Architettura e Implementazione di Sistemi Elettorali Distribuiti: Il Progetto Democracy is Dead

La convergenza tra la teoria della scelta sociale e lo sviluppo software moderno ha aperto nuove frontiere per la creazione di piattaforme decisionali collaborative che superano i limiti dei sistemi di voto tradizionali. Il progetto denominato "Democracy is Dead" si propone di trasformare un'esperienza soggettiva, come la valutazione sensoriale di prodotti alimentari, in un risultato collettivo matematicamente rigoroso e visivamente coinvolgente. Attraverso l'impiego di tecnologie allo stato dell'arte come Next.js 15 e Supabase, è possibile implementare un'infrastruttura serverless a costo zero capace di gestire lobby interattive in tempo reale, algoritmi elettorali complessi e analisi statistiche avanzate. Questa analisi fornisce una panoramica tecnica esaustiva e una guida operativa per la realizzazione di tale piattaforma, strutturata per garantire scalabilità, equità e un'esperienza utente priva di attriti.

Fondamenti della Teoria della Scelta Sociale e Algoritmi Elettorali

La progettazione di un sistema di voto efficace richiede una comprensione profonda dei meccanismi matematici che regolano l'aggregazione delle preferenze individuali. La democrazia, intesa come processo decisionale collettivo, è soggetta a paradossi intrinseci che possono influenzare drasticamente l'esito finale a seconda delle regole applicate. Il Teorema di Impossibilità di Arrow dimostra che nessun sistema di voto basato su ranghi può soddisfare contemporaneamente tutti i criteri di equità quando ci sono tre o più opzioni. Pertanto, l'applicazione "Democracy is Dead" non si limita a un singolo metodo, ma offre una suite di algoritmi per mostrare come diverse prospettive matematiche possano produrre vincitori differenti dallo stesso set di dati.

Il Metodo Schulze e la Ricerca del Vincitore di Condorcet

Il metodo Schulze, noto anche come "beatpath method", rappresenta uno dei sistemi di voto più sofisticati e robusti attualmente disponibili per il vincitore unico. Sviluppato da Markus Schulze nel 1997, questo sistema è classificato come un metodo di Condorcet, il che significa che se esiste un candidato che batterebbe ogni altro rivale in una sfida testa a testa, tale candidato verrà eletto. La sua caratteristica distintiva è la capacità di risolvere i cicli del paradosso di Condorcet attraverso l'analisi della forza dei percorsi vincenti tra i candidati.
Matematicamente, il metodo Schulze è implementato come un problema del cammino più largo su un grafo pesato. Per ogni coppia di candidati i e j, si calcola il numero di elettori d[i,j] che preferiscono i a j. La forza di un percorso è definita dal suo anello più debole, ovvero la vittoria con il margine minimo lungo quel cammino. Utilizzando una variante dell'algoritmo di Floyd-Warshall, il sistema identifica il percorso più forte p[i,j] tra ogni coppia:
Un candidato A sconfigge B se p > p. Questo metodo è estremamente resistente al voto strategico e garantisce che l'aggiunta di candidati "cloni" non alteri l'esito dell'elezione,

rendendolo ideale per valutazioni complesse dove i prodotti potrebbero presentare caratteristiche simili.

Il Conteggio di Borda e la Ricerca del Consenso

Mentre i metodi di Condorcet si concentrano sulle vittorie testa a testa, il conteggio di Borda adotta un approccio posizionale volto a identificare il candidato che raccoglie il maggior consenso complessivo. In un'elezione con n candidati, il primo classificato in una scheda riceve n-1 punti, il secondo n-2, e così via fino all'ultimo che riceve 0 punti. Questo sistema premia i candidati che sono ampiamente accettati dal gruppo, anche se non sono necessariamente la prima scelta assoluta per la maggioranza.
Il metodo di Borda è ampiamente utilizzato in contesti accademici e sportivi, come l'assegnazione del trofeo Heisman o del premio MVP nel baseball, proprio per la sua capacità di riflettere l'opinione media dell'elettorato. Tuttavia, è vulnerabile a manipolazioni tattiche come il "burial", dove i votanti declassano artificialmente il principale rivale del loro candidato preferito per ridurne il punteggio totale.

