**Project: Interview Online**

* **Implement Webhooks**
* http.ts :-

import { httpRouter } from "convex/server";

import { httpAction } from "./\_generated/server";

import { WebhookEvent } from "@clerk/nextjs/webhooks";

import { Webhook } from "svix";

import {api} from "./\_generated/api";

const http = httpRouter();

http.route({

    path: "/clerk-webhook",

    method: "POST",

    handler: httpAction(async (ctx, request)=>{

        const webhookSecret = process.env.CLERK\_WEBHOOK\_SECRET;

        if(!webhookSecret) {

            throw new Error("Missing CLERK\_WEBHOOK\_SECRET");

        }

        const svix\_id = request.headers.get("svix-id");

        const svix\_signature = request.headers.get("svix-signature");

        const svix\_timestamp = request.headers.get("svix-timestamp");

        if(!svix\_id || !svix\_signature || !svix\_timestamp) {

            new Response("Missing headers", { status: 400 });

        }

        const payload = await request.json();

        const body = JSON.stringify(payload);

        const wh = new Webhook(webhookSecret);

        let evt: WebhookEvent;

        try {

            evt = wh.verify(body,{

                "svix-id": svix\_id || "",

                "svix-signature": svix\_signature || "",

                "svix-timestamp": svix\_timestamp || "",

            }) as WebhookEvent

        } catch (error) {

            console.error("Error verifying webhook", error);

            return new Response("Invalid signature", { status: 401 });

        }

        const eventType = evt.type;

        if(eventType === "user.created"){

            const { id, email\_addresses, first\_name, last\_name, image\_url } = evt.data;

            const email = email\_addresses[0].email\_address;

            const name = `${first\_name || ""} ${last\_name || ""}`.trim();

            try {

                await ctx.runMutation(api.users.syncUser, {

                    clerkId : id,

                    email,

                    name,

                    image : image\_url,

                })

                // await ctx.db.insert("users", {

                //     id,

                //     email: email\_addresses[0]?.email\_address || "",

                //     name,

                //     imageUrl: image\_url || "",

                // });

            } catch (error) {

                console.error("Error creating user", error);

                return new Response("Error creating user", { status: 500 });

            }

        }

        return new Response("WebHook processed successfully",{status : 200});

    }),

})

export default http;

* users.ts
* import { v } from "convex/values";
* import { mutation } from "./\_generated/server";
* export const syncUser = mutation({
* args:{
* name: v.string(),
* email: v.string(),
* clerkId: v.string(),
* image: v.optional(v.string()),
* },
* handler: async(ctx, args)=>{
* const exitingUser = await ctx.db.query("users")
* .filter(q => q.eq(q.field("clerkId"), args.clerkId)).first()
* if(exitingUser) return
* return await ctx.db.insert("users",{
* ...args,
* role: "candidate"
* })
* }
* })