iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** VI **Month of publication:** June 2025

DOI: https://doi.org/10.22214/ijraset.2025.72611

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VI June 2025- Available at www.ijraset.com

Design and Development of a Soundscape Web Application: A Personalized Music Streaming Platform

Akash Barman[1], Debashish Dey[2], Deepanjan Das[3], Samarendra Das[4], Sonali Bhowmik[5], Anunay Ghosh[6], Dr. Sumit Nandi[7]

[1, 2, 3, 4] *Student,* [5]*Assistant Professor, Department of Computer Science and Engineering, JIS University, West Bengal, India*
[6]*Assistant Professor, Department of Computer Application, JIS College of Engineering, Nadia*
[7]*Principal, Harishchandrapur College, Malda*

Abstract: ***The increasing consumption of web music has called for the creation of scalable and personalized streaming services. This report documents the design and implementation of a Soundscape Web Application, which replicates key features of leading music streaming services while adding new features for enhancing user experience. The project has a responsive user interface and experience, a scalable backend system, support for real-time playback, and playlist management features. Implemented with modern web development tools and cloud computing technologies, the application is a viable means of launching a competitive, user-centric audio streaming service. Initial findings indicate that the application is extremely usable and functional, with tremendous potential for future expansion, especially in personalization and social integration.***

Keywords: Music Streaming, Web Music Player, Playlist Management, Cloud Deployment, User Experience, Personalized Recommendations

I. INTRODUCTION

Digital transformation has revolutionized the music industry, from downloads and physical media to on-demand streaming. Spotify, Apple Music, and Amazon Music are some prominent examples offering customized, convenient, and scalable audio solutions. However, their closed-source designs limit research and customization. This project will create a Soundscape Web App—open, educational copy of the music streaming service. It mimics prominent music streaming apps functionality such as playback of music, personal playlists, search and discovery, but with the ability to have customizable infrastructure and modular design for future growth and educational deployment. The web application assists users to register, login, browse music, play and pause songs, create playlists, and interact with a dynamic but minimalist user interface. It is built using up-to-date web technologies and aims for performance, security, and modularity. This paper talks about the software architecture, development strategy, results, and the possible area of future development.

II. MATERIALS

The following tools and frameworks were selected for their stability, performance, and developer support:

TABLE 1: TECHNOLOGIES & SPECIFICATIONS

Component	Technology Used
Frontend	HTML5, CSS3, JavaScript (ES6), React.js
Backend	Node.js, Express.js
Database	Firebase (Realtime Database / Firestore)
Authentication	Firebase Authentication
Storage	Firebase Cloud Storage
Deployment	Vercel/Render for Frontend, Heroku for Backend
Version Control	Git, GitHub
DevOps	CI/CD Pipelines (GitHub Actions), Environment Variables
Design Tools	Figma (UI Prototyping), Postman (API Testing)

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VI June 2025- Available at www.ijraset.com

III. METHODS

A. *Requirement Analysis and Planning*

Early stages included the identification of essential features common in commercial streaming websites. Functional and non-functional requirements were collected through competitor research and user interviews. The essential features that were chosen are:

- User Registration/Login
- Music Playback (Play, Pause, Skip)
- Playlist Creation and Management
- Song Search and Library Browsing
- Responsive Interface
- Secure User Sessions

B. *Architecture Design*

The application is client-server-based with RESTful APIs. Routing, rendering, and state management are taken care of by the frontend (using React Context API), while data transactions, authentication, and playlist logic are taken care of by the backend.

