CHAPTER 5

PROJECT IMPLEMENTATION

5.1. DEFINITION:

This guide outlines the implementation of a video conferencing application using **Next.js**, **Tailwind CSS,UI Acentity, Shad CN**, **Clerk** for authentication, and **GetStream** for real-time video capabilities. The application allows users to create, join, and manage video calls seamlessly.

5.2. Project Setup

- **Create a Next.js Application**:

  Command - **npx create-next-app callify-meet**
- **Install Required Packages**:

  Command - **npm install @clerk/nextjs @stream-io/video-react-sdk**

  @stream-io/node-sdk tailwindcss react-icons @headlessui/react
- **Initialize Tailwind CSS**: Follow the Tailwind CSS installation guide to set up Tailwind in your Next.js project.

5.3. Environment Configuration

- **Create a .env.local file** in the root directory and add the following environment variables:
- **In .env.local –**

  Command –

  NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=

  CLERK_SECRET_KEY=

  NEXT_PUBLIC_CLERK_SIGN_IN_URL=/sign-in

  NEXT_PUBLIC_CLERK_SIGN_UP_URL=/sign-up

  NEXT_PUBLIC_STREAM_API_KEY=

  STREAM_SECRET_KEY=

5.4. User Authentication with Clerk:

- **Set Up Clerk**:
  - Create a Clerk account and set up a new application.
  - Use Clerk's middleware to protect routes and manage user sessions.
- **Middleware Configuration**: Create a **middleware.ts** file:

```
import { clerkMiddleware, createRouteMatcher } from "@clerk/nextjs/se

const protectedRoutes = createRouteMatcher(["/facetime", "/dashboard"

export default clerkMiddleware((auth, req) => {
    if (protectedRoutes(req)) {
        auth().protect();
    }
});

export const config = {
    matcher: ["/((?!.*\\\\..*|_next). *)", "/", "/(api|trpc)(. *)"],
};
```

Figure 5.4.1 – Clerk Authentication

5.5. Integrating GetStream for Video Calls:

- **Set Up Stream Client**: Create a **StreamVideoProvider.tsx** in the **providers** folder:

```
"use client";
import { tokenProvider } from "@/actions/stream.actions";
import { StreamVideo, StreamVideoClient } from "@stream-io/video-reac
import { useState, ReactNode, useEffect } from "react";
import { useUser  } from "@clerk/nextjs";

const apiKey = process.env.NEXT_PUBLIC_STREAM_API_KEY!;

export const StreamVideoProvider = ({ children }: { children: ReactNo
    const [videoClient, setVideoClient] = useState<StreamVideoClient
    const { user, isLoaded } = useUser ();

    useEffect(() => {
        if (!isLoaded || !user || !apiKey) return;
        const client = new StreamVideoClient({
            apiKey,
            user: {
                id: user?.id,
                name: user?.primaryEmailAddress?.emailAddress,
```

```
12
13      useEffect(() => {
14          if (!isLoaded || !user || !apiKey) return;
15          const client = new StreamVideoClient({
16              apiKey,
17              user: {
18                  id: user?.id,
19                  name: user?.primaryEmailAddress?.emailAddress,
20                  image: user?.imageUrl,
21              },
22              tokenProvider,
23          });
24          setVideoClient(client);
25      }, [user, isLoaded]);
26
27      if (!videoClient) return null;
28
29      return <StreamVideo client={videoClient}>{children}</StreamVideo
30  };
```

Figure 5.5.1 – Get Stream Client

5.6. Building the User Interface:

- **Create Modals for Meeting Management**:
    - **CreateLink**: For creating new meetings.
    - **InstantMeeting**: For starting instant meetings.
    - **JoinMeeting**: For joining existing meetings.
    - **UpcomingMeeting**: To view scheduled meetings.

- **Example of CreateLink Modal**:

```
1  "use client";
2  import { Dialog, DialogTitle, DialogPanel, Transition } from "@headle
3  import { Fragment, useState } from "react";
4
5  export default function CreateLink({ enable, setEnable }) {
6      const [description, setDescription] = useState("");
7
8      const handleStartMeeting = async (e) => {
9          e.preventDefault();
10         // Logic to create a meeting
11     };
12
13     return (
14         <Transition appear show={enable} as={Fragment}>
15             <Dialog as="div" onClose={() => setEnable(false)}>
16                 <DialogPanel>
17                     <DialogTitle>Create a Meeting</DialogTitle>
18                     <form onSubmit={handleStartMeeting}>
19                         <input
20                             type="text"
21                             value={description}
```

Figure 5.6.1 – Create Link Model

5.7. Implementing Meeting Functionality

- **Creating and Joining Calls**: Use the Stream SDK to create and join calls

Figure 5.7.1 – Create and Joining Model

- **Joining a Call**: Implement a function to join a call using its ID:

Figure 5.7.2 – Joining Call

5.8. Styling with Tailwind CSS:

- **Apply Tailwind Classes**: Use Tailwind CSS classes to style your components for a modern and responsive design.
- **Apply Shad CN & UI Acentity Classes**: Use Tailwind CSS classes to style your components for a modern and responsive design.

5.9. Deployment:

- **Deploying on Vercel**:
    - Push your code to a GitHub repository.
    - Connect your repository to Vercel and deploy your application.
    - Provide all the environment variables**(.env.local)**