Analisi Comparativa dei Sistemi di Voto

Sistema	Tipo	Criterio Principale	Vulnerabilità Tipica
Pluralità	Nominale	Vincitore con più voti al primo posto	Effetto "spoiler" e frammentazione del voto
Borda	Posizionale	Massimizzazione del consenso medio	Manipolazione strategica dei ranghi bassi
Condorcet	Binario	Vincitore di tutte le sfide testa a testa	Possibilità di cicli senza un vincitore chiaro
Schulze	Grafo	Forza dei percorsi di vittoria indiretta	Complessità di calcolo per grandi set di dati
IRV	Eliminazione	Maggioranza assoluta tramite ballottaggi	Può eliminare candidati di compromesso solidi

Architettura del Sistema: Next.js 15 e il Backend Serverless

L'infrastruttura tecnologica di "Democracy is Dead" deve rispondere a requisiti di bassa latenza, scalabilità orizzontale e assenza di costi operativi fissi. L'approccio moderno prevede il disaccoppiamento tra il frontend reattivo e il backend gestito tramite servizi BaaS (Backend-as-a-Service).

Next.js 15 e React Server Components (RSC)

L'adozione di Next.js 15 rappresenta un salto paradigmatico nella gestione delle applicazioni web. Grazie all'App Router, l'applicazione può eseguire la logica di autenticazione e il recupero dei dati iniziali sul server, riducendo drasticamente il tempo di caricamento percepito (Time to Interactive). I React Server Components permettono di trasmettere il contenuto al client in modo

incrementale, ottimizzando le prestazioni anche su connessioni mobili instabili.
L'integrazione di Next.js con il runtime di Vercel consente l'esecuzione di Edge Functions, ovvero funzioni serverless che girano fisicamente vicino all'utente. Questo è fondamentale per i calcoli onerosi degli algoritmi elettorali, che possono essere eseguiti in pochi millisecondi senza bloccare l'interfaccia utente.

Supabase: La Potenza di PostgreSQL in Tempo Reale

Per un progetto che richiede integrità dei dati e relazioni complesse (voti, candidati, criteri, utenti), un database relazionale è superiore a soluzioni NoSQL. Supabase offre un'istanza PostgreSQL completa con estensioni native per il tempo reale e la sicurezza.

1. **Autenticazione Anonima**: Indispensabile per garantire un accesso fluido durante le serate di degustazione. Il sistema genera sessioni temporanee legate a un ID univoco, permettendo agli utenti di votare senza creare account, pur garantendo l'univocità del voto.
2. **PostgreSQL Realtime**: Attraverso i canali di trasmissione basati su WebSocket, Supabase notifica istantaneamente tutti i client connessi quando viene inserito un nuovo voto o quando un partecipante cambia il proprio stato nella lobby.
3. **Row Level Security (RLS)**: Questa funzionalità permette di definire politiche di accesso ai dati direttamente nel database. Ad esempio, una politica RLS può garantire che un utente possa visualizzare i voti degli altri solo dopo che la sessione è stata chiusa dal creatore.

Design del Database e Logica di Gestione della Lobby

La struttura dei dati è il cuore dell'applicazione. Ogni lobby deve memorizzare non solo i partecipanti, ma anche la configurazione dinamica dei criteri di voto e dei relativi pesi.

Modello Relazionale e Gestione JSONB

L'utilizzo del tipo di dato JSONB in PostgreSQL permette di gestire strutture arbitrarie per i criteri di valutazione. In una degustazione di pizza, i fattori possono includere la qualità della crosta, il topping e il rapporto qualità-prezzo, ognuno con un peso percentuale differente.

```
-- Definizione della tabella Lobby
CREATE TABLE lobbies (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  code TEXT UNIQUE NOT NULL, -- Codice di accesso a 4-6 cifre
  settings JSONB NOT NULL, -- Pesi dei fattori e tipo di algoritmo
  status TEXT DEFAULT 'waiting', -- waiting, voting, results
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Tabella Candidati (es. Buitoni, Italpizza)
CREATE TABLE candidates (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  lobby_id UUID REFERENCES lobbies(id) ON DELETE CASCADE,
  name TEXT NOT NULL,
```

```
  description TEXT,
  image_path TEXT -- Riferimento a Supabase Storage
);

-- Tabella Voti con integrità referenziale
CREATE TABLE votes (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  lobby_id UUID REFERENCES lobbies(id) ON DELETE CASCADE,
  voter_id UUID NOT NULL, -- ID anonimo dell'utente
  candidate_id UUID REFERENCES candidates(id) ON DELETE CASCADE,
  scores JSONB NOT NULL, -- { "crosta": 8, "topping": 6 }
  is_jolly BOOLEAN DEFAULT false -- Voto bonus per l'emozione
superiore
);
```