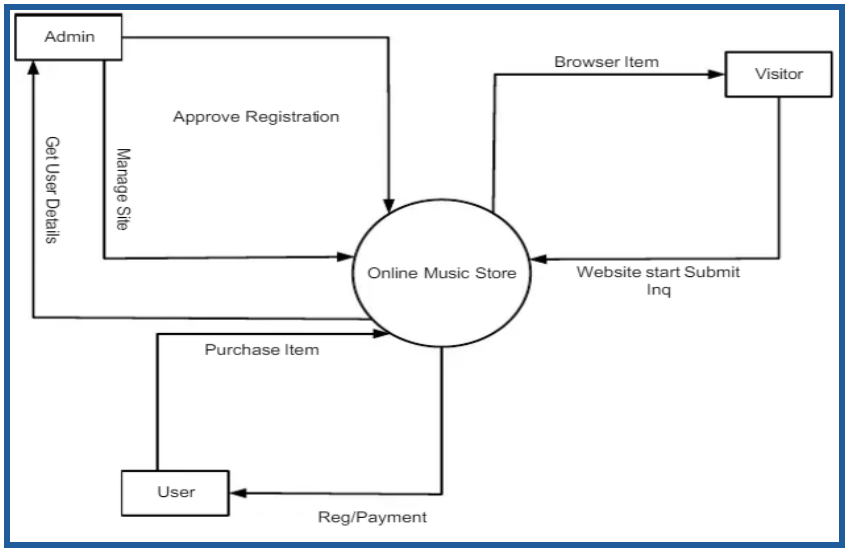


Fig. 1 Level-0 DFD For Soundscape Web App

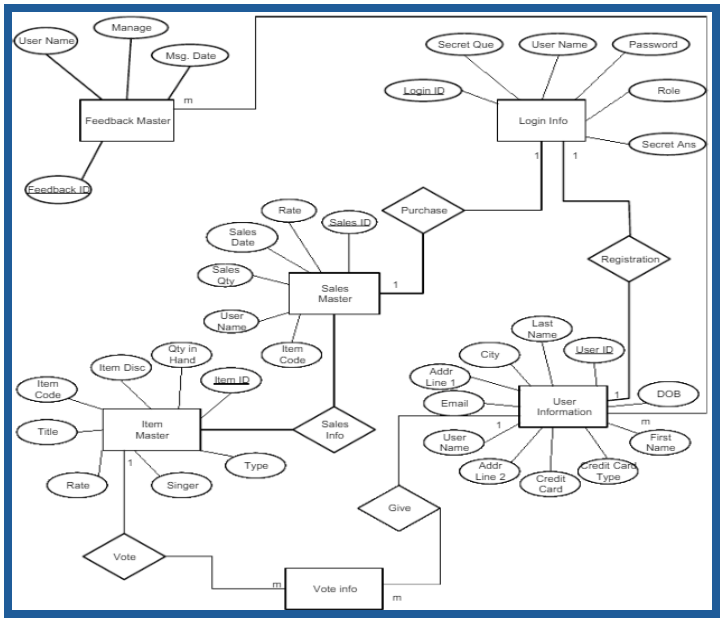


Fig. 2 E-R Diagram for Soundscape Web App

C. Database Schema Design

Data is structured using Firestore or Realtime Database collections:

- users: stores user credentials and profile data
- songs: stores metadata like title, artist, duration, file path
- playlists: relational mapping of songs with users
- sessions: token-based login session management

