ECM3408 Enterprise Computing – Endpoint Designs

James Thomas

February 2025

1 AudD Microservice

The AudD microservice (`audd.py`) provides a service to be used for querying song fragments. The service is used by the User microservice to obtain information about songs and song fragments. AudD is separated as its own microservice to ensure loose coupling and provide the ability to easily exchange the audio recognition engine if another service is chosen in later versions.

1.1 POST /audd

The AudD `POST` endpoint accepts WAV64 encoded song files and responds with the song information, if the song is in the AudD database, otherwise an error code and corresponding message. Details of the endpoint for a successful request can be seen in Figure 1; details of an unsuccessful request can be seen in Figure 2.

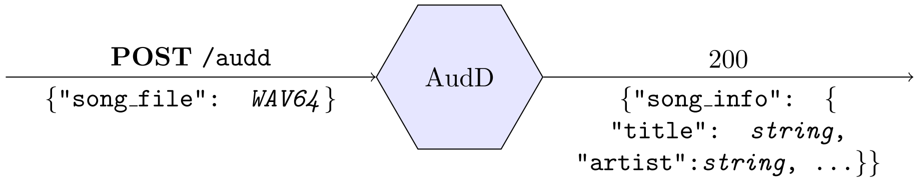

Figure 1: The AudD audio recognition engine endpoint – Success Case

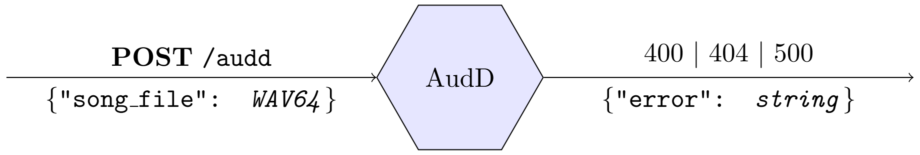

Figure 2: The AudD audio recognition engine endpoint – Failure Case

2 Shamzam Admin Microservice

The Admin microservice (`shamzam_admin.py`) provides the Admin functionality for the Shamzam application. The service has three endpoints allowing for addition and removal of songs, as well as querying what songs are stored in the Shamzam database. This microservice is split from the User microservice to ensure durability and access for the service, but the services share a database to ensure data consistency.

2.1 POST /admin/songs

The Admin `POST` endpoint allows for the addition of songs to the Shamzam database. The songs are accepted in a WAV64 encoded format, with accompanying string fields for the artist and title of the song. It is important that *the song title and artist must match that of the audio engine*. If the title and artist provided match a pre-existing song, the stored song will be updated to the new provided song; preventing duplication to bolster data integrity. This ensures data is consistent, accurate, and that songs cannot overwrite each other. This endpoint provides functionality for user story *S1*. Details of the endpoint for a successful request can be seen in Figure 3; details of an unsuccessful request can be seen in Figure 4.

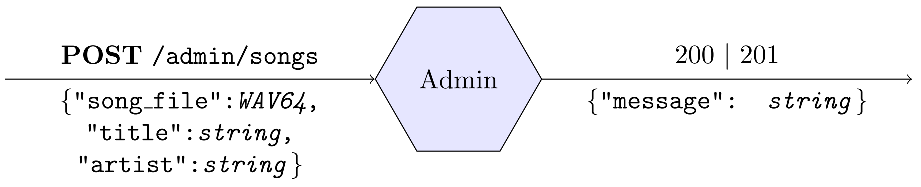

Figure 3: The Shamzam admin song adding endpoint – Success Case

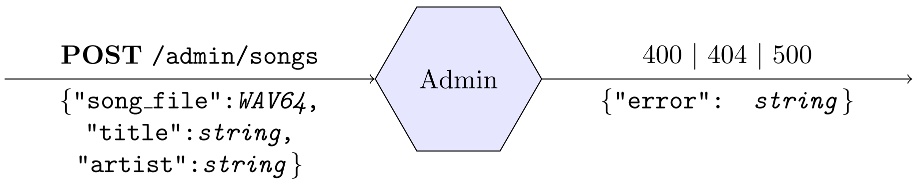

Figure 4: The Shamzam admin song adding endpoint – Failure Case

2.2 DELETE /admin/songs

The Admin `DELETE` endpoint allows songs to be removed by providing JSON containing the song title and artist. Songs can only be removed when *perfectly matching title and artists are provided.* The endpoint requires *both* title and artist of a song to be provided to ensure the correct version of a song is removed. This endpoint provides functionality for user story *S2*. Details of the endpoint for a successful request can be seen in Figure 5; details of an unsuccessful request can be seen in Figure 6.

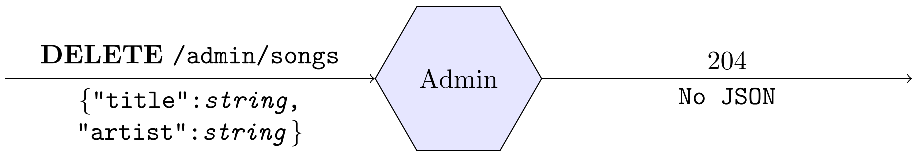

Figure 5: The Shamzam Admin song removal endpoint – Success Case

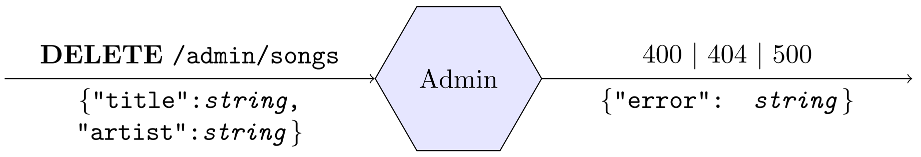

Figure 6: The Shamzam Admin song removal endpoint – Failure Case

2.3 GET /admin/songs

The Admin `GET` endpoint responds with JSON containing a list of lists of all song title-artist pairs within the Shamzam database currently. Each sublist is of the format `[title,artist]`. The list is sorted *alphabetically* on the song artist, to group songs by the artist. This endpoint provides functionality for user story *S3*. Details of the endpoint for a successful request can be seen in Figure 7; details of an unsuccessful request can be seen in Figure 8.

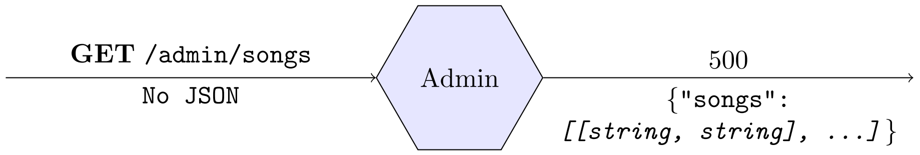

Figure 7: The Shamzam Admin song title listing endpoint – Success Case

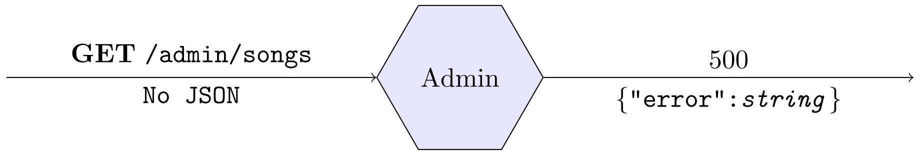

Figure 8: The Shamzam Admin song title listing endpoint – Failure Case

3 User Microservice

The User microservice (`shamzam_user.py`) provides the User functionality for the Shamzam application. The User microservice is split from the Admin microservice to ensure the services' availability is distinct. Even if Admin fails, Users can use the application, and vice versa.

3.1 POST /user

The User `POST` endpoint allows users to request the full version of a song from the Shamzam database. If the song fragment given matches a song in the Shamzam database, the JSON response contains the WAV64, as well as basic information about the song. If the song fragment matches a song in the AudD database but not the Shamzam database, a message conveys this and the basic information on the song is still provided, ensuring users still gain functionality from the application even if their requested song is not in the Shamzam database: this outcome is shown in Figure 10. This endpoint provides functionality for user story *S4*. Details of the endpoint for a successful request can be seen in Figure 9; details of unsuccessful requests can be seen in Figures 10 and 11.

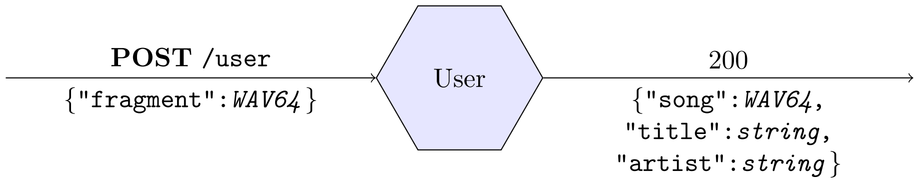

Figure 9: The Shamzam user query endpoint – Success Case

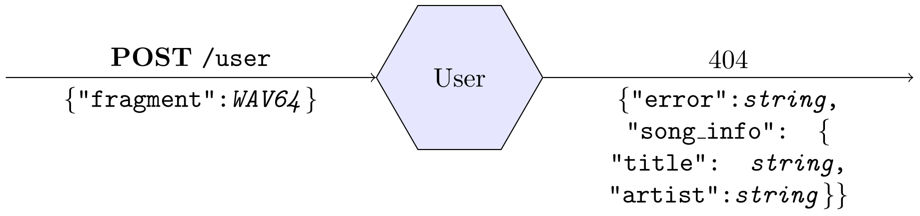

Figure 10: The Shamzam user query endpoint – Recognised but not in DB

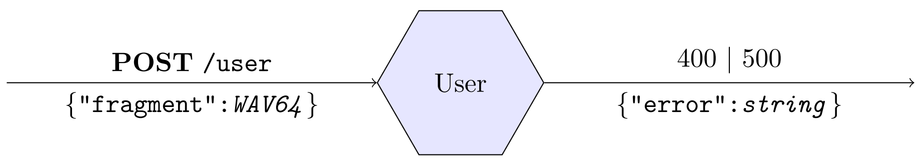

Figure 11: The Shamzam user query endpoint – Failure Case