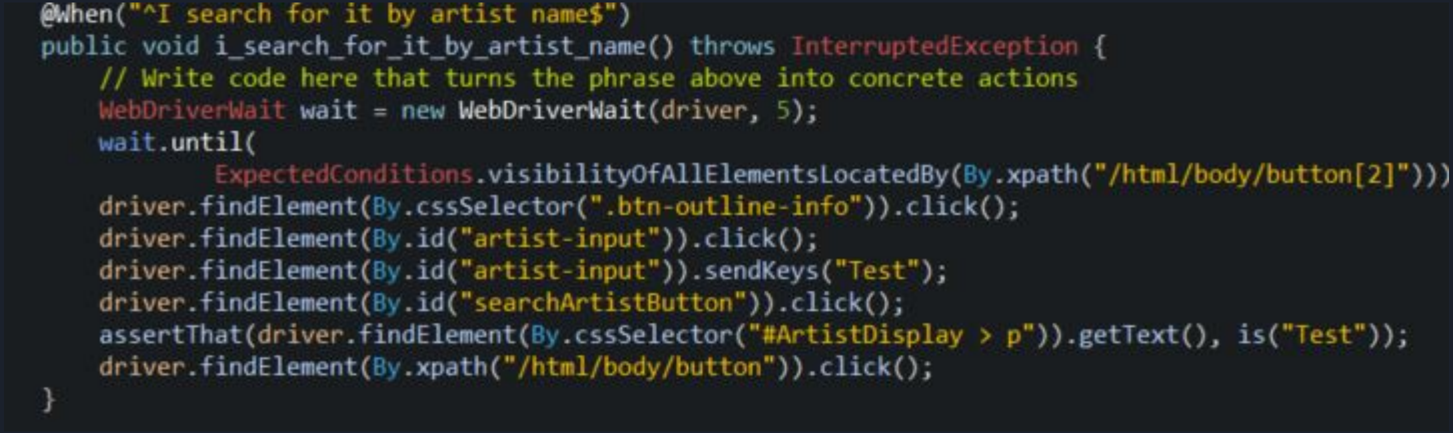
```
@Test
void createTest() throws Exception {
    when(repo.save(TEST_1)).thenReturn(TEST_1);
    assertThat(service.create(TEST_1)).isEqualTo(this.mapToDTO(TEST_1));
    verify(repo, atLeastOnce()).save(TEST_1);
}
```

Automated Testing (SG)



```
@When("^I create an artist$")
public void i_create_an_artist() throws InterruptedException {
    // Write code here that turns the phrase above into concrete actions
     driver.findElement(By.cssSelector(".nav-item:nth-child(2) > .nav-link")).click();
        //Create an artist
     WebDriverWait wait = new WebDriverWait(driver, 5);

     wait.until(
                ExpectedConditions.elementToBeClickable(By.xpath("//button[@id='createArtistButton']")));
        driver.findElement(By.xpath("//button[@id='createArtistButton']")).click();
        driver.findElement(By.xpath("//*[@id='createArtistButton']")).click();
        driver.get("http://localhost:8082/html/create_artist.html");
        driver.findElement(By.id("name")).click();
        driver.findElement(By.id("name")).sendKeys("Test");
        wait.until(
                ExpectedConditions.elementToBeClickable(By.cssSelector("#submitArtistButton")));
        assertEquals("http://localhost:8082/html/create_artist.html", driver.getCurrentUrl());
        driver.findElement(By.xpath("//*[@id='submitArtistButton']")).click();
```

