

Choonz

...

By Team 4

Introduction

Process of Development

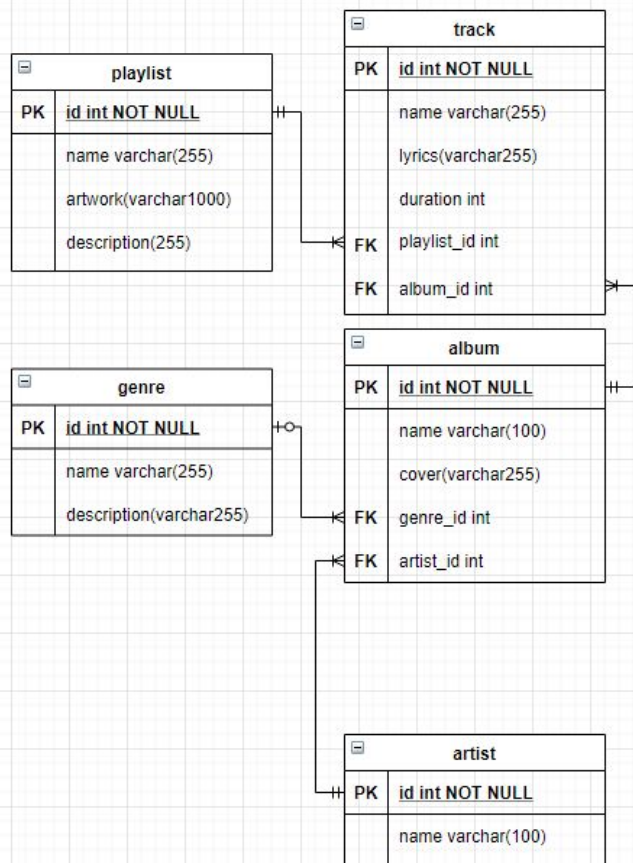
- Looked at current state of the project (API and Front-End)
- Created a risk assessment for project
- Decided on the priority of the features for final product (to quickly provide a MVP)
- Split the user stories into sprints
- Added project to Git
- Worked using main/dev/feature branch model.

Risk Assessment

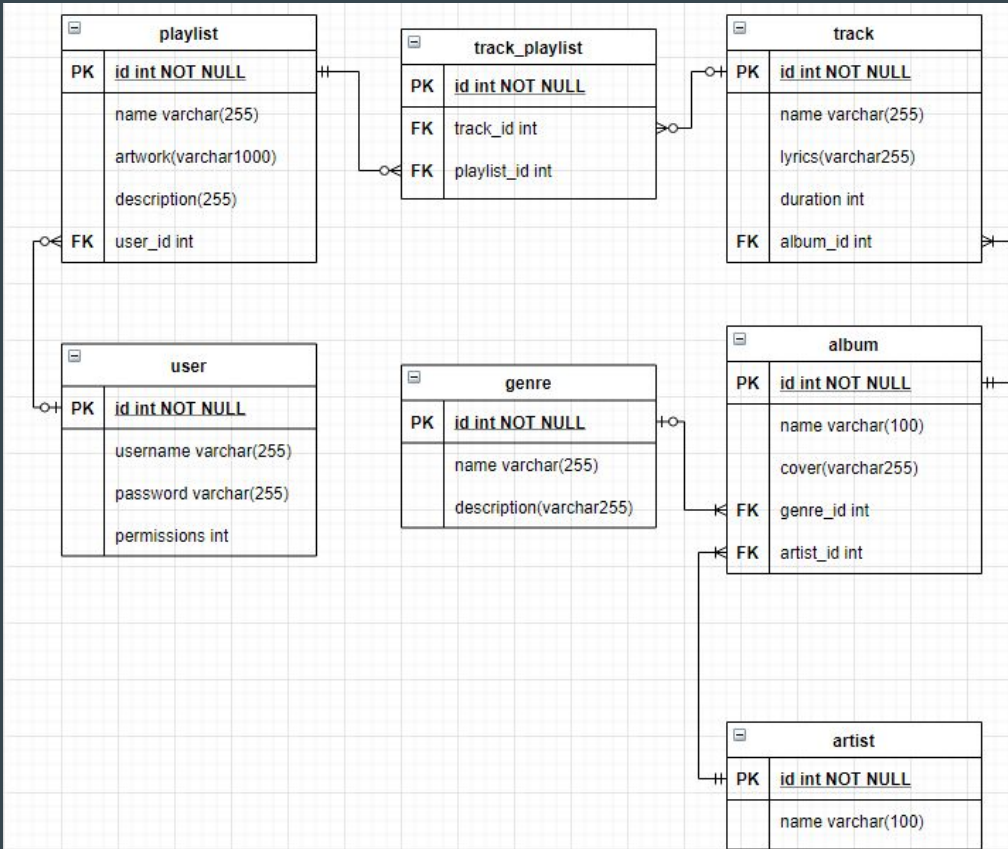
ID	Risk Description	Cause	Effect	Likelihood (1-5)	Impact (1 – 5)	Risk Rating (1-25)	Action
1	Lack of time	Improper time management	Not Finishing Project	3	5	15	Daily Planning and daily targets / stick to MVP
2	Version Control not correctly utilised	Not pushing to correct Git branches regularly	Unable to <u>rollback</u> to a working verion of the system, resulting in no working program	2	2	4	Use Feature/Branch model and push regularly
3	Login Leak	Password Information gets stolen	Information leak and loss of trust	2	5	10	Encrypt Data after stored in database
4	Inaccessible work equipment	Failure of equipment	Losing access to project	1	3	3	Have access to back-up equipment
5	Become Unfit to work	Catch an Illness (such as Covid-19)	Unable to work	3	2	6	Social Distance
6	Over working	Fall behind schedule	Not finishing project on time	1	1	1	Take regular breaks and manage

Diagrams

- ERD at the start of the project
- No User Table
- Track contained a playlist ID



- ERD after refactoring
- User table was added
- A user owns a playlist
- A user can be admin
- Track_playlist table was added



Sprint 1

CPB Sprint 1 7 issues		9.5	0	11	Plan sprint ▾	⋮
18/Jan/21 10:00 AM • 22/Jan/21 5:30 PM						
📌	As a User, I want to be able to view an album page, so that I can view the tracks in the album.	Albums	CPB-14	↑	4	
📌	As a User, I want to be able to click on a track in the album, so that I can view the track page	Tracks	CPB-15	↑	4	
📌	As a User, I want to be able to click an artist, so that I can see the artist page	Artists	CPB-16	↑	4	
✓	User Backend	Log In	CPB-34	↑		
📌	As a User, I want to be able to register as a user, so I can log in.	Log In	CPB-11	↑	3	
📌	As a User, I want to be able to log in, so that I can use the site.	Log In	CPB-12	↑	5.5	
✓	Unit Testing	JUnit / Mockito	CPB-61	↑		

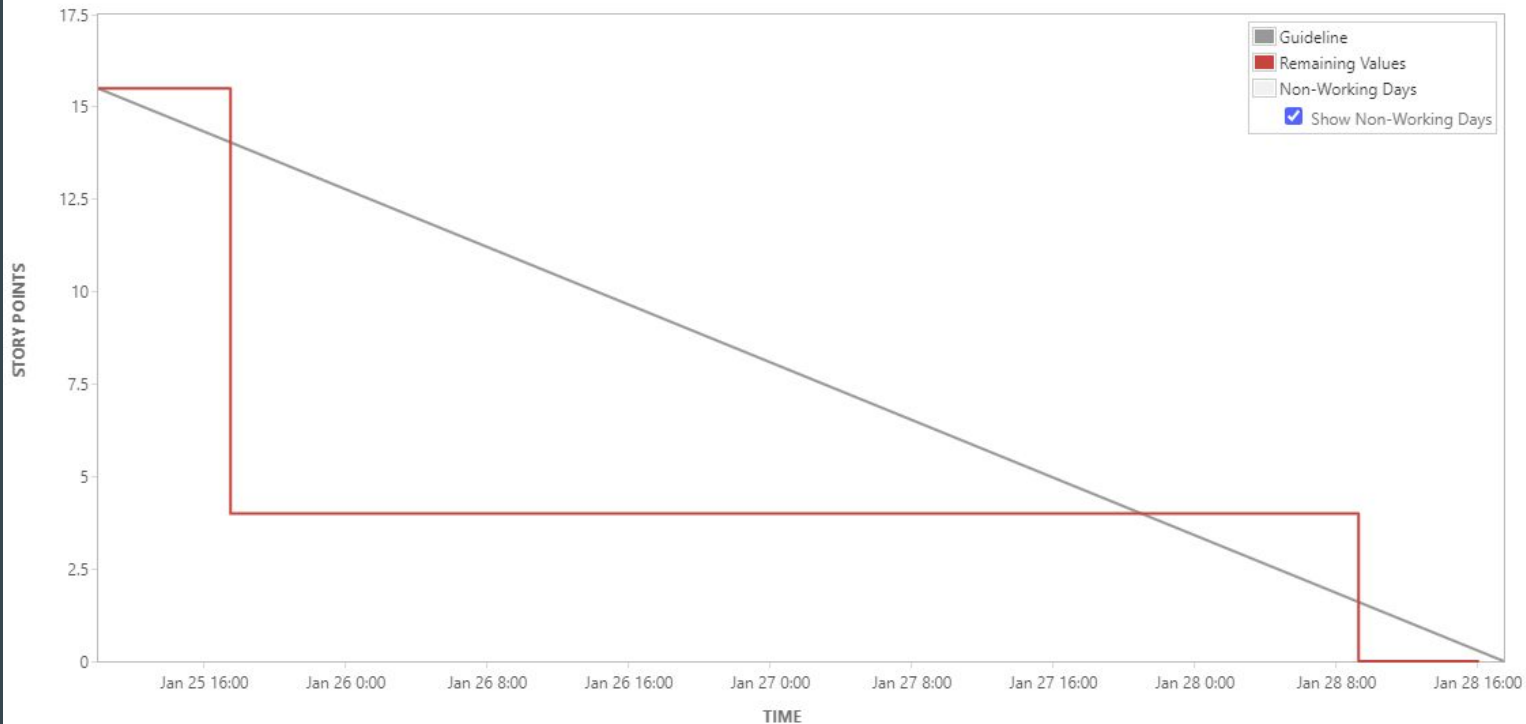
Sprint 1: Retrospective

- Because this sprint was mainly to quickly produce a MVP, this was completed relatively quickly.
- There was no delete from playlist function. This was before we decided to implement the track_playlist table.
- Real data should be used rather than using test data
- The only testing that was implemented was Unit testing and Postman. This was taken into consideration for the next sprint where testing was performed regularly.

Sprint 2

CPB Sprint 2 10 issues		0	0	15.5	Plan sprint ▾	⋮
25/Jan/21 10:00 AM • 28/Jan/21 5:30 PM						
📌	As a User, I want to see cards for each playlist so that I can perform CRUD on them.	Home Page	EPB-13	↑	4	
📌	As a user I need to see all tracks in a genre, so that I can click on a track.	Genres	EPB-20	↑	5	
📌	As a user I need to be able to create a playlist, so that only I can edit it.	Playlists	EPB-21	↑	6.5	
✓	Integration Testing	JUnit / Mockito	EPB-60	↑		
✓	Feature Testing	Selenium / Cucumber	CPB-73	↑		
✓	JavaScript Testing	Selenium / Cucumber	EPB-72	↑		
✓	Soak Testing	JMeter	EPB-75	↑		
✓	Spike Testing	JMeter	EPB-76	↑		
✓	Load Testing	JMeter	EPB-77	↑		
✓	Stress Testing	JMeter	EPB-78	↑		

Sprint 2

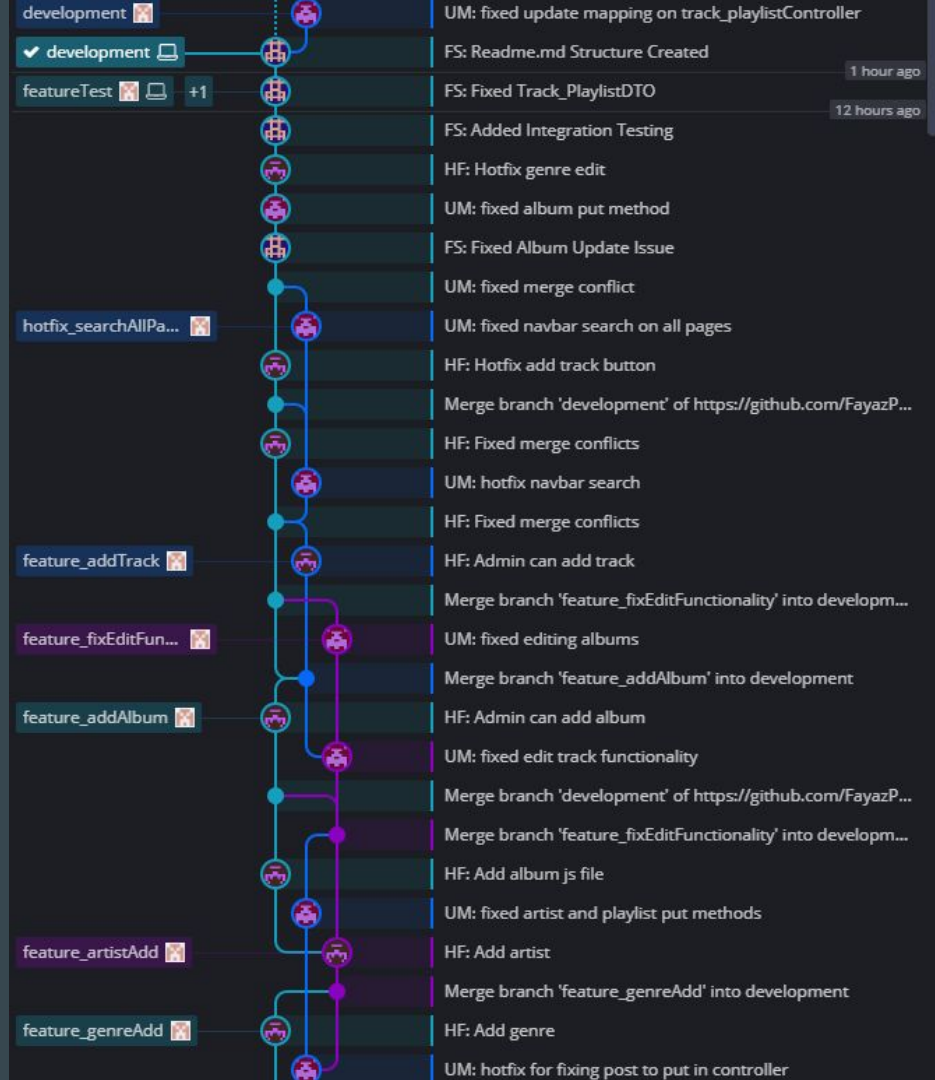


Sprint 2: Retrospective

- Was mainly to do with CRUD functions on the playlist and allowing an admin to perform CRUD functions on every entity.
- This meant adding permissions to a user; 0 for normal, 1 for admin.
- Creating Intermediary many to many table to track-playlist.
- Fixing bugs found through testing.
- Updating documentation for end users.

Git

- Never fully used Git in a team environment, with different developers on different branches.
- Started with a few merge conflicts in the beginning, but we all understood it relatively quickly.
- Git is now an **essential part of development**.



