Shamzam Microservices API Design

1. Catalogue Service API

Handles adding, removing, and listing tracks.

Endpoints:

Verb	Resource	Response Codes
POST	`/tracks`	201 (Created), 400 (Bad Request)
DELETE	`/tracks/{id}`	200 (OK), 404 (Not Found)
GET	`/tracks`	200 (OK)
GET	`/tracks/{id}`	200 (OK), 404 (Not Found)
GET	`/tracks?title={title}&artist={artist}`	200 (OK), 404 (Not Found)

Example Requests & Responses

1. Add a Track

Request:

```
POST /tracks
{
  "title": "Blinding Lights",
  "artist": "The Weeknd",
  "file_path": "/audio/blinding_lights.mp3"
}
```

Response:

```
{
  "id": 1,
  "message": "Track added successfully"
}
```

2. Get All Tracks

Request:

```
GET /tracks
```

Response:

```
[
  { "id": 1, "title": "Blinding Lights", "artist": "The Weeknd" },
  { "id": 2, "title": "Don't Look Back in Anger", "artist": "Oasis" }
]
```

2. Audio Recognition Service API

Handles matching a music fragment to a track.

Endpoints:

Verb	Resource	Response Codes
POST	`/recognise`	200 (OK), 400 (Bad Request)

Example Requests & Responses

1. Recognise Audio Fragment

Request:

```
POST /recognise
{
  "file_path": "/uploads/fragment.wav"
}
```

Internal API Call to Audd.io:

```
POST https://api.audd.io/
{
  "api_token": "your_api_key",
  "audio": "/uploads/fragment.wav"
}
```

Response from Audd.io:

```
{
  "result": {
    "title": "Blinding Lights",
    "artist": "The Weeknd"
  }
}
```

Audio Recognition Service queries Catalogue Service to check if track exists:

```
GET /tracks?title=Blinding Lights&artist=The Weeknd
```

Final Response:

```
{
  "track_id": 1,
  "title": "Blinding Lights",
  "artist": "The Weeknd",
  "file_path": "/audio/blinding_lights.mp3"
}
```

3. Database Service API (Internal Only)

Handles direct database interactions.

Endpoints:

Verb	Resource	Response Codes
POST	`/db/tracks`	201 (Created), 400 (Bad Request)
DELETE	`/db/tracks/{id}`	200 (OK), 404 (Not Found)
GET	`/db/tracks`	200 (OK)
GET	`/db/tracks?title={title}&artist={artist}`	200 (OK), 404 (Not Found)

Example Requests & Responses

1. Add a Track to Database (Internal)

Request:

```
POST /db/tracks
{
  "title": "Blinding Lights",
  "artist": "The Weeknd",
  "file_path": "/audio/blinding_lights.mp3"
}
```

Response:

```
{
  "id": 1,
  "message": "Track added to database"
}
```

4. Microservice Communication Flow

Adding/Removing Tracks

1. User sends **POST/DELETE** request to **Catalogue Service**.
2. **Catalogue Service** forwards the request to **Database Service**.
3. **Database Service** updates the SQLite database and responds.
4. **Catalogue Service** confirms the operation to the user.

Retrieving a Track from a Snippet

1. User uploads an audio fragment to **Audio Recognition Service**.
2. **Audio Recognition Service** sends the fragment to **Audd.io**.
3. **Audd.io** responds with track details (title, artist).
4. **Audio Recognition Service** queries **Catalogue Service** using the track title and artist.
5. **Catalogue Service** asks **Database Service** for track details using the title and artist.
6. **Database Service** returns track details.
7. **Catalogue Service** sends final response to **Audio Recognition Service**, which then responds to the user.