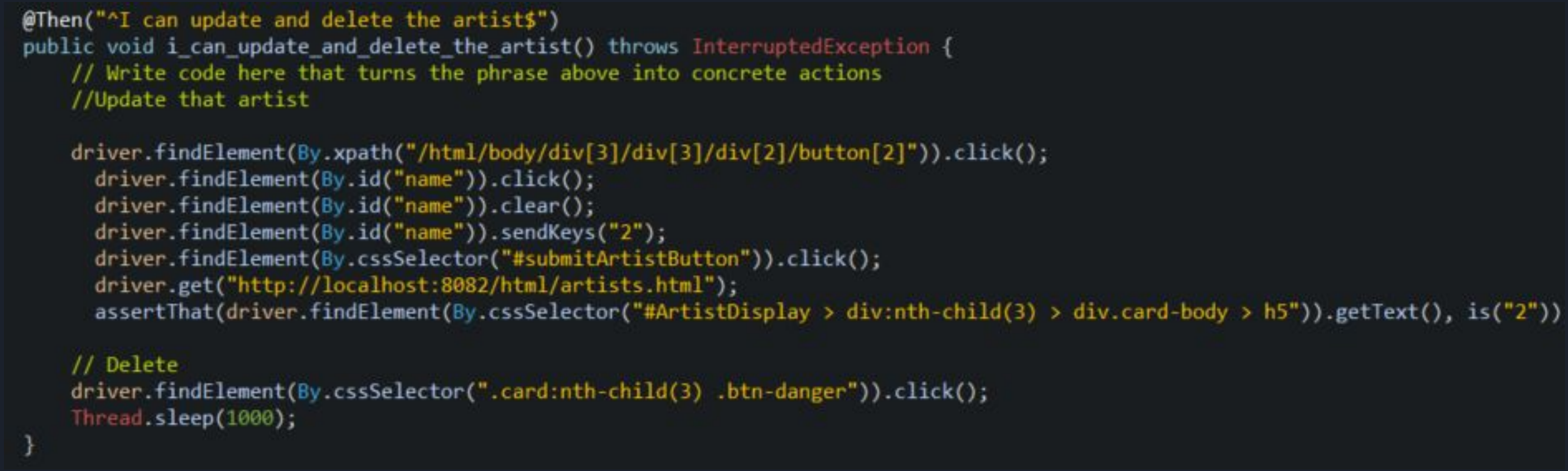
Automated Testing (SG)

- 

```
@When("^I search for it by artist name$")
public void i_search_for_it_by_artist_name() throws InterruptedException {
    // Write code here that turns the phrase above into concrete actions
    WebDriverWait wait = new WebDriverWait(driver, 5);
    wait.until(
            ExpectedConditions.visibilityOfAllElementsLocatedBy(By.xpath("/html/body/button[2]")))
    driver.findElement(By.cssSelector(".btn-outline-info")).click();
    driver.findElement(By.id("artist-input")).click();
    driver.findElement(By.id("artist-input")).sendKeys("Test");
    driver.findElement(By.id("searchArtistButton")).click();
    assertThat(driver.findElement(By.cssSelector("#ArtistDisplay > p")).getText(), is("Test"));
    driver.findElement(By.xpath("/html/body/button")).click();
}
```

Automated Testing (SG)

-



```
@Then("^I can update and delete the artist$")
public void i_can_update_and_delete_the_artist() throws InterruptedException {
    // Write code here that turns the phrase above into concrete actions
    //Update that artist

    driver.findElement(By.xpath("/html/body/div[3]/div[3]/div[2]/button[2]")).click();
      driver.findElement(By.id("name")).click();
      driver.findElement(By.id("name")).clear();
      driver.findElement(By.id("name")).sendKeys("2");
      driver.findElement(By.cssSelector("#submitArtistButton")).click();
      driver.get("http://localhost:8082/html/artists.html");
      assertThat(driver.findElement(By.cssSelector("#ArtistDisplay > div:nth-child(3) > div.card-body > h5")).getText(), is("2"))

    // Delete
    driver.findElement(By.cssSelector(".card:nth-child(3) .btn-danger")).click();
    Thread.sleep(1000);
}
```

