IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

Video Interview SAAS Platform

Prof. Vivek R. Shelke[1] Hariom Ingle[2], Chetan Rathod[3], Mahesh Jadhav[4], Pratik Khose[5]
Professor, Department of Computer Engineering[1]
Student, Department of Computer Engineering[2-5]
Government College of Engineering Yavatamal, Maharashtra, India

Abstract: *The surge in remote work and online recruitment has generated the need for expert platforms beyond regular-purpose video conferencing software. This project offers a Video Calling Interview Platform, a complete-stack web application built with Next.js, TypeScript, Stream, Convex, and Clerk. The platform supports formal remote interviews via capabilities such as real-time video calls, screen sharing, session recording, role-based authentication, and interview feedback.*
In contrast to conventional solutions, this platform is built with interview process workflows in mind, enabling interviewers and candidates to participate in interactive, secure sessions. It features dynamic routing, server-client component management, and state handling through a serverless backend. Integrating up-to-date front-end development methodologies with cloud-native infrastructure, this solution provides a user-focused, extensible, and scalable solution for remote technical recruit.

Keywords: Serverless backend, User-centric, Convex Database, Clerk, role-based authentication, scalability, infrastructure, interview preparation

I. INTRODUCTION

In today's remote-first world, companies are rapidly shifting from traditional in-person interviews to virtual interviewing solutions that offer flexibility, efficiency, and scalability. But all available platforms are either too generic (such as Zoom or Google Meet) or do not have features specific to technical interviews, e.g., session tracking, feedback logging, screen sharing with context, or role-based access.The Video Calling Interview Platform is a contemporary, full-stack web app that seeks to simplify remote technical interviews. It facilitates frictionless interaction between interviewers and candidates via live video calls, screen sharing, and live feedback. Developed using the latest technologies such as Next.js, TypeScript, Convex, Clerk, and Stream, the platform provides a seamless, secure, and scalable experience for end-users and administrators alike.
This platform not only enables communication but also accommodates functionalities such as screen capture, interview feedback, role-based authentication, and dynamic session management, which make it the perfect solution for technical interviews being held remotely by organizations. The architecture is modular and serverless, providing high availability, low latency, and quick development.

1.1 Context

This project was built to model a production-grade, real-world SaaS application using a modern full-stack architecture. It incorporates a number of key features to ensure technical interviews are more interactive and informative, such as:

- Live video conferencing using Stream's SDK
- Live video conferencing via Stream's SDK
- Role-based authorization through Clerk
- Realtime backend through Convex for near-instant updates, presence, and session state
- Screen recording and sharing for walkthroughs of code and reviewing sessions
- Feedback from interviews in the form of ratings and comments
- Reusable layouts & dynamic routing with Next.js App Router

DOI: 10.48175/568

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

The objective was to create a serverless, scalable setup that illustrates complex front-end and back-end integration, optimized data transfer, and robust user experience design with Tailwind CSS and Shadcn UI.

1.2 Problem Statementt

Traditional video conferencing platforms such as Zoom, Google Meet, or Microsoft Teams were not designed for technical interviews. They tend to be lacking in the structure, interactivity, and post-interview analytics essential for an easy hiring process. Consequently, organizations find themselves handling a variety of tools simply to conduct simple tasks such as scheduling, offering feedback, recording sessions, or monitoring performance.

Organizations conducting remote interviews face challenges such as:

- No built-in mechanism for candidate/interviewer role management
- No real-time feedback or formal rating mechanisms
- No screen recording integration and session playback
- Interrupted experience due to frequent toggling between multiple tools
- No centralized management of interview data or audit trails

Preparation for technical interviews also does not come easy for the candidates. Most depend on books or theory-based websites, which are not able to provide real-world, hands-on practice required to crack the interview. Without a live, interactive place to hone their craft, even talented applicants may feel unready. That's why a specialized platform is increasingly necessary—one that accommodates live coding, screen sharing, playback of sessions, structured feedback, and safe access—all at once, making remote hiring easier and more efficient for both parties.