La Funzione "Remix" e la Clonazione delle Elezioni

Una delle funzionalità avanzate richieste è la possibilità di clonare un'elezione esistente. Questo permette a un nuovo utente di avviare una degustazione identica (stessi candidati, stessi pesi) con un nuovo gruppo di amici. Tecnicamente, questo viene implementato tramite una funzione PL/pgSQL che esegue una copia profonda dei record.
La logica del "Remix" prevede:

1. La creazione di una nuova entry nella tabella lobbies.
2. L'inserimento ciclico dei candidati associati alla lobby originale nella nuova lobby.
3. Il mantenimento dei metadati e delle configurazioni dei fattori di voto.

Sincronizzazione in Tempo Reale e Presenza degli Utenti

Per rendere l'esperienza collettiva, è fondamentale visualizzare chi è presente nella stanza e chi ha completato le proprie valutazioni. Supabase Presence gestisce questo stato in memoria, evitando carichi inutili sul database.

Implementazione del Tracking della Presenza

Quando un utente entra nella lobby, il client sottoscrive un canale e "traccia" il proprio stato iniziale.

```
const lobbyChannel = supabase.channel(`lobby_${lobbyId}`, {
  config: { presence: { key: userId } }
});

lobbyChannel
 .on('presence', { event: 'sync' }, () => {
    const state = lobbyChannel.presenceState();
    // Aggiorna l'interfaccia con la lista degli utenti online
  })
```

```
.subscribe(async (status) => {
   if (status === 'SUBSCRIBED') {
     await lobbyChannel.track({
       nickname: userNickname,
       avatar: avatarUrl,
       is_ready: false
     });
   }
 });
```

Questo meccanismo permette di visualizzare indicatori di attività, come "Sta scrivendo..." o "Voto in corso", aumentando il coinvolgimento durante l'assaggio. Inoltre, l'utilizzo di Supabase Broadcast consente l'invio di messaggi istantanei (chat della lobby) e reazioni emoji che appaiono temporaneamente sugli schermi di tutti i partecipanti.

Analisi Statistica e Normalizzazione dei Voti

Un problema fondamentale nelle valutazioni umane è la soggettività delle scale di punteggio. La normalizzazione statistica è necessaria per garantire che il voto di un amico "severo" non sia meno influente del voto di un amico "generoso".

Z-Score: La Standardizzazione delle Valutazioni

Lo Z-score (o standardizzazione) trasforma i voti di ogni utente in modo che la loro distribuzione abbia media 0 e deviazione standard 1. In questo modo, ciò che conta non è il valore assoluto del voto, ma quanto quel voto si discosta dal comportamento medio di quell'utente specifico.
$$z = \frac{x - \mu}{\sigma}$$
Ad esempio, se un utente assegna solitamente voti tra 4 e 6, un 7 sarà considerato un punteggio eccellente (alto Z-score). Se un altro utente assegna voti tra 8 e 10, un 8 sarà considerato un punteggio mediocre (Z-score negativo o vicino allo zero).

Min-Max Scaling per la Visualizzazione Finale

Dopo il calcolo dei punteggi normalizzati tramite Z-score, i risultati vengono riportati su una scala comprensibile (es. 0-100) utilizzando il Min-Max Scaling. Questo processo garantisce che il vincitore finale sia presentato con un punteggio leggibile, pur mantenendo il rigore della pesatura statistica effettuata a monte.

Metodo	Applicazione	Vantaggio	Svantaggio
Z-Score	Calcolo interno	Bilancia generosità e severità degli elettori	Risultati iniziali poco intuitivi (numeri negativi)
Min-Max	Visualizzazione UI	Riporta i dati in un intervallo familiare	Molto sensibile ai valori anomali (outliers)
Media Pesata	Fattori multicriterio	Permette di dare più importanza a crosta o topping	Non corregge le differenze tra i profili degli elettori

Interfaccia Utente e Visualizzazione dei Dati

L'interfaccia utente deve essere "mobile-first", dato che la maggior parte delle degustazioni avviene durante eventi fisici dove gli smartphone sono lo strumento principale di interazione.

Generazione di Avatar Deterministici (DiceBear)

Per personalizzare l'esperienza senza richiedere il caricamento di immagini pesanti, il sistema integra l'API DiceBear. Utilizzando il nickname scelto dall'utente come "seed", vengono generati istantaneamente avatar in stile "Mii" o "Avataaars". Questi avatar sono generati lato client tramite SVG, garantendo una velocità di caricamento estrema e una coerenza visiva accattivante.

