

Spark!Bytes: Documentation

Introduction

Spark!Bytes is a web application designed to reduce food waste on Boston University's campus by connecting leftover food from events with BU community members looking for free snacks or meals. The system allows event organizers to create postings for leftover food and helps students or faculty easily find and claim that food. In doing so, Spark!Bytes aims to foster a more sustainable environment and ensure fewer perfectly good meals go to waste.

Objectives

1. **Identify Overlooked Food Sources:** Provide a platform for BU event organizers to quickly post leftover snacks, beverages, or meals.
2. **Connect with the BU Community:** Let students, faculty, and staff discover where free food is currently available.
3. **Reduce Food Waste:** Prevent perfectly good food from heading to landfills when it could be used by others.
4. **Manage & Secure:** Maintain role-based access (admin vs. standard users), secure signups, and handle user preferences.

System Requirements

1. **Identify Sources of Food:** Allows authorized event organizers to create posts about leftover or upcoming free food.
2. **Eligible Users:** Only Boston University-affiliated users (with @bu.edu email) can create an account, sign in, and access the site.
3. **Notification of Availability:** Spark!Bytes issues notifications to subscribed users when new events are posted (if they have not disabled them).
4. **Manage Food Availability:** Organizers can update or remove events, set quantity and details on leftover food, and track them in real-time.
5. **Security and Access Control:** Role-based access: Admin vs. student user. Only admins can access the `/admin` dashboard to manage roles and oversee all application data.

