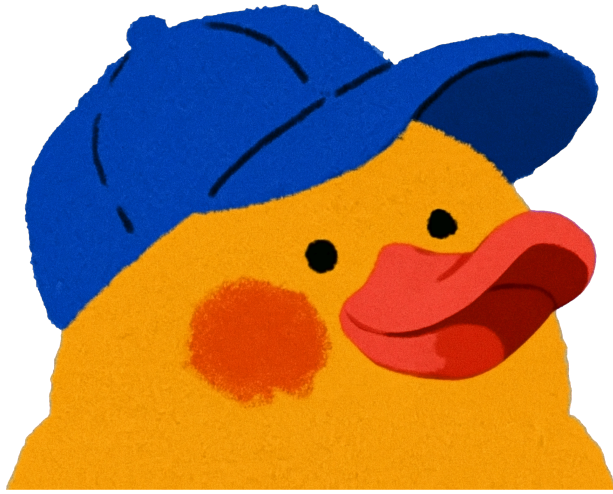


# Quack!: Technical Design



**Eng POC:** Andre Santiago-Neyra · **Status:** Draft · **Timeline:** One semester (~14 weeks)

**Team Members:** Pratiksha Vilas Pawar, Zhong Zhangniantong

## Overview

Quack! lets Stevens students pin a spontaneous meetup on a campus map, such as "Studying for CS102!" at the dining hall at 7, so anyone nearby can join.

Meeting people in college means joining clubs, which demand schedules and commitment, or relying on class, which leaves two minutes to talk. Nothing covers the low-commitment middle: four classmates would join that study session if they knew it existed. Group chats only reach friends, stories broadcast without coordinating, and event platforms assume weeks of planning.

## Objective and Scope

Ship a deployed, tested app that proves the post, discover, and join loop for Stevens students.

### Goals

- Complete the post, discover, and join loop with real persistence and real accounts
- Posting an event takes under 30 seconds of user time: location, time, one line of detail
- Discovery is location-first. The map is the home screen
- Only students of the target campus can sign in
- Deployed on a public URL with seeded demo data, not a localhost prototype
- Architecture a four person team can build, test, and reason about in 14 weeks

### Non-goals

- Horizontal scale. A single application instance and a single database is sufficient
- Real-time push. Polling and refresh on navigation are acceptable
- Native iOS and Android clients
- Integration with the university's own identity provider (see Alternatives Considered)
- Messaging, feeds, payments, ticketing, recurring events, check-in, ratings
- Moderation tooling and admin dashboards
- Offline support and multi-campus tenancy

## Requirements

### Functional

| ID | Requirement                                                                                    | Priority |
|----|------------------------------------------------------------------------------------------------|----------|
| F1 | A student can sign in with their school account, and only school accounts are accepted         | P0       |
| F2 | An authenticated user can create an event with location, date, time, category, and description | P0       |
| F3 | A user can view events near them on a map, bounded by the visible viewport                     | P0       |
| F4 | A user can open an event and see its details and attendee list                                 | P0       |
| F5 | A user can join and leave an event                                                             | P0       |
| F6 | A host can edit or cancel an event they created                                                | P1       |
| F7 | A host can set event visibility to public or link only                                         | P1       |
| F8 | A user can step the map one day back or one day forward from today                             | P1       |

| ID | Requirement                                                  | Priority |
|----|--------------------------------------------------------------|----------|
| F9 | A host can see view and join counts for each of their events | P1       |

### **Non-functional**

Map queries return in under 500ms at p95 with 2,000 seeded events. Target scale is 500 users, 2,000 events, and 50 concurrent sessions. Availability is best effort, but the demo build must be live and seeded in week 14.

No passwords stored, HTTPS only, no secrets in the repository. Coordinates are never logged next to a user id. Personal data is limited to email, display name, avatar URL, and provider subject id.

Forms are keyboard navigable and labelled, and a list view backs up the map.

