



# Choonz

...

Team 1 Presentation

# Project Management - Jira & Risk Assessment

## CMQ Project Risk Assessment Version 1.0

### Risk Category Key:

Probability (how likely is risk to occur)

1 = Unlikely 2 = Possible 3 = Probable

Severity (what is the impact of the risk if it occurs)

1 = Minor 2 = Moderate 3 = Major

Overall risk (probability x severity)

1-2 = Low 3-4 = Moderate 5-7 = High 8-9 = Very high

### Initial Risk Assessment:

ID	Risk Description	Cause	Outcome	Probability (1-3)	Severity (1-3)	Overall risk (1-9)	Mitigating actions
1	Lack of time	Tight timescales / time management	Incomplete project	2	3	6	Focus on MVP. Plan daily / sprint tasks and set goals. Review progress. Ensure non-code deliverables are uploaded after sufficient progress even if they may be amended later.
2	Lack of experience	New to some of the tools and methodologies. Not fully familiar with other areas	Incomplete project due to impact on time taken to complete some aspects of project	2	3	6	Additional practice in other less familiar areas. Ask for trainer help if stuck. Team members to take tasks in areas of expertise where possible.
3	Utilities failure	Power and/or internet unavailable	No shared access & minimal project progress during outage	2	2	4	Ensure laptop and phone charge maintained and local copy of resources for each team member where possible.

### Final Risk Assessment:

ID	Risk Description	Cause	Outcome	Probability (1-3)	Severity (1-3)	Overall risk (1-9)	Mitigating actions
1	Lack of time	Tight timescales / time management	Incomplete project	2	3	6	Focus on MVP. Plan daily / sprint tasks and set goals. Review progress. Ensure non-code deliverables are uploaded after sufficient progress even if they may be amended later.
2	Lack of experience	New to some of the tools and methodologies. Not fully familiar with other areas	Incomplete project due to impact on time taken to complete some aspects of project	2	3	6	Additional practice in other less familiar areas. Ask for trainer help if stuck. Team members to take tasks in areas of expertise where possible.
3	Utilities failure	Power and/or internet unavailable	No shared access & minimal project progress during outage	2	2	4	Ensure laptop and phone charge maintained and local copy of resources for each team member where possible.
4	External disruption to resource availability	COVID-19 / home environment	Possible reduced internet speeds (more remote workers) / illness	3	2	6	Link to the router via cable to minimize issues, turn off camera if internet slow, use data/hotspot if required and available. Regular hand washing etc. to reduce risk of illness. Asking other team members for support
5	Lack of progress	Overtired / overworking / overcomplicating project	Stress / difficulty completing project	1	2	2	Schedule breaks and spend some of each outside weather permitting, stay hydrated.
6	Unstable / non-working system	Insufficient testing	Substandard project	2	3	6	Allocate adequate testing time, save tested working copy before working on should/could-have features.
7	Team Management	Sub-par communication and conflicting views or understanding	Incomplete or sub-par project. Friction could break the group	2	3	6	Be respectful of other members without talking over each other. Disagreements are settled as and when they occur.

https://jiraqa.atlassian.net/secure/RapidBoard.jspa?rapidView=4&projectKey=CMQ&view=planning&selectedIssue=CMQ-69&issueLimit=100

Jira Software Your work Projects Filters Dashboards People Apps Create

Choonz-Music-QA Classic software project

Projects / Choonz-Music-QA / CMQ board

Backlog

CMQ board Board

Roadmap Projects / Choonz-Music-QA / CMQ board

Backlog

Only My Issues Recently Updated

VERSIONS

EMCS

As a user, I can sign up to the system along side other users, so that I can have access

As a user I want to see a readme.md so I can find out more about Choonz and how to set it up

As a user I can browse the system without logging in but will not be able to CRUD so that I can see what is available on the system

As a user, I can CRUD albums, artists, tracks and genres.

As a user, I can CRUD as many playlists as I want on my home screen, so that I can manage my playlists.

As a client, I should see an 80% test coverage for backend testing, so that the industry-standard is met

As a user, I should see consistent styling throughout the entire site

As a team, we must utilise the feature-branch model within a version control repository

As a developer, I should use a static analysis tool, so that code smells, bugs, and vulnerabilities are reduced.

As a client, the website latency should always be less than 2 seconds with 10,000 concurrent users.