Development with Testing

The screenshot displays a REST client interface with a POST request to `http://localhost:8082/trackPlaylist/create`. The request body is a JSON object with the following structure:

```
1 {
2   ... "tracks": {
3     ... "id": 1
4   },
5   ... "playlists": {
6     ... "id": 1
7   }
8 }
```

The response status is 201 Created, with a time of 14 ms and a size of 431 B. The response body is a JSON object with the following structure:

```
1 {
2   ... "id": 52,
3   ... "tracks": {
4     ... "id": 1,
5     ... "name": null,
6     ... "trackPlaylists": [],
7     ... "duration": 0,
8     ... "lyrics": null
9   },
10  ... "playlists": {
11    ... "id": 1,
12    ... "name": null,
13    ... "description": null,
14    ... "artwork": null,
15    ... "trackPlaylists": []
16  }
17 }
```

Unit & Integration testing

74% Test Coverage

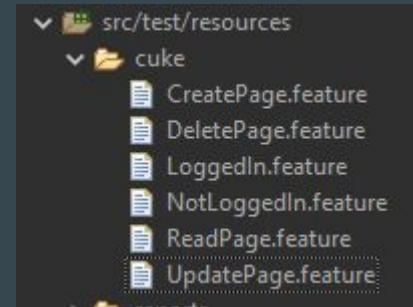
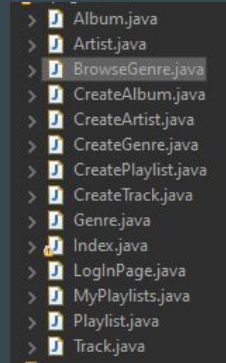
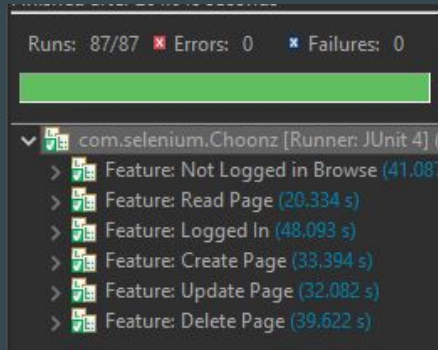
80% Industry Standard

```

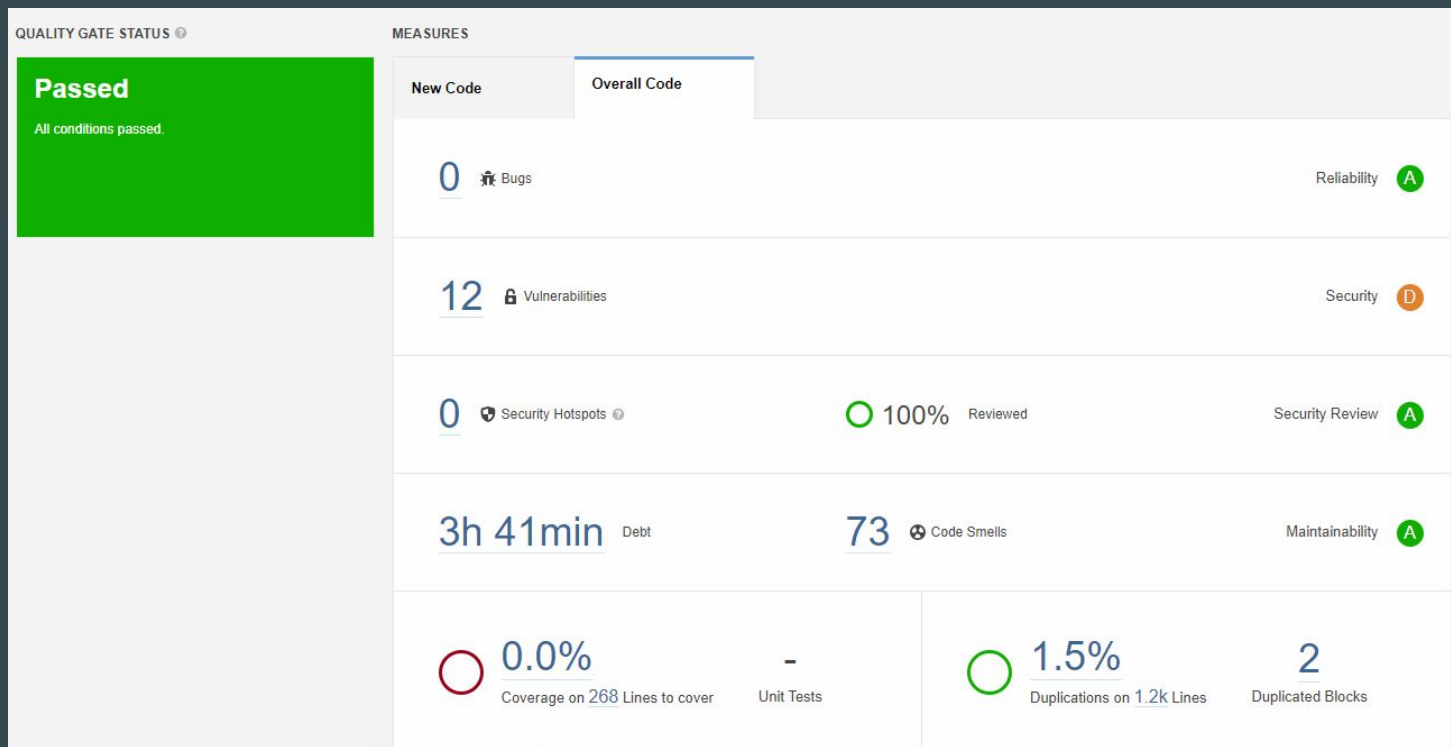
└─ java 79% classes, 74% lines covered
  └─ com.qa.choonz 79% classes, 74% lines covered
    └─ config 100% classes, 100% lines covered
    └─ exception
    └─ persistence 100% classes, 76% lines covered
      └─ domain 100% classes, 76% lines covered
        └─ Album 100% methods, 100% lines covered
        └─ Artist 100% methods, 100% lines covered
        └─ Genre 100% methods, 100% lines covered
        └─ Playlist 80% methods, 65% lines covered
        └─ Track 90% methods, 95% lines covered