## **System Architecture**

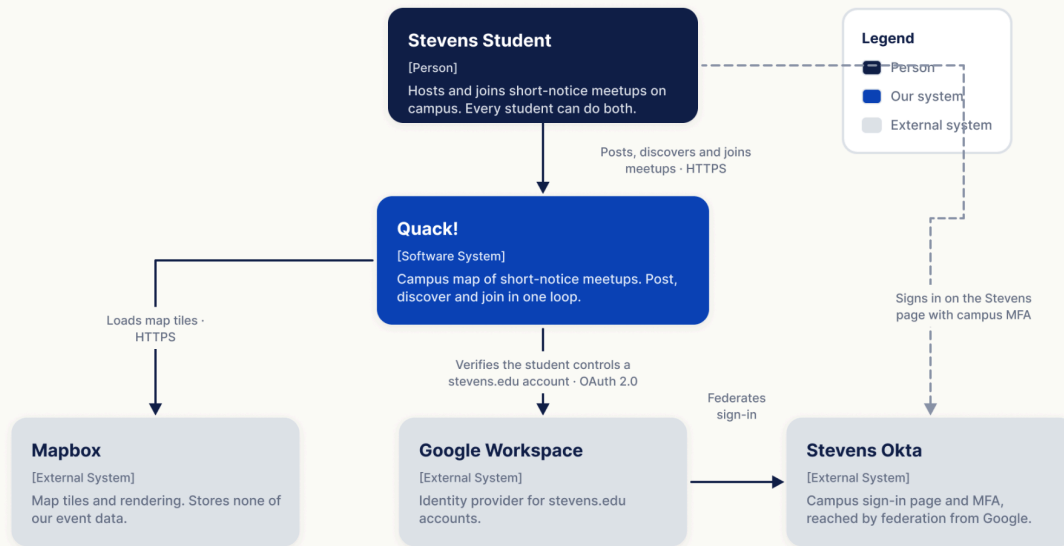
Three tiers: a client that renders the map and every screen, a stateless Spring Boot API that owns logic, authorization, and persistence, and Postgres with PostGIS. The client loads tiles from Mapbox directly; the API talks to Google only during sign-in.

No queues, caches, workers, or microservices. Each adds a failure mode we cannot afford in 14 weeks.

### **C4 Level 1: System Context**

Who uses Quack! and what it depends on. Every student can both host and attend, so there is one actor.