As a developer, the application should be: spike, load, stress, and soak tested so that performance is managed.

As a user, I should have quick server interactions (response times should be <10 milliseconds).

As a client, I must see unit, integration, user-acceptance, functional and non-functional testing

Fixing back-end functionality

As a user, the throughput rate should be >300/s.

As a developer, RAM allocation should be minimal, with few memory leaks so the user experience is satisfactory regardless of other applications running at the same time.

Set up separate sql environment to aid testing (allow running h2 or local sql)

As a client I should see a documentation folder, so that I can see the development of the system

Backlog 12 issues

As a user, it would be nice to be able to search for a specific track, album, or artist.

As a user, I should see each album on it's own page so that I can view album details.

General CMQ-1

Project Requirements CMQ-43

General CMQ-40

User Home Screen CMQ-10

Playlist CMQ-12

Project Requirements CMQ-35

General CMQ-3

Project Requirements CMQ-42

Project Requirements CMQ-38

Non-functional testing CMQ-29

Non-functional testing CMQ-31

Non-functional testing CMQ-19

Project Requirements CMQ-36

CMQ-46

Non-functional testing CMQ-32

Non-functional testing CMQ-30

CMQ-69

Project Requirements CMQ-70

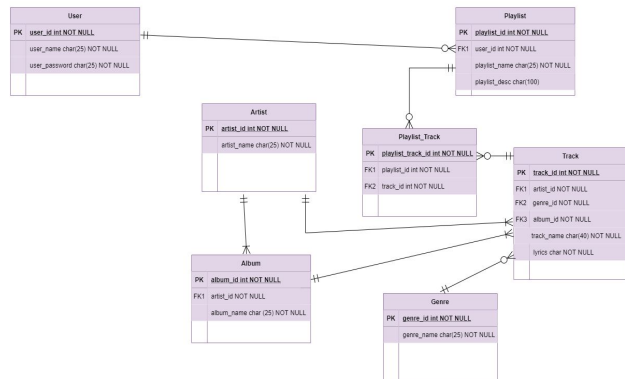
Create sprint

General Quickets

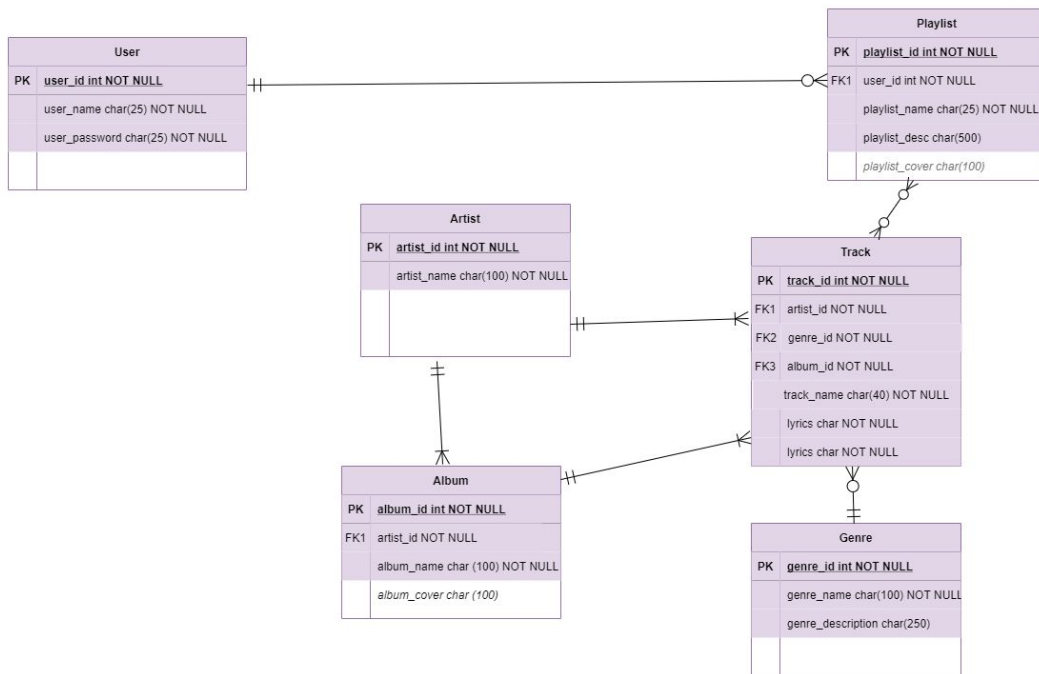
Albums CMQ-12

# Design - ERD from Client Requirements

CMQ - Entity Relationship Diagram



CMQ - Entity Relationship Diagram v1-1



## Additional Design

- User Stories (Jira)
- Sketches of page layouts
- MVC Design Pattern

# Security

## In place














- Minimise input fields
- Appropriate access

## Future possibilities

- Secure DB root name/password and avoid standards in production
- Consider cloud environment
- Forgotten password service
- Hash passwords (e.g. Bcrypt) rather than encrypt (in progress)
- Prevent SQL injection in input fields

# Choonz Application - Development approach and Demo Time



 .mvn/wrapper	CMQ-42 #comment EC: initial upload of files to the repo to the main b...	11 days ago
 Documentation	CMQ-70 #comment BO: jmx test file for jmeter	2 hours ago
 bin	CMQ-64 #comment BO: fixed Genres by adding G's S's	11 days ago
 src	CMQ-70 #comment EC: add Jmeter reports zip folder	12 hours ago
 .gitignore	CMQ-42 #comment EC: initial upload of files to the repo to the main b...	11 days ago
 README.md	BS: Testing	11 days ago
 UMD-Diagram.png	CMQ-42 #comment EC: initial upload of files to the repo to the main b...	11 days ago
 UMD-Diagram.ucls	CMQ-42 #comment EC: initial upload of files to the repo to the main b...	11 days ago
 debug.log	CMQ-65 #comment BO: Selenium setup and nav test	9 days ago
 lombok.config	CMQ-68 #comment BS: fixed recursion issues	8 days ago
 mvnw	CMQ-42 #comment EC: initial upload of files to the repo to the main b...	11 days ago
 mvnw.cmd	CMQ-42 #comment EC: initial upload of files to the repo to the main b...	11 days ago
 pom.xml	BS: Fixing merge issues from refactorDB to developer	3 days ago

# Testing strategy

## Functional Back end Testing 85%

- Unit Testing (JUnit, SpringBoot)
- Integration Testing
  - Spring MVC test framework (MockMVC - controller testing)

## Functional Front end Testing

- Selenium
- POM (Page Object Model ) Design Pattern

## User Acceptance Testing

- Cucumber /Gherkin
  - Given-When-Then
  - Testing against user stories

## Non-Functional Testing

- Performance Testing
  - Load, Stress, Spike, Soak (JMeter)
- Security Testing
  - Identify vulnerabilities (SonarQube)

# Functional Back End Unit Testing (Sprint 1)

```
// create
@Test // Test 1
void createTest() throws Exception {
    when(this.service.create(TEST_ALBUM_1)).thenReturn(this.mapToDTO(TEST_ALBUM_1));
    assertThat(new ResponseEntity<AlbumDTO>(this.mapToDTO(TEST_ALBUM_1), HttpStatus.CREATED))
        .isEqualTo(this.controller.create(TEST_ALBUM_1));
    verify(this.service, atLeastOnce()).create(TEST_ALBUM_1);
}

// Read one
@Test // Test 2
void readOneTest() throws Exception {
    when(this.service.readOne(TEST_ALBUM_1.getId())).thenReturn(this.mapToDTO(TEST_ALBUM_1));
    assertThat(new ResponseEntity<AlbumDTO>(this.mapToDTO(TEST_ALBUM_1), HttpStatus.OK))
        .isEqualTo(this.controller.readOne(TEST_ALBUM_1.getId()));
    verify(this.service, atLeastOnce()).readOne(TEST_ALBUM_1.getId());
}

// Read all
@Test // Test 3
void readAllTest() throws Exception {
    List<AlbumDTO> dtoList = ALBUM_LIST.stream().map(this::mapToDTO).collect(Collectors.toList());
    when(this.service.readAll()).thenReturn(dtoList);
    assertThat(this.controller.readAll()).isEqualTo(new ResponseEntity<>(dtoList, HttpStatus.OK));
    verify(this.service, atLeastOnce()).readAll();
}
```







```
// Update
@Test // Test 4
void updateTest() throws Exception {
    // rules WHEN... THEN DO....
    when(this.service.update(this.mapToDTO(TEST_ALBUM_1), TEST_ALBUM_1.getId()))
        .thenReturn(this.mapToDTO(TEST_ALBUM_1));

    // processes Store variables / perform requests....
    // assertions
    assertThat(new ResponseEntity<AlbumDTO>(this.mapToDTO(TEST_ALBUM_1), HttpStatus.ACCEPTED))
        .isEqualTo(this.controller.update(this.mapToDTO(TEST_ALBUM_1), TEST_ALBUM_1.getId()));

    // verification Check how many times rule has been applied
    verify(this.service, atLeastOnce()).update(this.mapToDTO(TEST_ALBUM_1), TEST_ALBUM_1.getId());
}

// Delete
@Test // Test 5
void deleteTest() throws Exception {
    when(this.service.delete(TEST_ALBUM_1.getId())).thenReturn(false);
    assertThat(this.controller.delete(TEST_ALBUM_1.getId()))
        .isEqualTo(new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR));
    verify(this.service, atLeastOnce()).delete(TEST_ALBUM_1.getId());
}

// Delete
@Test // Test 6
void deleteTest2() throws Exception {
    when(this.service.delete(TEST_ALBUM_1.getId())).thenReturn(true);
    assertThat(this.controller.delete(TEST_ALBUM_1.getId())).isEqualTo(new ResponseEntity<>(HttpStatus.NO_CONTENT));
    verify(this.service, atLeastOnce()).delete(TEST_ALBUM_1.getId());
}
```

