ChatGPT

Tools and technologies relevant to the MTG tool project

The Magic:-The-Gathering (MTG) project described by the user likely uses a modern **full-stack JavaScript/TypeScript stack** together with card-data sources. Below is a consolidated research report summarizing each technology or service, its purpose and important features. Unless noted, all dates refer to content accessed on **14 September 2025** (user's time zone: Europe/London).

1. Supabase platform

Supabase is an open-source alternative to Firebase. It bundles managed PostgreSQL, authentication, storage, serverless functions and real-time messaging. Because it provides a complete backend, it is often used with front-end frameworks like React and Next.js.

1.1 Supabase Database (PostgreSQL)

- Every Supabase project has its own **PostgreSQL database**. The database is *portable* and can be extracted or migrated at any time. It integrates with row-level security (RLS) policies and JSON Web Tokens (JWT) so that data is private by default [1]. The platform's **SQL editor and dashboard** allow developers to create and manage tables and to run SQL queries from the browser [2].
- Supabase automatically generates RESTful APIs for database tables and views, meaning the app can query and mutate data via HTTP without writing custom server code [2].

1.2 Authentication (Supabase Auth)

- Supabase Auth provides a **complete user-management system** that is tightly integrated with the PostgreSQL policy engine [3]. It supports email/password login, magic-link sign in, social logins (Google, GitHub, Apple, Twitter and more) and enterprise-level SAML/OIDC providers [4].
- User data is stored in the project's own PostgreSQL database, and row-level policies are used to restrict access, making authentication and authorization consistent across the stack [4].
- The SDK includes functions such as `signUp()`, `signInWithPassword()` and `getUser()` to manage sign-up/sign-in flows and fetch the current session [3].

1.3 Edge Functions

- Supabase Edge Functions let developers deploy **serverless JavaScript/TypeScript functions** that run on Deno. They are fully managed and globally deployed, offering low-latency execution and automatic scaling [5]. Edge functions integrate with GitHub for CI/CD and can be tested locally with `supabase functions serve` [6].
- Functions have access to environment variables and are protected by the same authentication mechanism as other Supabase services. Because they run close to users, they are suited for tasks such as sending transactional emails or performing server-side validations.

1.4 Realtime

- Supabase **Realtime** uses WebSockets to synchronize client state across all connected users. It supports listening to insert/update/delete events on specific database tables, tracking connected clients via **presence** (e.g., to show who is online), and **broadcast** channels for sending custom messages to subscribers [7]. These features make it straightforward to implement chat or collaborative editing.

1.5 Storage

- Supabase Storage is an open-source object store that is **S3-compatible**, integrates with Supabase Auth/Postgres, and uses row-level policies to protect files [8] [9]. Developers can upload, download, list, move or delete files through the SDK. Files can be served via a built-in CDN and images can be resized, compressed or converted on the fly [10].
- The dashboard offers an object explorer with drag-and-drop uploading, path navigation, multi-select actions and file previews [9].

2. Card-data APIs and libraries

MTG applications need reliable card and set data. Three main resources are commonly used:

2.1 Scryfall API

- **Scryfall** is described by EDHREC as a powerful Magic card search engine that lets users search for groups of cards meeting specific criteria, see the categories a card belongs to (e.g., ability words or creature types) and view official card rulings [11]. A Microsoft Learn connector page notes that Scryfall indexes *official* Magic cards that can be collected or used in formats [12].
- The API exposes many actions (endpoints) to retrieve data. According to the connector documentation, developers can fetch information about individual cards, lists of cards, ability words, artifact types, artist names, creature types, and sets [13]. Search queries support advanced syntax such as `type:wizard` or `cmc=3` (converted mana cost) [14].
- When used in a client application, Scryfall helps build card lookup, deck-building and auto-complete features.

2.2 MTGJSON

- **MTGJSON** is an open-source project that aggregates Magic:-The-Gathering data from multiple sources and publishes daily-updated **JSON and CSV files**. The project aims to provide portable formats of card data, including card pricing, descriptive properties, simplified searching and TypeScript type definitions [15]. Patrons get access to a GraphQL API.
- MTGJSON is useful when developers need a local copy of the entire card database, offline support or integration with analytic tools.

2.3 MTG Developers API (mtg-cards API)

- The Magic:-The-Gathering Developers API (sometimes referred to as `mtg` API) allows developers to fetch card and set information via REST endpoints (`/cards`, `/sets`, etc.). The API documentation states that **rate limits** are enforced (up to 5 000 requests per hour) and multiple community SDKs exist (Ruby, Python, JavaScript, C#, etc.) [16]. Developers should adhere to usage guidelines and error handling described in the docs [17].
- While less comprehensive than Scryfall, this API can be a lightweight option for simple card lookups.

3. Front-end and UI technologies

Modern MTG tools often run in the browser, so they depend on frameworks and CSS utilities that improve developer experience and performance.