II. LITERATURE SURVEY

With the competitive tech job market today, interview preparation is more difficult than ever. Applicants are not just rehearsing technical skills but also communication, problem-solving, and adjusting to new, often remote, interview styles. Conventional prep strategies such as in-person mock interviews or simple video calls are insufficient. They are not personalized, delayed in feedback, and unstructured.

With remote work now the new normal, most interviews are now done remotely. However, basic video platforms like Zoom or Google Meet are not designed for technical interviews. They lack crucial functionalities such as live coding support, feedback tracking, and session history—making it necessary for recruiters and candidates to use multiple tools.

This project solves that issue by providing a specialized video interview platform. Developed with cutting-edge technologies such as Next.js, Stream, Convex, and Clerk, it consolidates all of it into one spot—real-time communication, safe authentication, organized interview flows, and smooth feedback. The objective is straightforward: enable smarter, smoother, and more efficient remote interviews for all parties.

Limitations of General Video Platforms

Tools such as Zoom, Google Meet, and Microsoft Teams provide high video quality and reliability but are not suitable for interviews. These tools:Lack built-in support for storing interview metadata.

- Do not support real-time interview evaluation
- Rely on third-party plugins or manual systems for feedback
- Do not provide real-time evaluation of an interview
- Are not optimized for role-based interaction (candidate vs interviewer)

2. Existing Interview Platforms

Solutions like HackerRank, CoderPad, and Interviewing.io are specialized solutions for remote technical interviews, having code editors integrated with video calls. But:

- They are largely proprietary and commercial offerings
- Provide limited flexibility and customization options
- Do not expose complete access to integrate with custom hiring pipelines

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

- Are monolithic or rigid backend architectures

3. Authentication and User Management

Strong user authentication is necessary for safe, role-based interview access. Clerk offers a contemporary solution with such features as:

- Smooth onboarding (email/SMS/social login)
- Session management and RBAC (Role-Based Access Control)
- Developer-friendly APIs to integrate with frameworks such as Next.js

4. Real-Time and Serverless Backends

Serverless and reactive backends are being increasingly utilized by modern applications to enhance scalability and developer speed. Convex provides:

- Real-time data syncing without having to build APIs manually
- Shared programming model for backend logic and data storage
- Streamlined state management for UI components

5. Real-Time Video Infrastructure

Stream is a platform for developers that offers:

- High-performance video call
- Screen sharing, recording, and participant roles features
- Secure handling of rooms and session APIs
- Smooth integration with frontend frameworks

III. PROJECT ARCHITECHTURE

The application's system architecture is built as a contemporary, modular full-stack web application using React, Next.js, and serverless APIs. It is based on a client-server architecture, optimized for scalability, maintainability, and performance.

The application has three primary layers:

- Frontend Architecture (Client-side UI)
- Backend Architecture (API Layer & Business Logic)
- Database Layer (Persistent Storage)

These parts are loosely coupled and communicate with each other via HTTP-based RESTful APIs or direct server-rendering mechanisms.

1. **Frontend Architecture**

The frontend is developed with React + Next.js, serving as the core user interface. It's designed to be responsive, dynamic, and strongly coupled with Clerk and Convex for real-time data synchronization and user identity management.

Key Components:

- **React + Next.js Pages:**
- Pages are organized with Next.js routing (pages/ or app/ directories). Server-side rendering (SSR) is enabled for SEO-critical pages, while most interactions are client-side.

Clerk Integration:

- Clerk's React SDK is bootstrapped in a layout or root component with <ClerkProvider>.
- Authentication flows (sign-up, sign-in, sign-out, password reset) are managed through Clerk's pre-built UIs or custom flows with Clerk hooks (e.g., useUser(), useSignIn()).

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

- Protected routes are guarded with Clerk's withServerSideAuth or client-side guards.

State & Data Management with Convex React:

- Convex delivers a reactive data layer through useQuery() and useMutation().
- Frontend bits subscribe to live data streams, updating UI automatically on database changes—no refetch logic needed.
- Real-time updates are done out of the box through WebSocket connections.

Benefits:

- State sync is done automatically between sessions/devices.
- No REST or GraphQL boilerplate — Convex takes care of data transport.
- Zero-client-config integration with Clerk + Convex (auth tokens passed transparently).

2. Backend Architecture (Convex Functions)

The logic is implemented as Convex Functions, provisioned and executed in a serverless environment hosted by Convex.

Features:

Typesafe Functions:

- Implemented in TypeScript.
- Split into query (read) and mutation (write) functions.

Auth Middleware:

- Convex has native integration with Clerk. Upon a client invoking a backend function, Convex validates the Clerk JWT and exposes user identity through ctx.auth.
- Example: const identity = await ctx.auth.getUserIdentity();

Access Control:

- Role-based access (admin, user, guest) is implemented in the logic layer.
- "only owners can update their document" logic is baked into mutations using the identity.subject field (Clerk user ID).

Edge Performance:

- Functions are rolled out globally, offering low-latency access close to the user.
- Optimized for high concurrency and low cold start times.

3. Database Layer (Convex Database)

- Convex offers a proprietary document database optimized for reactivity and ACID-compliant transactions.

Features

Data Model:

- Collections: JSON-like documents grouped into collections.
- Example: users, posts, comments, messages.
- Schema Enforcement (optional): Runtime type enforcement or developer-provided schemas via zod or user-defined types are supported by Convex.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

Features:

- Strong Consistency: Convex provides consistent reads and writes across clients even under high concurrency.
- Transactions All the writes are done within transactions, so more than one related change can be rolled back atomically. Developers code logic in mutation functions to hide business rules.
- Reactive Subscriptions: useQuery() creates a live subscription to data. When ever any client causes a mutation, all other clients subscribed to impacted queries automatically update.
- User-Scoped Data: User data is generally keyed by Clerk's own ID (e.g., userId: identity.subject). Queries are scoped by auth context, which cannot be forged client-side.

Deeper Integration: Clerk + Convex

- The interaction between Clerk and Convex produces a zero-boilerplate full-stack system:
- Clerk handles sessions, tokens, and user roles.
- Convex reads and authenticates Clerk tokens automatically.
- Functions acquire identity context securely (userId, email, roles).
- Developer authors auth-aware logic once and uses it across all endpoints

IV. IMPLEMENTATION ARCHITECTURE

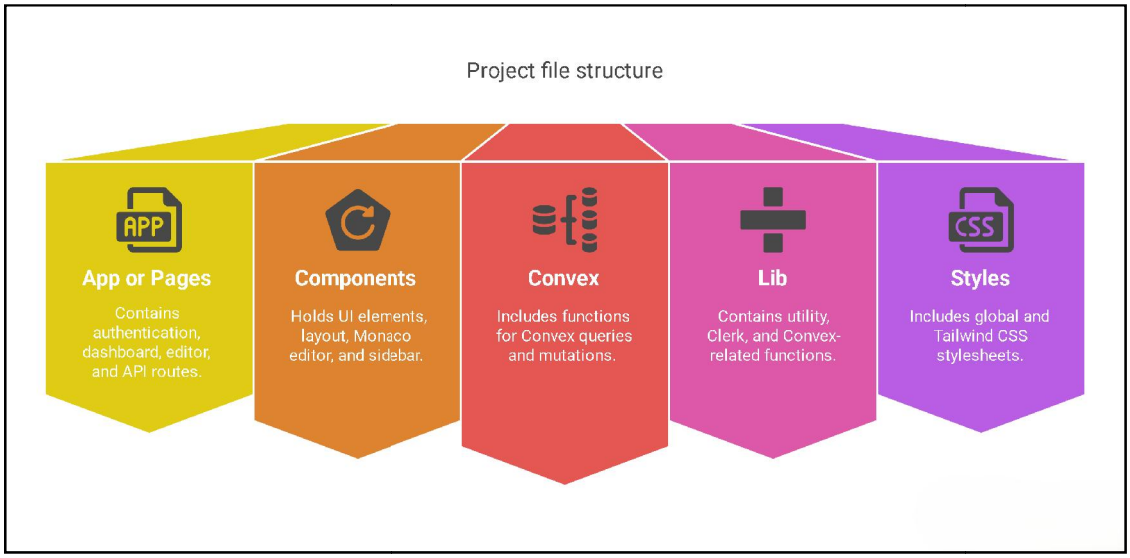

Fig 1. Implementation Architecture

Authentication with Clerk

Clerk handles full user identity, authentication, and session management.

Setup:

```
// _app.tsx or layout.tsx
import { ClerkProvider } from "@clerk/nextjs";
export default function App({ Component, pageProps }) {
 return (
  <ClerkProvider>
   <Component {...pageProps} />
  </ClerkProvider>
 );
```

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

```
}
```

Convex Database & Backend Logic

Convex replaces traditional server + DB with reactive, typesafe functions.

Schema:

```
// convex/schema.ts
import { defineSchema, defineTable } from "convex/server";
import { v } from "convex/values";

export default defineSchema({
  documents: defineTable({
    title: v.string(),
    content: v.string(),
    userId: v.string(),
    updatedAt: v.number(),
  }),
});
```

Editor Integration with Monaco

The Monaco Editor provides code-editing capabilities similar to VS Code.

```
// components/MonacoEditor.tsx
import Editor from "@monaco-editor/react";
export default function MonacoEditor({ value, onChange }) {
  return (
    <Editor
      height="100%"
      defaultLanguage="javascript"
      theme="vs-dark"
      value={value}
      onChange={onChange}
    />
  );
}
```

Real-Time Collaboration (Convex Reactive Layer):

Use useQuery() to subscribe to a document:

```
const doc = useQuery(api.documents.getById, { id: docId });
return <MonacoEditor value={doc?.content} />;
```

UI and Layouts (Radix + Tailwind + ShadCN-style):

Use Radix UI primitives for accessibility-compliant components:

```
import * as Dialog from "@radix-ui/react-dialog";
<Dialog.Root>
  <Dialog.Trigger>Edit</Dialog.Trigger>
  <Dialog.Content>
    <h2>Edit Document</h2>
  </Dialog.Content>
</Dialog.Root>
```

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

Algorithms & Data Structures:

- While Convex abstracts much of the backend complexity, these core concepts are used
- Document-based Data Modeling: Similar to MongoDB: each document has an _id, userId, content.
- Diffing & Syncing: Convex uses internal data diffing algorithms (similar to CRDT/OT).
- Real-Time Subscription Trees: Queries are subscribed reactively using dependency trees and WebSocket channels.

V. MODULES DESCRIPTION

Core Frameworks & Utilities:

Module	Description
next	Framework for React used for building full-stack web applications. Provides SSR, routing, and API handling.
react & react-dom	Core React libraries for building and rendering the UI.
typescript	Adds static typing to JavaScript, improving maintainability and reducing bugs.

Authentication:

Module	Description
@clerk/nextjs	Provides pre-built components and hooks for authentication, user management, and session handling in Next.js apps.

Video Conferencing:

Module	Description
@stream-io/node-sdk	Backend SDK for interacting with Stream APIs (chat, video, etc.).
@stream-io/video-react-sdk	React components for integrating Stream's video calling into the frontend UI. Provides features like call setup, participants view, and screen sharing.

UI Framework & Styling:

Module	Description
tailwindcss	A utility-first CSS framework for fast UI development.
tailwindcss-animate	Adds animation utilities to Tailwind for enhanced UI transitions.
tailwind-merge	Resolves Tailwind class conflicts dynamically (e.g., bg-red-500 bg-blue-500).
next-themes	Enables dark/light theme switching using Tailwind and Next.js.
clsx	Conditional class name utility, useful for managing dynamic styles.
class-variance-authority	Helps define and manage variant-based classNames in Tailwind.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

Radix UI (Headless UI Components)

Module	Description
@radix-ui/react-avatar	For rendering user profile avatars.
@radix-ui/react-dialog	Modal/dialog window primitives.
@radix-ui/react-dropdown-menu	Dropdown menu logic.
@radix-ui/react-label	Accessible label component for inputs.
@radix-ui/react-scroll-area	Scrollable container for overflowing content.
@radix-ui/react-select	Headless dropdown select component.
@radix-ui/react-slot	Allows composition of custom components.
@radix-ui/react-switch	Toggle switch component (e.g., for settings).

Editor Integration

Module	Description
@monaco-editor/react	Embeds the Monaco code editor (used in VS Code) into React apps. Enables live coding/collaboration features.

State, Time, and Layout Utilities

Module	Description
convex	Real-time backend-as-a-service used for serverless functions, database, and data syncing.
date-fns	Lightweight date manipulation library (e.g., formatting timestamps, calculating durations).
react-day-picker	Provides calendar/date picker components.
react-resizable-panels	Enables resizable panels, perfect for IDE-like UI with split panes.

Feedback and Notifications

Module	Description
react-hot-toast	Modern toast notification library to show success, error, or info messages.

Webhooks

Module	Description
svix	Handles secure webhook delivery and event processing. Useful for integrating services like Stripe, Clerk, etc.

Dev Dependencies

Module	Description
@types/*	TypeScript typings for Node, React, and DOM. Enables better IntelliSense and error checking.
postcss	Tool for transforming CSS with plugins. Required for Tailwind.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

VI. TECHNOLOGH STACK ARCHITECTURE

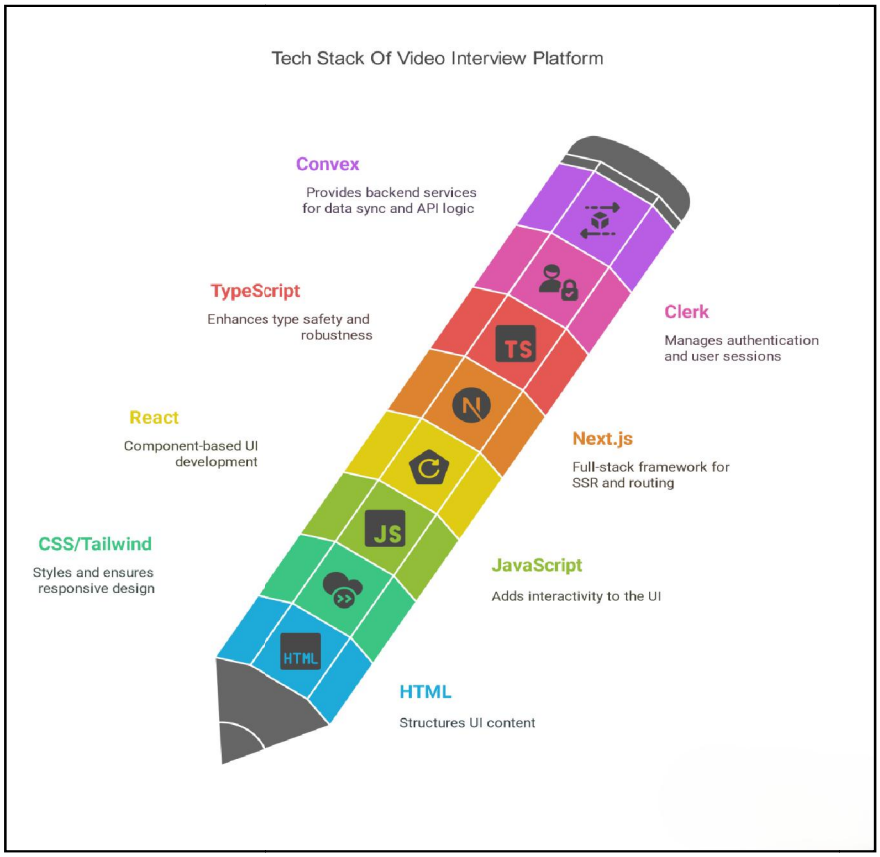

Fig 2. Technology stack Architecture

VII. RESULT

The developed Video Interview SAAS Platform successfully demonstrated its capability to conduct real-time technical interviews, integrating live video, screen sharing, and secure authentication. Our testing confirmed the system's efficiency in managing interview sessions and logging feedback, showcasing a robust and scalable solution for remote hiring. The platform consistently delivered a smooth user experience, validating its design for streamlined technical assessments

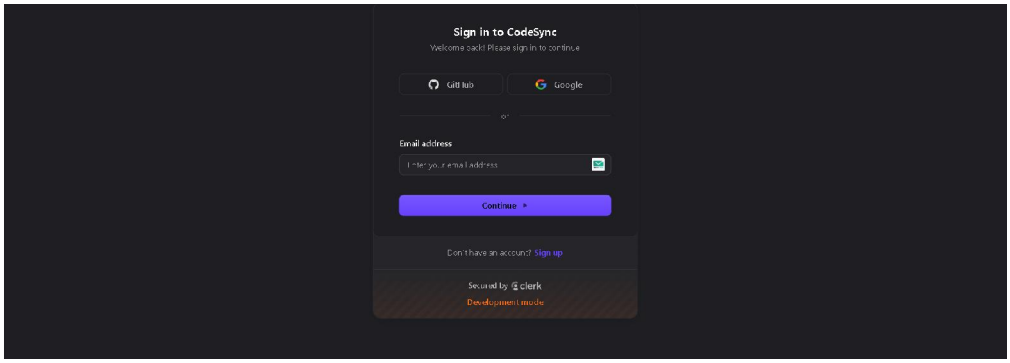

Fig 3. Login Page

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

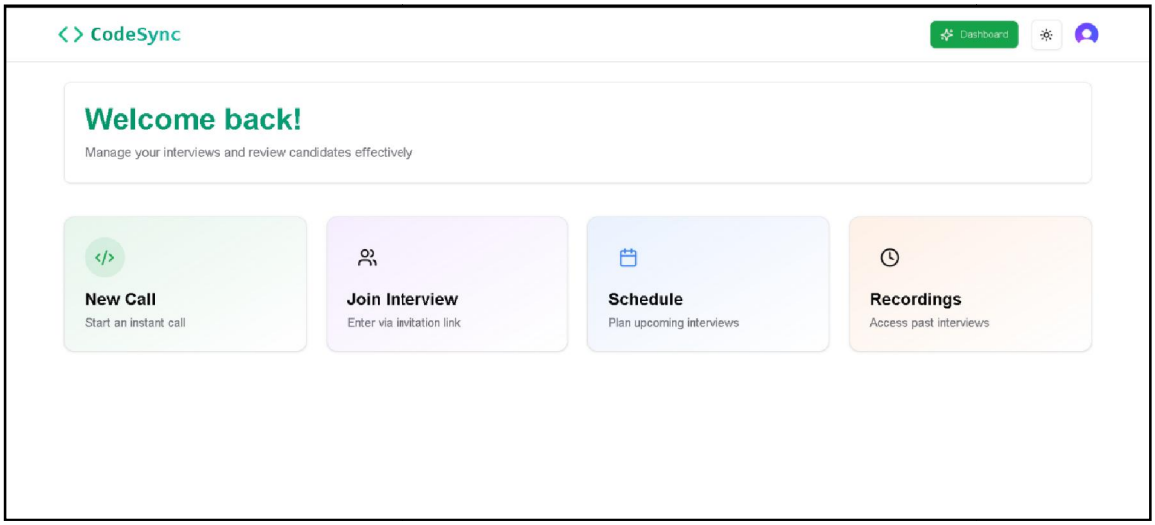

Fig 4. Home page

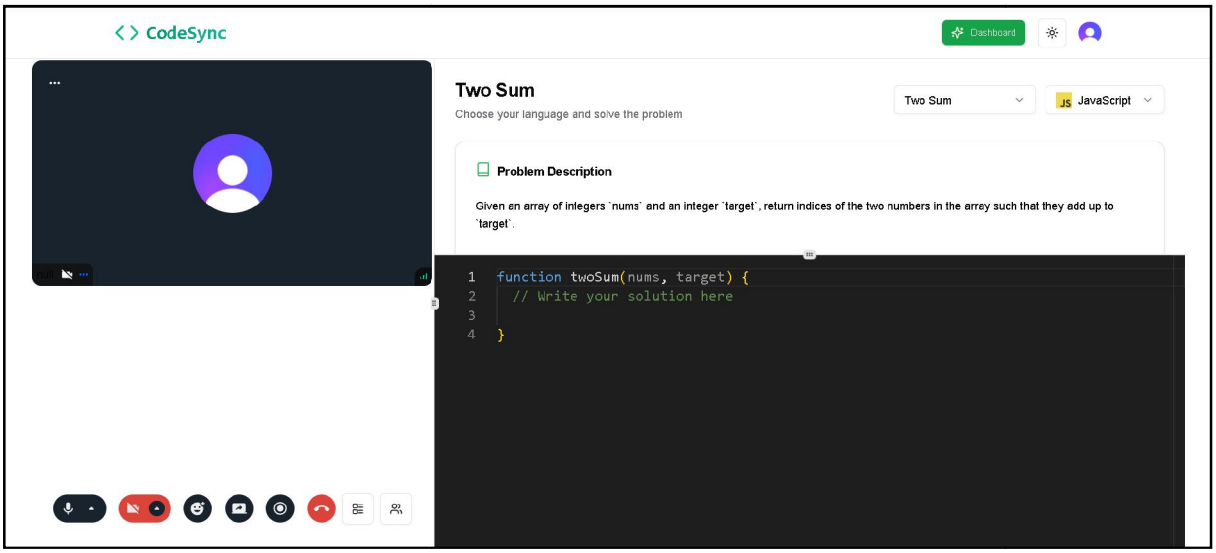

Fig 5. Interview Page

VIII. CONCLUSION

The Video Calling Interview Platform effectively solves the shortcomings of conventional video conferencing software by providing a customized, end-to-end solution specifically engineered for technical interviewing. Through the implementation of contemporary technologies like Next.js, TypeScript, Stream, Convex, Clerk, and Tailwind CSS, the platform offers a frictionless and secure experience both for candidates and interviewers.

This project illustrates the potential of developer tools and cloud services to be used synergistically in developing a scalable, user-friendly, and production-quality application. It not just addresses the immediate requirements of remote recruitment but also sets the foundation for further development such as coding tests, analytics boards, and AI-powered evaluation.

DOI: 10.48175/568

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, June 2025
Impact Factor: 7.67

IX. ACKNOWLEDGMENT

The title of the Acknowledgment section and the References section should not be numbered. Causal Productions thanks Michael Shell and other authors for creating and sustaining the IEEE LaTeX style files that have been employed in the preparation of this template.

X. FUTURE SCOPE

The Video Interview SAAS Platform, though strong in its present configuration, has wide scope for future development. Some of the major improvements would be to incorporate AI-driven practice sessions to offer individualized feedback and mock interview exposure to candidates. Adding aptitude tests to the platform would further enhance its functionality, making it a more complete initial screening tool. In addition, ongoing work will focus on optimizing its scalability so that the platform can easily handle an ever-growing user base and increasing number of interview sessions with optimal performance and stability.

REFERENCES

[1]. Bhat N.& Desai, R. (2022). Enhancing Code Quality with SonarQube. Journal of Software Engineering.

[2]. Auth0. (2023). Role-Based Access Control (RBAC): A Comprehensive Guide. Retrieved from https://auth0.com/docs/get-started/authentication-and-authorization/role-based-access-control-rbac (Note: Replace with a more specific, academic source if available, or a date accessed if using a general web resource).

[3]. Google Cloud. (2024). Serverless Computing Explained. Retrieved from https://cloud.google.com/serverless (Note: Similar to above, if an academic paper on serverless architectures is found, use that).

[4]. Next.js Documentation. (2024). App Router. Retrieved from https://nextjs.org/docs/app (Note: While documentation is useful, for a formal paper, try to find a survey or research paper discussing the benefits of framework features like the App Router if possible).

[5]. Shah, S., & Agarwal, P. (2021). Real-time Communication in Web Applications using WebRTC. International Journal of Computer Applications, 182(4), 1-6.

[6]. Smith, J., & Brown, A. (2023). The Impact of Modern Frontend Frameworks on Web Application Development. Journal of Web Development, 10(2), 45-60.