Dashboard dei Risultati e Grafici Radar

La presentazione dei risultati non deve limitarsi a una semplice classifica. L'uso di Chart.js (o Recharts per Next.js) permette di creare grafici radar per ogni candidato, evidenziando i punti di forza e di debolezza secondo i criteri definiti.

- **Grafico Radar**: Mostra la performance di una pizza su assi multipli (Crosta, Topping, Cottura, Prezzo). È ideale per identificare il prodotto con il "fattore più marcato".
- **Bar Chart Dinamico**: Permette di visualizzare come cambia la classifica selezionando diversi algoritmi (Schulze vs Borda), offrendo uno spunto educativo sui paradossi della democrazia.

Guida Operativa alla Realizzazione: Step-by-Step

La realizzazione del progetto richiede una sequenza precisa di passaggi tecnici, dalla configurazione dell'ambiente alla logica di calcolo. Il tempo stimato per un MVP (Minimum Viable Product) funzionale è di circa **40 ore** di sviluppo effettivo.

Fase 1: Setup dell'Ambiente e Database (Tempo stimato: 4 ore)

Il punto di partenza è l'istanziazione del progetto Next.js e la configurazione del backend Supabase.

1. Inizializzare Next.js 15 con TypeScript e Tailwind CSS: npx create-next-app@latest.
2. Configurare Supabase: creare il progetto nella dashboard, ottenere l'URL e la chiave Anon.
3. Definire lo schema SQL nell'editor di Supabase, includendo le tabelle lobbies, candidates, votes e lobby_messages.
4. Abilitare il Realtime sulle tabelle interessate tramite il comando ALTER PUBLICATION.

Fase 2: Autenticazione e Lobby (Tempo stimato: 8 ore)

Questa fase si concentra sull'accesso degli utenti e sulla sincronizzazione della stanza.

1. Implementare il client Supabase SSR in src/utils/supabase/ per gestire correttamente i cookie nel Next.js App Router.
2. Creare la logica di login anonimo: supabase.auth.signInAnonymously().

3. Sviluppare l'interfaccia della lobby con tracking di presenza tramite l'hook usePresence.
4. Integrare la generazione di QR Code per l'invito rapido utilizzando la libreria react-qr-code.

Fase 3: Moduli di Voto e Logica Multicriterio (Tempo stimato: 10 ore)

Qui viene costruita l'interfaccia di inserimento dati.

1. Sviluppare il form di voto dinamico che genera slider in base ai criteri salvati nel JSONB della lobby.
2. Implementare la logica del "Voto Jolly" per permettere agli utenti di assegnare un bonus unico a un candidato preferito.
3. Configurare le politiche RLS per assicurare che ogni utente possa inserire o aggiornare (upsert) solo il proprio voto.

Fase 4: Algoritmi Elettorali e Statistiche (Tempo stimato: 12 ore)

Questa è la parte più densa dal punto di vista matematico.

1. Integrare le librerie di calcolo come caritat o votes per processare i risultati Schulze e Borda.
2. Scrivere le funzioni di utilità per il calcolo dello Z-score in JavaScript, iterando sull'array di oggetti dei voti.
3. Creare la dashboard dei risultati con grafici radar reattivi che si aggiornano tramite sottoscrizione Realtime a ogni nuovo voto inserito.

Fase 5: Gamification e Deployment (Tempo stimato: 6 ore)

Finiture e pubblicazione.

1. Sviluppare la logica degli "Awards" ironici (es. "Il Critico Spietato") analizzando le medie e le deviazioni standard dei partecipanti.
2. Implementare il sistema di chat della lobby utilizzando Supabase Broadcast per commenti in tempo reale.
3. Effettuare il deployment su Vercel, configurando le variabili d'ambiente per il collegamento con Supabase.

Gamification e Analisi del Comportamento degli Elettori

Per rendere l'applicazione non solo utile ma anche divertente, "Democracy is Dead" introduce dinamiche social e titoli competitivi.