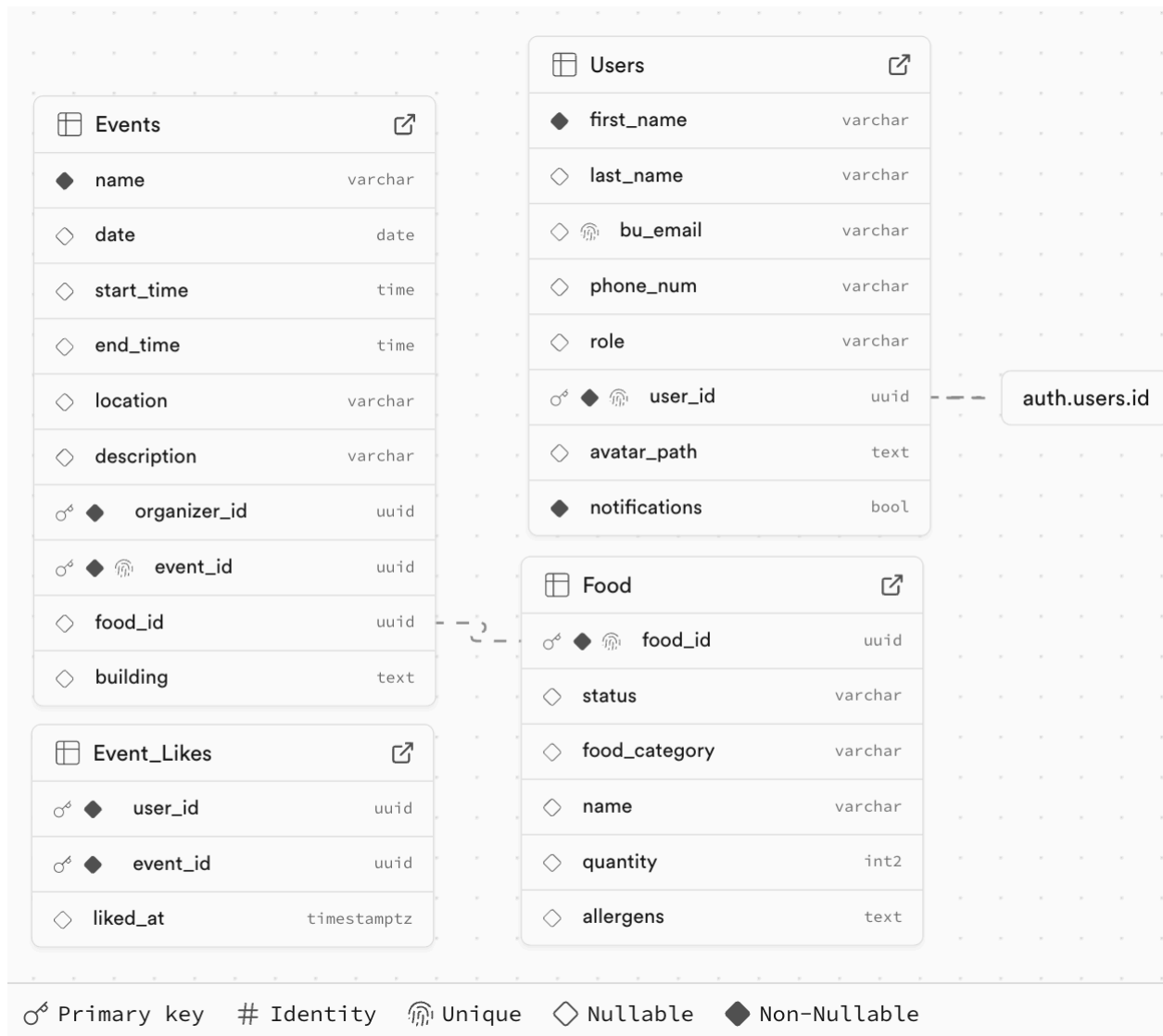
Architecture and Technology Stack

Overall Stack





- **Front-End Framework:** [Next.js 13+ with React]
- **Styling:** Tailwind CSS & custom CSS modules
- **Backend/Server:** Supabase (PostgreSQL database + authentication + storage)
- **Deployment:** Vercel

Database Layer (Supabase)

Tables:



RLS Policies:

<div><div> Users</div><div><div>SELECT</div><div>Enable users to view their own data only Applied to: <code>authenticated</code> role</div></div><div><div>INSERT</div><div>For Insert Applied to: <code>public</code> role</div></div><div><div>UPDATE</div><div>Update user settings Applied to: <code>public</code> role</div></div></div>	<div><div> Food</div><div><div>INSERT</div><div>Enable insert for authenticated users only Applied to: <code>authenticated</code> role</div></div><div><div>SELECT</div><div>Enable read access for all users Applied to: <code>public</code> role</div></div><div><div>UPDATE</div><div>Enable update for users Applied to: <code>public</code> role</div></div></div>
<div><div> Events</div><div><div>DELETE</div><div>Enable delete for users based on <code>user_id</code> Applied to: <code>public</code> role</div></div><div><div>INSERT</div><div>Enable insert for users based on <code>user_id</code> Applied to: <code>public</code> role</div></div><div><div>SELECT</div><div>Enable read access for all users Applied to: <code>public</code> role</div></div><div><div>UPDATE</div><div>Enable update for users based on email Applied to: <code>authenticated</code> role</div></div></div>	<div><div> Event_Likes</div><div><div>DELETE</div><div>Enable delete for users based on <code>user_id</code> Applied to: <code>public</code> role</div></div><div><div>INSERT</div><div>Enable insert for authenticated users only Applied to: <code>authenticated</code> role</div></div><div><div>SELECT</div><div>Enable read access for all users Applied to: <code>public</code> role</div></div></div>

Users RLS:

SELECT	INSERT	UPDATE
<pre>alter policy "Update user settings" on "public"."Users" to public using (true with check (true);</pre>	<pre>alter policy "For Insert" on "public"."Users" to public with check ((auth.uid() = user_id));</pre>	<pre>alter policy "Update user settings" on "public"."Users" to public using (true with check (true);</pre>

Food RLS:

SELECT	INSERT	UPDATE
alter policy "Enable read access for all users" on "public"."Food" to public using (true);	alter policy "Enable insert for authenticated users only" on "public"."Food" to authenticated with check (true);	alter policy "Enable update for users" on "public"."Food" to public using (true);

Events RLS:

SELECT	INSERT	UPDATE	DELETE
alter policy "Enable read access for all users" on "public"."Events" to public using (true);	alter policy "Enable insert for users based on user_id" on "public"."Events" to public with check (((SELECT auth.uid() AS uid) = organizer_id));	alter policy "Enable update for users based on email" on "public"."Events" to authenticated using (((SELECT auth.uid() AS uid) = organizer_id) with check (((SELECT auth.uid() AS uid) = organizer_id));	alter policy "Enable delete for users based on user_id" on "public"."Events" to public using (((SELECT auth.uid() AS uid) = organizer_id));

Event_Likes RLS:

SELECT	INSERT	UPDATE
alter policy "Enable read access for all users" on "public"."Event_Likes" to public using (true);	alter policy "Enable insert for authenticated users only" on "public"."Event_Likes" to authenticated with check (true);	alter policy "Enable delete for users based on user_id" on "public"."Event_Likes" to public using (((SELECT auth.uid() AS uid) = user_id));

Features & Functionality

Authentication & Roles

- Signup / Login / Reset Password
 - `signup/page.tsx`, `login/page.tsx`, `reset/page.tsx`: Provide full user authentication flows with validation (including BU email checks).
 - Users must have an `@bu.edu` email to register.
- Role Management
 - Admin: Access `/admin` page, update roles, see all user data.
 - Student: Can only manage their own profile, view or create events (if given permission by the system design).

Admin Dashboard

- **Purpose:** Lists all users, supports searching by name, toggling user roles between admin and student, etc.
- **Core Functions:**
 - `toggleAdmin`: Switches a user's role from `admin` to `student` and vice versa.
 - `fetchAllUsers`, `userRole`: Supabase queries for retrieving user lists and role verification.

Event Listings & Filtering

- **Purpose:**
 - Show paginated list of upcoming or leftover food events.
 - Support searching by event name and advanced filtering by location, date, and dietary restrictions.
- **Key Functions**
 - `fetchEvents`: Retrieves events from Supabase, merges them with the `Food` table data, and calculates "like" counts.
 - `EventCard` and `EventFilter`: Provide a UI for filtering events (by date range, location codes, allergens) and toggling "likes."
 - Pagination controls (`currentPage`, `totalPages`).

Map Integration

- **Purpose:** Displays a Google Map with static markers for major BU buildings.
- **Dependencies:** [@vis.gl/react-google-maps](https://vis.gl/react-google-maps) or Google Maps JavaScript API.
- **Implementation:**
 - Hardcoded `STATIC_LISTINGS` with known lat/lng for BU locations.
 - `CustomMarker` used for popup info windows or advanced marker customization.

Profile Management

- **Purpose:** Let users update personal info, including name, email, password, profile photo, or notification settings.
- **Functions:**
 - `updateUserName`, `updateEmail`, `updatePassword`: Helper methods to handle data changes.
 - `ProfilePictureUpload`: Feature to upload a new avatar to Supabase Storage.
 - `NotificationToggle`: Toggle event notifications on/off.

Landing Page/About Page

- **Purpose:** Provide marketing or informational pages for new visitors:
 - **Landing Page:** Summarizes the application's mission, how to get started, and call-to-action (CTA) for signup.
 - **About Page:** Highlights the project team (images, bios, and LinkedIn links).

Conclusion

Spark!Bytes successfully addresses the challenge of leftover food on BU's campus, ensuring that rather than going to waste, free meals become easily accessible. By implementing secure role-based management, robust event features, and a user-friendly interface, Spark!Bytes provides a sustainable, community-driven solution. Future iterations can build on these foundations to expand feature sets, incorporate real-time updates, and strengthen the user experience even further.