



Duo Sync Auth

Secure Authentication & Real-Time Data Synchronization

Fall 2025 Senior Design Project

Knight Foundation School of Computing and Information Sciences – FIU

Students: Matthew Chaar, Eduardo Parrado Arango, Joshua Aranzazu, Nelson Rodriguez, Joshua Conte

Instructor/Faculty: Seyed Masoud Sadjadi



What is Duo Sync Auth?

- A modern, scalable web application that simulates push-based 2FA (like Duo)
- Implements a secure authentication flow and real-time data sync using Supabase.
- Provides a clean “School Portal” demo so users can see how 2FA approvals work in real time.



Why 2FA Matters

- Single-factor logins (just username + password) are no longer enough.
- Modern attacks include phishing, credential stuffing, and data breaches that regularly expose passwords.
- Industry best practice now is Two-Factor Authentication (2FA) to build real trust.
- Push-based authentication (like Duo) balances strong security with a smooth user experience.





Project Summary

- Goal: build a full-stack web app that accurately simulates a real-time, push-based 2FA workflow.
- Demonstrates three main skills:
- Modern Frontend Development – responsive interfaces for desktop login and mobile authenticator.
- Real-Time Data Synchronization – instant updates using Supabase Realtime instead of polling.
- Enterprise UX Design – simple, trustworthy UI that feels like a real security product.





2FA Security Demo – How It Looks

- Desktop Login – simulate logging in from a computer.
- Mobile Authenticator – review and approve or deny login requests in real time.

How it works:

- Enter email on the desktop login page.
- Open mobile authenticator to see pending request.
- Approve or deny the request.
- Watch the desktop UI update instantly when the status changes.





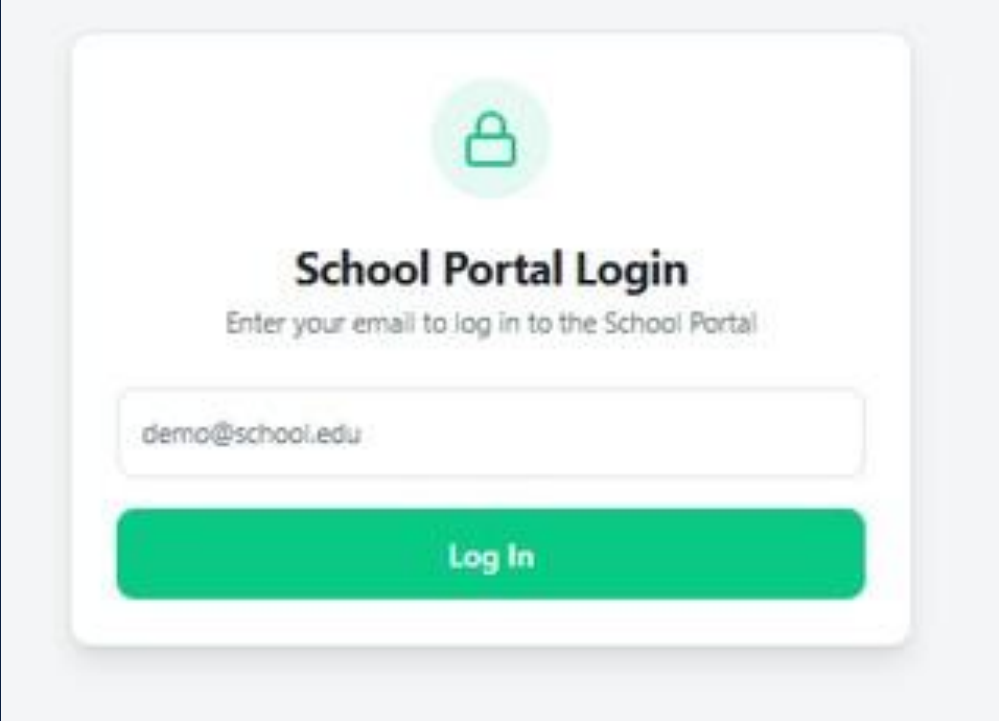
Technology Stack

- Poster (1)
- Frontend
 - React 18 (Single Page App)
 - TypeScript for type safety
 - Vite for fast HMR and production bundling
 - Tailwind CSS for styling
 - shadcn/ui for UI components
- Backend / BaaS
 - Supabase platform
 - PostgreSQL database
 - GoTrue Authentication
 - Realtime (WebSockets) for push updates



