Description of application development for this project:

For this project, I will be developing a backend java API by utilizing two microservices for managing user profiles and favorite playlists. To accomplish this, two databases will be used for the backend, namely, mongoDB and neo4j. The neo4j database will store user profile data and interactions/relationships, while mongoDB will store a collection of songs. After setting up the backend, the microservices will be implemented to include a number of REST API endpoints used to enable communication between the two backend servers via CRUD operations. Maven will be used to compile and run these microservices, where it will then be possible to use these services by utilizing http requests via the port number specified.

An example of a New Feature:

Add a property to playlists called 'Access' (with possible values: public or private), and add an endpoint that allows user profiles to subscribe to publicly accessible playlists. Default access will be private upon every creation of a playlist. Users must give public access privileges before any profile user can subscribe to their playlists. There should also be an endpoint implemented to fetch possible playlists that have public access.

Type of endpoints needed to implement this new feature:

To implement this new feature, we would need **GET** and **PUT** endpoints as follows:

- **GET** /getPublicPlaylists{userName}
  Description: Retrieves playlists created by user that are publicly accessible
  URL Parameters:
  - ➔ userName - the userName of profile whose publicly accessible playlists will be fetched

  Expected Response:
  - ➔ "status"
    - ➢ "OK", if the data was retrieved successfully
    - ➢ "NOT_FOUND" if the user doesn't have any publicly accessible playlists
    - ➢ "NOT_FOUND" if the user does not exist
  - ➔ "data"

```
{
 "data":
    { "akeem": [
          “plName” : “akeempl”
          “songName” : “34+35”,
          “songArtistFullName” : “Ariana Grande”
          “songAlbum” : “Positions(Deluxe)”
      ],
      "akeem": [
           “plName” : “akeempl2”
          “songName” : “Unbreak My Heart”,
          “songArtistFullName” : “Toni Braxton”
          “songAlbum” : “Secrets(1996)”
      ]
    },
   "status": "OK"
}
```

- **PUT** /changeAccess

  Description: Allows a profile to change access privileges to the associated playlists created by the profile.

  Body Parameters:

  ➔ userName - the username of the profile

  ➔ plName - the username of the playlist **created** by the associated profile

  Example Body:

```
“userName” : “Akeem”
“plName” : “akeempl”
```

  Expected Response:

  ➔ “status”

  ➢ “OK”, if the user is able to change the access privilege of the associated playlists

  ➢ “NOT_FOUND” if the profile-plName pair doesn’t exist

- **PUT** /subscribe

  Description: Allows a profile to subscribe to a publicly accessible playlist

  Body Parameters:

  ➔ userName - the username of the profile associated with the playlist to which the user wants to subscribe

  ➔ plName - the specific publicly accessible playlist name the user profile wants to subscribe to **Example Body:**

“userName” : “Akeem” “plName” : “akeempl”

Expected Response:

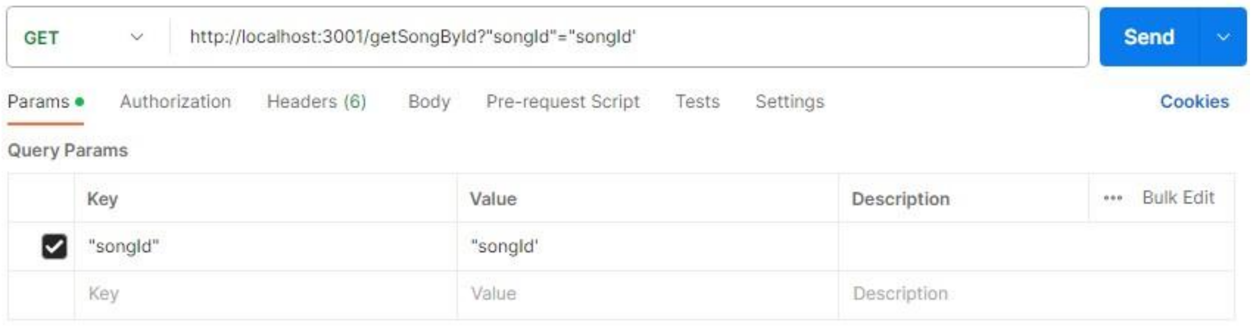
➔ “status”

➢ “OK”, if the user is able to subscribe to the playlist

➢ “NOT_FOUND” if the playlist does not have public access, or if the profile-playlist pair doesn’t exist.

Testing of project components:

I will be testing each feature for my project using Postman software. Each of the endpoints can be tested by sending http requests to the respective microservice. For example, by using Postman, we can send a GET request to the getsongByID endpoint as follows:



GET http://localhost:3001/getSongById?"songId"="songId'

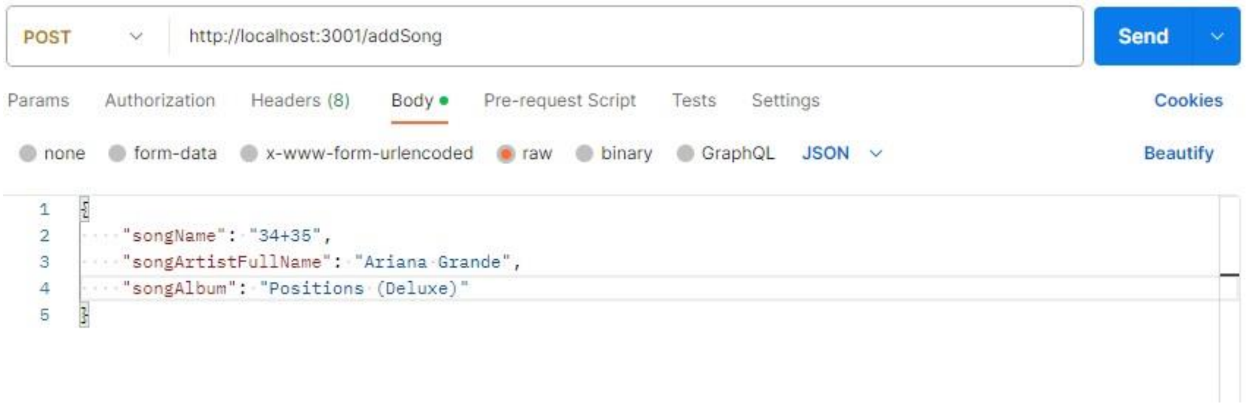
Params ● Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description
✓	"songId"	"songId'	
	Key	Value	Description

From the screenshot above, notice that we can test endpoints that require parameters, by adding the (Key, Value) pair to the table shown.