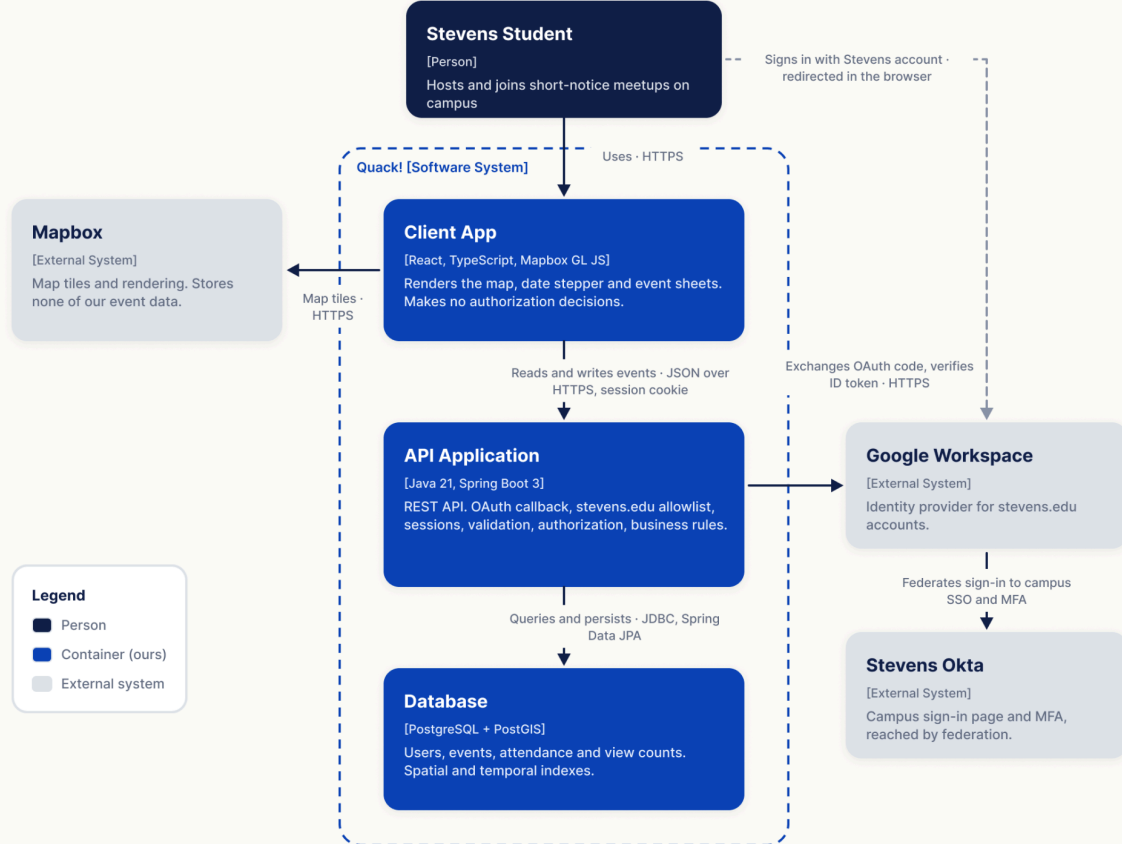
## Quack! · C4 Level 1: System Context



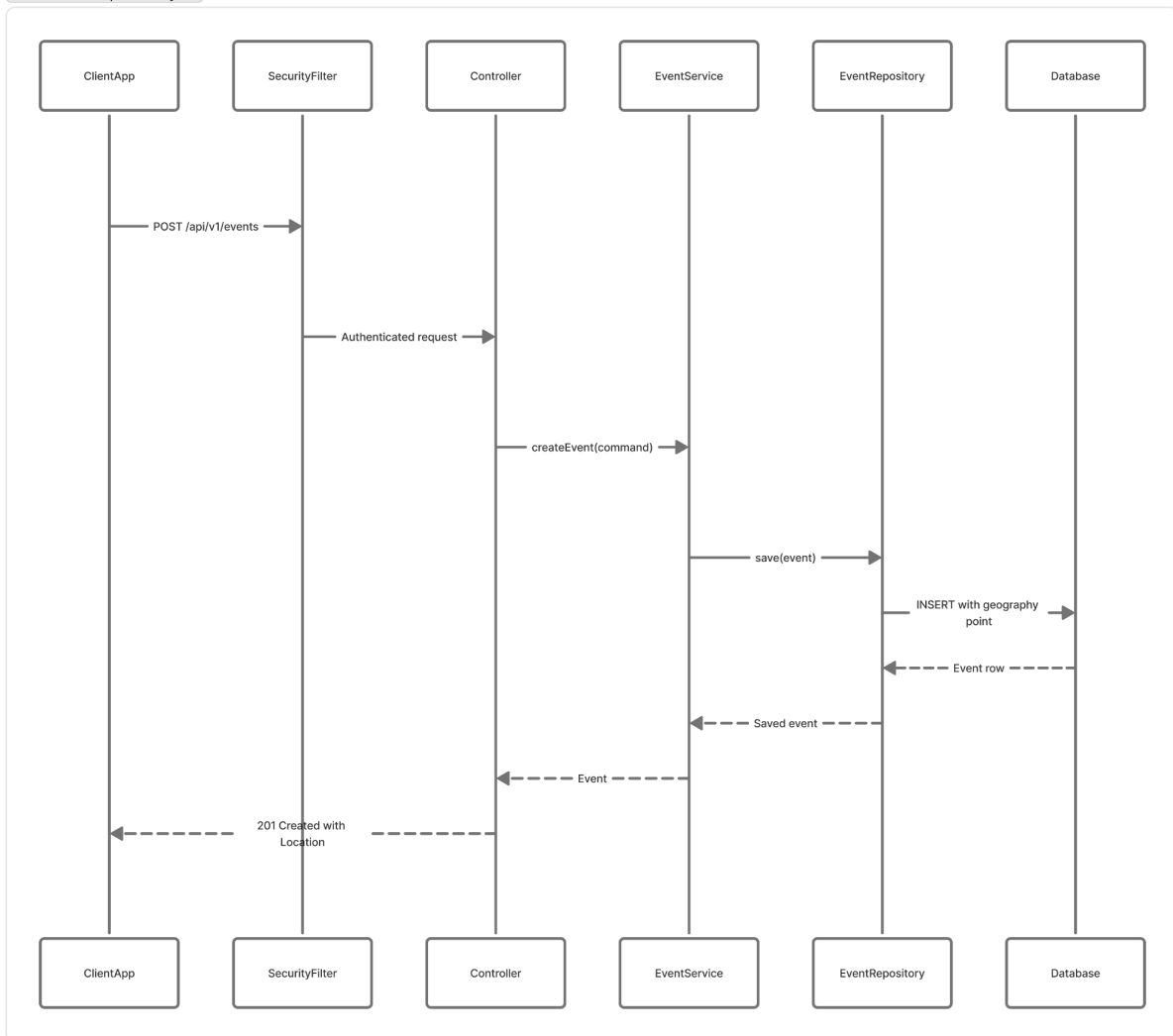
## C4 Level 2: Container

The deployable pieces inside Quack!, their technology, and how they communicate.

## Quack! · C4 Level 2: Containers



Create Event Request Lifecycle



## Component responsibilities

| Component       | Responsibility                                                                                              | Explicitly not responsible for                                       |
|-----------------|-------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| React client    | Rendering, map interaction, form input, optimistic UI                                                       | Authorization decisions, domain verification, visibility enforcement |
| Spring Boot API | OAuth callback handling, domain allowlist, sessions, validation, authorization, business rules, persistence | Rendering, tile fetching, geocoding                                  |
| Postgres        | Durable storage,                                                                                            | Business logic in triggers                                           |

| Component      | Responsibility                                               | Explicitly not responsible for                                  |
|----------------|--------------------------------------------------------------|-----------------------------------------------------------------|
|                | uniqueness constraints, spatial and temporal indexes         | or stored procedures                                            |
| Mapbox         | Map tiles and rendering                                      | Storing any of our event data, geocoding (not used)             |
| OAuth provider | Authenticating that the user controls a school email account | Authorization. Membership of the school domain is checked by us |

## Technology Stack

|                    |                                                                                                          |
|--------------------|----------------------------------------------------------------------------------------------------------|
| <b>Layer</b>       | Decision                                                                                                 |
| <b>Client</b>      | React Native, TypeScript                                                                                 |
| <b>Server</b>      | Java 21, Spring Boot 3, Spring Web                                                                       |
| <b>Persistence</b> | Spring Data JPA, Flyway migrations, hibernate-spatial                                                    |
| <b>Database</b>    | PostgreSQL with the PostGIS extension, on Railway or Supabase                                            |
| <b>Auth</b>        | spring-boot-starter-oauth2-client against Google or Microsoft, with a server-side email domain allowlist |
| <b>Map SDK</b>     | Mapbox GL JS v3 via react-map-gl                                                                         |
| <b>App hosting</b> | Render web service, plus Render static site or Vercel for the client                                     |
| <b>CI</b>          | GitHub Actions                                                                                           |
| <b>Testing</b>     | JUnit 5, Testcontainers with a PostGIS image, Spring Boot test slices                                    |

