Contents

[Code Snippet: 1](#_Toc209453240)

[Logs: 4](#_Toc209453241)

# Code Snippet:

// Create Supabase session: try recovery then magiclink; else use internal fallback

let sessionData: any = null;

try {

  const genLinkRecRequest: any = { type: 'recovery', email: normEmail, options: { redirectTo: undefined } };

  console.log('Generating link for recovery type request: ', JSON.stringify(genLinkRecRequest));

  const recRes = await client.auth.admin.generateLink(genLinkRecRequest);

  console.log('Recovery link generated at:', new Date().toISOString());

  console.log('Generating link for recovery type response: ', JSON.stringify(recRes));

  if (recRes.error) {

    console.warn('generateLink(recovery) failed:', recRes.error.message);

  } else {

    sessionData = await redeemLink(recRes.data);

  }

  if (!sessionData) {

    const genLinkMagicRequest: any = { type: 'magiclink', email: normEmail, options: { redirectTo: undefined } };

    console.log('Generating link for magiclink type request: ', JSON.stringify(genLinkMagicRequest));

    const magicRes = await client.auth.admin.generateLink(genLinkMagicRequest);

    console.log('Magic link generated at:', new Date().toISOString());

    console.log('Generating link for magiclink type response: ', JSON.stringify(magicRes));

    if (magicRes.error) {

      console.warn('generateLink(magiclink) failed:', magicRes.error.message);

    } else {

      sessionData = await redeemLink(magicRes.data);

    }

  }

} catch (e) {

  console.warn('Session generation error:', (e as any)?.message || e);

}

// Helper: parse and redeem a link (prefer code exchange, fallback to verifyOtp)

const redeemLink = async (linkResult: any) => {

  const actionLink = linkResult?.properties?.action\_link as string | undefined;

  let linkType: string | null = null;

  let authCode: string | null = null;

  let recoveryToken: string | null = (linkResult as any)?.email\_otp?.token ?? null;

  if (actionLink) {

    try {

      const url = new URL(actionLink);

      authCode = url.searchParams.get('code');

      if (!recoveryToken) recoveryToken = url.searchParams.get('token');

      linkType = url.searchParams.get('type');

      const host = url.host;

      console.log('Auth link parsed:', { linkType, hasCode: !!authCode, hasToken: !!recoveryToken, host });

    } catch { }

  }

  if (authCode) {

    const { data: ex, error: exErr } = await publicClient.auth.exchangeCodeForSession(authCode);

    if (!exErr && ex?.session?.access\_token && ex?.session?.refresh\_token) {

      return {

        access\_token: ex.session.access\_token,

        refresh\_token: ex.session.refresh\_token,

        expires\_in: (ex.session as any).expires\_in ?? 3600,

        token\_type: (ex.session as any).token\_type ?? 'bearer',

        user: ex.user ?? authUser,

      };

    }

    if (exErr) console.warn('exchangeCodeForSession failed:', exErr.message);

  }

  if (recoveryToken) {

    const verifyType = (linkType === 'recovery' || linkType === 'magiclink' || linkType === 'invite' || linkType === 'signup') ? (linkType as any) : 'recovery';

    const { data: verified, error: verifyErr } = await publicClient.auth.verifyOtp({ email: normEmail, token: recoveryToken, type: verifyType });

    console.log('verifyOTP completed at:', new Date().toISOString());

    if (!verifyErr && verified?.session?.access\_token && verified?.session?.refresh\_token) {

      return {

        access\_token: verified.session.access\_token,

        refresh\_token: verified.session.refresh\_token,

        expires\_in: (verified.session as any).expires\_in ?? 3600,

        token\_type: (verified.session as any).token\_type ?? 'bearer',

        user: verified.user ?? authUser,

      };

    }

    if (verifyErr) console.warn('verifyOtp failed:', verifyErr.message);

  }

  return null;

};

# Logs:

Generating link for recovery type request: {

  "type": "recovery",

    "email": "sagar.varma465@gmail.com",

      "options": { }

}

Recovery link generated at: 2025 -09 - 22T21: 58: 30.321Z

Generating link for recovery type response: {

  "data": {

    "properties": {

      "action\_link": "https://xxxxxxxxxxxxxxxxxxxx.supabase.co/auth/v1/verify?token=7088e68f75ec4efeb6afd563576b65c19349b321f0d5253a9226718b&type=recovery&redirect\_to=http://localhost:3000",

        "email\_otp": "615496",

          "hashed\_token": "7088e68f75ec4efeb6afd563576b65c19349b321f0d5253a9226718b",

            "redirect\_to": "http://localhost:3000",

              "verification\_type": "recovery"

    },

    "user": {

      "id": "4f6ab3ae-7e1f-4235-a412-281b532ded42",

        "aud": "authenticated",

          "role": "authenticated",

            "email": "sagar.varma465@gmail.com",

              "email\_confirmed\_at": "2025-09-20T20:32:01.879418Z",

                "phone": "",

                  "confirmed\_at": "2025-09-20T20:32:01.879418Z",

                    "recovery\_sent\_at": "2025-09-22T21:58:29.164235559Z",

                      "last\_sign\_in\_at": "2025-09-22T04:01:39.281005Z",

                        "app\_metadata": {

        "provider": "email",

          "providers": [

            "email"

          ]

      },

      "user\_metadata": {

        "email": "sagar.varma465@gmail.com",

          "email\_verified": true,

            "phone\_verified": false,

              "sub": "4f6ab3ae-7e1f-4235-a412-281b532ded42"

      },

      "identities": [

        {

          "identity\_id": "66def39d-b03f-44a7-9c18-6c351a4c3b1e",

          "id": "4f6ab3ae-7e1f-4235-a412-281b532ded42",

          "user\_id": "4f6ab3ae-7e1f-4235-a412-281b532ded42",

          "identity\_data": {

            "email": "sagar.varma465@gmail.com",

            "email\_verified": true,

            "phone\_verified": false,

            "sub": "4f6ab3ae-7e1f-4235-a412-281b532ded42"

          },

          "provider": "email",

          "last\_sign\_in\_at": "2025-09-20T20:26:11.614567Z",

          "created\_at": "2025-09-20T20:26:11.614622Z",

          "updated\_at": "2025-09-20T20:26:11.614622Z",

          "email": "sagar.varma465@gmail.com"

        }

      ],

        "created\_at": "2025-09-20T20:26:11.568058Z",

          "updated\_at": "2025-09-22T21:58:29.167416Z",

            "is\_anonymous": false

    }

  },

  "error": null

}

verifyOTP completed at: 2025 -09 - 22T21: 58: 30.401Z

verifyOtp failed: Token has expired or is invalid

Generating link for magiclink type request: {

  "type": "magiclink",

    "email": "sagar.varma465@gmail.com",

      "options": { }

}

Magic link generated at: 2025 -09 - 22T21: 58: 30.515Z

Generating link for magiclink type response: {

  "data": {

    "properties": {

      "action\_link": "https://xxxxxxxxxxxxxxxxxxxx.supabase.co/auth/v1/verify?token=c10fdd5c683a373467f7687b211b4e0e39ac1ecb51e67e25253210cf&type=magiclink&redirect\_to=http://localhost:3000",

        "email\_otp": "532917",

          "hashed\_token": "c10fdd5c683a373467f7687b211b4e0e39ac1ecb51e67e25253210cf",

            "redirect\_to": "http://localhost:3000",

              "verification\_type": "magiclink"

    },

    "user": {

      "id": "4f6ab3ae-7e1f-4235-a412-281b532ded42",

        "aud": "authenticated",

          "role": "authenticated",

            "email": "sagar.varma465@gmail.com",

              "email\_confirmed\_at": "2025-09-20T20:32:01.879418Z",

                "phone": "",

                  "confirmed\_at": "2025-09-20T20:32:01.879418Z",

                    "recovery\_sent\_at": "2025-09-22T21:58:29.371827856Z",

                      "last\_sign\_in\_at": "2025-09-22T04:01:39.281005Z",

                        "app\_metadata": {

        "provider": "email",

          "providers": [

            "email"

          ]

      },

      "user\_metadata": {

        "email": "sagar.varma465@gmail.com",

          "email\_verified": true,

            "phone\_verified": false,

              "sub": "4f6ab3ae-7e1f-4235-a412-281b532ded42"

      },

      "identities": [

        {

          "identity\_id": "66def39d-b03f-44a7-9c18-6c351a4c3b1e",

          "id": "4f6ab3ae-7e1f-4235-a412-281b532ded42",

          "user\_id": "4f6ab3ae-7e1f-4235-a412-281b532ded42",

          "identity\_data": {

            "email": "sagar.varma465@gmail.com",

            "email\_verified": true,

            "phone\_verified": false,

            "sub": "4f6ab3ae-7e1f-4235-a412-281b532ded42"

          },

          "provider": "email",

          "last\_sign\_in\_at": "2025-09-20T20:26:11.614567Z",

          "created\_at": "2025-09-20T20:26:11.614622Z",

          "updated\_at": "2025-09-20T20:26:11.614622Z",

          "email": "sagar.varma465@gmail.com"

        }

      ],

        "created\_at": "2025-09-20T20:26:11.568058Z",

          "updated\_at": "2025-09-22T21:58:29.373952Z",

            "is\_anonymous": false

    }

  },

  "error": null

}

verifyOTP completed at: 2025 -09 - 22T21: 58: 30.583Z

verifyOtp failed: Token has expired or is invalid