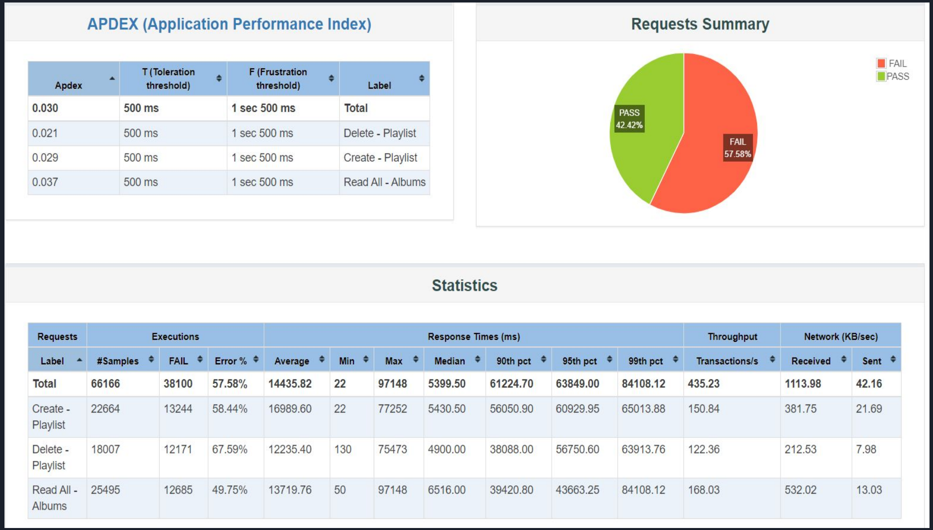
Performance Testing (SM)

- Load Testing
- Spike Testing
- Soak Testing
- Stress Testing

Performance Testing: Load Testing

- Load Testing All CRUD functionality
- Load Testing user Journey.

Throughput	435.23
Latency	5000 - 18000 ms



APDEX (Application Performance Index)

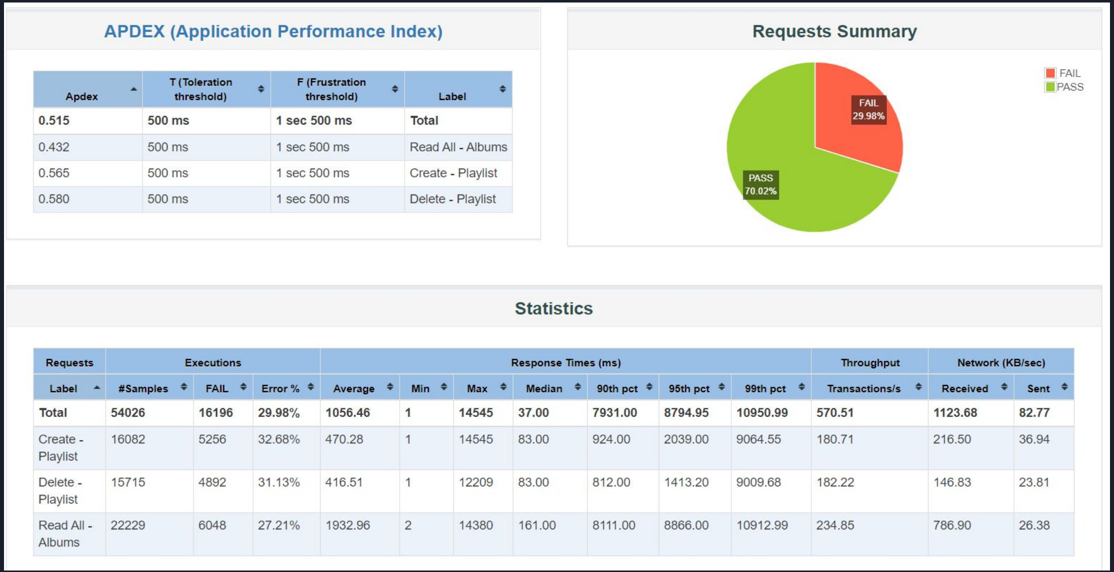
Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.030	**500 ms**	**1 sec 500 ms**	**Total**
0.021	500 ms	1 sec 500 ms	Delete - Playlist
0.029	500 ms	1 sec 500 ms	Create - Playlist
0.037	500 ms	1 sec 500 ms	Read All - Albums

Requests Summary

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**66166**	**38100**	**57.58%**	**14435.82**	**22**	**97148**	**5399.50**	**61224.70**	**63849.00**	**84108.12**	**435.23**	**1113.98**	**42.16**
Create - Playlist	22664	13244	58.44%	16989.60	22	77252	5430.50	56050.90	60929.95	65013.88	150.84	381.75	21.69
Delete - Playlist	18007	12171	67.59%	12235.40	130	75473	4900.00	38088.00	56750.60	63913.76	122.36	212.53	7.98
Read All - Albums	25495	12685	49.75%	13719.76	50	97148	6516.00	39420.80	43663.25	84108.12	168.03	532.02	13.03