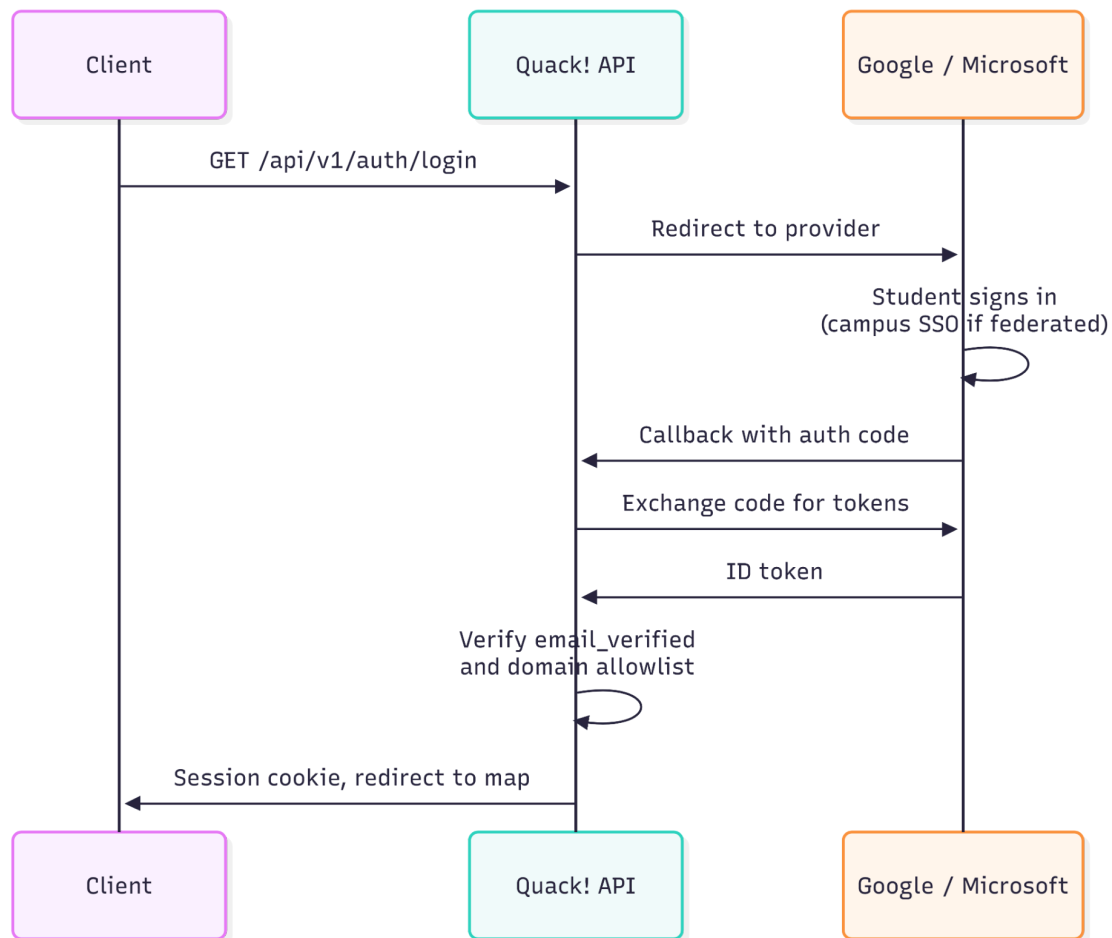
Render's free tier sleeps after 15 idle minutes, with 30 to 50 second cold starts plus JVM startup, so warm the API before demoing. Set the JVM max heap explicitly to stay under 512 MB.

Keep Spring shallow: controller, service, repository. No single-implementation interfaces, DTO mapper frameworks, or abstract base services.

## Authentication and Identity

Students sign in with their Stevens Google account. The server accepts only

stevens.edu. We register our own Google OAuth client, so campus IT is not involved.



Stevens federates Google to Okta, so students get the Stevens Okta page and campus MFA without us registering on the institutional tenant.

### Rules

The domain check runs server side on the ID token. The allowlist is configuration, so adding a campus needs no code change.

No passwords means no registration, reset, or recovery flows. Users are keyed on provider subject id, since schools can reassign email addresses.

Instructors and TAs also use stevens.edu, so the allowlist is a single entry.