3.1 React

- **React** is a JavaScript library for building user interfaces. Its core concept is the **component**, a modular, self-contained UI unit that can be composed to build complex screens. The Flatirons article notes that React's component-based architecture promotes code reusability and maintainability; components are small pieces of UI that can be reused across an application [18].
- React uses a **unidirectional data flow**: data moves from parent components to children, simplifying state management and making the flow of data predictable [19]. This pattern helps developers reason about how state changes affect the user interface.
- A signature feature is **virtual DOM reconciliation**. Instead of directly modifying the browser DOM, React maintains a virtual representation of the UI and efficiently calculates which parts of the real DOM need updates. This minimises DOM operations and enhances performance [20]. The Claritee article similarly emphasises that React's virtual DOM reduces costly direct DOM manipulations, enabling smooth UI updates [21].
- Other features often highlighted include JSX (a syntax that combines JavaScript and HTML), the introduction of hooks (such as `useState` and `useEffect`) for state and side-effect management, and strong community and ecosystem support [22]. Because React is **just a library** and not a full framework, it can be paired with routing, state-management or backend technologies of the developer's choice [23].

3.2 Next.js

- **Next.js** is a popular React framework that provides a complete solution for building web applications. A technology overview notes that Next.js supports:

- **Server-Side Rendering (SSR)** and **Static Site Generation (SSG)** for improved performance and SEO [24].

- **Incremental Static Regeneration (ISR)**, which allows updating static pages on demand without a full rebuild [24].
- **File-based routing**, where pages are created simply by adding files to the `pages` directory; nested folders define dynamic routes [24].
- Built-in **CSS/Sass** support and **API routes**, so developers can write backend logic and front-end code in the same project [24].

- Automatic **code splitting** and **hot module replacement (HMR)**, which optimise performance and improve developer experience [24].

- Next.js integrates seamlessly with Supabase and Prisma, making it a strong choice for full-stack applications.

3.3 Tailwind CSS

- **Tailwind CSS** is a utility-first CSS framework. The official tagline describes it as a framework "packed with classes like `flex`, `pt-4`, `text-center` and `rotate-90` that can be composed to build any design" [25].

- A technology overview explains that Tailwind's **utility-first classes** enable rapid UI development by allowing developers to build custom designs directly in markup; **responsive design utilities** and **extensive customisation options** are built in [26]. Because styles are created by composing small classes, Tailwind encourages a **component-based approach** with reusable design components [26].
- The same article lists **pros** such as quick development, responsive design and maintainability, and **cons** including a steeper learning curve and verbose HTML [27]. Popular alternatives include Bootstrap, Bulma and Foundation [27].

3.4 Prisma ORM (optional but common)

- **Prisma** is a type-safe ORM (Object-Relational Mapper) for Node.js and TypeScript. It generates a **Prisma schema**, which lets developers declare database tables in a human-readable DSL (domain-specific language). From this schema, Prisma generates TypeScript types, a query builder and SQL migrations [28].
- Prisma's **automated migrations** create SQL migration files from schema changes, and the **Prisma Client** offers fully type-safe queries with auto-completion [29] [30]. The strong type-safety ensures compile-time checks for database queries and reduces runtime errors [30].
- A typical workflow involves modeling data in a `.prisma` file, running migrations and using the generated client in server code or serverless functions. Because Prisma provides an intuitive API and integrates with Next.js and Supabase, many full-stack projects use it for database access.

4. Summary and relevance to the MTG project

The MTG tool the user refers to likely uses Supabase for database storage, authentication, storage of deck images and real-time features, plus serverless functions for custom logic. Front-end technologies include React for building the UI, Next.js for routing and SSR/SSG, and Tailwind CSS for styling. Data about Magic cards is fetched from external services: Scryfall (detailed card search), MTGJSON (bulk data files) or the MTG Developers API (simple REST queries). Prisma may be used as a type-safe ORM when connecting to Supabase's PostgreSQL database. Understanding each of these tools and their features helps ChatGPT provide accurate explanations and troubleshooting assistance for the project.

---

[1] [2] Database | Open source SQL Database
https://supabase.com/database

[3] [4] Auth | Built-in user management
https://supabase.com/auth

[5] [6] Supabase Edge Functions - Deploy JavaScript globally in seconds
https://supabase.com/edge-functions

[7] Realtime | Sync your data in real time
https://supabase.com/realtime

[8] [9] [10] Storage | Store any digital content
https://supabase.com/storage

[11] Guide to Scryfall Syntax | EDHREC
https://edhrec.com/guides/guide-to-scryfall-syntax

[12] [13] Scryfall (Independent Publisher) - Connectors | Microsoft Learn
https://learn.microsoft.com/en-us/connectors/scryfallip/

14 Searching with Scryfall: Magic at your Fingertips — Lucky Paper
https://luckypaper.co/articles/searching-with-scryfall-magic-at-your-fingertips/

15 MTGJSON · MTGJSON.com · Portable formats for all Magic: The Gathering data
https://mtgjson.com/

16 17 Getting Started
https://docs.magicthegathering.io/

18 19 20 22 What is React or React.js? A Guide in 2025
https://flatirons.com/blog/what-is-reactjs/

21 What is React? A Comprehensive Introduction to the Popular JavaScript Library - Claritee – Design Any App in Minutes with AI
https://claritee.io/blog/what-is-react-a-comprehensive-introduction-to-the-popular-javascript-library/

23 React
https://react.dev/

24 Understanding Next.js: Key Features, Benefits, and Use Cases - Prismetric
https://www.prismetric.com/understanding-next-js/

25 Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.
https://tailwindcss.com/

26 27 Tailwind CSS: Utility-First CSS Framework | Magnet
https://magnet.co/technologies/tailwind-css

28 30 TypeScript & Prisma | TypeScript ORM for SQL Databases
https://www.prisma.io/typescript

29 Prisma | Next-generation ORM for Node.js & TypeScript
https://www.prisma.io/orm