▼  QA-Specialism-Project-Choonz-Team1		48.0 %	802	868	1,670
>  src/main/java		23.3 %	234	770	1,004
>  src/test/java		85.3 %	568	98	666



# Functional Back End Unit and Integration Testing (Sprint 2)

```
// Create test
@Test
void createTest() throws Exception {
    GenreDTO albumDTO = mapToDTO(new Genre("50s Rock n Roll", "Rock n Roll"));
    String testDTOAsJSON = this.jsonifier.writeValueAsString(albumDTO);

    RequestBuilder request = post(URI + "/create").contentType(MediaType.APPLICATION_JSON).content(testDTOAsJSON);

    ResultMatcher checkStatus = status().isCreated();

    GenreDTO testSavedDTO = mapToDTO(new Genre("50s Rock n Roll", "Rock n Roll"));
    testSavedDTO.setId(2L);
    String testSavedDTOAsJSON = this.jsonifier.writeValueAsString(testSavedDTO);

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

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

```
// create
@Test // Test 1
void createTest() throws Exception {
    when(this.service.create(TEST_GENRE_1)).thenReturn(this.mapToDTO(TEST_GENRE_1));
    assertThat(new ResponseEntity<GenreDTO>(this.mapToDTO(TEST_GENRE_1), HttpStatus.CREATED))
        .isEqualTo(this.controller.create(TEST_GENRE_1));
    verify(this.service, atLeastOnce()).create(TEST_GENRE_1);
}

// Read one
@Test // Test 2
void readOneTest() throws Exception {
    when(this.service.readOne(TEST_GENRE_1.getId())).thenReturn(this.mapToDTO(TEST_GENRE_1));
    assertThat(new ResponseEntity<GenreDTO>(this.mapToDTO(TEST_GENRE_1), HttpStatus.OK))
        .isEqualTo(this.controller.readOne(TEST_GENRE_1.getId()));
    verify(this.service, atLeastOnce()).readOne(TEST_GENRE_1.getId());
}

// Read all
@Test // Test 3
void readAllTest() throws Exception {
    List<GenreDTO> dtoList = GENRE_LIST.stream().map(this::mapToDTO).collect(Collectors.toList());
    when(this.service.readAll()).thenReturn(dtoList);
    assertThat(this.controller.readAll()).isEqualTo(new ResponseEntity<>(dtoList, HttpStatus.OK));
    verify(this.service, atLeastOnce()).readAll();
}
```

▼ QA-Specialism-Project-Choonz-Team1	<div><div></div></div> 96.0 %	4,841	202	5,043
> src/main/java	<div><div></div></div> 85.4 %	1,180	202	1,382
> src/test/java	<div><div></div></div> 100.0 %	3,661	0	3,661

# Functional Front End Testing (Selenium)

## Front End Functionality Covered:

- Site navigation
- User login
- Track read
- Playlist read
- Artist read
- Playlist creation