Performance Testing: Spike Test



APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.515	**500 ms**	**1 sec 500 ms**	**Total**
0.432	500 ms	1 sec 500 ms	Read All - Albums
0.565	500 ms	1 sec 500 ms	Create - Playlist
0.580	500 ms	1 sec 500 ms	Delete - Playlist

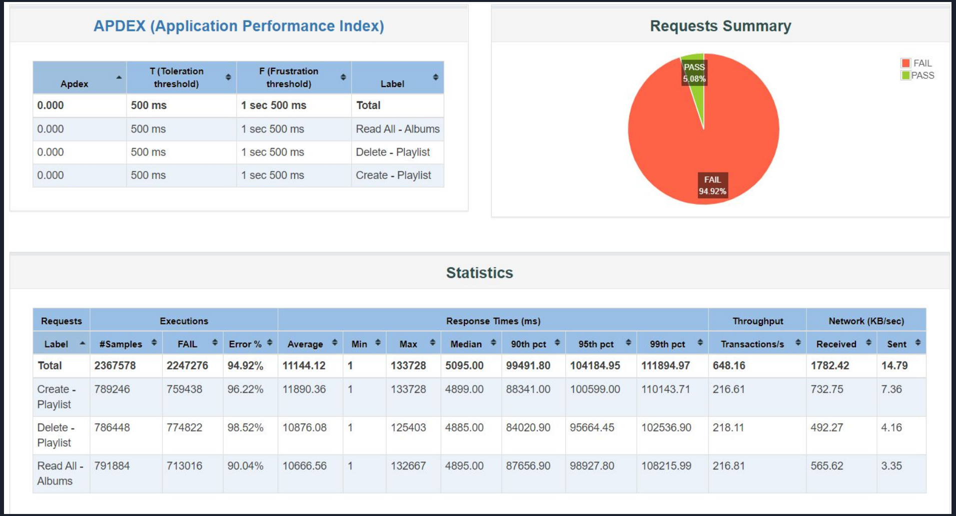
Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**54026**	**16196**	**29.98%**	**1056.46**	**1**	**14545**	**37.00**	**7931.00**	**8794.95**	**10950.99**	**570.51**	**1123.68**	**82.77**
Create - Playlist	16082	5256	32.68%	470.28	1	14545	83.00	924.00	2039.00	9064.55	180.71	216.50	36.94
Delete - Playlist	15715	4892	31.13%	416.51	1	12209	83.00	812.00	1413.20	9009.68	182.22	146.83	23.81
Read All - Albums	22229	6048	27.21%	1932.96	2	14380	161.00	8111.00	8866.00	10912.99	234.85	786.90	26.38

Throughput	570.51
Latency	85 - 2396 ms

Performance Testing: Soak Test

Throughput	64816
Latency	5000-25000



APDEX (Application Performance Index)

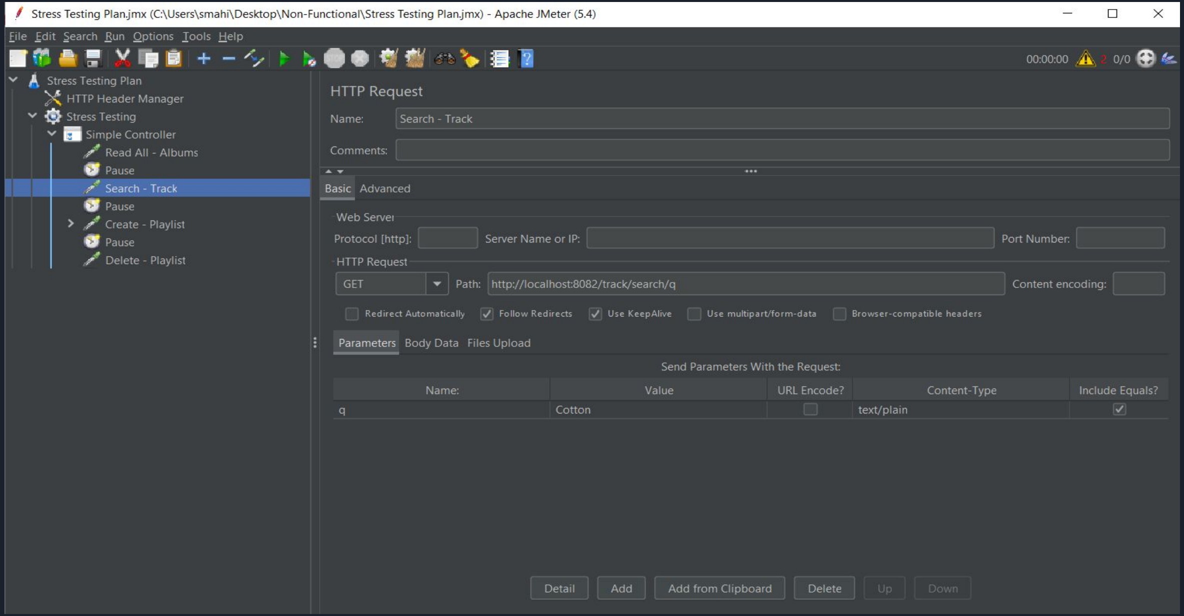
Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.000	**500 ms**	**1 sec 500 ms**	**Total**
0.000	500 ms	1 sec 500 ms	Read All - Albums
0.000	500 ms	1 sec 500 ms	Delete - Playlist
0.000	500 ms	1 sec 500 ms	Create - Playlist