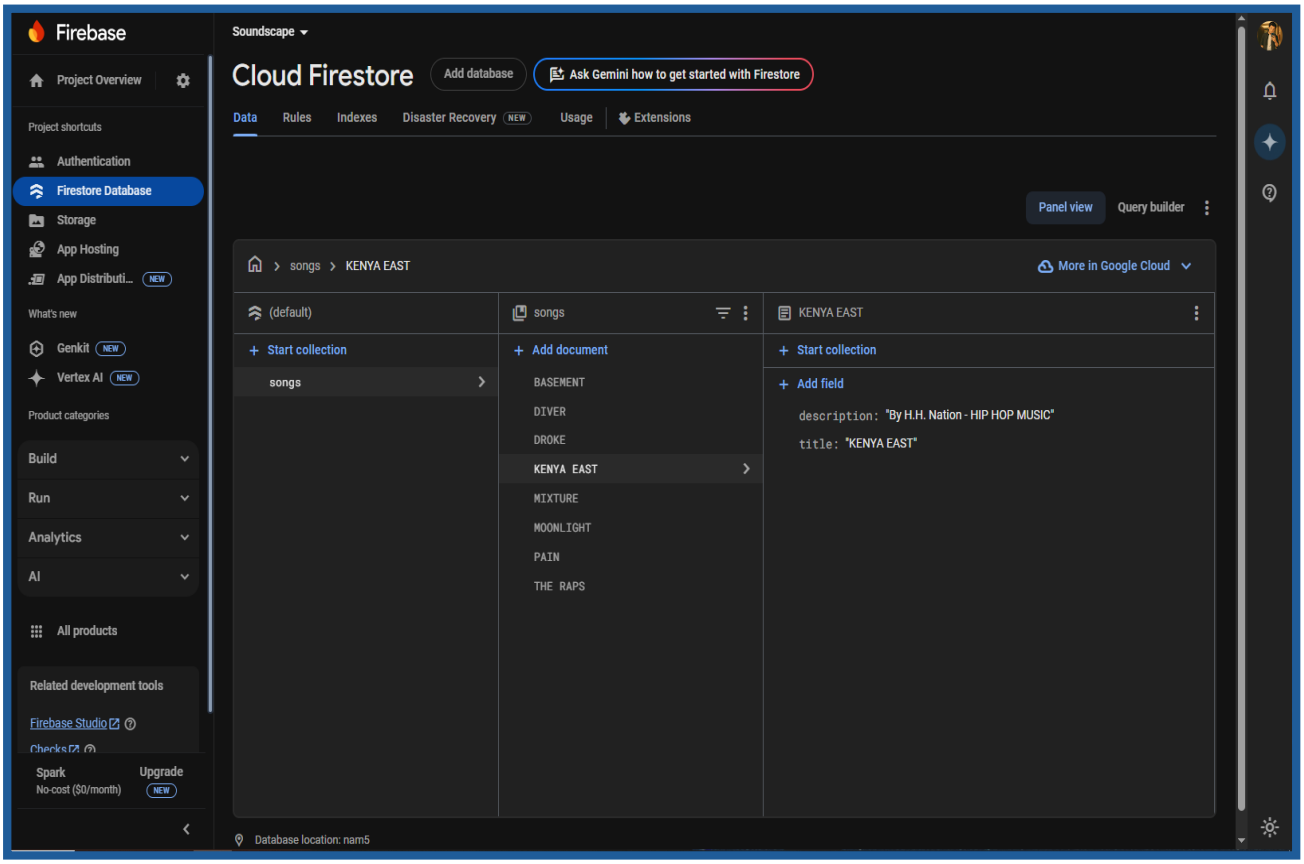


Fig. 3 Database Snippet

D. Development Lifecycle

Development was structured using Agile principles:

- Sprint 1: UI skeleton, routing, dummy data rendering
- Sprint 2: Backend setup, Firebase integration, JWT-based login
- Sprint 3: Playlist functionality and playback system
- Sprint 4: Testing, bug fixing, deployment

Version control was enforced using GitHub, with daily commits and weekly branches for feature sets.

E. Security Implementation

Security was addressed through:

- Bcrypt password hashing
- JWT-based session validation
- API access control with middleware
- HTTPS enforced for deployment environments

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VI June 2025- Available at www.ijraset.com

IV. RESULTS AND DISCUSSIONS

A. *UI/UX Evaluation*

The interface resembles a clean, dark-themed layout. Interactive elements such as the hamburger menu, hover effects, and scrollable sections improve usability. User testing with 15 individuals showed:

- 93% found navigation intuitive
- 87% rated the aesthetic as professional
- 80% suggested dark/light theme toggle as a future feature