```
@BeforeEach
public void setUp() {
    System.setProperty("webdriver.chrome.driver", "src/test/resources/chromedriver.exe"); // set driver path
    driver = new ChromeDriver();
    driver.manage().window().setSize(new Dimension(1366, 768));
    driver.get("http://localhost:8082/html/form.html");
}

@AfterEach
public void tearDown() {
    driver.close();
}

@Test
public void testHome() throws InterruptedException {
    WebDriverWait wait = new WebDriverWait(driver, 30);
    wait.until(ExpectedConditions.titleIs("Choonz Playlist"));
    assertEquals("Choonz Playlist", driver.getTitle());
    String uName = "JohnSmith";
    driver.findElement(By.id("username")).sendKeys(uName);
    String pWord = "passwordJS";
    driver.findElement(By.id("password")).sendKeys(pWord);
    WebElement submitButton = driver.findElement(By.id("btn"));
    submitButton.submit();
    WebDriverWait wait2 = new WebDriverWait(driver, 30);
    wait2.until(ExpectedConditions.titleIs("Choonz Secret Music"));
    assertEquals("Choonz Secret Music", driver.getTitle());
}
```

# SonarQube

## Sprint 0 Initial Report:

[Chooz](#) [Home](#) [Master](#) [1](#) [Project Settings](#) [Project Information](#)

Overview Issues Security Hotspots Measures Code Activity

**My Issues All** [Blank Change](#) [+ -](#) to select issues [-- --](#) to navigate [↺ ↻](#) 10 issues 1h 40min effort

### Filters

[Clear All Filters](#)

- Type **VULNERABILITY** [OKAY](#)
- Severity
  - Critical 10
  - High 0
  - Minor 0
- Scope
- Resolution
- Status
- Security Category
  - SonarSource 10
  - Others
- DWASP Top 10
- SANS Top 25
- CWE
- Creation Date
- Language
- Rule
- Tag
- Directory
- File
- Assignee

Issue ID	Title	Severity	Open	Assigned	Effort	Action
wid_...com/jacg/choonz/hust/controllerAlbumController.java	Replace this persistent entity with a simple POJO or DTO object. Why is this an issue?	Critical	Open	Not assigned	10min effort	4 days ago • 133 • <a href="#">View</a> <a href="#">exp=as5, spring</a>
wid_...com/jacg/choonz/hust/controllerArtistController.java	Replace this persistent entity with a simple POJO or DTO object. Why is this an issue?	Critical	Open	Not assigned	10min effort	4 days ago • 145 • <a href="#">View</a> <a href="#">exp=as5, spring</a>
wid_...com/jacg/choonz/hust/controllerGenreController.java	Replace this persistent entity with a simple POJO or DTO object. Why is this an issue?	Critical	Open	Not assigned	10min effort	
wid_...com/jacg/choonz/hust/controllerPlaylistController.java	Replace this persistent entity with a simple POJO or DTO object. Why is this an issue?	Critical	Open	Not assigned	10min effort	