Requests Summary

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**2367578**	**2247276**	**94.92%**	**11144.12**	**1**	**133728**	**5095.00**	**99491.80**	**104184.95**	**111894.97**	**648.16**	**1782.42**	**14.79**
Create - Playlist	789246	759438	96.22%	11890.36	1	133728	4899.00	88341.00	100599.00	110143.71	216.61	732.75	7.36
Delete - Playlist	786448	774822	98.52%	10876.08	1	125403	4885.00	84020.90	95664.45	102536.90	218.11	492.27	4.16
Read All - Albums	791884	713016	90.04%	10666.56	1	132667	4895.00	87656.90	98927.80	108215.99	216.81	565.62	3.35

Performance Testing: Stress Testing



Stress Testing Plan.jmx (C:\Users\smahi\Desktop\Non-Functional\Stress Testing Plan.jmx) - Apache JMeter (5.4)

HTTP Request

Name: Search - Track

Comments:

Web Server

Protocol [http]: Server Name or IP: Port Number:

HTTP Request

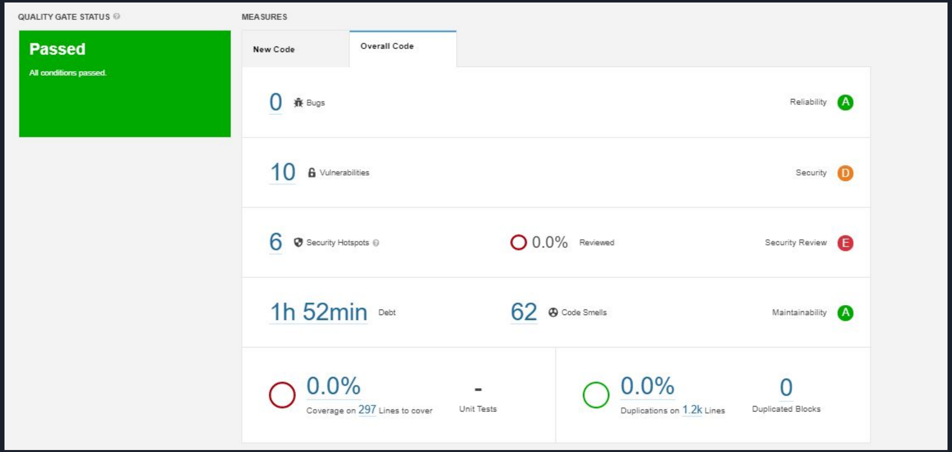
GET Path: http://localhost:8082/track/search/q Content encoding:

☐ Redirect Automatically ✓ Follow Redirects ✓ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
q	Cotton	☐	text/plain	✓

SonarQube and Code Quality (MB)


QUALITY GATE STATUS

Passed
All conditions passed.