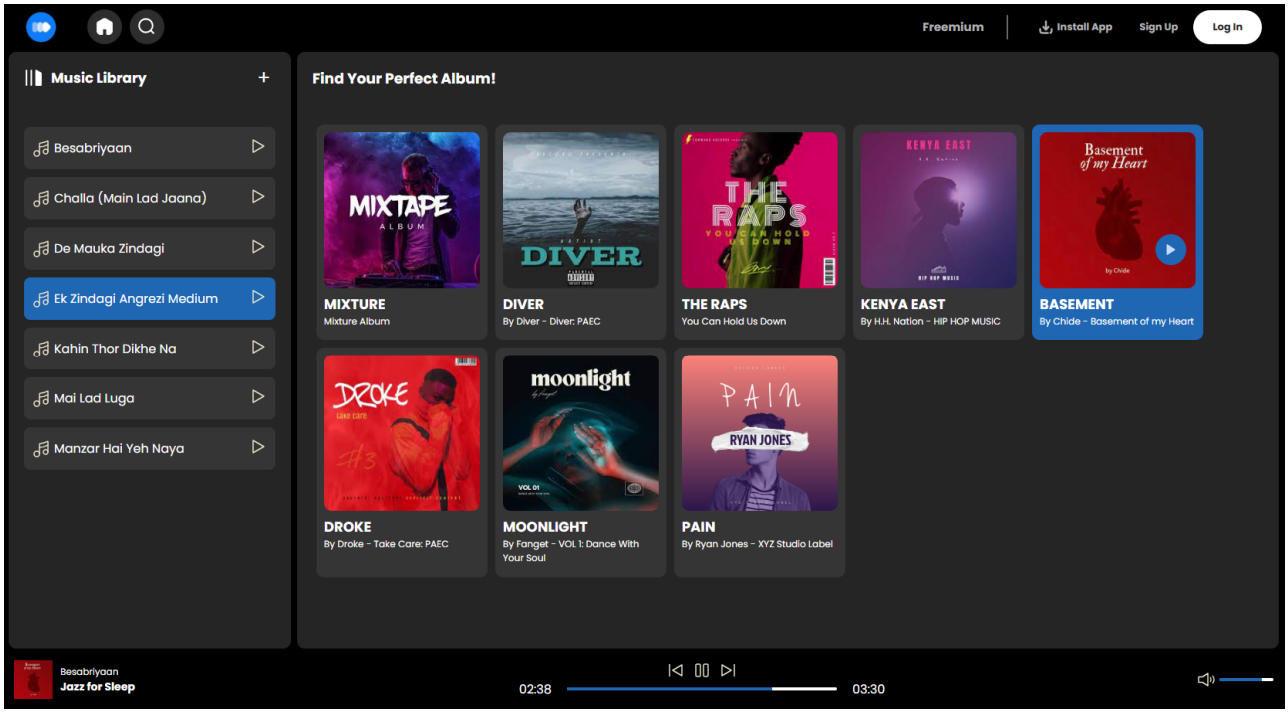
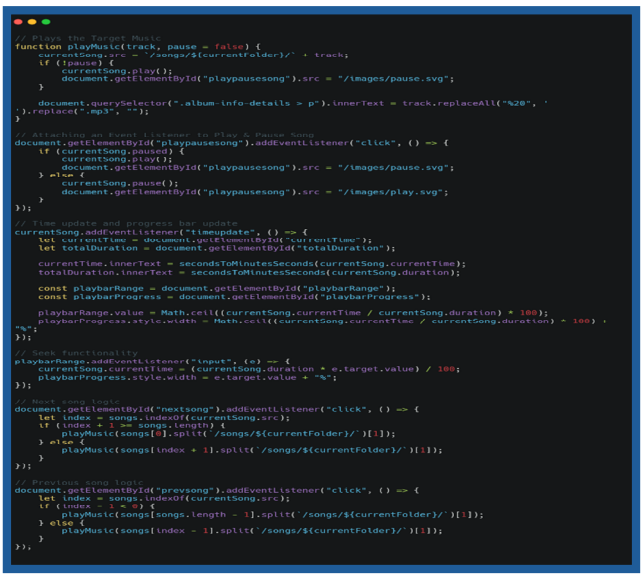