wid_...com/jacg/choonz/hust/controllerTrackController.java	Replace this persistent entity with a simple POJO or DTO object. Why is this an issue?	Critical	Open	Not assigned	10min effort	

**QUALITY GATE STATUS** ⓘ

**Passed**

All conditions passed.

10 of 10 shown

The screenshot displays the SonarQube dashboard for a project named '0.0.1-SNAPSHOT'. The top navigation bar includes links for Overview, Issues, Security Hotspots, Measures, Code, and Activity. The main section is titled 'MEASURES' and is divided into 'New Code' and 'Overall Code' tabs. The 'Overall Code' tab is active, showing a summary of code quality metrics. A green box on the left indicates the project is 'Passed' with 14 conditions passed. The metrics are as follows:

- Reliability:** 0 bugs (Green A)
- Security:** 10 vulnerabilities (Yellow D)
- Security Hotspots:** 5 (Yellow D), 0.0% Reviewed (Red E)
- Code Smells:** 55 (Yellow D), 5h 45min Debt (Yellow D)
- Maintainability:** 18 duplicated blocks (Green A)
- Unit Tests:** 0.0% Coverage on 514 Lines to cover (Red E)
- Duplications:** 32.0% Duplications on 1.5k Lines (Red E)

The bottom section shows the 'Overall Code' summary with a date of January 28, 2021, 4:50 PM. The metrics are:

- Reliability:** 0 bugs (Green A)
- Security:** 10 vulnerabilities (Yellow D)
- Security Hotspots:** 5 (Yellow D), 0.0% Reviewed (Red E)
- Code Smells:** 63 (Yellow D), 5h 45min Debt (Yellow D)
- Maintainability:** 18 duplicated blocks (Green A)
- Unit Tests:** 0.0% Coverage on 514 Lines to cover (Red E)
- Duplications:** 32.0% Duplications on 1.5k Lines (Red E)

The bottom right corner shows the date and time: January 28, 2021, 4:50 PM, and the project name: 0.0.1-SNAPSHOT.

# Jmeter

## Goals:

- Load, Stress, Spike, and Soak Tests implementation
- Stress Test: 12k users with throughput >300
- Soak Test: 60-80% of peak load
- Load Test: 10K Thread count, 40000 (4 methods \* 10000 users) total when implementing CRUD

# Load Testing (playlists)

## Load test conditions:

### Thread Properties

Number of Threads (users): 10000

Ramp-up period (seconds): 60

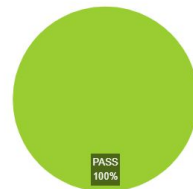
Loop Count: ☐ Infinite 1

☒ Same user on each iteration

### APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.801	500 ms	1 sec 500 ms	Total
0.003	500 ms	1 sec 500 ms	Transaction Controller
1.000	500 ms	1 sec 500 ms	read
1.000	500 ms	1 sec 500 ms	create
1.000	500 ms	1 sec 500 ms	update
1.000	500 ms	1 sec 500 ms	delete

### 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	40000	0	0.00%	2.63	0	359	2.00	3.00	3.00	3.00	332.98	95.77	78.70

# Stress Testing (Albums)

Stress test conditions:

Thread Properties

Number of Threads (users):

Ramp-up period (seconds):

Loop Count: ☐ Infinite

☒ Same user on each iteration

☐ Delay Thread creation until needed

☐ Specify Thread lifetime

HTML Report:



# Stress Testing To Failure (Albums)

## Stress test conditions:

Thread Properties

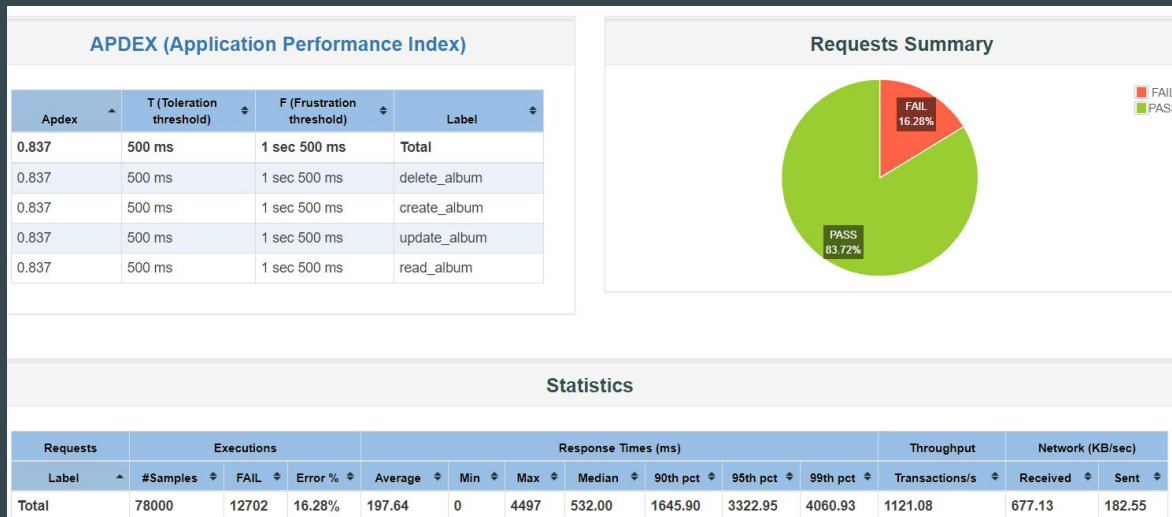
Number of Threads (users): 6500

Ramp-up period (seconds): 60

Loop Count: ☐ Infinite 3

☒ Same user on each iteration

## HTML Report:



# Spike Testing (Tracks)

Spike test conditions:

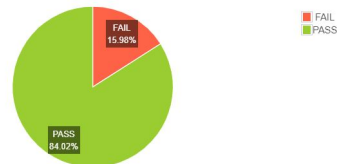
Threads Schedule					
Start Threads Count	Initial Delay, sec	Startup Time, sec	Hold Load For, sec	Shutdown Time	
1000	1	1	1	1	1
2000	10	1	1	1	1
10000	20	1	1	1	1
1000	30	1	1	1	1
Add Row		Copy Row		Delete Row	

HTML Report:

## APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.807	500 ms	1 sec 500 ms	Total
0.727	500 ms	1 sec 500 ms	create_track
0.857	500 ms	1 sec 500 ms	delete_track
0.863	500 ms	1 sec 500 ms	read_track
0.868	500 ms	1 sec 500 ms	update_track

## 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	50976	8148	15.98%	224.93	0	2917	230.00	618.00	824.00	1079.98	1617.77	1026.38	264.65		
create_track	20911	4594	21.97%	265.05	1	2917	177.00	673.00	843.00	1900.99	663.63	511.89	104.36		
delete_track	8374	1028	12.28%	204.14	0	1639	157.00	460.00	583.25	785.00	270.50	119.69	43.80		
read_track	11834	1418	11.98%	185.14	0	1554	153.00	395.00	492.00	670.65	380.65	220.45	53.99		
update_track	9857	1108	11.24%	205.27	0	1396	163.00	447.00	554.00	836.00	317.84	182.27	65.02		

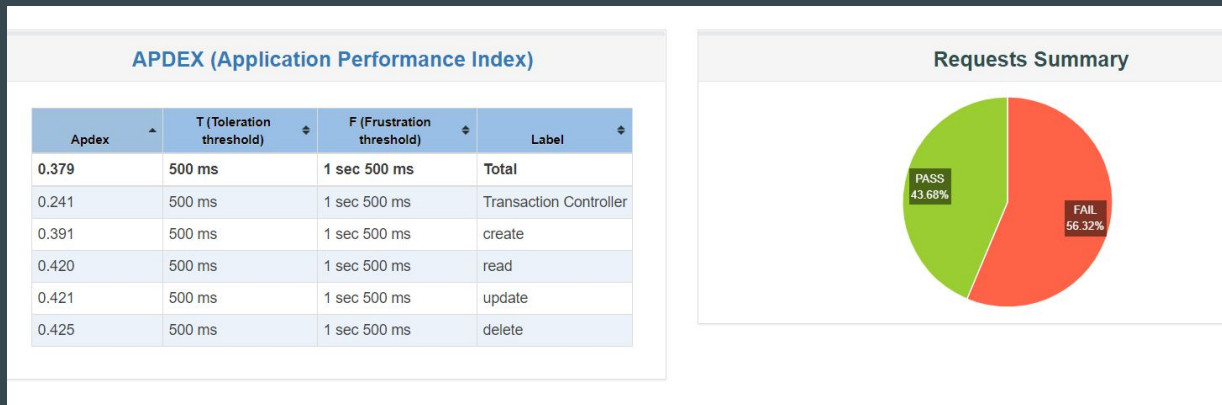


# Soak Testing (Playlists)

Soak test conditions:

Threads Schedule				
Start Threads Count	Initial Delay, sec	Startup Time, sec	Hold Load For, sec	Shutdown Time
2000	10	120	600	120

HTML Report:



# Ideal future Jmeter testing

- Soak test (24 Hours+)
- Distributed testing on a hosted remote server (Jenkins)
- Jmeter testing user login and CRUD, and log out.

# Design - Future Scope

- Extend / specialised API for in-house / music industry updates to core-data
- Collect MI on plays per track within Choonz - feed into Featured Tracks (could also be used to reduce supplier costs or provide income stream)
- Monetise by adverts and/or paid subscription options
- Include dynamic picture links (currently hardcoded)
- Allow users to share a copy of their playlist(s) with other Choonz users
- Incorporate podcasts
- Implement functionality from original specification, not addressed during first two sprints due to time and prioritisation constraints

**Thank you for listening!**

**Q&A session**