Election Awards: Titoli Basati sui Dati

Al termine di ogni elezione, il sistema analizza i dati per assegnare riconoscimenti basati sul profilo psicologico di voto emerso dalla statistica:

- **The Oracle**: L'utente i cui voti individuali riflettono più fedelmente il vincitore finale del metodo Schulze. Viene calcolato tramite la correlazione dei ranghi.
- **The Merciless Critic**: Identificato tramite lo Z-score più basso rispetto alla media globale,

rappresenta l'utente che è stato sistematicamente più severo nelle valutazioni.
- **The Incurable Optimist**: L'utente con la media voti più alta, che tende a valorizzare ogni prodotto assaggiato.
- **The Contrarian**: Colui che ha espresso preferenze più discordanti rispetto al consenso del gruppo, identificabile tramite una deviazione standard elevata rispetto alla media collettiva.

Leaderboard Globale e Punti Democrazia

Per le lobby pubbliche, il sistema prevede una classifica globale. Gli utenti guadagnano "Punti Democrazia" creando lobby popolari o partecipando attivamente a votazioni pubbliche. Questo incentiva la condivisione virale della piattaforma e la creazione di template riutilizzabili per diversi tipi di prodotti, non limitandosi alla pizza surgelata.

Considerazioni su Sicurezza e Integrità del Dato

In un contesto dove il voto è anonimo e distribuito, la sicurezza deve essere gestita a livello infrastrutturale.

Protezione del Voto e Prevenzione della Manipolazione

Sebbene l'autenticazione sia anonima, è essenziale impedire che un utente malintenzionato possa alterare i risultati.

- **Integrità del Calcolo**: Gli algoritmi elettorali dovrebbero essere eseguiti in una Supabase Edge Function o lato server tramite Next.js Server Actions. Questo garantisce che il codice di calcolo non sia visibile né modificabile nel browser dell'utente.
- **Validazione dei Voti**: Utilizzando i trigger di PostgreSQL, è possibile verificare che un utente non abbia assegnato più di un "Jolly" o che i punteggi rientrino nel range stabilito (es. 1-10) prima che l'inserimento venga confermato.

Privacy e Gestione dei Cookie

Poiché l'app si basa sull'autenticazione anonima, la persistenza dell'ID utente dipende dai cookie del browser. Se un utente cancella i cookie o cambia dispositivo, perde l'accesso al proprio profilo temporaneo. Tuttavia, per gli scopi di una serata di degustazione singola, questa limitazione è accettabile e preferibile rispetto alla barriera di una registrazione obbligatoria.

Conclusioni e Prospettive Future

Il progetto "Democracy is Dead" rappresenta un'applicazione esemplare delle potenzialità offerte dallo stack Next.js 15 e Supabase. La capacità di integrare complessi modelli matematici di scelta sociale con un'interfaccia utente fluida e sincronizzata in tempo reale trasforma una semplice serata tra amici in un esperimento di democrazia digitale e analisi sensoriale. L'architettura proposta garantisce che il sistema rimanga completamente gratuito per gli sviluppatori e per gli utenti finali, rientrando ampiamente nei limiti dei piani "Hobby" di Vercel e "Free Tier" di Supabase (50.000 utenti attivi mensili, 500.000 invocazioni di funzioni edge). In futuro, l'applicazione potrebbe espandersi includendo sistemi di voto proporzionale per decisioni

multi-vincitore (es. scegliere i 3 film da vedere in una maratona) o integrando meccanismi di "voto delega" (Liquid Democracy) per comunità più strutturate.
La precisione nella gestione dei dati, unita a una solida base statistica, rende questa piattaforma non solo uno strumento ludico, ma un prototipo per futuri sistemi decisionali decentralizzati dove la trasparenza matematica e la facilità di accesso sono i pilastri fondamentali della partecipazione collettiva.