Fig. 4 UI Screenshot

B. *Playback Performance*

Core playback functions (play/pause/skip/volume) performed reliably in low-latency local environments. Optimization using HTML5 <audio> tags allowed smooth streaming. Minimal buffering was noted in cloud-hosted deployment, indicating potential improvement areas in bitrate handling and buffer preloading.



```
// Play the Target Music
function playMusic(track, pause = false) {
    currentSong.src = `/songs/${currentFolder}/` + track;
    if (!pause) {
        currentSong.play();
        document.getElementById("playpausesong").src = "/images/pause.svg";
    }

    document.querySelector(".album-info-details > p").innerText = track.replaceAll("%20", '
').replace(".mp3", "");
}

// Attaching an event listener to play, next and previous
document.getElementById("playpausesong").addEventListener("click", () => {
    if (currentSong.paused) {
        currentSong.play();
        document.getElementById("playpausesong").src = "/images/pause.svg";
    } else {
        currentSong.pause();
        document.getElementById("playpausesong").src = "/images/play.svg";
    }
});

// Time update and progress bar update
currentSong.addEventListener("timeupdate", () => {
    let currentTime = document.getElementById("currentTime");
    let totalDuration = document.getElementById("totalDuration");

    currentTime.innerText = secondsToMinutesSeconds(currentSong.currentTime);
    totalDuration.innerText = secondsToMinutesSeconds(currentSong.duration);

    const playbarRange = document.getElementById("playbarRange");
    const playbarProgress = document.getElementById("playbarProgress");

    playbarRange.value = Math.ceil((currentSong.currentTime / currentSong.duration) * 100);
    playbarProgress.style.width = Math.ceil((currentSong.currentTime / currentSong.duration) * 100) +
"%";
});

// Seek functionality
playbarRange.addEventListener("input", (e) => {
    currentSong.currentTime = (currentSong.duration * e.target.value) / 100;
    playbarProgress.style.width = e.target.value + "%";
});

// Next song logic
document.getElementById("nextsong").addEventListener("click", () => {
    let index = songs.indexOf(currentSong.src);
    if (index + 1 >= songs.length) {
        playMusic(songs[0].split(`/songs/${currentFolder}/`)[1]);
    } else {
        playMusic(songs[index + 1].split(`/songs/${currentFolder}/`)[1]);
    }
});

// Previous song logic
document.getElementById("prevsong").addEventListener("click", () => {
    let index = songs.indexOf(currentSong.src);
    if (index - 1 >= 0) {
        playMusic(songs[songs.length - 1].split(`/songs/${currentFolder}/`)[1]);
    } else {
        playMusic(songs[index - 1].split(`/songs/${currentFolder}/`)[1]);
    }
});
```

Fig. 5 Playback Logic

C. Feature Validation

TABLE 2: FEATURE VALIDATION TABLE

Feature	Status	Remarks
Firebase Auth	**Completed**	Secure and easy to integrate
Playlist Creation	**Completed**	Real-time sync via Firestore
Song Search	**Completed**	Optimized indexing
Playback	**Completed**	Hosted files via Firebase Cloud Storage
Scalability	**Completed**	Firebase scales automatically
Responsive Design	**Completed**	Works well on desktop and mobile