For this project, there are also endpoints where body parameters are required. Below is an example of how body parameters are included into the request:



POST http://localhost:3001/addSong

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

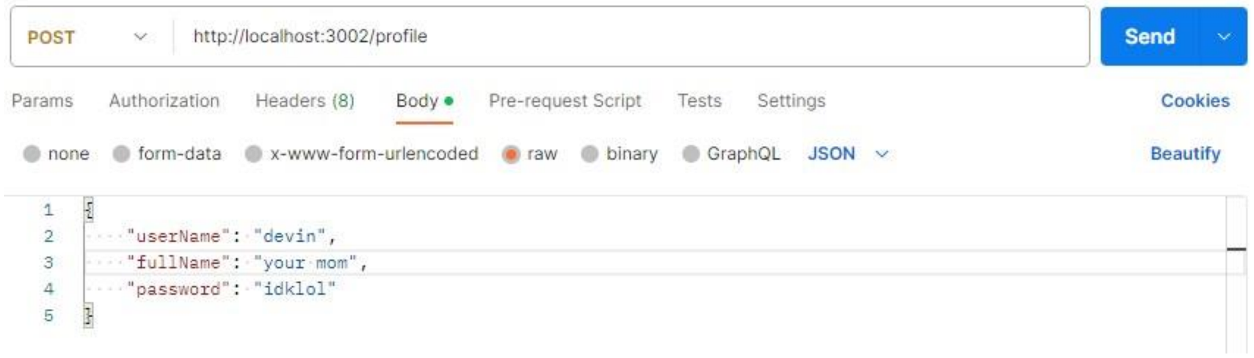
none form-data x-www-form-urlencoded ● raw binary GraphQL JSON Beautify

```
{
    "songName": "34+35",
    "songArtistFullName": "Ariana Grande",
    "songAlbum": "Positions (Deluxe)"
}
```

This is a POST request for the /addSong endpoint where we can see that the body parameters will be sent as a JSON object.

These two screenshots above, show examples of how the song microservice can be tested. Also notice how the URL provides the port number for which the song microservices is running on: **3001**.

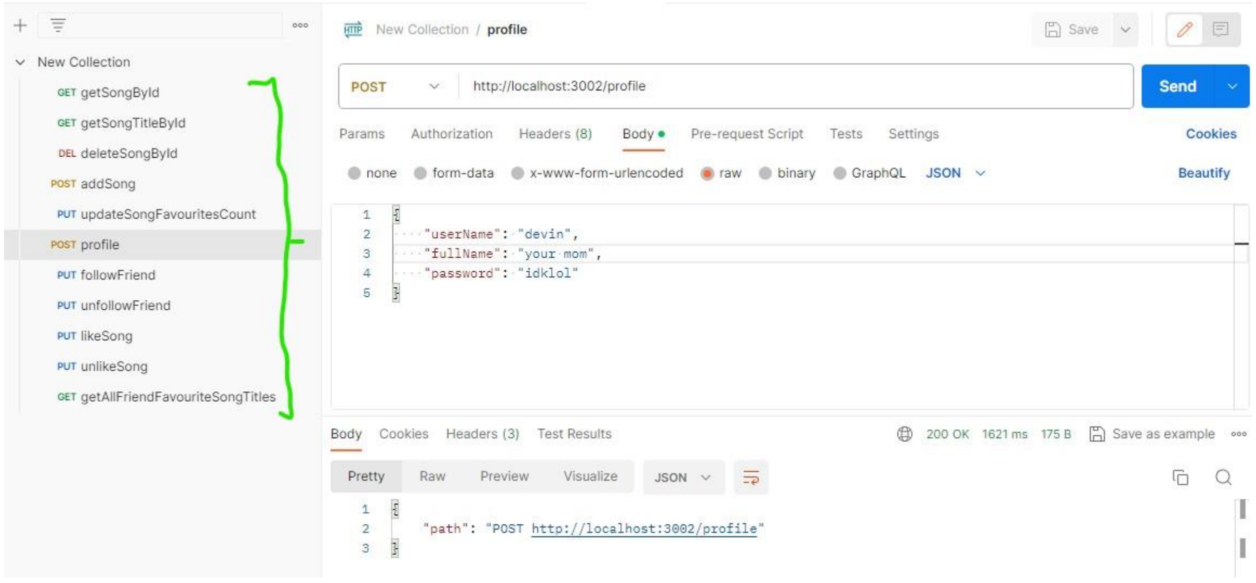
Here is an example of how to test an endpoint from the profile microservice, which runs on port **3002**.



http://localhost:3002/profile

```
{
    "userName": "devin",
    "fullName": "your mom",
    "password": "idklol"
}
```

Below are the list of endpoints that will be tested extensively for this project(in green):



GET getSongById
GET getSongTitleById
DEL deleteSongById
POST addSong
PUT updateSongFavouritesCount
POST profile
PUT followFriend
PUT unfollowFriend
PUT likeSong
PUT unlikeSong
GET getAllFriendFavouriteSongTitles

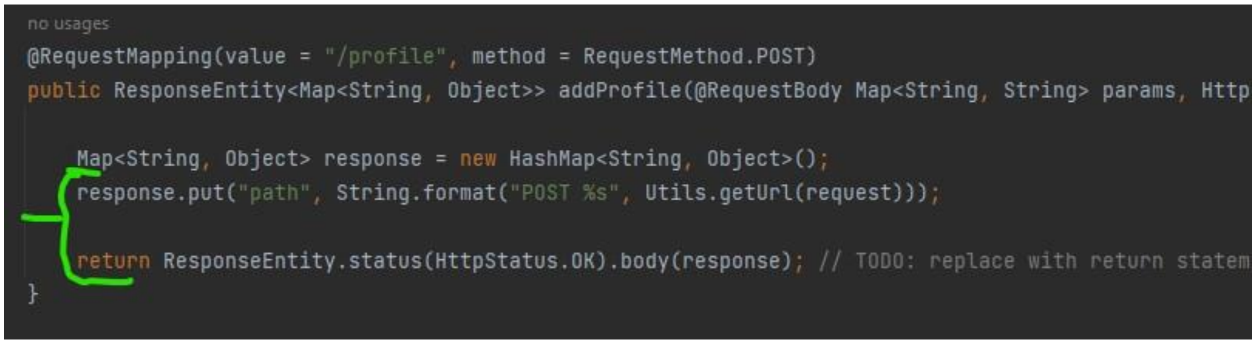
http://localhost:3002/profile

```
{
    "userName": "devin",
    "fullName": "your mom",
    "password": "idklol"
}
```