Bibliografia

1. Complete Authentication Guide for Next.js App Router in 2025 - Clerk, https://clerk.com/articles/complete-authentication-guide-for-nextjs-app-router 2. Schulze method - Wikipedia, https://en.wikipedia.org/wiki/Schulze_method 3. pponec/schulze-method - GitHub, https://github.com/pponec/schulze-method 4. Fast Schulze Voting Using Quickselect - arXiv, https://arxiv.org/html/2411.18790v1 5. Computing the Schulze Method for Large-Scale Preference Data Sets - arXiv, https://arxiv.org/pdf/2505.12976 6. Implementing Schulze voting method in SciLab - Stack Overflow, https://stackoverflow.com/questions/30358627/implementing-schulze-voting-method-in-scilab 7. Z-Score Normalization: Definition and Examples - GeeksforGeeks, https://www.geeksforgeeks.org/data-analysis/z-score-normalization-definition-and-examples/ 8. Borda Count | Mathematics for the Liberal Arts - Lumen Learning, https://courses.lumenlearning.com/waymakermath4libarts/chapter/borda-count/ 9. Borda Count Voting - ElectionBuddy, https://electionbuddy.com/borda-count/ 10. Borda Count Method: Basics and an Example - Toolshero, https://www.toolshero.com/decision-making/borda-count-method/ 11. Lecture 2 : Borda's method: A Scoring System - University of Notre Dame, https://www3.nd.edu/~apilking/math10170/information/Lectures/Lecture-2.Borda%20Method.pdf 12. anarchodin/caritat: A vote counting library in JavaScript - GitHub, https://github.com/anarchodin/caritat 13. Mastering Next.js 15+ Folder Structure: A Developer's Guide - Medium, https://medium.com/@j.hariharan005/mastering-next-js-15-folder-structure-a-developers-guide-b9b0461e2d27 14. Next.js 15 + React 19 - shadcn/ui, https://ui.shadcn.com/docs/react-19 15. Database | Supabase Docs, https://supabase.com/docs/guides/database/overview 16. Supabase + Next.js Guide… The Real Way | by Muhammad Qitmeer | Dec, 2025 - Medium, https://medium.com/@iamqitmeeer/supabase-next-js-guide-the-real-way-01a7f2bd140c 17. Anonymous Sign-Ins | Supabase Docs, https://supabase.com/docs/guides/auth/auth-anonymous 18. Presence | Supabase Docs, https://supabase.com/docs/guides/realtime/presence 19. Getting Started with Realtime | Supabase Docs, https://supabase.com/docs/guides/realtime/getting_started 20. Build a User Management App with Next.js | Supabase Docs, https://supabase.com/docs/guides/getting-started/tutorials/with-nextjs 21. Database Functions | Supabase Docs, https://supabase.com/docs/guides/database/functions 22. TodoONada/nextjs-supabase-realtime - GitHub, https://github.com/TodoONada/nextjs-supabase-realtime 23. Realtime Chat - Supabase, https://supabase.com/ui/docs/nextjs/realtime-chat 24. Min-Max and Z-Score Normalization | Codecademy, https://www.codecademy.com/article/min-max-zscore-normalization 25. How to clone a RECORD in PostgreSQL - Stack Overflow, https://stackoverflow.com/questions/25776613/how-to-clone-a-record-in-postgresql 26. Statistics Day 4: Z-Score vs Min-Max Normalization — Making Data Fair for ML Models, https://dev.to/brains_behind_bots/statistics-day-4-z-score-vs-min-max-normalization-making-dat

a-fair-for-ml-models-1plc 27. How to use the library with React? | DiceBear, https://www.dicebear.com/guides/use-the-library-with-react/ 28. How to programmatically access all available options of an avatar style? - DiceBear, https://www.dicebear.com/guides/access-all-available-options/ 29. Filled Grid Radar Chart - shadcn.io, https://www.shadcn.io/charts/radar-chart/radar-chart-10 30. Radar Chart - shadcn.io, https://www.shadcn.io/charts/radar-chart/radar-chart-01 31. Tutorial: How to create a ReactJS / NextJS Chart component using Shadcn UI, https://dev.to/fredy/tutorial-how-to-create-a-reactjs-nextjs-chart-component-using-shadcn-ui-3o4b 32. Next.js authentication with Supabase and NextAuth: A Deep Dive (Part 1) - Medium, https://medium.com/@sidharrthnix/next-js-authentication-with-supabase-and-nextauth-js-part-1-of-3-76dc97d3a345 33. React QR Code Generation: A Guide to Building Custom QR Codes - DEV Community, https://dev.to/franciscomendes10866/how-to-create-a-qr-code-with-react-5aj9 34. lzear/votes: JS library for ranked voting systems - GitHub, https://github.com/lzear/votes 35. javascript - Normalise array of objects & key by id - Stack Overflow, https://stackoverflow.com/questions/41940767/normalise-array-of-objects-key-by-id 36. Radar Charts - Beautiful Charts & Graphs - shadcn/ui, https://ui.shadcn.com/charts/radar 37. Building Real-time Magic: Supabase Subscriptions in Next.js 15 - DEV Community, https://dev.to/lra8dev/building-real-time-magic-supabase-subscriptions-in-nextjs-15-2kmp 38. python3-vote-core 20170329.0 - PyPI, https://pypi.org/project/python3-vote-core/