MEASURES

New Code | Overall Code

0 Bugs — Reliability A

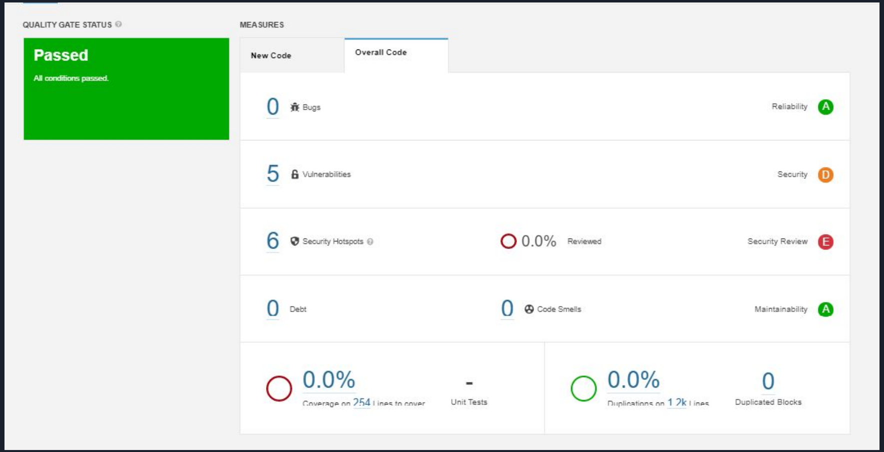
10 Vulnerabilities — Security D

6 Security Hotspots — 0.0% Reviewed — Security Review E

1h 52min Debt — 62 Code Smells — Maintainability A

0.0% Coverage on 297 Lines to cover — - Unit Tests

0.0% Duplications on 1.2k Lines — 0 Duplicated Blocks


QUALITY GATE STATUS

Passed
All conditions passed.

MEASURES

New Code | Overall Code

0 Bugs — Reliability A

5 Vulnerabilities — Security D

6 Security Hotspots — 0.0% Reviewed — Security Review E

0 Debt — 0 Code Smells — Maintainability A

0.0% Coverage on 254 Lines to cover — - Unit Tests

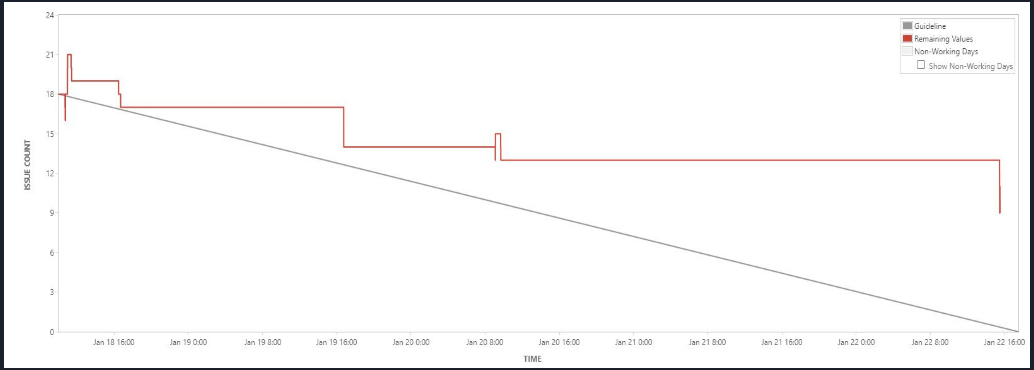
0.0% Duplications on 1.2k Lines — 0 Duplicated Blocks

Demonstration

Sprint Reviews

Sprint 1 (MB)
(18/01/2021 - 22/01/2021)

- In total this Sprint had an issue count of 27
- We managed to end the sprint with a final issue count of 9
- Scope changes occured to add missing functionality



Sprint 2 (MB)
(25/01/2021 - 29/01/2021)

- After meeting with PO we made some changes to design of the website
- We were left with 27 Issues
- By Thursday night we were left with x issues
- Heavy Focus on Security using SpringSecurity but ultimately scrapped due to complexity

Conclusions

- Developed a fully automated functional testing suite for both the front and back end
- Developed a website with in-memory database with full CRUD functionality
- Looking ahead we would implement better security measures (Spring Security was our first choice for securing API by roles and setting up users).
- Increase the application's ability to handle high volume of users.
- Host performance tests on GCP for greater computational power.

Any Questions?