        └─ Track_Playlist 16% methods, 12% lines covered
        └─ User 70% methods, 45% lines covered
      └─ repository
        └─ AlbumRepository
        └─ ArtistRepository
        └─ GenreRepository
        └─ PlaylistRepository
        └─ Track_PlaylistRepository
        └─ TrackRepository
        └─ UserRepository
      └─ rest 93% classes, 82% lines covered
        └─ controller 100% classes, 80% lines covered
          └─ AlbumController 85% methods, 90% lines covered
          └─ ArtistController 85% methods, 90% lines covered
          └─ GenreController 100% methods, 100% lines covered
          └─ PlaylistController 75% methods, 83% lines covered
          └─ RouteController 0% methods, 25% lines covered
          └─ Track_PlaylistController 14% methods, 36% lines covered
          └─ TrackController 85% methods, 90% lines covered
          └─ UserController 87% methods, 90% lines covered
        └─ dto 85% classes, 87% lines covered
      └─ service 100% classes, 69% lines covered
        └─ AlbumService 100% methods, 100% lines covered
        └─ ArtistService 85% methods, 88% lines covered
        └─ GenreService 100% methods, 100% lines covered
        └─ PlaylistService 25% methods, 28% lines covered
        └─ Track_PlaylistService 12% methods, 23% lines covered
        └─ TrackService 100% methods, 100% lines covered
        └─ UserService 66% methods, 61% lines covered
      └─ utils 0% classes, 0% lines covered
    └─ ChoonzApplication 0% methods, 33% lines covered
```

