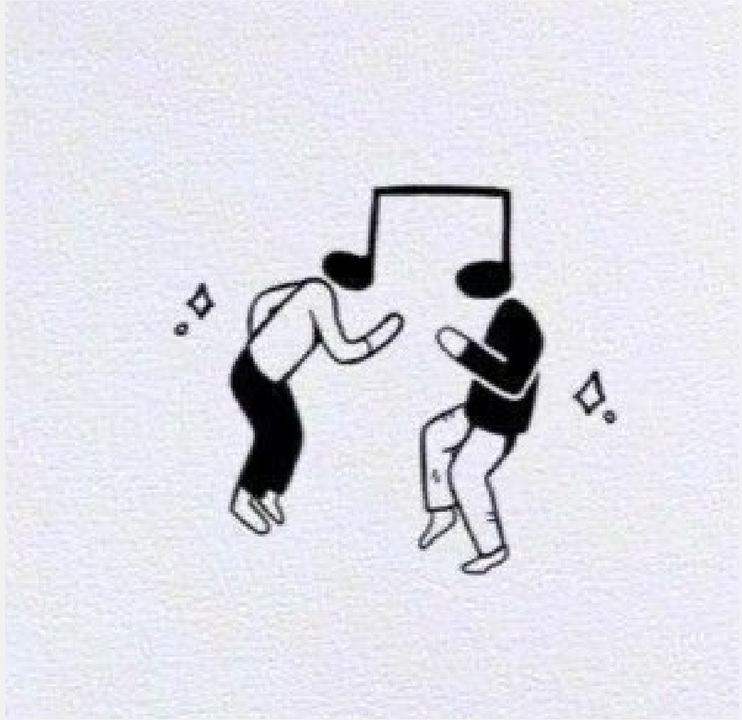
09 MAY 2025 SRI SHASHANK KATARI PROF. MIHAELA SABIN INFORMATION TECHNOLOGY

BEATS & CHATS: INTEGRATING SPOTIFY AND YOUTUBE APIS WITH REACT & FIREBASE

GOAL

The goal of this study was to explore how cloud web technologies and serverless architecture can be combined to build a socially interactive music platform. It focused on learning full-stack development by integrating APIs, real-time databases, and responsive frontend design.



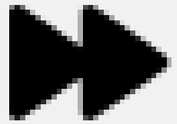
IDEA

- Inspired by group listening & movie sessions
- Wanted social + music together
- No chat in Spotify
- Aimed for fun, shared experience
- Not live, but still connected

PROJECT OBJECTIVES

LEARN API AUTHENTICATION AND THIRD-PARTY DATA INTEGRATION

APPLY FRONTEND FRAMEWORKS FOR CLEAN UI AND STATE HANDLING

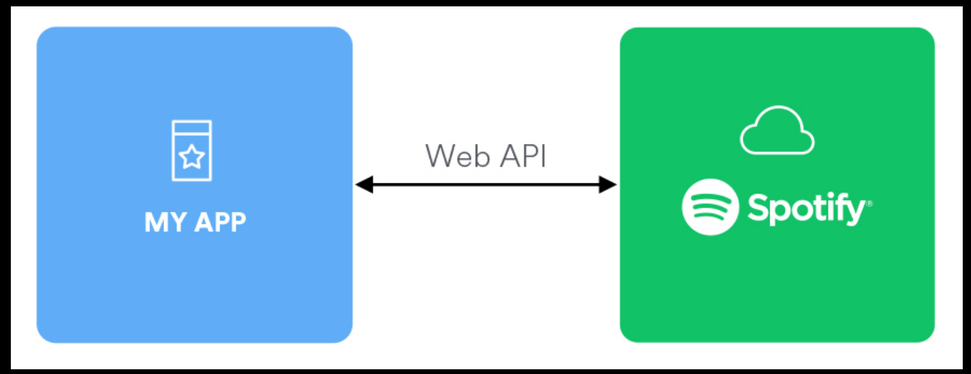


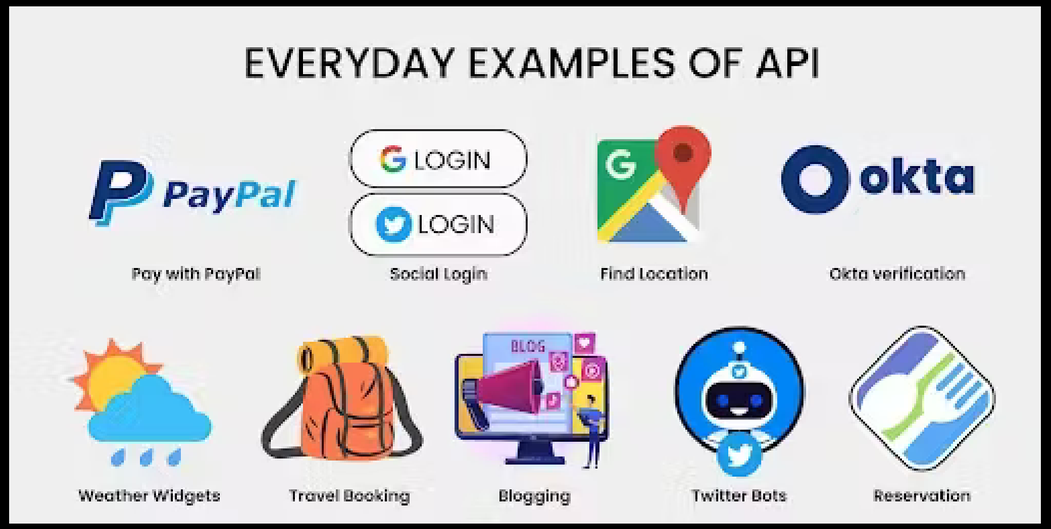
SOLVE PLAYBACK LIMITATIONS

DESIGN REAL-TIME CLOUD DATABASE:

ENSURE MULTI-USER EXPERIENCE AND INTERACTION

PRACTICE SERVERLESS ARCHITECTURE:





WHAT ARE APIS AND WHY THEY MATTER ?

REST REQUEST

GET, PUT, POST

REST RESPONSE

XML/JSON

API ARCHITECTURE TYPES

REST, GraphQL, GRPC, SOAP

AUTHENTICATION & SPOTIFY INTEGRATION

```
CONST GETUSERPLAYLISTS = (TOKEN) => {
                RETURN AXIOS
.GET('HTTPS://API.SPOTIFY.COM/V1/ME/PLAYLISTS', {
                  HEADERS: {
        AUTHORIZATION: `BEARER ${TOKEN}`,
```

- Albums
- Artists
- Audiobooks
- Categories
- Chapters
- Episodes
- Genres
- Markets
- Player
- Playlists
- Search
- Shows
- Tracks
- Users

OAUTH 2.0 IMPLICIT GRANT FLOW

Direct token exchange via browser-based login.

ACCESS TOKEN PARSING AND STORAGE

Extracted tokens from URL and saved locally.

USER SESSION RESTORATION ON RELOAD

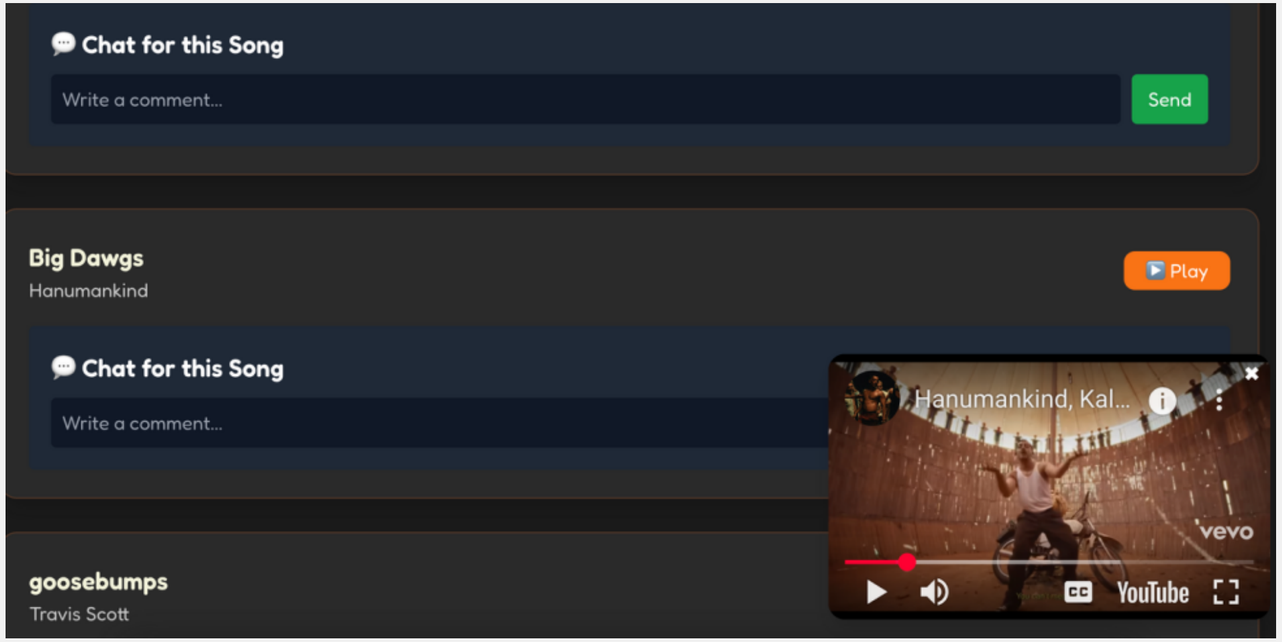
Restored token and display name from cache.

FETCHING DATA WITH API CALLS

Used token to retrieve playlists and profile, API endpoints to fetch tracks.

```
CONST SEARCHYOUTUBE = ASYNC (QUERY) => {
AXIOS.GET(`HTTPS://WWW.GOOGLEAPIS.COM/YOUTUBE/V3/SEARCH`, {
                        PARAMS: {
                      PART: 'SNIPPET',
                       MAXRESULTS: 1,
                          Q: QUERY,
                    KEY: YOUTUBE_API_KEY,
                       TYPE: 'VIDEO',
```

YOUTUBE API FOR PLAYBACK



NEED FOR ALTERNATIVE PLAYBACK SOURCE

Spotify limits full playback to Premium users.

DYNAMIC SEARCH QUERY CONSTRUCTION

Used track name and artist for results.

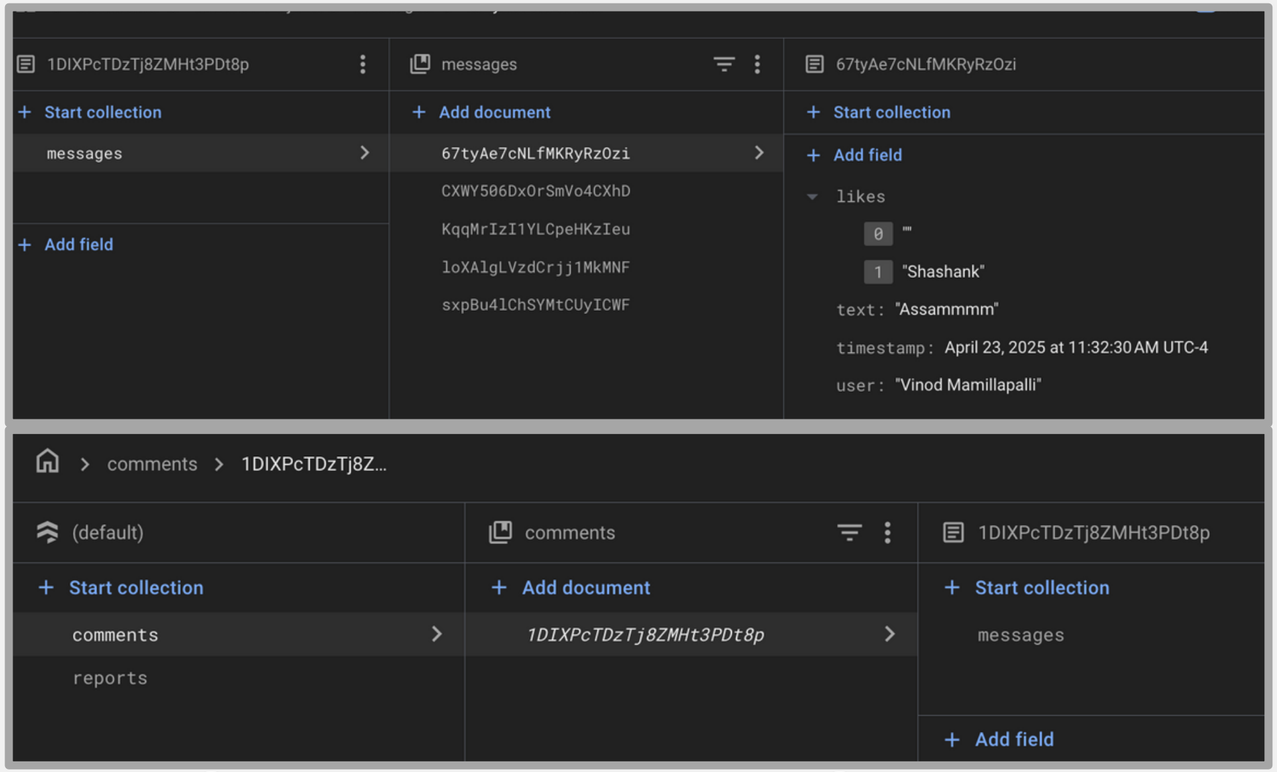
EMBEDDING YOUTUBE VIDEOS VIA IFRAME

Displayed videos directly inside React player.

FALLBACK COMPATIBILITY ACROSS ALL USERS

Ensured songs playable for non-Spotify users.

FIREBASE FIRESTORE FOR CHAT



REAL-TIME UPDATES VIA ONSNAPSHOT()

Auto-syncs messages without page refresh

HIERARCHICAL DATA STRUCTURE

songID → messages → replies per comment

SOCIAL FEATURES ENABLED

Users can reply, like, and report messages

SERVERLESS, SCALABLE, FRONTEND-FRIENDLY

No backend setup; uses secure cloud access.

WHY REACT FOR FRONTEND DEVELOPMENT?

REACT USES JSX

Blend of JS and HTML-like syntax
Embed JS expressions

COMPONENT BASED

Broken into reusable pieces
arbitrary inputs (props)
return React elements

STATEFUL COMPONENTS

Hooks like ‘useState’ and ‘useEffect’ let you control how data changes the UI

VIRTUAL DOM

Track UI changes and update only what’s needed

UI & CODE ARCHITECTURE

```
src/
├── components/
│   ├── ChatBox.jsx          // Real-time comment system with like/reply/report
│   ├── Library.jsx          // Displays user-specific Spotify playlists
│   └── Player.jsx           // Embeds YouTube video for selected track
│
├── pages/
│   ├── Home.jsx             // Animated landing page with hero text and navigation
│   └── PlaylistDetails.jsx// Lists tracks from selected playlist + ChatBox
│
├── utils/
│   ├── auth.js              // Spotify OAuth token handling
│   ├── spotify.js           // API calls to fetch playlists and tracks
│   └── youtube.js           // Builds search query & gets playable YouTube video
│
├── App.js                   // Main component with routing and session logic
├── firebase.js              // Firebase config and Firestore DB instance
└── index.html               // Base HTML with Tailwind and font setup
```

REACT RENDER TREE LOGIC

Minimized unnecessary re-renders, conditional rendering based on props, Dependency Trees

REACT PATTERNS

React Router, Dynamic Rendering, Functional components with React Hooks, Different props

TAILWIND CSS

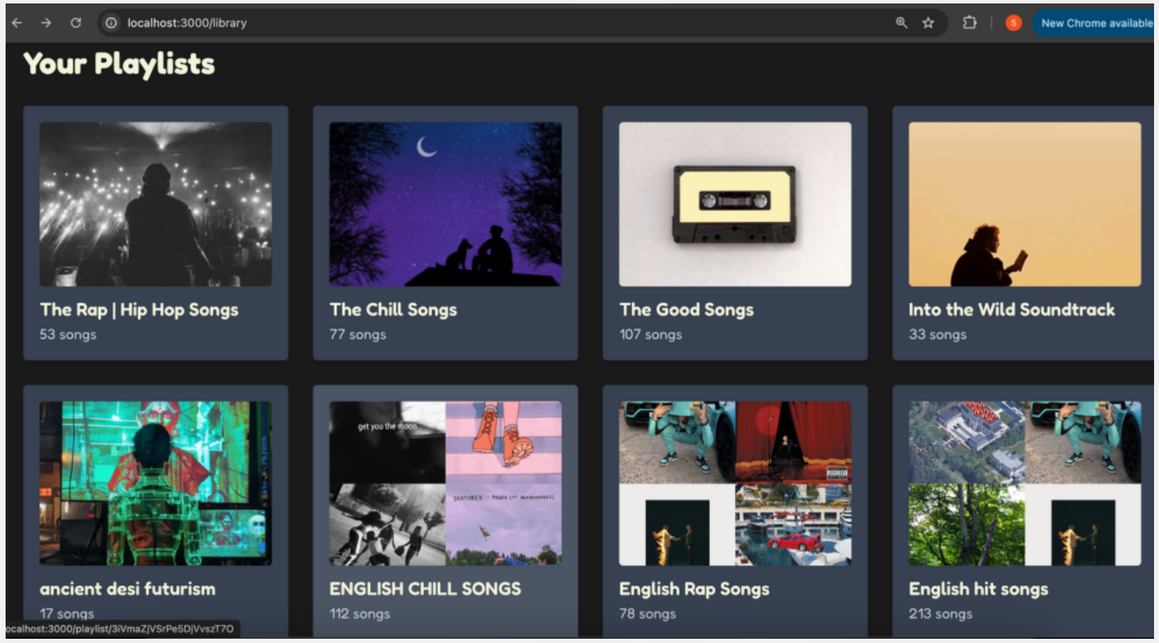
Utility-first styling approach, Built-in layout utilities, Easy hover transitions, No external CSS

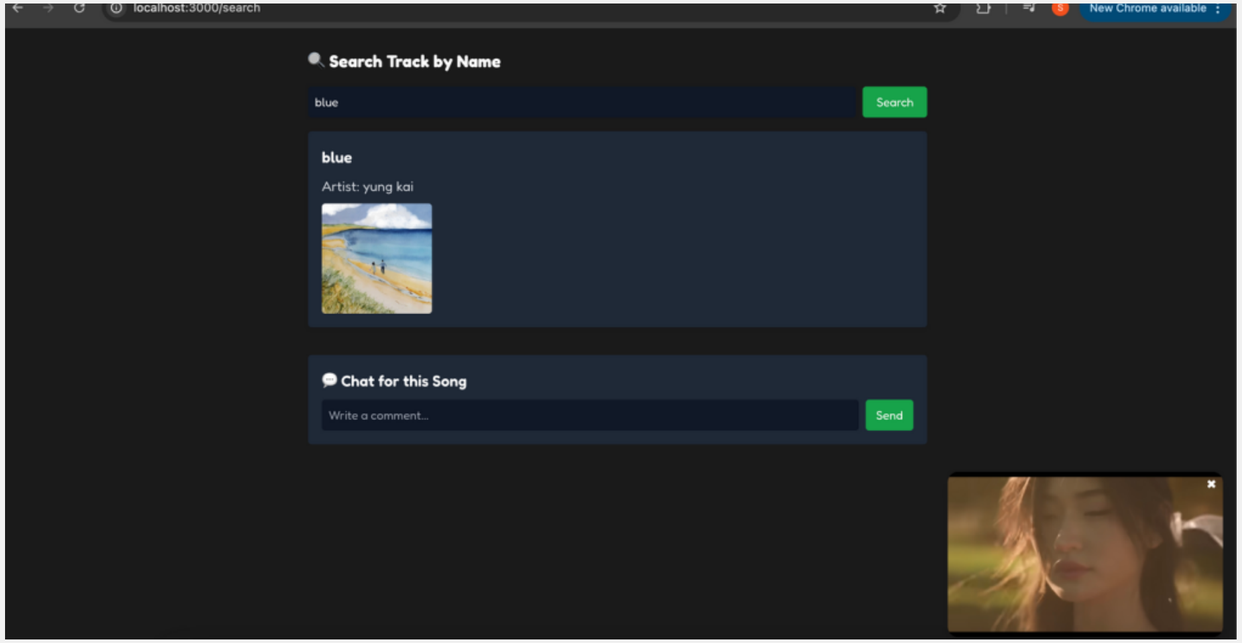
CODE FOLDER STRUCTURE

Components, Pages, Utils. New Tracksearch.jsx

RESULTS

1. USER'S LIBRARY PAGE
2. CHATBOX
3. SEARCH BAR





EVALUATION

CHALLENGES	LIMITATIONS	KEY LEARNINGS	FUTURE ENHANCEMENTS
MULTI ACCOUNT LOGINS CAUSED TOKEN CONFLICTS REPEATED CONSENT SCREENS	NO NATIVE USER SYSTEM	IMPLEMENTING OAUTH-BASED AUTHENTICATION, API INTEGRATION	SYNCHRONIZED LISTENING, LYRICS INTEGRATION
PLAYBACK RESTRICTIONS AND ABUSE REPORTING	MINIMAL ADMIN MODERATION	REACT STATE LOGIC & ROUTING	SHAREABLE SONG CARDS
UI CLUTTER AND OVERLAP	AUTHENTICATION CONSTRAINTS, LIMITED TESTING COVERAGE	DESIGNING FIRESTORE NOSQL SCHEMA	REAL-TIME NOTIFICATIONS

REFERENCES

[1] SPOTIFY FOR DEVELOPERS. (2024). SPOTIFY WEB API. RETRIEVED FROM HTTPS://DEVELOPER.SPOTIFY.COM/DOCUMENTATION/EB-API/

[2] Google Developers. (2023). YouTube Data API v3. Retrieved from https://developers.google.com/youtube/v3

[3] Firebase. (2024). Cloud Firestore Documentation. Retrieved from https://firebase.google.com/docs/firestore

[4] Tailwind CSS. (2024). Tailwind CSS Documentation. Retrieved from https://tailwindcss.com/docs

[5] Meta (React Team). (2024). React – A JavaScript library for building user interfaces. Retrieved from https://reactjs.org/

[6] Stack Overflow Community. (2023). Handling OAuth token flow in React applications. Retrieved from https://stackoverflow.com/questions/tagged/oauth+reactjs

[7] W3C. (2011). Embedding external content (iframes). Retrieved from https://www.w3.org/TR/html5/the-iframe-element.html

[8] OpenJS Foundation. (2023). JavaScript Event Loop Explained. Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop

[9] Axios. (2023). Axios – Promise-based HTTP client for the browser and Node.js. Retrieved from https://axios-http.com/docs/intro

[10] React Router. (2024). Declarative Routing for React Apps. Retrieved from https://reactrouter.com/en/main/start/tutorial

[11] Firebase. (2023). Realtime Updates with onSnapshot. Retrieved from https://firebase.google.com/docs/firestore/query-data/listen

[12] Meta (React Team). (2024). React useState and useEffect Hooks. Retrieved from https://react.dev/learn/state-a-component

[13] Google Developers. (2023). Understanding OAuth 2.0 for Web Applications. Retrieved from Your paragraph text

[14] Firebase. (2024). Security Rules for Firestore. Retrieved from https://firebase.google.com/docs/firestore/security/get-started

[15] DigitalOcean. (n.d.). An Introduction to Server-Side Rendering with React. DigitalOcean Community Tutorials. Retrieved May 8, 2025, from https://www.digitalocean.com/community/tutorials/react-server-side-rendering

THANK YOU

I’M HAPPY TO TAKE ANY QUESTIONS/SUGGESTIONS

09 MAY 2025

SRI SHASHANK KATARI

PROF. MIHAELA SABIN

INFORMATION TECHNOLOGY