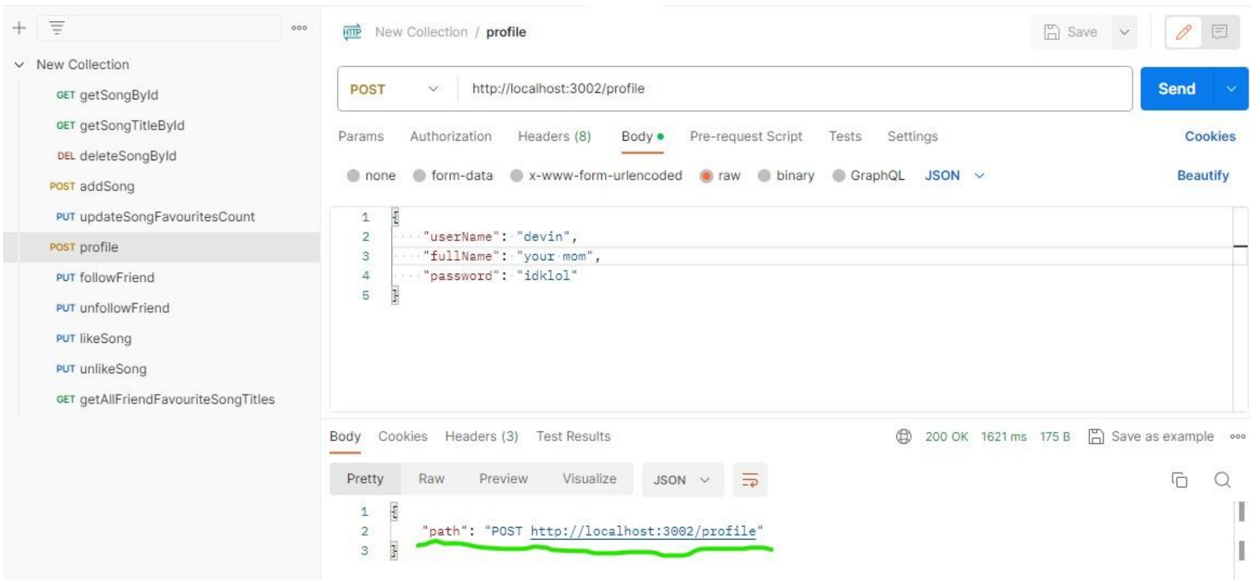
200 OK 1621 ms 175 B

```
{
    "path": "POST http://localhost:3002/profile"
}
```

To validate that the microservices work as intended, we can simply check the response message in ‘Body’ after sending the request. For example, if I send a **POST** request to the profile microservice, the /profile endpoint receives this request and processes the response by sending a string of the http request and the URL used to contact the service, as shown below(underlined/bracketed in green):



```
no usages
@RequestMapping(value = "/profile", method = RequestMethod.POST)
public ResponseEntity<Map<String, Object>> addProfile(@RequestBody Map<String, String> params, Http

    Map<String, Object> response = new HashMap<String, Object>();
    response.put("path", String.format("POST %s", Utils.getUrl(request)));

    return ResponseEntity.status(HttpStatus.OK).body(response); // TODO: replace with return statem
}
```



New Collection / profile

POST http://localhost:3002/profile

```
{
    "userName": "devin",
    "fullName": "your mom",
    "password": "idklol"
}
```

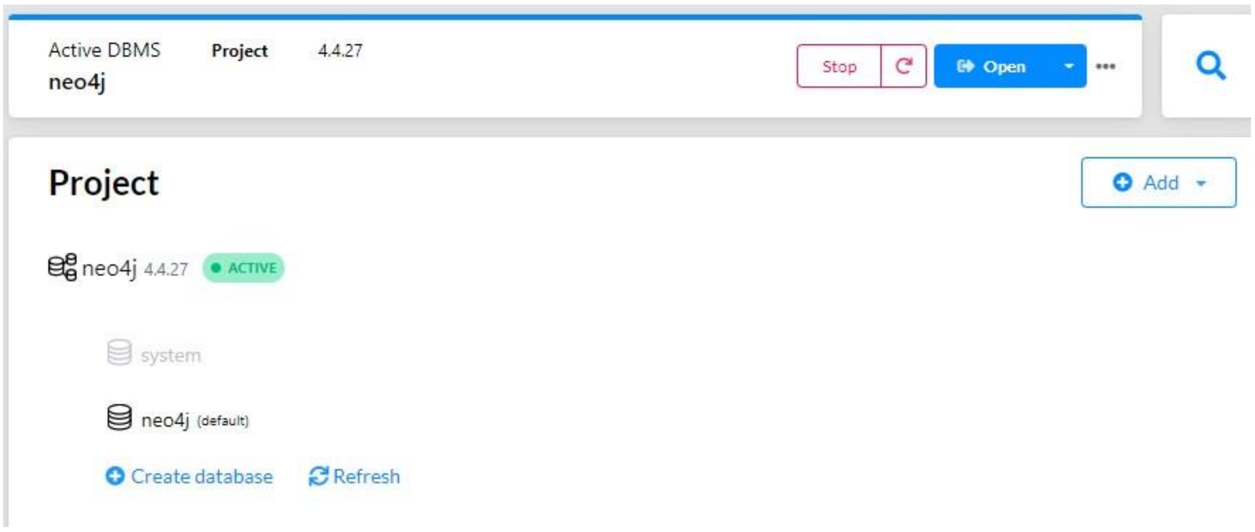
200 OK 1621 ms 175 B

```
{
    "path": "POST http://localhost:3002/profile"
}
```

Note: After implementing the components for this project, the response messages for all endpoints will be modified to match the expected responses specified in the requirements. For this phase, I am testing to ensure the project was set up properly and the microservices are running as intended.

Project Setup:

Screenshots showing neo4j database setup:


Active DBMS neo4j Project 4.4.27

Stop Open

Project

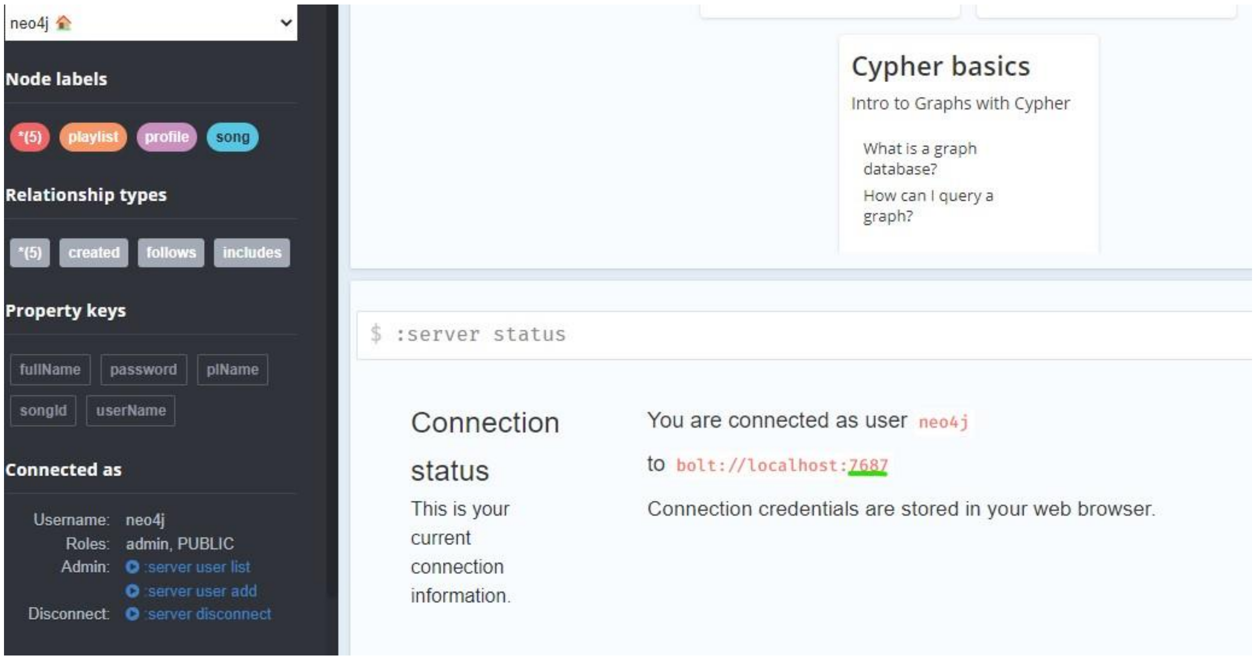
Add

neo4j 4.4.27 ACTIVE

system

neo4j (default)