Functional Testing - Selenium

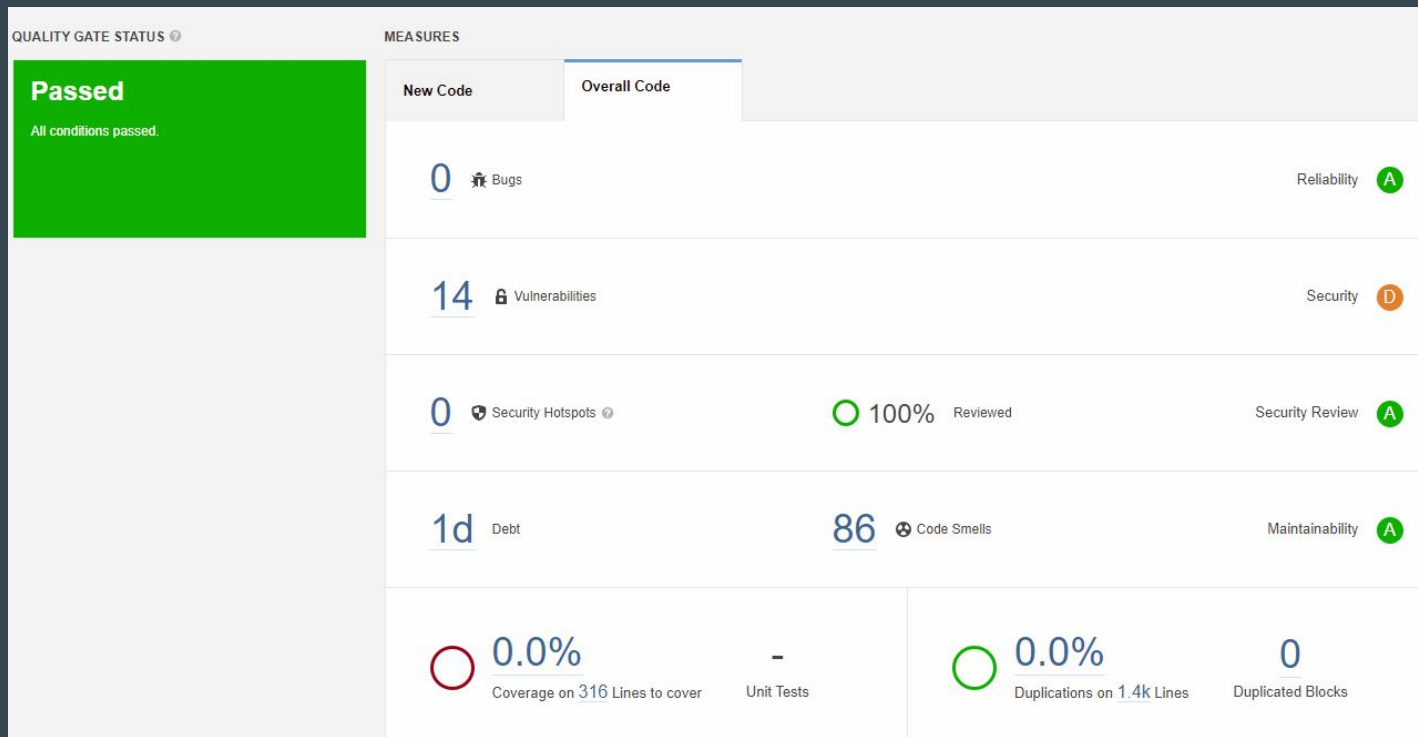
- Used Selenium to test the front-end and see if all the functions performed correctly.
- Also allowed us to see if changes in the API affected the front-end.
- Testing started with testing the difference in the site when logged in or not, then went on to test all CRUD.
- Used Cucumber and Gherkin to write the feature files
- And in the step definition file, used Selenium with POM



Static Analysis - SonarQube

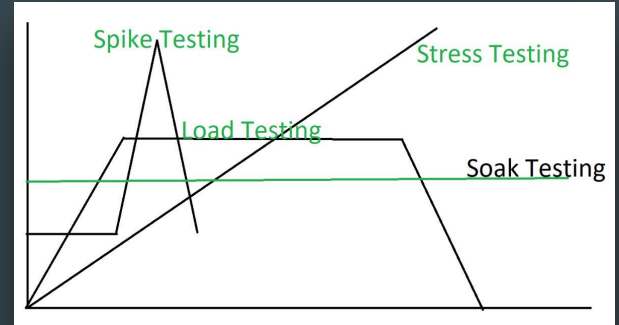


Static Analysis - SonarQube



Non-Functional Testing

- Used JMeter for measuring performance of web application and api
- Performed Spike, Load, Stress & Soak Tests
- Performance hindered by personal hardware



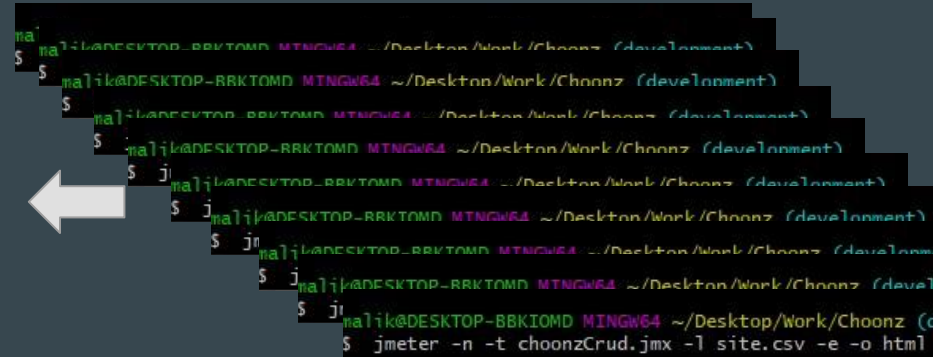
Non-Functional Testing

Utility Batch Files

Created Batch files to save writing command lines

Helps individuals unfamiliar with JMeter testing suite run tests easily

Name	Date modified	Type	Size
crudTESTS	28/01/2021 17:33	File folder	
csv	28/01/2021 17:33	File folder	
html	28/01/2021 17:33	File folder	
pageTESTS	28/01/2021 17:33	File folder	
jmeter	28/01/2021 17:33	Text Document	291 KB
runLOADTests(around_13.5min)	28/01/2021 17:33	Windows Batch File	1 KB
runSOAKTests(around_15min)	28/01/2021 17:33	Windows Batch File	1 KB
runSPIKETests(around_5min)	28/01/2021 17:33	Windows Batch File	1 KB
runSTRESSTests(around_8.5min) - without hmPage	28/01/2021 17:33	Windows Batch File	1 KB
runSTRESSTests(around_8.5min)	28/01/2021 17:33	Windows Batch File	1 KB

A screenshot of a Windows command prompt terminal window. The prompt is "malik@DESKTOP-BBKTOMD MINGW64 ~\Desktop\Work\Choonz (development)". The user has entered the command "jmeter -n -t choonzCrud.jmx -l site.csv -e -o html". A large white arrow points from the left towards the terminal window.

```
malik@DESKTOP-BBKTOMD MINGW64 ~\Desktop\Work\Choonz (development)
$ jmeter -n -t choonzCrud.jmx -l site.csv -e -o html
```

Non-Functional Testing



Example Test Result:

Spike Test on Album page with API crud operation.

Tested with 1000 threads within 1 minute

Throughput
Transactions/s
836.33
420.73
420.65
418.72

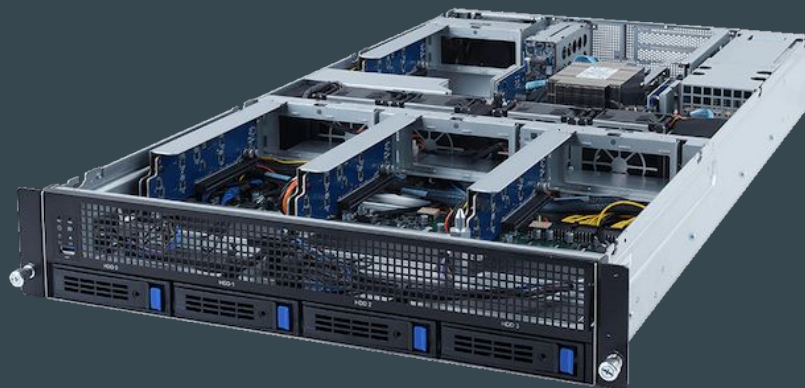
Response Times (ms)						
Average	Min	Max	Median	90th pct	95th pct	99th pct
82.98	1	2500	111.00	201.00	271.00	731.98

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.639	500 ms	1 sec 500 ms	Total
0.626	500 ms	1 sec 500 ms	Transaction Controller
0.641	500 ms	1 sec 500 ms	Albums Read
0.649	500 ms	1 sec 500 ms	Album Page

Non-Functional Testing

Solution to meet Spec Requirements:

Hardware



Demonstration

Questions