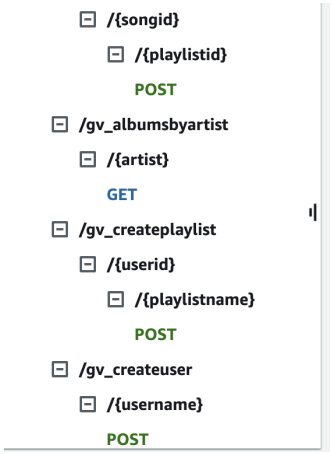
Gillian Gracey, Annabel Edwards, Varoon Enjeti, Jenna Kopp
CS 310 - Prof. Hummel
Final Project Write Up - GrooveVault

Description:

For our final project, we decided to create a music app that would store songs, users, and playlists. GrooveVault is a music app designed to allow users to make and edit playlists. Our group created a client-side Python application with AWS Lambda functions and API Gateway.

Server-Side using Lambda nad API Gateway:

On the server side, GrooveVault uses AWS services, using Lambda functions for serverless computing and API Gateway for request routing. The Lambda functions provide a flexible and scalable solution for processing user requests and managing database interactions without the need for dedicated server management. These functions are triggered by events from the client and then routed through API Gateway, which manages and directs traffic to the corresponding Lambda function. This setup ensures efficient, secure, and cost-effective backend operations, allowing GrooveVault to handle varying user loads. This is an example of what some functions in our API look like and a sample lambda function that is uploading a song to S3.




```
#
# generate unique filename in preparation for the S3 upload:
#
print("**Uploading local file to S3**")

basename = pathlib.Path(og_filename).stem
extension = pathlib.Path(og_filename).suffix

if extension != ".mp3" :
  raise Exception("expecting og_filename to have .mp3 extension")

bucketkey = "groovevault/" + basename + "-" + str(uuid.uuid4()) + ".mp3"

print("S3 bucketkey:", bucketkey)

#
# add a jobs record to the database BEFORE we upload, just in case
# the compute function is triggered faster than we can update the
# database:
#
print("**Adding jobs row to database**")

sql = """
  INSERT INTO songs(songname, artist, album, originalsongfile, songfilekey)
                VALUES(%s, %s, %s, %s, %s);
"""

datatier.perform_action(dbConn, sql, [songname, artist, album, og_filename, bucketkey])

#
```

Database in RDS:

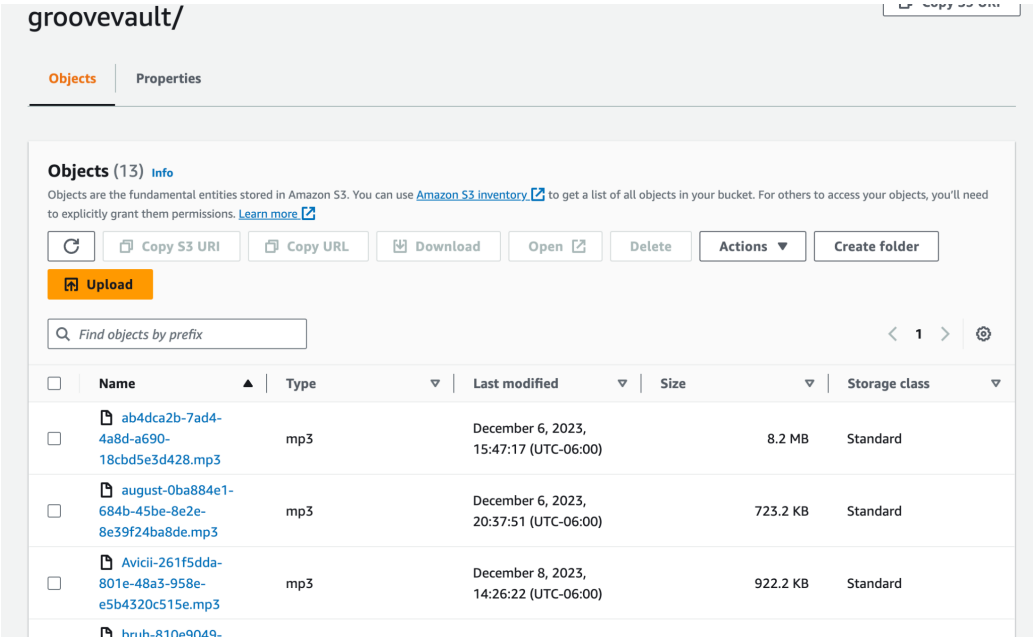
The database is managed using RDS through AWS services. Using SQL with tables for users, songs, playlists, and playlist-songs, we populated the DB. This setup ensures efficient data handling and integrity. Our SQL code creates a comprehensive structure for the GrooveVault database. It includes tables for users, songs, and playlists each with a primary id key and then another table called playlistsongs with two foreign keys attaching songs to playlists - each table with its specific fields and constraints to ensure data consistency and reliability.

```
CREATE TABLE users
(
    userid       int not null AUTO_INCREMENT,
    username     varchar(64) not null,
    PRIMARY KEY  (userid),
    UNIQUE       (username)
);

ALTER TABLE users AUTO_INCREMENT = 1;  -- starting value

CREATE TABLE songs
(
    songid       int not null AUTO_INCREMENT,
    songname     varchar(64) not null,
    artist       varchar(64) not null,
    album        varchar(64) not null,
    originalsongfile    varchar(256) not null,  -- original name from user
    songfilekey         varchar(256) not null,  -- filename in the bucket
    PRIMARY KEY  (songid),
    UNIQUE       (songfilekey)
);

ALTER TABLE songs AUTO_INCREMENT = 1;  -- starting value

CREATE TABLE playlists
(
    playlistid       int not null AUTO_INCREMENT,
    playlistname     varchar(256) not null,
```

```
ALTER TABLE songs AUTO_INCREMENT = 1;  -- starting value

CREATE TABLE playlists
(
    playlistid       int not null AUTO_INCREMENT,
    playlistname     varchar(256) not null,
    userid           int not null,
    PRIMARY KEY (playlistid),
    FOREIGN KEY (userid) REFERENCES users(userid)
);

ALTER TABLE playlists AUTO_INCREMENT = 1;  -- starting value

CREATE TABLE playlistsongs
(
  playlistid         int not null,
  songid             int not null,
  FOREIGN KEY (playlistid) REFERENCES playlists(playlistid),
  FOREIGN KEY (songid) REFERENCES songs(songid)
)

--
-- Insert some users to start with:
--
-- PWD hashing: https://phppasswordhash.com/
--
```

GrooveVault Bucket in S3:

GrooveVault uses AWS S3 for storing songs, utilizing its scalability and security for efficient data management. Song uploads are processed through AWS Lambda functions, which handle validation and transfer to the S3 bucket. We chose Amazon S3 for storing objects in GrooveVault due to its high durability, scalability, and secure data hosting capabilities, ensuring reliable and efficient management of our music content.



Client-Side:

The client-side of GrooveVault is built using Python in Replit, offering a text-based interface where users can interact with various features of the app. Users can create a user profile and a playlist by creating a playlist and then adding songs to it. The application allows users to search for songs by artist, album, or in each playlist.

```
>> Enter a command:
   0 => end
   1 => get user's playlists
   2 => songs on playlist
   3 => songs by artist
   4 => albums by artist
   5 => songs in album
   6 => song stats in DB
   7 => playlist stats in DB
   8 => download and play playlist
   9 => download and play song
   10 => add song to playlist
   11 => create user
   12 => create playlist
   13 => upload song to library
   14 => get users
3
Enter artist>
Taylor Swift
Taylor Swift has the following songs:
  Song 1: August
  Song 2: The 1
  Song 3: Evermore
  Song 4: Hoax
```

```
>> Enter a command:
   0 => end
   1 => get user's playlists
   2 => songs on playlist
   3 => songs by artist
   4 => albums by artist
   5 => songs in album
   6 => song stats in DB
   7 => playlist stats in DB
   8 => download and play playlist
   9 => download and play song
   10 => add song to playlist
   11 => create user
   12 => create playlist
   13 => upload song to library
   14 => get users
2
Enter playlistid>
80009
Playlist 80009 has the following songs:
  Song 1: Evermore
  Song 2: Driver's License
  Song 3: Hoax
```

```
>> Enter a command:
   0 => end
   1 => get user's playlists
   2 => songs on playlist
   3 => songs by artist
   4 => albums by artist
   5 => songs in album
   6 => song stats in DB
   7 => playlist stats in DB
   8 => download and play playlist
   9 => download and play song
   10 => add song to playlist
   11 => create user
   12 => create playlist
   13 => upload song to library
   14 => get users
4
Enter artist>
Taylor Swift
Artist Taylor Swift has the following albums (# of times it appear in the Database):
  Album 1: Folklore
  Album 2: Folklore
  Album 3: Evermore
  Album 4: Folklore

>> Enter a command:
   0 => end
   1 => get user's playlists
   2 => songs on playlist
   3 => songs by artist
```

Users can also find the amount of songs in the database, the playlist stats, and the users in the database so far.

```
Playlist ID: 80005
  Playlist Name: emo era
  User ID: 2

Playlist ID: 80006
  Playlist Name: test1
  User ID: 1

Playlist ID: 80007
  Playlist Name: taylor
  User ID: 2

Playlist ID: 80008
  Playlist Name: Taylor
  User ID: 7

Playlist ID: 80009
  Playlist Name: Sad Girl
  User ID: 4
```

```
Song ID: 9
  Song Name: Evermore
  Artist Name: Taylor Swift
  Album Name: Evermore
  Song File Name: evermore.mp3

Song ID: 10
  Song Name: Driver's License
  Artist Name: Olivia Rodrigo
  Album Name: Sour
  Song File Name: drivers_license.mp3

Song ID: 11
  Song Name: Hoax
  Artist Name: Taylor Swift
  Album Name: Folklore
  Song File Name: hoax.mp3
```

Lastly, users can upload songs (up to 1 minute) to the S3 bucket where they are stored (which also then enter the database). We recognized that the API gateway has a 10 MB limit of how much data we can transmit across the server, so we had to reduce our original vision of playing entire songs and instead just playing clips. However, the user can still download and play songs or play the entire playlist directly through the client application. This integration provides a convenient listening experience on the client side.

```
>> Enter a command:
   0 => end
   1 => get user's playlists
   2 => songs on playlist
   3 => songs by artist
   4 => albums by artist
   5 => songs in album
   6 => song stats in DB
   7 => playlist stats in DB
   8 => download and play playlist
   9 => download and play song
   10 => add song to playlist
   11 => create user
   12 => create playlist
   13 => upload song to library
   14 => get users
8
Enter playlistid>
80008
Playing The 1 by Taylor Swift
Enter p to pause. Enter s to skip song>
```

```
>> Enter a command:
   0 => end
   1 => get user's playlists
   2 => songs on playlist
   3 => songs by artist
   4 => albums by artist
   5 => songs in album
   6 => song stats in DB
   7 => playlist stats in DB
   8 => download and play playlist
   9 => download and play song
   10 => add song to playlist
   11 => create user
   12 => create playlist
   13 => upload song to library
   14 => get users
13
Enter MP3 filename>
Avlcli.mp3
Enter song's name>
Wake me up
Enter the song's artist>
Avlcli
Enter song's album>
Wake me up
MP3 uploaded, song id = 12
```