Create database Refresh


neo4j

Node labels

*(5) playlist profile song

Relationship types

*(5) created follows includes

Property keys

fullName password plName songId userName

Connected as

Username: neo4j
Roles: admin, PUBLIC
Admin: :server user list
:server user add
Disconnect: :server disconnect

Cypher basics
Intro to Graphs with Cypher
What is a graph database?
How can I query a graph?

```
$ :server status
```

Connection status

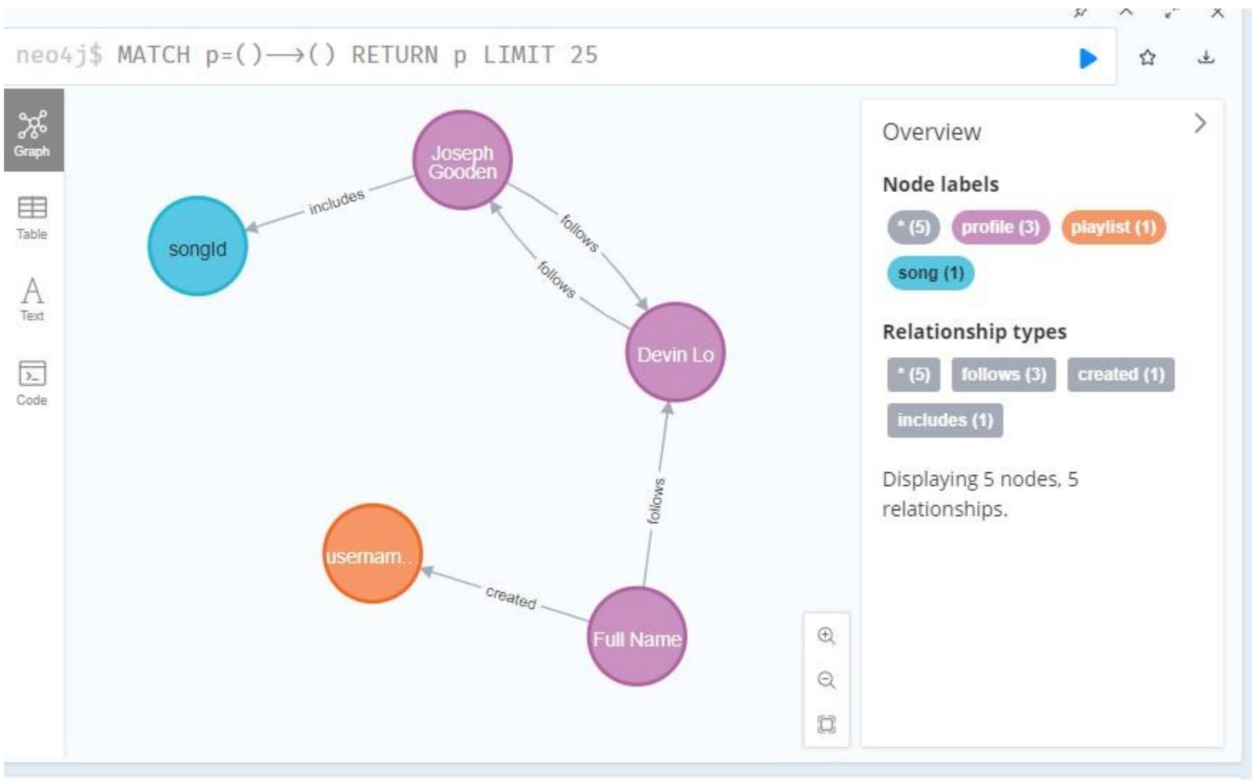
This is your current connection information.

You are connected as user `neo4j`

to `bolt://localhost:7687`

Connection credentials are stored in your web browser.

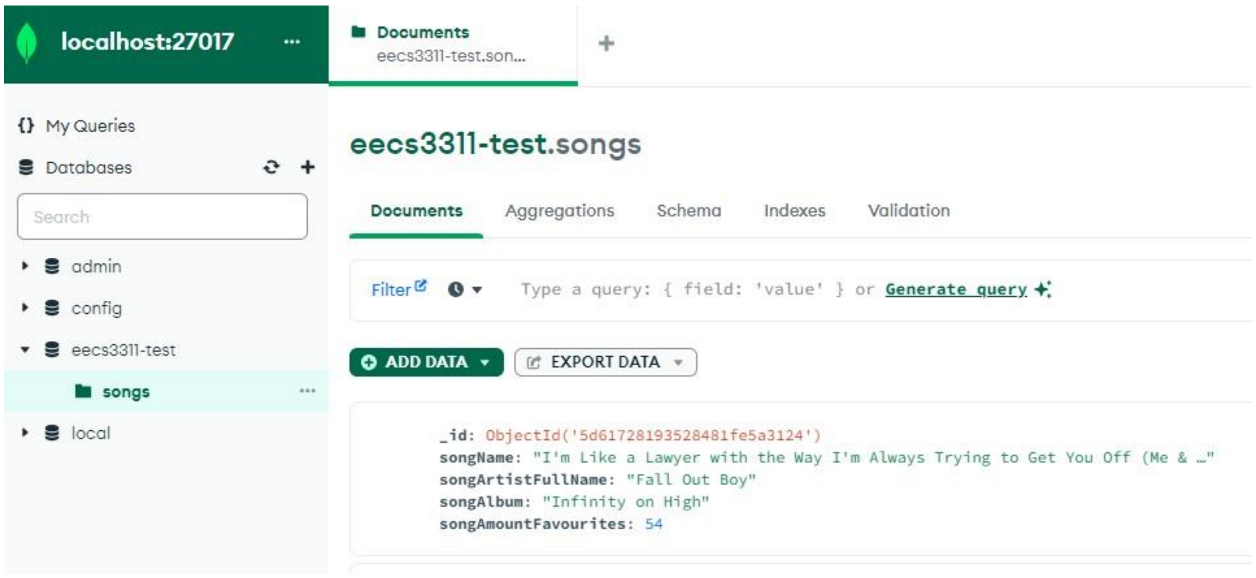
Neo4j server runs on port **7687**



```
neo4j$ MATCH p=()⟶() RETURN p LIMIT 25
```

