Microscaled Spotify Design Development framework
Barebones:

- Audio Delivery
- Metadata management
- Object Storage*

Requirements & Considerations:
(On a microscale)
100 songs
50 users

Core Requirements:
Artist upload their songs
Users can search and play songs
Users can create and manage playlists
Users can maintain profiles
Basic monitoring and observability (health checks, error tracking, performance metrics)

Audio formats:
Ogg and AAC files with different bitrates for adaptive streaming
- 64kbps for mobile data saving
- 128kbps for standard quality
- 320kbps for premium users
- Average for one song at normal audio quality (standard bitrate) takes 3MB of storage

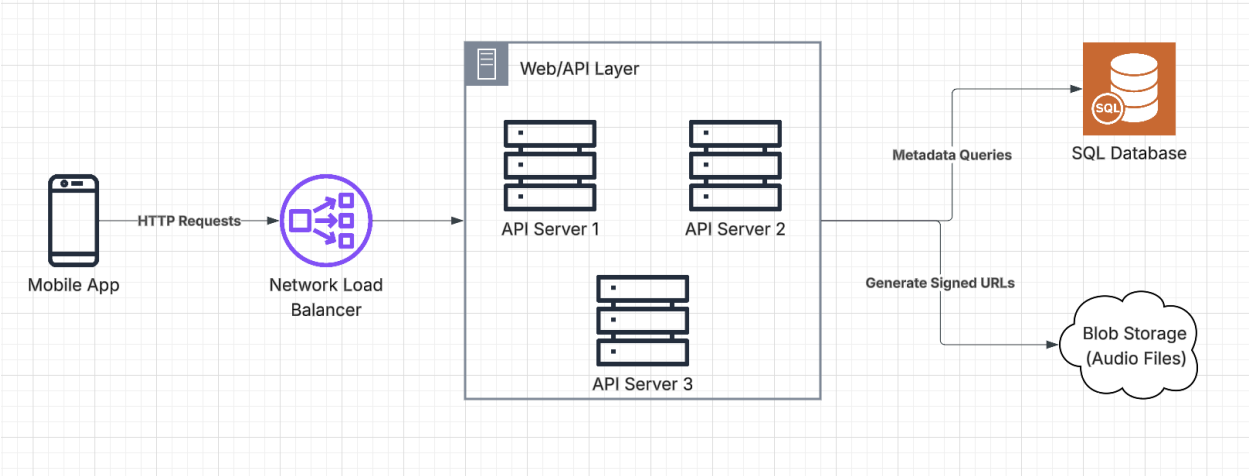
Capacity Planning:
- 3MB x 100 songs = 300MB of raw audio data
- Does not include replicas across different regions
- Does not include versioning overhead when artists re-upload songs
- With features not included we instead increase the amount by 2-3x

User metadata:
- User profiles, preferences, and playlist data are: roughly 1 KB x 50 users = 50KB

Song metadata:
- Each song needs a title, artist references, duration, file URLs etc.
- Roughly 100 bytes per song x 100 songs = ~10KB
- Very small compared to the actual audio data size

Daily bandwidth:
- Average listening time is 3.5min at 128-160kbps; roughly 3-4MB per stream
- Assuming each user streams 10-15 songs daily
- Leads to significant egress costs (How much it would cost to transfer data out of AWS/Azure)

High Level Architecture Overview (Made with lucidchart)



Mobile App
- Requires a dev team to prod team.
- Role delegations required to Mobile app development