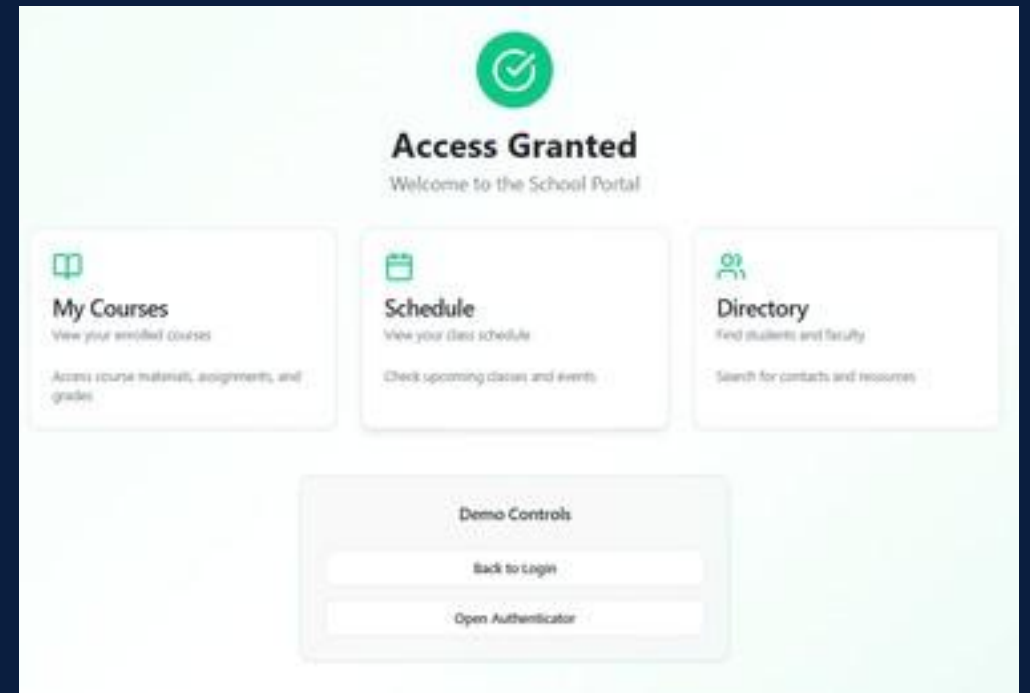
School Portal Login

- User enters their school email and submits.
- App validates the email and creates a new row in auth_requests with status pending.
- UI switches to a “Waiting for approval” state with a spinner and toast notification.
- From this moment on, the page is listening for real-time updates from Supabase.

A mockup of a 'School Portal Login' form. At the top is a green circular icon containing a white padlock. Below the icon, the title 'School Portal Login' is displayed in bold black text, followed by the subtitle 'Enter your email to log in to the School Portal' in a smaller, lighter font. A text input field contains the email address 'demo@school.edu'. Below the input field is a large, rounded green button with the text 'Log In' in white.

Access Granted – School Portal

- When the mobile authenticator approves the request:
- Supabase updates the auth_requests row to approved.
- The real-time subscription receives the update.
- React redirects the user to the “Access Granted” dashboard automatically.
- Dashboard cards simulate a real school portal:
- My Courses, Schedule, Directory, plus demo controls.



System Architecture

- Client Layer (Browser):
 - React SPA handles routing, UI, and user interactions.
 - TanStack Query (React Query) manages data fetching and caching.
- Supabase Backend:
 - GoTrue Auth issues JWT sessions and enforces Row Level Security.
 - PostgreSQL stores 2FA requests in auth_requests.
 - Realtime service broadcasts changes to subscribed clients via WebSockets.
 - No custom backend server – the client talks directly to Supabase over HTTPS/WebSockets.





Database Design (auth_requests)

- Single main table: auth_requests in the public schema.
- Columns:
 - id – unique ID for each auth request (string).
 - email – user's email address.
 - status – "pending", "approved", or "rejected".
 - created_at – timestamp of when the request was created.
- Strong TypeScript types ensure:
 - Valid shape for Row, Insert, and Update operations.
 - Fewer runtime errors when talking to Supabase.





Supabase Client Setup

- Bullets:
 - `createClient<Database>` from `@supabase/supabase-js` is used to create a typed client.
- Environment variables:
 - `VITE_SUPABASE_URL` – project URL.
 - `VITE_SUPABASE_PUBLISHABLE_KEY` – public key used on the frontend.
- Auth config:
 - Uses `localStorage` to store session.
 - `persistSession: true` so the user stays logged in between refreshes.
 - `autoRefreshToken: true` to keep JWTs fresh in the background.
- Client is imported across the app as:
 - `Import { supabase } from "@integrations/supabase/client";`





How the Login Flow Works (Code Behind)

- `handleLogin`:
 - Prevents default form submit.
 - Validates that email is not empty and contains "@".
 - Inserts { email, status: "pending" } into `auth_requests`.
 - Saves returned id in `requestId` and shows "Waiting for approval..." toast.
- Realtime subscription (`useEffect`):
 - Subscribes to `postgres_changes` on the `auth_requests` table.
 - Filters by `id = requestId` so only the current request is tracked.
 - If `status === "approved"` → success toast + redirect to `/dashboard`.
 - If `status === "rejected"` → error toast + reset waiting state.
- Cleanup:
 - Channel is removed on unmount to avoid memory leaks.
 - If you want a snippet on the slide, show just the `supabase.channel('auth-status-changes').on(...).subscribe()` part.





UX Improvements with Custom Hooks

- seToast
 - Centralized toast system with an internal reducer.
 - Limits active toasts with `TOAST_LIMIT = 1` to avoid spamming the user.
 - Supports actions for add, update, dismiss, and remove.
- Used for:
 - Invalid email warnings,
 - “Waiting for approval...” info,
 - “Access approved” / “Access denied” alerts.
 - `useIsMobile`
 - Uses `window.matchMedia('(max-width: 767px)')`.
 - Tracks `isMobile` so layout and spacing can adapt on smaller screens.
 - Helps keep the 2FA demo responsive across devices.





Example Test Cases

- Login Validation Test
 - Input: invalid email (missing @).
 - Expected: toast “Please enter a valid email”, no row created.
- Pending to Approved Flow
 - Input: user submits email, then mobile marks request approved.
 - Expected: status changes to approved, desktop redirects to dashboard.
- Pending to Rejected Flow
 - Input: user submits email, mobile hits rejected.
 - Expected: error toast, login state resets, user can try again.
- Realtime Subscription Test
 - Input: manually update auth_requests row in Supabase.
 - Expected: desktop UI reacts immediately without refresh.





Limitations & Future Improvements

- Ran the flow end-to-end:
- Desktop login → pending state → mobile approval/denial → dashboard redirect.
- Tested approved vs rejected scenarios to ensure correct UI behavior.
- Verified that:
 - auth_requests rows are created correctly in PostgreSQL.
 - Realtime updates arrive only for the correct requestId.
 - Old subscriptions are cleaned up when leaving the page.





Conclusion

- Duo Sync Auth shows a complete push-based 2FA simulation using modern web tech.
- Combines React + TypeScript + Tailwind + shadcn/ui on the frontend with Supabase (Auth, PostgreSQL, Realtime) on the backend.
- Demonstrates:
 - Real-time data sync,
 - Clean UX,
 - Strong architecture and typed database access.
- Questions?