Sample screenshot showing nodes and their corresponding relationships

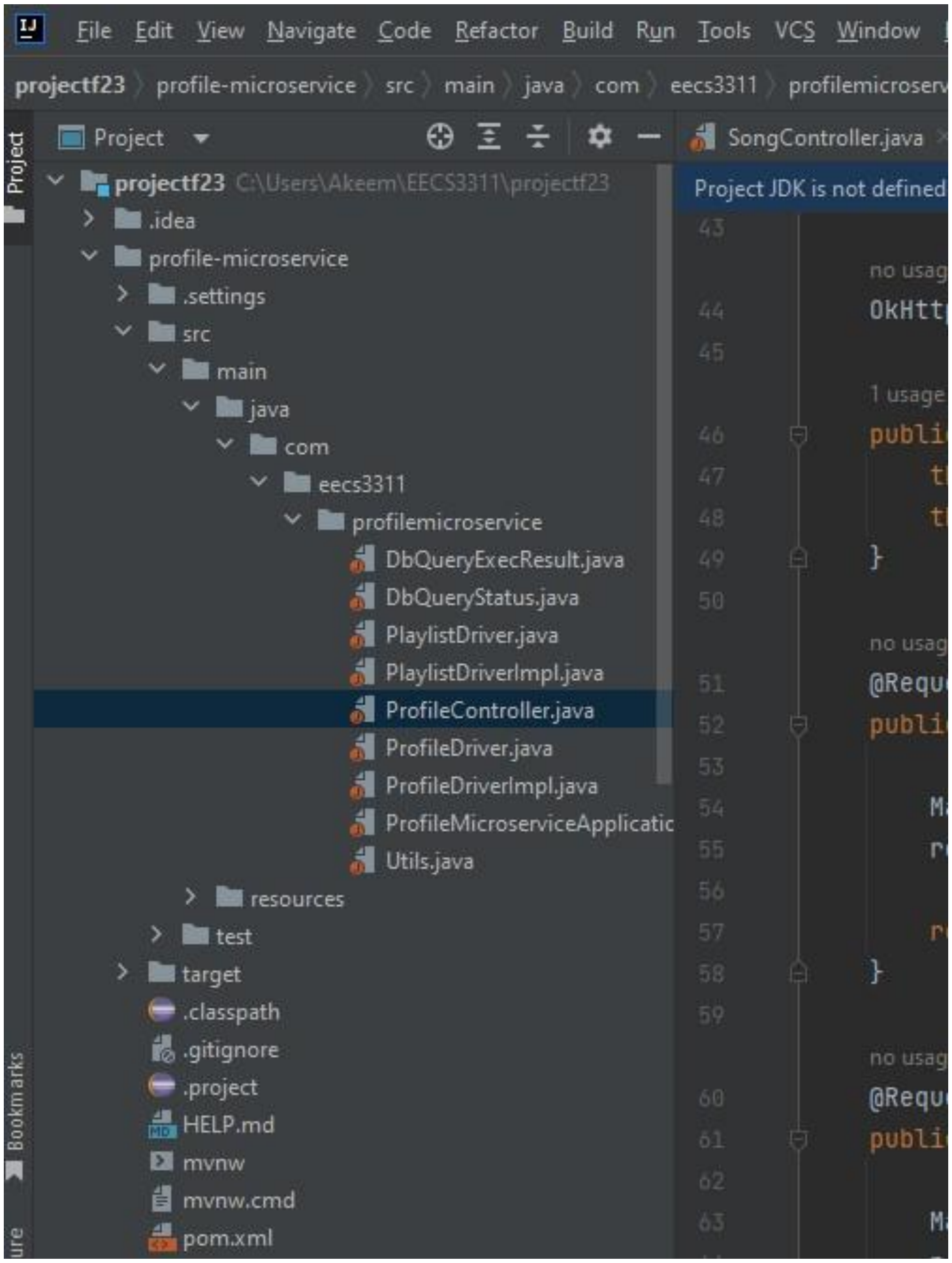
Screenshot showing MongoDB database setup:



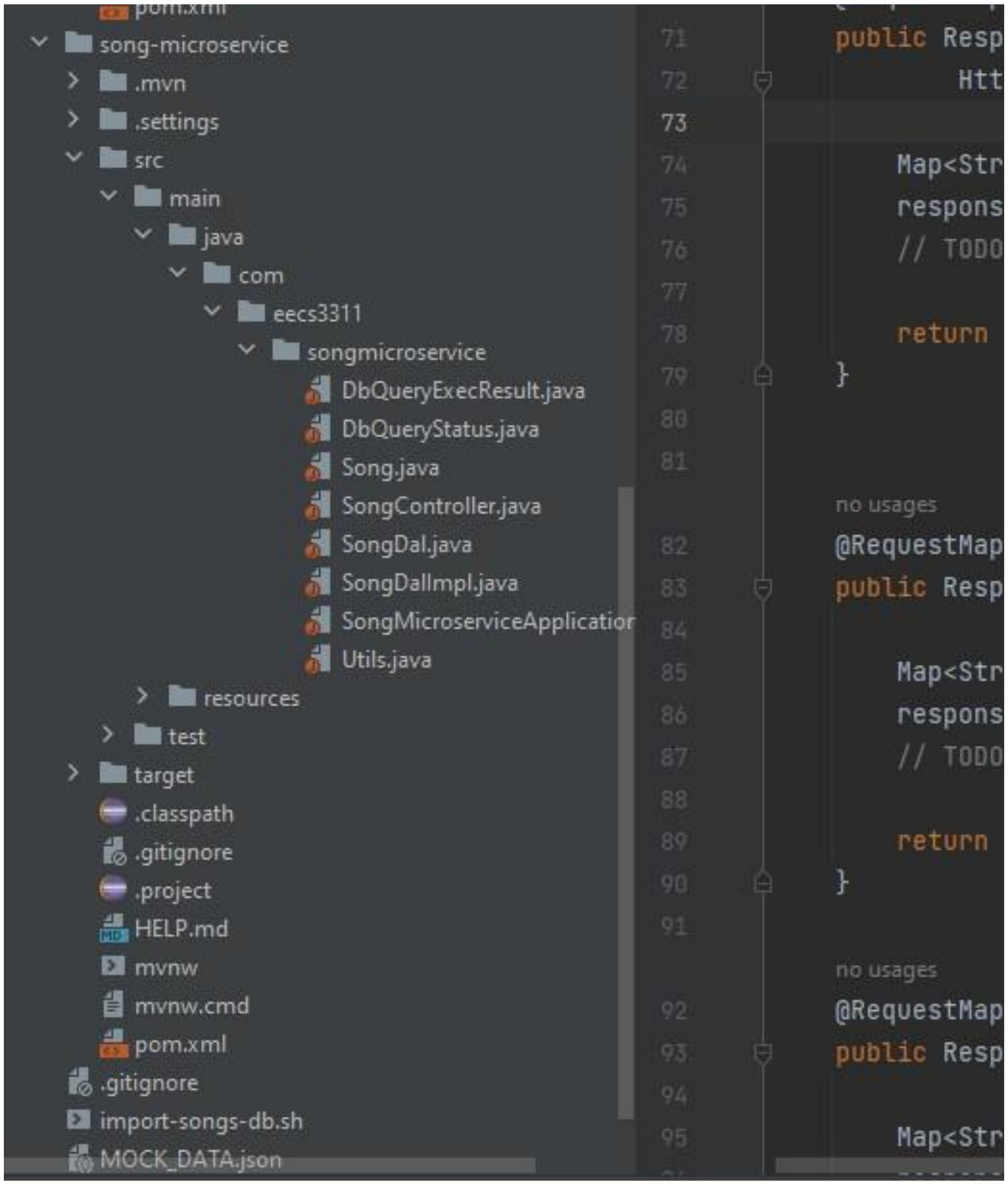
```
_id: ObjectId('5d61728193528481fe5a3124')
songName: "I'm Like a Lawyer with the Way I'm Always Trying to Get You Off (Me & …"
songArtistFullName: "Fall Out Boy"
songAlbum: "Infinity on High"
songAmountFavourites: 54
```

MongoDB server running on port **27017**

IDE Setup: IntelliJ IDE is used for this project



Screenshot above showing project structure for profile-microservice



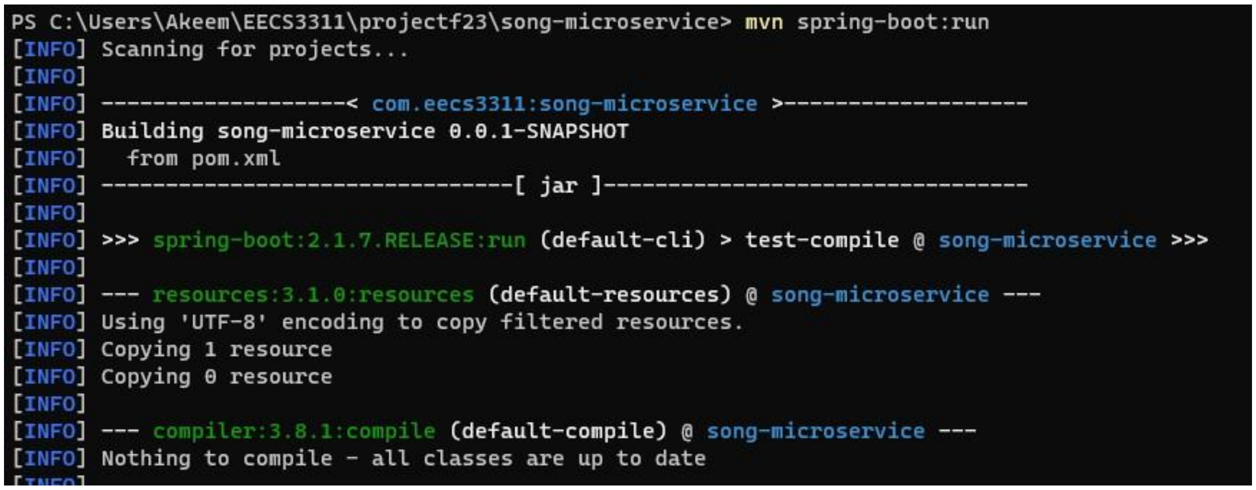
```
pom.xml
song-microservice
  .mvn
  .settings
  src
    main
      java
        com
          eecs3311
            songmicroservice
              DbQueryExecResult.java
              DbQueryStatus.java
              Song.java
              SongController.java
              SongDal.java
              SongDalImpl.java
              SongMicroserviceApplicatio
              Utils.java
      resources
    test
  target
  .classpath
  .gitignore
  .project
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
.gitignore
import-songs-db.sh
MOCK_DATA.json
```

```
public Resp
        Htt

    Map<Str
    respons
    // TODO

    return
}


no usages
@RequestMap
public Resp

    Map<Str
    respons
    // TODO

    return
}

no usages
@RequestMap
public Resp

    Map<Str
```

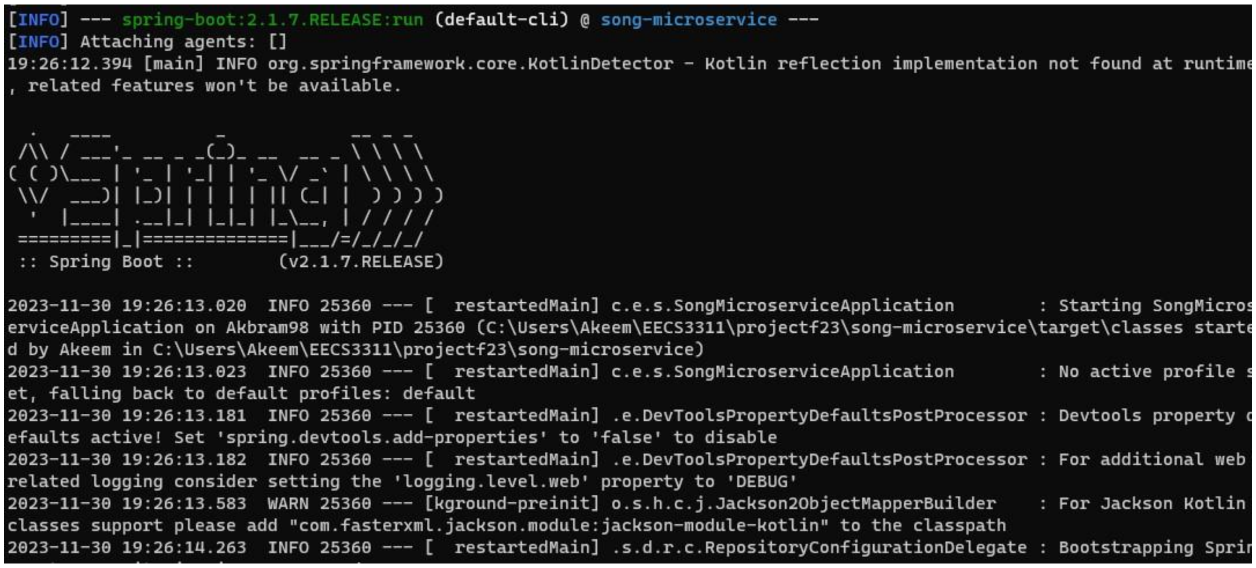
Screenshot above showing project structure for song-microservice

Running maven cmds to deploy microservices:
mvn compile mvn spring-boot:run

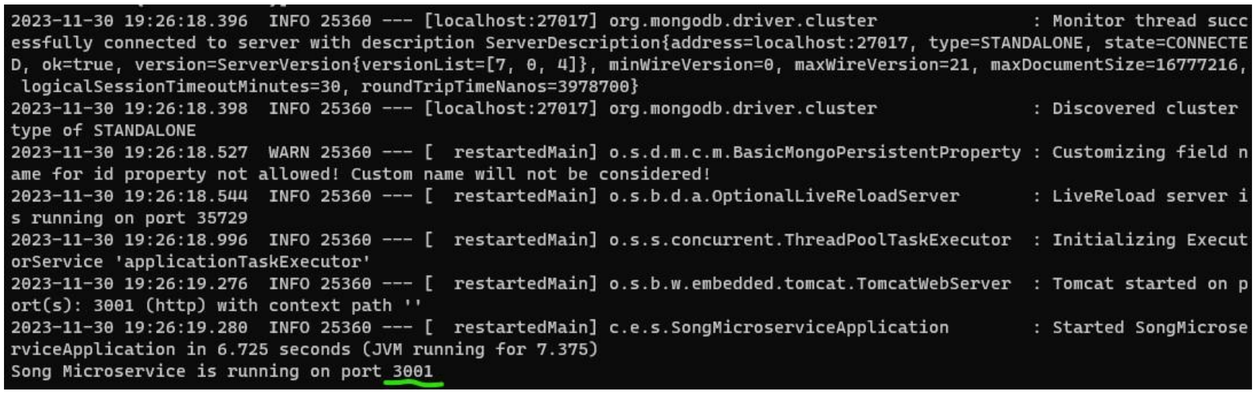
Sample screenshot of song microservice running:



```
PS C:\Users\Akeem\EECS3311\projectf23\song-microservice> mvn spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------< com.eecs3311:song-microservice >-------------------
[INFO] Building song-microservice 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] >>> spring-boot:2.1.7.RELEASE:run (default-cli) > test-compile @ song-microservice >>>
[INFO]
[INFO] --- resources:3.1.0:resources (default-resources) @ song-microservice ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 0 resource
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ song-microservice ---
[INFO] Nothing to compile - all classes are up to date
```

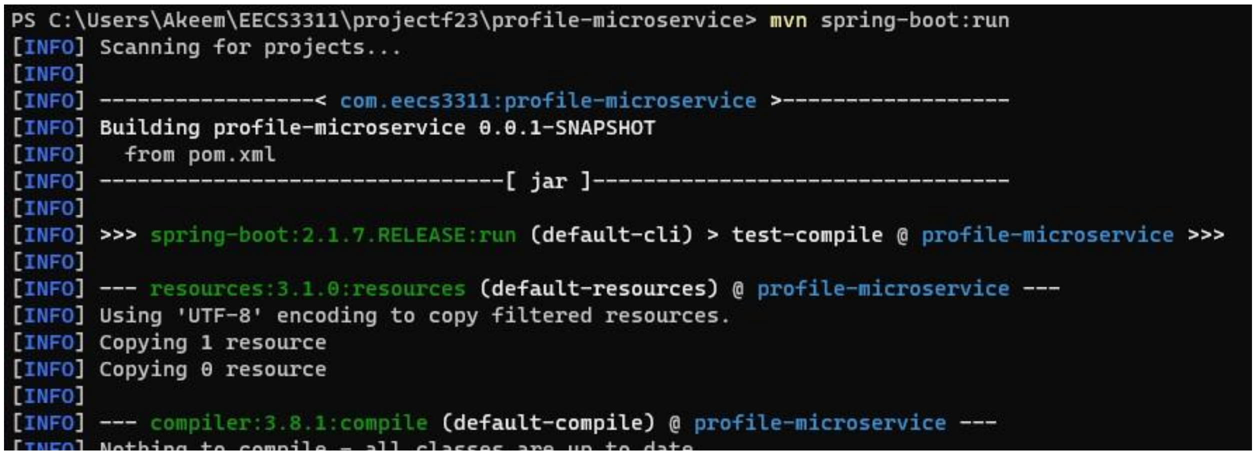


```
[INFO] --- spring-boot:2.1.7.RELEASE:run (default-cli) @ song-microservice ---
[INFO] Attaching agents: []
19:26:12.394 [main] INFO org.springframework.core.KotlinDetector - Kotlin reflection implementation not found at runtime
, related features won't be available.

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.7.RELEASE)

2023-11-30 19:26:13.020  INFO 25360 --- [  restartedMain] c.e.s.SongMicroserviceApplication        : Starting SongMicros
erviceApplication on Akbram98 with PID 25360 (C:\Users\Akeem\EECS3311\projectf23\song-microservice\target\classes starte
d by Akeem in C:\Users\Akeem\EECS3311\projectf23\song-microservice)
2023-11-30 19:26:13.023  INFO 25360 --- [  restartedMain] c.e.s.SongMicroserviceApplication        : No active profile s
et, falling back to default profiles: default
2023-11-30 19:26:13.181  INFO 25360 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property d
efaults active! Set 'spring.devtools.add-properties' to 'false' to disable
2023-11-30 19:26:13.182  INFO 25360 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web
related logging consider setting the 'logging.level.web' property to 'DEBUG'
2023-11-30 19:26:13.583  WARN 25360 --- [kground-preinit] o.s.h.c.j.Jackson2ObjectMapperBuilder    : For Jackson Kotlin
classes support please add "com.fasterxml.jackson.module:jackson-module-kotlin" to the classpath
2023-11-30 19:26:14.263  INFO 25360 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Sprin
```

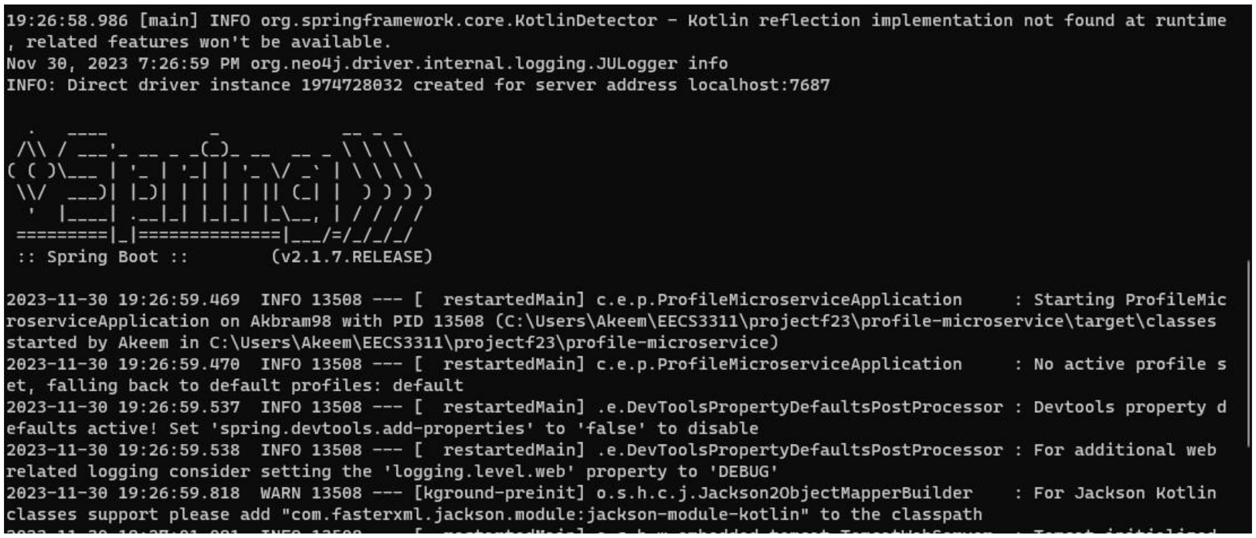


```
2023-11-30 19:26:18.396  INFO 25360 --- [localhost:27017] org.mongodb.driver.cluster               : Monitor thread succ
essfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTE
D, ok=true, version=ServerVersion{versionList=[7, 0, 4]}, minWireVersion=0, maxWireVersion=21, maxDocumentSize=16777216,
 logicalSessionTimeoutMinutes=30, roundTripTimeNanos=3978700}
2023-11-30 19:26:18.398  INFO 25360 --- [localhost:27017] org.mongodb.driver.cluster               : Discovered cluster
type of STANDALONE
2023-11-30 19:26:18.527  WARN 25360 --- [  restartedMain] o.s.d.m.c.m.BasicMongoPersistentProperty : Customizing field n
ame for id property not allowed! Custom name will not be considered!
2023-11-30 19:26:18.544  INFO 25360 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server i
s running on port 35729
2023-11-30 19:26:18.996  INFO 25360 --- [  restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing Execut
orService 'applicationTaskExecutor'
2023-11-30 19:26:19.276  INFO 25360 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on p
ort(s): 3001 (http) with context path ''
2023-11-30 19:26:19.280  INFO 25360 --- [  restartedMain] c.e.s.SongMicroserviceApplication        : Started SongMicrose
rviceApplication in 6.725 seconds (JVM running for 7.375)
Song Microservice is running on port 3001
```

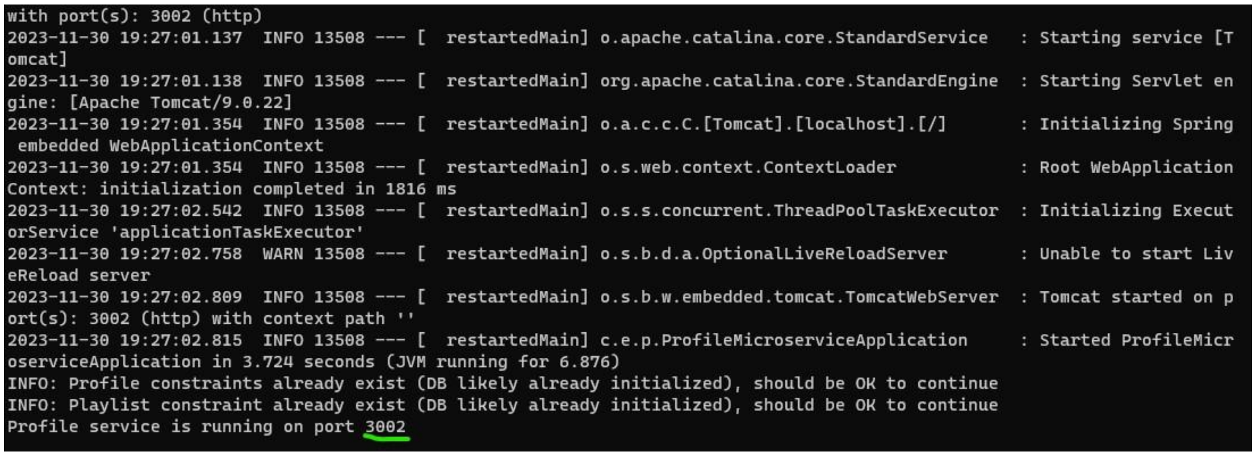
Sample screenshot of profile microservice running:



```
PS C:\Users\Akeem\EECS3311\projectf23\profile-microservice> mvn spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------< com.eecs3311:profile-microservice >------------------
[INFO] Building profile-microservice 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] >>> spring-boot:2.1.7.RELEASE:run (default-cli) > test-compile @ profile-microservice >>>
[INFO]
[INFO] --- resources:3.1.0:resources (default-resources) @ profile-microservice ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 0 resource
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ profile-microservice ---
[INFO] Nothing to compile - all classes are up to date
```



```
19:26:58.986 [main] INFO org.springframework.core.KotlinDetector - Kotlin reflection implementation not found at runtime
, related features won't be available.
Nov 30, 2023 7:26:59 PM org.neo4j.driver.internal.logging.JULogger info
INFO: Direct driver instance 1974728032 created for server address localhost:7687

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.7.RELEASE)

2023-11-30 19:26:59.469  INFO 13508 --- [  restartedMain] c.e.p.ProfileMicroserviceApplication     : Starting ProfileMic
roserviceApplication on Akbram98 with PID 13508 (C:\Users\Akeem\EECS3311\projectf23\profile-microservice\target\classes
started by Akeem in C:\Users\Akeem\EECS3311\projectf23\profile-microservice)
2023-11-30 19:26:59.470  INFO 13508 --- [  restartedMain] c.e.p.ProfileMicroserviceApplication     : No active profile s
et, falling back to default profiles: default
2023-11-30 19:26:59.537  INFO 13508 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property d
efaults active! Set 'spring.devtools.add-properties' to 'false' to disable
2023-11-30 19:26:59.538  INFO 13508 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web
related logging consider setting the 'logging.level.web' property to 'DEBUG'
2023-11-30 19:26:59.818  WARN 13508 --- [kground-preinit] o.s.h.c.j.Jackson2ObjectMapperBuilder    : For Jackson Kotlin
classes support please add "com.fasterxml.jackson.module:jackson-module-kotlin" to the classpath
2023-11-30 19:27:01.001  INFO 13508 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized
```



```
with port(s): 3002 (http)
2023-11-30 19:27:01.137  INFO 13508 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [T
omcat]
2023-11-30 19:27:01.138  INFO 13508 --- [  restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet en
gine: [Apache Tomcat/9.0.22]
2023-11-30 19:27:01.354  INFO 13508 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring
 embedded WebApplicationContext
2023-11-30 19:27:01.354  INFO 13508 --- [  restartedMain] o.s.web.context.ContextLoader            : Root WebApplication
Context: initialization completed in 1816 ms
2023-11-30 19:27:02.542  INFO 13508 --- [  restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing Execut
orService 'applicationTaskExecutor'
2023-11-30 19:27:02.758  WARN 13508 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : Unable to start Liv
eReload server
2023-11-30 19:27:02.809  INFO 13508 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on p
ort(s): 3002 (http) with context path ''
2023-11-30 19:27:02.815  INFO 13508 --- [  restartedMain] c.e.p.ProfileMicroserviceApplication     : Started ProfileMicr
oserviceApplication in 3.724 seconds (JVM running for 6.876)
INFO: Profile constraints already exist (DB likely already initialized), should be OK to continue
INFO: Playlist constraint already exist (DB likely already initialized), should be OK to continue
Profile service is running on port 3002
```