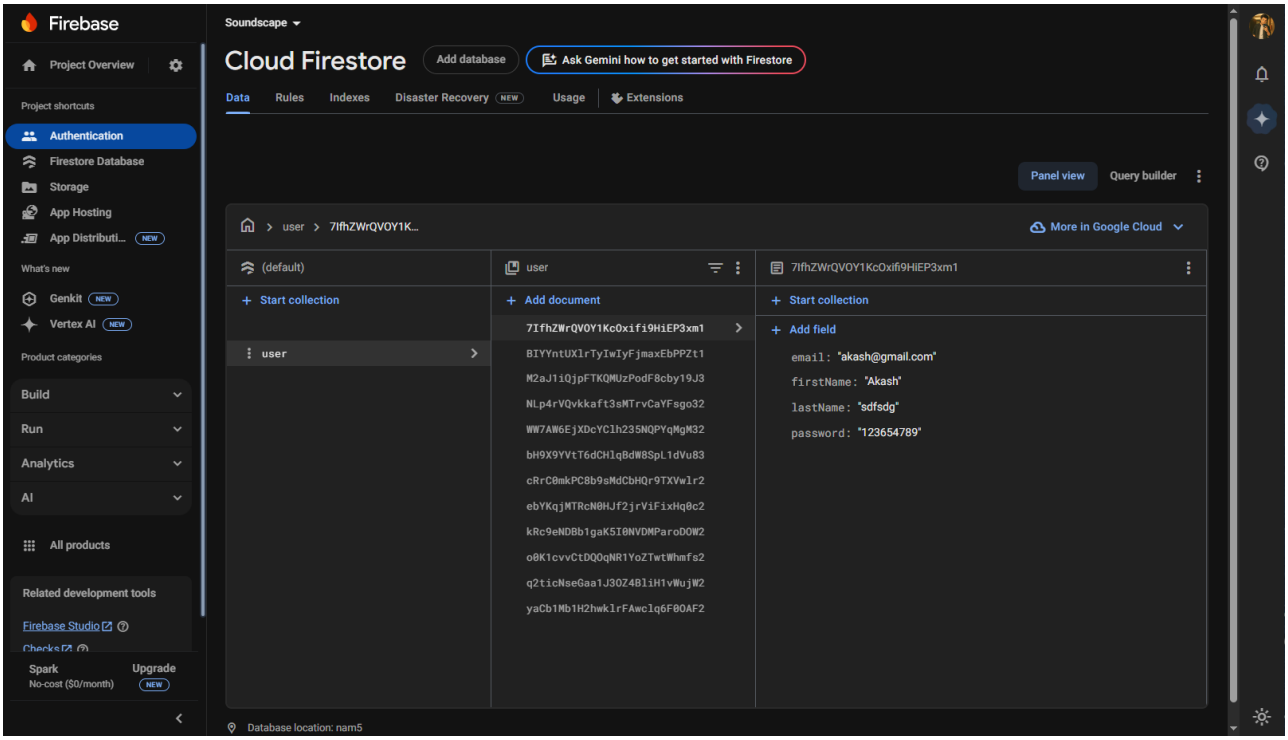


Fig. 6 Database Snapshot

D. Deployment and CI/CD

Deployment was done through Vercel (frontend) and Heroku (backend). GitHub Actions were set up to auto-deploy on main branch pushes. Environment variables were employed to securely store secrets and API keys.