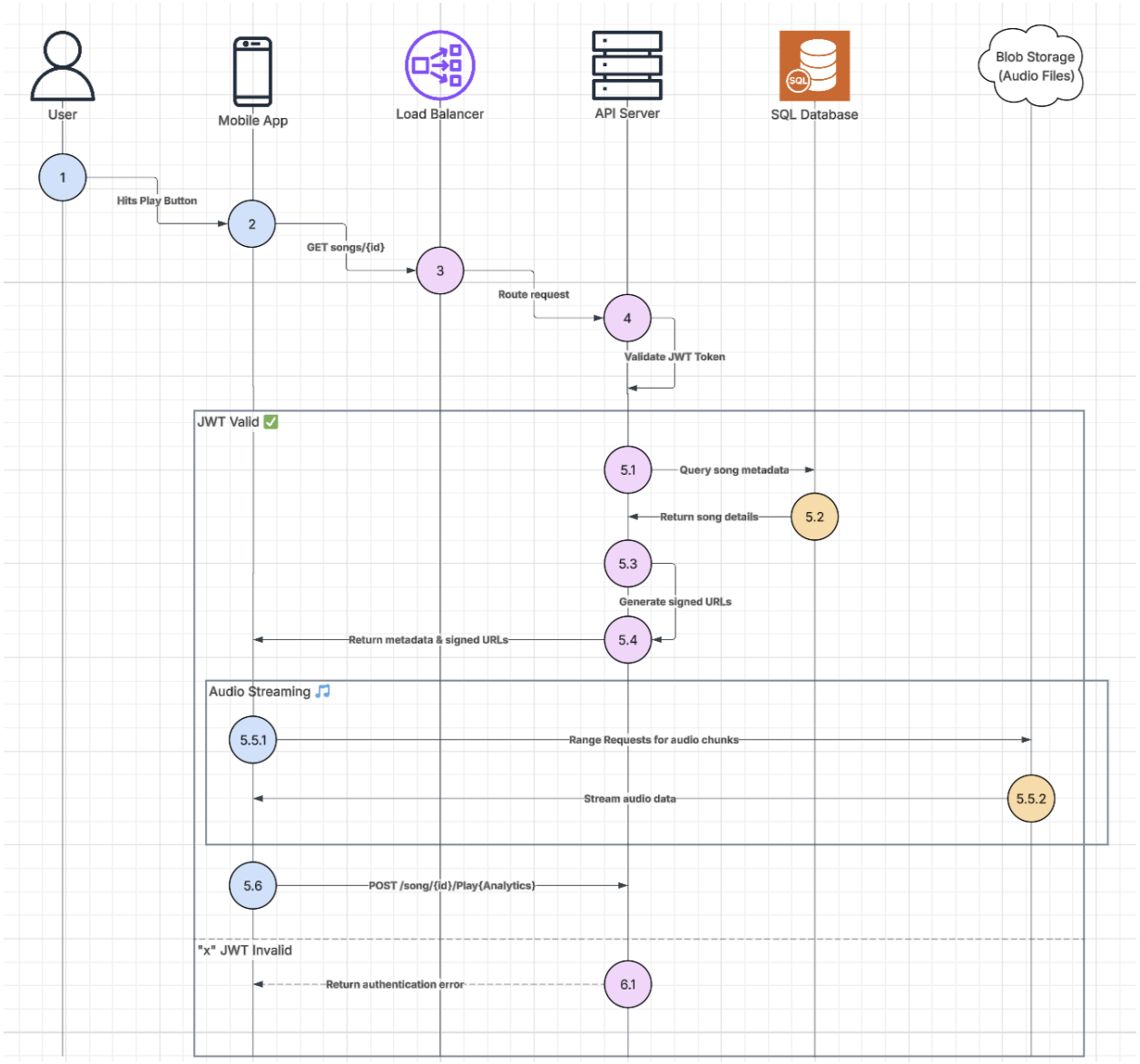
Load Balancer
- User either Round Robin or List Connections Algorithm
- Run Health Checks on the API Servers for High Availability

API Servers
- Restful APIs
- Use JWT tokens to query metadata of a song and urls

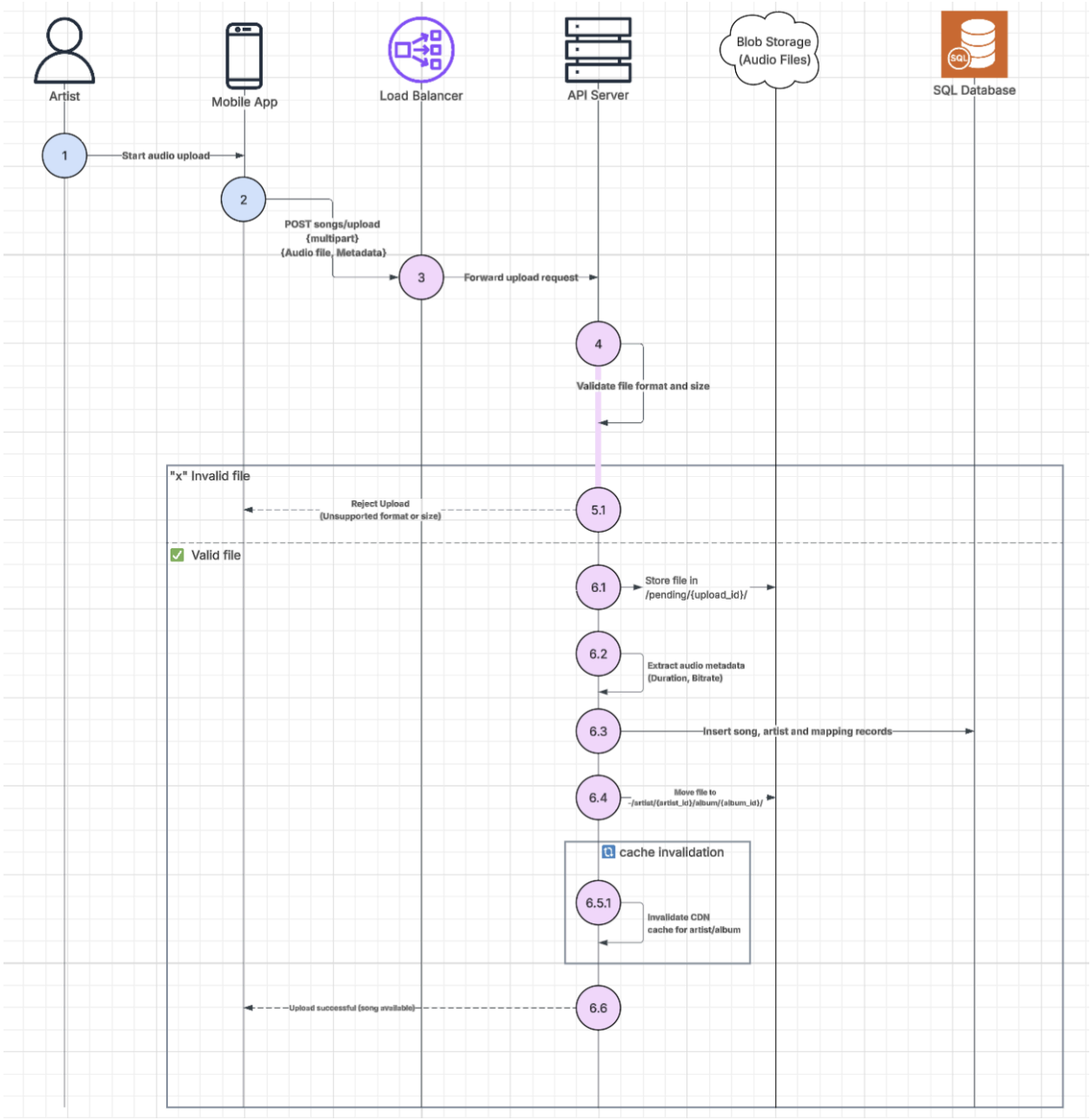
SQL Database
- Song metadata + URLs

Blob Storage (AWS S3)
- Holds all audio files (Largest system file size)
- Queries generated signed URLs (urls that expire after a set of time & differs)
- Unlimited Scalability, built in durability and cost effectiveness

User System Workflow (Lucidchart)



Artist System Workflow (lucid chart)



API Design

(REST API URL endpoints)
Search & Discovery
- Search different content types with pagination:
GET /search?q={query}&type=song,artist&limit=20&offset=0
- Get trending songs, optionally filtered by genre
GET /songs/trending?genre={genre}&limit=50
- Get all songs by a specific artist
GET /artists/{id}/songs?limit=50

Content Access

- Get song metadata and streaming URL
GET /songs/{id}
- Direct streaming endpoint (alternative to signed URLs)
GET /songs/{id}/stream
- Get playlist details with an optional song list
GET /playlists/{id}?include_songs=true

User Actions

- Create a new playlist
POST /playlists
{
“name”: “My Favorites”,
“Is_public”: false
}

- Add songs to the playlist
PUT /playlists/{id}/songs
{
“Songs_id”: [123, 456, 789],
“position”: 5
}

- Remove a song from playlist
DELETE
/playlist/{id}/songs/{song_id}

- Like/unlike a song
POST /songs/{id}/like

User management

- Get the current user’s playlist
GET /users/me/playlists

- Get songs liked by the user
GET /users/me/liked-songs?
limit=50&offset=0

- Follow an artist
POST /users/me/follow/{artist_id}

Data storage:

S3

The system uses the blob storage to store audio files.

- The audio files are immutable; they rarely change once uploaded.
- We'd organize them with a sensible folder structure
- /artist/{artistId}/album/{albumId}/{songId}.ogg

Relational Database

playlistitems

PlalylistID	bigint pk
Position	int
AddedAt	timestamptz
SongID	bigint plk

playlists

PlaylistID	bigintpk
OwnerID	bigint
Name	text
Created At	timestamptz

users

UserID	bigint pk
Email	citext unique
Password Hash	text
Created At	timestamptz
Last Login	timestamptz
SubscriptionType	text
Country	text

artistsongs

SongID	bigint pk
ArtistID	bigint pk

songs

SongID	bigint pk
Title	text
Duration	int
Release Date	date
FileURL	text
CreatedAt	timestamptz
ArtistID	bigint

artists

ArtistID	bigint pk
Name	text
Country	text
Bio	text
ImageURL	text
CreateAt	timestamptz not null
UpdatedAt	timestamptz not null

~Scalability: