**Frontend (React)**

React.js – UI rendering

React Router – Routing

Redux – State management

Socket.IO-client – Real-time updates

Web Audio API or Tone.js – Audio effects/visualization

**Backend (Node.js + Express)**

Node.js – Server runtime

Express.js – RESTful API framework

Socket.IO – Real-time communication (comments, live duets)

fluent-ffmpeg – Audio/Video encoding and processing

**Databases**

MongoDB – Core NoSQL database (recordings, users, comments)

**Machine Learning & Feeds**

TensorFlow.js – Basic ML inference on backend

Python + TensorFlow or scikit-learn

**1. User Opens App / Logs In**

User enters credentials → POST /api/auth/login

Validates credentials

Issues **JWT token**

MongoDB → users collection

**2. User Records a Song (Solo or Duet)**

Uses **Web Audio API** or **WebRTC** to capture audio/video

Shows real-time effects (Auto-tune, pitch using Tone.js or AudioContext)

Uploads via multipart/form-data → POST /api/media/upload

**3. User Waits or Browses Feed**

React fetches GET /api/feed

Receives list of songs (latest or recommended)

MongoDB aggregation or Redis cache

Optional ML scoring using TensorFlow.js

**4. User Likes / Comments / Follows**

Like: POST /api/social/like

Comment: POST /api/social/comment

Follow: POST /api/social/follow

Update MongoDB

Publishes event → Sends real-time updates via Socket.IO

**Socket.IO**:

Other users receive notifications in real time

**5. User Joins a Duet**

Loads original video via signed S3 URL

Starts duet session using **WebRTC (PeerJS)**

**6. Playback for Others**

Fetches media URL → plays .m3u8 with **HLS.js**

Cloudflare/Vercel serves content quickly worldwide

**7. Recommendation Engine**

Analyze user interactions (likes, follows, time spent)

Compute recommendations

Stores results in MongoDB

**8. User Gets Notifications**

Via **WebSocket (Socket.IO)** or polling every X seconds

New likes/comments shown in UI in real-time