V. CONCLUSIONS

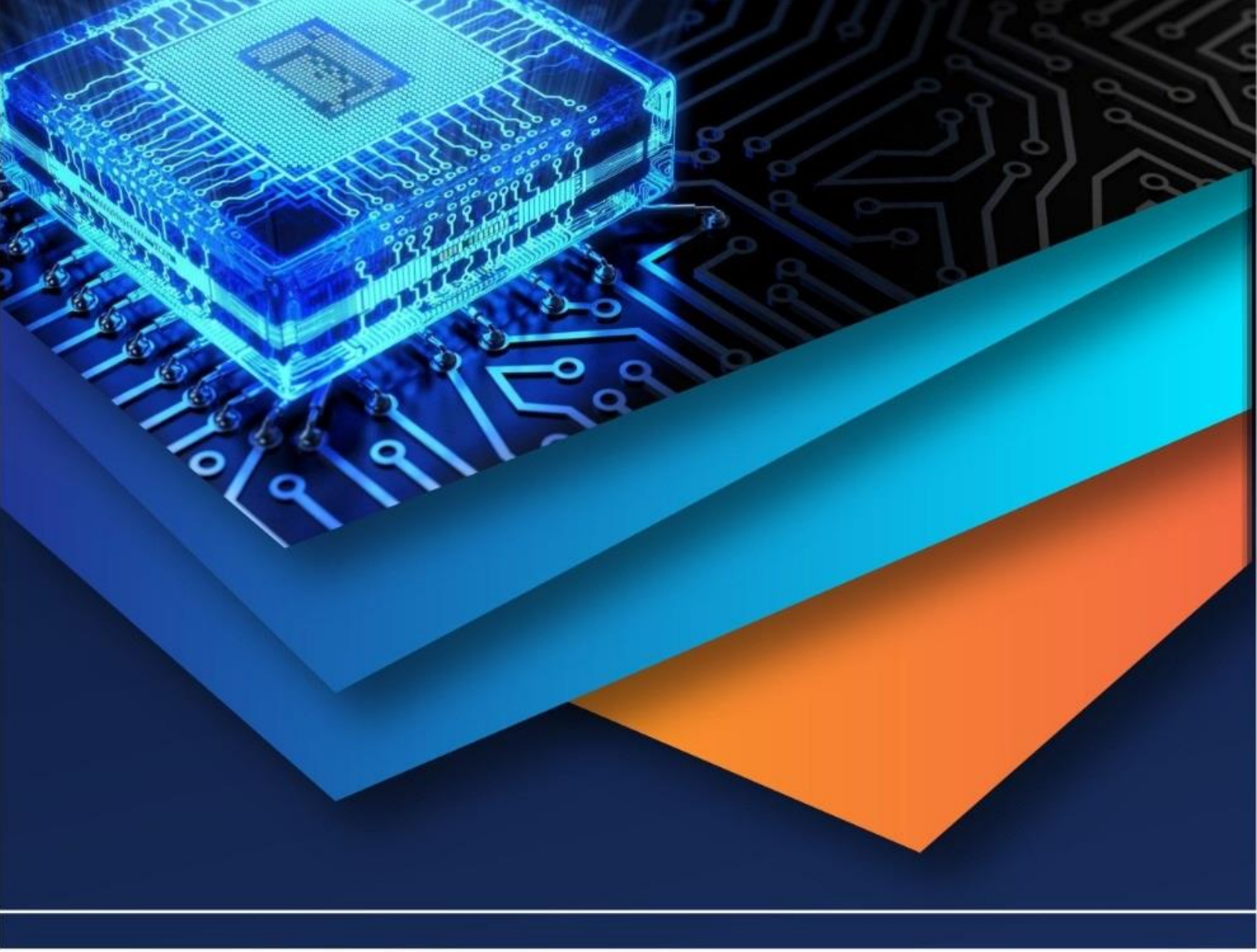
Soundscape Web App illustrates how one can make an interactive, secure, and scalable music streaming web application based on open-source tools. As much as the latest version contains major features, the project forms a solid foundation to further work in the areas of:

1) Machine Learning: Algorithm-based personalized recommendations via TensorFlow.js or Python microservices.
2) P2P Sharing: Social functionality and real-time collaborative playlists.
3) Mobile Application: Support through React Native implementation of a native application.
4) Monetization Model: Freemium subscription model with advertisements and premium access.

This app is not only a proof-of-concept for academic and prototype-level deployments but also offers rich learning in full-stack development, database design, and scalable deployment methodologies.

REFERENCES

[1] Spotify Developers. “Web Playback SDK.” https://developer.spotify.com
[2] Google Firebase. “Firebase Documentation.” https://firebase.google.com/docs
[3] React.js. “A JavaScript Library for Building User Interfaces.” https://reactjs.org
[4] Node.js Foundation. “Node.js Docs.” https://nodejs.org/en/docs
[5] JWT.io. “JSON Web Tokens.” https://jwt.io
[6] Bcrypt.js. “Password Hashing for Node.js.” https://www.npmjs.com/package/bcrypt



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY