

Rewind: Event Campaign Site and Signup Platform

Purpose

Rewind was The Forum's back-to-school party for SFU students, held at Celebrities Nightclub on September 20, 2026. The club needed one site to run the whole campaign: collect a mailing list before tickets went on sale, run two capped discount guest lists distributed privately by promoters, switch to selling tickets the moment the drop happened, and measure which ads were producing signups.

General ticket sales had to run through TicketWeb, which holds the ticketing contract with the venue and takes a large cut of every ticket. Discounted presale and friends-and-family entry was ours to handle directly, and that is the part the site owns outright: door entry at \$5 with no fees, against \$30 all-in through the platform. So it is two systems: a link out to TicketWeb for general sales, and the system of record for everything sold around it.

My role

I was the only person writing code, and I owned the architecture, the database schema, the API, the front end, the deployment and the image pipeline. Everyone else was non-technical: they supplied copy, poster art, photography and event logistics, and they were the source of most requirement changes. Turning those into something that would not break when they changed shaped the design more than any single technical decision.

What I built

A React 19, TypeScript and Vite single-page app with a scroll-driven landing experience, deployed on Vercel, with serverless API routes backed by Postgres (Supabase), Brevo for email, and Meta's Pixel and Conversions API for attribution. Roughly 15,000 lines of TypeScript and 850 of SQL. The schema is managed as versioned, re-runnable migrations, and of the 397 tests, some read the SQL files directly and fail if a database function's signature stops matching the code that calls it.

Challenges

The event was rebranded halfway through, and the artwork did not exist yet. It started as "Back 2 School" and became "Rewind", a Y2K-themed night, after the site was already built. The posters were coming from an external designer and were not ready while I was designing the pages that would eventually carry them. I could not wait, so I built for the substitution: the key art is a swappable layer rather than a section in the page flow, every string and price lives in one config file, and the palette is a token set I re-sampled from the poster once it arrived. The rebrand landed as content changes rather than a rewrite. The two hero screenshots in the appendix are the same codebase six weeks apart.

Releasing it in pieces while the campaign was already running. Everything had to be live before students came back, and the club wanted it up as early as possible. I was writing finals during the same weeks, and the people just hired to help were not onboarded yet, because there was no time to bring anyone up to speed on a project this size before launch. The work stayed mine start to finish.

So instead of waiting until all of it was done, I released in the order the campaign actually needed things. The presale guest list went up first, then the general sale through TicketWeb, then friends and family. That only works if a new piece can go live without disturbing the ones already taking signups, which is why the lists are configuration rather than code. Adding the third weeks later was one config entry and one migration, not a change to two lists already taking real traffic.

Two people taking the last spot on a capped list. Each guest list had 75 spots, and the links went out in group chats, so people signed up in bursts rather than one at a time. That breaks the obvious approach. If the code counts the existing signups and then inserts a new one, two requests arriving together both count 74, both decide there is room, and the list ends up at 76. Every spot over the cap is a real cost, because the club absorbs the difference at the door.

I put the whole operation into a single database function that locks the list's row before it does anything else. The duplicate check, the signup window, the age check, rate limiting, the capacity check and the insert all happen while that lock is held, so two people submitting at the same moment get handled one after the other. Different lists never block each other. The email and tracking calls run after the lock is released, because waiting on another company's server while holding a lock

would stall everyone queued behind it. When friends and family was added I gave it its own pool, so whichever link circulated faster could not take all 75 spots from both audiences.

An email platform that rejected whole contacts. Brevo refuses a contact outright when its phone number collides with one already on another contact, or is not a real line. It rejects the entire record, so one phone conflict was keeping someone off the mailing list when the email address was the only part actually needed. The sync now detects that specific rejection and retries once without the phone number, under a single deadline covering both attempts so the retry cannot double the response time.

Behind that, I treated Postgres as the source of truth rather than the email platform. The signup row is committed before any third party is contacted, and each row carries a flag recording whether the Brevo sync actually succeeded, so an outage leaves a row marked unsynced for backfill instead of costing a signup. Before the door I exported that table to CSV so the executives could cross-reference it against Brevo directly, rather than either system being trusted on its own.

A form that vanished while people were typing in it. On mobile, tapping a field in the bottom-pinned signup block opened the keyboard, and the browser scrolled the document to lift the input clear of it. On the tickets page that scroll landed on the steepest part of the animation: roughly 15 pixels stopped the block taking taps, and roughly 150 removed it from the screen entirely, mid-word. The fix holds the animation in place while focus is inside the block and restores the scroll position afterwards, since that scroll was never asked for, while a deliberate swipe or drag still lets the page move normally.

Communicating technical trade-offs to people who do not build software. Most requests reached me as a finished conclusion rather than a problem to solve, and some of them cut against basic usability. I could not simply refuse: they knew the audience far better than I did, and it was their event. What worked was laying out the trade-off in plain language: what a choice would cost, what it would buy, and what I expected to happen, all before anyone committed. The principle I settled on is that there is no wrong choice, only a misinformed one. Once a decision was made that way it held, and I could build on it instead of revisiting it later.

Outcome

The site ran the full campaign end to end, from the pre-drop mailing list through the ticket drop to the event on September 20, which sold more than 600 tickets. All of it ran in production against real traffic, which is where most of the problems above were found.

What I would do differently

More of this should have happened before I started writing code. I would spend considerably more time on requirements with the club, and ask more questions rather than fewer, even when one feels obvious. The habit I picked up too late is to book a second meeting after attempting the implementation, because issues always surface once something real exists that nobody thought of while it was only being described.

I would also give the planning phases more room, both the visual design and the database schema. The schema in particular decided what I could report on later, and I would design it to capture more than I think I need. It is much easier to ignore a column than to go back for information that was never recorded.

The pages that were live

rewind.theforumuniversity.com The landing page. Before the ticket drop it collected the mailing list and counted down; from the drop onward it forwards to the ticket page.

rewind.theforumuniversity.com/tickets The sales page after the drop. It opens directly on the campaign artwork with the ticket link and a short mailing-list form, with no hero or countdown in front of it.

rewind.theforumuniversity.com/list-rphjpp4n The promoter presale guest list: 75 discounted door spots, closed August 25.

rewind.theforumuniversity.com/list-8h7uimig The friends and family guest list. Same page and same door rate, but its own separate pool of 75, closed September 19.

rewind.theforumuniversity.com/vip A VIP guest form hosted by Brevo and served as a static file outside the app entirely. Closed at 8:00 PM on September 18.

rewind.theforumuniversity.com/confirmed Where every guest list submission lands after a successful signup.

rewind.theforumuniversity.com/vip-confirmed Where the VIP form redirects after submission, set from Brevo's dashboard rather than from the app.

rewind.theforumuniversity.com/privacy The privacy policy, linked from the consent line above every submit button. The exact sentence each person agreed to is stored with their row.

ticketweb.ca/event/rewind-sfus-biggest-back-to-celebrities-nightclub-tickets/15038423 Where general ticket sales actually happened, off-site.

Appendix: screenshots

Captured from the repository at the commits named, run locally with the browser clock set to the campaign dates so each page renders the state it showed at the time.



August 4, "Back 2 School"



September 18, "Rewind"

The rebrand, six weeks apart in the same codebase. The finished hero is a draggable split comparison between two versions of the character, controlled by mouse, touch, or the arrow keys. The date in the first shot still reads Sept 26, a stale value later corrected to the real door date.

We will be playing your favourite songs throughout the last decades, where you will be able to sing along!

Sunday, September 20. Celebrities Nightclub. Doors 9:30 PM.

REGULAR DOOR ENTRY	WITH DISCOUNTED GUESTLIST CHECKED IN BEFORE 10:30 PM
\$30 With fees	\$5 No fees

SAVE \$25

CHECKED IN AFTER 10:30 PM
You will be charged the regular discounted guestlist rate at \$15. That is still half off!

GET ON THE LIST

Enter the name you will give at the door.

FIRST NAME _____ LAST NAME _____

EMAIL _____

Please avoid using an @sfu.ca address.

PHONE _____

DATE OF BIRTH _____

Month _____ Day _____ Year _____

The discounted guest list: price panel and the start of the form

This is the page the capacity logic protects, and the reason the club wanted it: \$5 at the door with no fees against \$30 all-in through the platform. Every price is computed from one figure in config rather than written into copy, so an unconfirmed fee could move without leaving a wrong number on the page.


THE FORUM

[TICKETS]



#

REWIND

SFU'S BIGGEST BACK TO SCHOOL PARTY

SEPT 20

GET TICKETS

ACCESS EXCLUSIVE PRE-SALES, DISCOUNTS & GIVEAWAYS FOR FUTURE EVENTS

First name

Email

JOIN

By signing up, you are agreeing to our Privacy Policy, and agree to receiving promotional emails from The Forum. You can unsubscribe anytime.

THE FORUM UNIVERSITY 2026

PRIVACY POLICY

The same site after the drop instant, on /tickets

No hero and no countdown: this route opens on the artwork with the ticket link and mailing strip already on screen. The landing page redirects here from the drop onward and back the other way before it, off a single shared predicate, so a tab left open at 11:59 moves itself at noon without a deploy.

SFU's Biggest Back to School Party
Turn up with 800+ students at Vancouver's favourite nightclub.

TICKETS DROP IN
4^d 00^h 00^m 00^s

RSVP ↓



Mobile at 390px, the layout the keyboard bug was found on