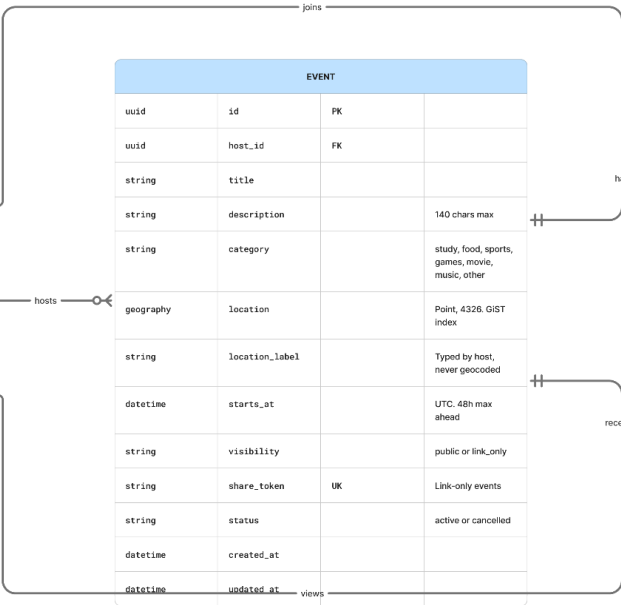
## Data Model

| USER     |                     |    |                             |
|----------|---------------------|----|-----------------------------|
| uuid     | id                  | PK |                             |
| string   | provider            |    | google                      |
| string   | provider_subject_id | UK | Identity key, with provider |
| string   | email               | UK | Always stevens.edu          |
| string   | email_domain        |    | Allowlist check             |
| string   | display_name        |    |                             |
| string   | avatar_url          |    | From Google profile         |
| bool     | is_seed             |    | Seed users cannot sign in   |
| datetime | created_at          |    |                             |
| datetime | last_login_at       |    |                             |

| EVENT     |                |    |                                                 |
|-----------|----------------|----|-------------------------------------------------|
| uuid      | id             | PK |                                                 |
| uuid      | host_id        | FK |                                                 |
| string    | title          |    |                                                 |
| string    | description    |    | 140 chars max                                   |
| string    | category       |    | study, food, sports, games, movie, music, other |
| geography | location       |    | Point, 4326. GIST index                         |
| string    | location_label |    | Typed by host, never geocoded                   |
| datetime  | starts_at      |    | UTC. 48h max ahead                              |
| string    | visibility     |    | public or link_only                             |
| string    | share_token    | UK | Link-only events                                |
| string    | status         |    | active or cancelled                             |
| datetime  | created_at     |    |                                                 |
| datetime  | updated_at     |    |                                                 |

| EVENT_ATTENDANCE |           |        |                      |
|------------------|-----------|--------|----------------------|
| uuid             | event_id  | PK, FK |                      |
| uuid             | user_id   | PK, FK |                      |
| datetime         | joined_at |        | Row deleted on leave |

| EVENT_VIEW |           |    |                        |
|------------|-----------|----|------------------------|
| uuid       | event_id  | FK |                        |
| uuid       | viewer_id | FK |                        |
| datetime   | viewed_at |    | One per viewer per day |



## User

1. id primary key
2. provider and provider\_subject\_id, unique together. The real identity key
3. email, unique, case-insensitive. Always a stevens.edu address
4. email\_domain, denormalized for the allowlist check and for future multi-campus support
5. display\_name, seeded from the provider profile, editable by the user
6. avatar\_url, the profile picture supplied by Google, used for the host bubbles on clustered pins
7. is\_seed boolean. Seeded demo users have no provider identity and can never sign in
8. created\_at, last\_login\_at

## Event

- id primary key
- host\_id foreign key to User
- title and description, the one-line detail
- category, enum of (study, food, sports, games, movie, music, other)
- location, a PostGIS geography(Point, 4326) column holding the pin. The API accepts and returns latitude and longitude as separate numbers; conversion happens at the persistence boundary
- location\_label, typed by the host, for example "Main Dining Hall". Never geocoded (see Frontend Design)
- starts\_at, stored in UTC
- visibility, enum of public or link\_only
- share\_token, generated for link-only events, unguessable, unique
- status, enum of active or cancelled
- created\_at, updated\_at

## EventAttendance

- event\_id, user\_id, composite unique constraint
- joined\_at. Leaving deletes the row. No in-scope story needs join and leave history

## EventView

- event\_id, viewer\_id, viewed\_at
- Deduplicated per viewer per event per day, so the host stat in F9 means something

## Indexes and query shape

- GiST index on location, which is what PostGIS uses for both bounding box and radius queries
- Index on starts\_at, since every map query filters by time window as well as bounds
- Index on host\_id for the my events screen
- Unique index on share\_token, and on (provider, provider\_subject\_id)

Pins are stored as PostGIS geography(Point, 4326). Viewport queries become a bounding box intersection on a GiST index, and a future distance filter is one ST\_DWithin call in metres. geography over geometry avoids managing projections. Setup is CREATE EXTENSION postgis, hibernate-spatial, JTS types on the entity, and a

PostGIS Testcontainers image: about a day now, versus a migration and test rework if added in week 11.

## Data retention

Cancelling sets status to cancelled instead of deleting, so attendees see what happened and the day-back view stays intact.

## Category Statistics

Category popularity is a query, not stored data: `SELECT category, COUNT(*) FROM events WHERE starts_at >= ? AND starts_at < ? GROUP BY category`. At 2,000 events this returns instantly, so there is no counter table and no rollup to keep in sync or backfill. Since events are never hard deleted, the full history stays available to re-aggregate. Add a rollup only if the query ever gets slow, which it will not at campus scale.

category is an enum column rather than its own table. A lookup table would only pay off if categories needed adding without a deploy, or carried metadata such as an icon or colour. Converting later is a small migration, and the stored values survive it either way.

The same column is what the P2 category filter needs, so this is a prerequisite for that stretch story rather than a detour.

## API Design

REST over HTTPS with JSON, RFC 3339 UTC timestamps, and every path under /api/v1. Auth is an httpOnly session cookie set by the OAuth callback. Conventions follow [RFC 9457](#) for errors, [Google AIP](#) for resource naming, and GeoJSON axis order for bounding boxes. Deviations are noted.

## Conventions

Paths are versioned at /api/v1 so breaking changes remain possible.

Resources are plural nouns and methods carry the verb: /events, /events/{id}, never /getEvents.

Errors use RFC 9457 Problem Details (application/problem+json) with type, title, status, detail, and instance, plus a stable code the client branches on. Field-level validation failures go in errors. Spring Boot 3's ProblemDetail provides this out of the box.

Status codes follow RFC 9110: 200, 201, 400 malformed, 401 unauthenticated, 403 forbidden, 404 not found or not visible, 409 conflict, 422 semantically invalid, 429 rate limited with Retry-After.

A link-only event without a valid token returns 404, not 403, so its existence is not leaked.

GET, PUT, and DELETE are idempotent, which is why joining is a PUT.

The map query returns an ETag; clients send If-None-Match and get a 304 when nothing

changed.

Lists return an object rather than a bare array and page by cursor: limit in, nextCursor out.

The OpenAPI spec is generated by springdoc and committed, so it cannot drift from the code.

## Endpoints

| Method and path                    | Purpose                                                     | Auth     | Story  |
|------------------------------------|-------------------------------------------------------------|----------|--------|
| GET<br>/api/v1/auth/login          | Begin OAuth.<br>Redirects to the provider                   | None     | F1     |
| GET<br>/api/v1/auth/callback       | OAuth callback.<br>Domain check, user upsert, session issue | None     | F1     |
| POST<br>/api/v1/auth/logout        | Invalidate the session                                      | Required | F1     |
| GET /api/v1/me                     | Current user, for session bootstrap on page load            | Required | F1     |
| POST<br>/api/v1/events             | Create an event.<br>Returns 201 with a Location header      | Required | F2     |
| GET /api/v1/events                 | Map query. Params: bbox, from, to, limit, cursor            | Required | F3, F8 |
| GET<br>/api/v1/events/{id}         | Event detail.<br>Optional token for link-only               | Required | F4     |
| PATCH<br>/api/v1/events/{id}       | Partial update. Host only                                   | Required | F6     |
| POST<br>/api/v1/events/{id}/cancel | Cancel. Host only. A custom method, see below               | Required | F6     |
| GET<br>/api/v1/events/{id}/        | Attendee list, paged. Attendees                             | Required | F4     |

| Method and path                          | Purpose                                                | Auth     | Story |
|------------------------------------------|--------------------------------------------------------|----------|-------|
| attendees                                | only                                                   |          |       |
| PUT<br>/api/v1/events/{id}/attendance    | Join. Idempotent by definition of PUT                  | Required | F5    |
| DELETE<br>/api/v1/events/{id}/attendance | Leave. Idempotent                                      | Required | F5    |
| GET<br>/api/v1/me/events                 | Events hosted by the caller, with view and join counts | Required | F9    |

Attendance is a sub-resource: PUT joins and DELETE leaves, both idempotent by definition.

Cancel is a custom method, POST /events/{id}/cancel, because the record persists after cancelling. This matches Stripe and GitHub; Google's AIP-136 would use :cancel.

As on Partiful, everyone sees the attendee count but only attendees see names; GET /events/{id}/attendees returns 403 otherwise. A separate endpoint also keeps event detail small.

## Example: map query

GET

/api/v1/events?bbox=-73.9950,40.8390,-73.9830,40.8480&from=2026-09-19T00:00:00Z&to=2026-09-20T00:00:00Z&limit=100

200 OK

ETag: "a1b2c3"

```
{
  "events": [
    {
      "id": "evt_8kq2",
      "title": "Studying for CS102!",
      "latitude": 40.8421,
      "longitude": -73.9401,
      "locationLabel": "Main Dining Hall",
      "category": "study",
      "startsAt": "2026-09-19T23:00:00Z",
      "attendeeCount": 3,
      "hostDisplayName": "Andre",
      "status": "active"
    }
  ],
  "nextCursor": null
}
```

```
}
```

bbox is west, south, east, north: the GeoJSON order and what Mapbox bounds return, so the client passes it through unchanged.

Error response:

400 Bad Request

Content-Type: application/problem+json

```
{
  "type": "https://quack.example/problems/invalid-bbox",
  "title": "Malformed bounding box",
  "status": 400,
  "detail": "bbox must be four comma-separated numbers in
west,south,east,north order",
  "instance": "/api/v1/events",
  "code": "INVALID_BBOX"
}
```

Map results are a thin projection. Descriptions load on detail, attendees from their own endpoint.

When several events share a spot, the map shows three host-avatar bubbles plus a fourth "..." bubble that opens a Show all list. Nothing is hidden and the map stays readable.

Avatars come from avatar\_url, taken from the Google profile at sign-in.

## Validation rules

- Start time at most 48 hours out, matching the map's one-day-forward limit
- Description at most 140 characters
- Category required, and one of the enum values. other exists so nobody force-fits an event into the wrong bucket
- Coordinates inside the Stevens campus bounding box; dorms count as on campus
- Map query dates limited to yesterday, today, or tomorrow; anything else is rejected
- Started events can be edited; cancelled ones cannot

## Backend Architecture

Controllers validate and map requests, services own business rules and all authorization, repositories own queries. Keeping authorization in services means a new controller cannot bypass it.

## Visibility and authorization matrix

| Caller                | Public event                           | Link-only, no token | Link-only, valid token |
|-----------------------|----------------------------------------|---------------------|------------------------|
| Unauthenticated       | 401                                    | 401                 | 401                    |
| Any signed-in student | View, join, leave                      | 404                 | View, join, leave      |
| Host                  | All of the above, plus edit and cancel | Full access         | Full access            |

Link-only events never appear in the map query, even for the host. They are reachable only by link.

## View counting

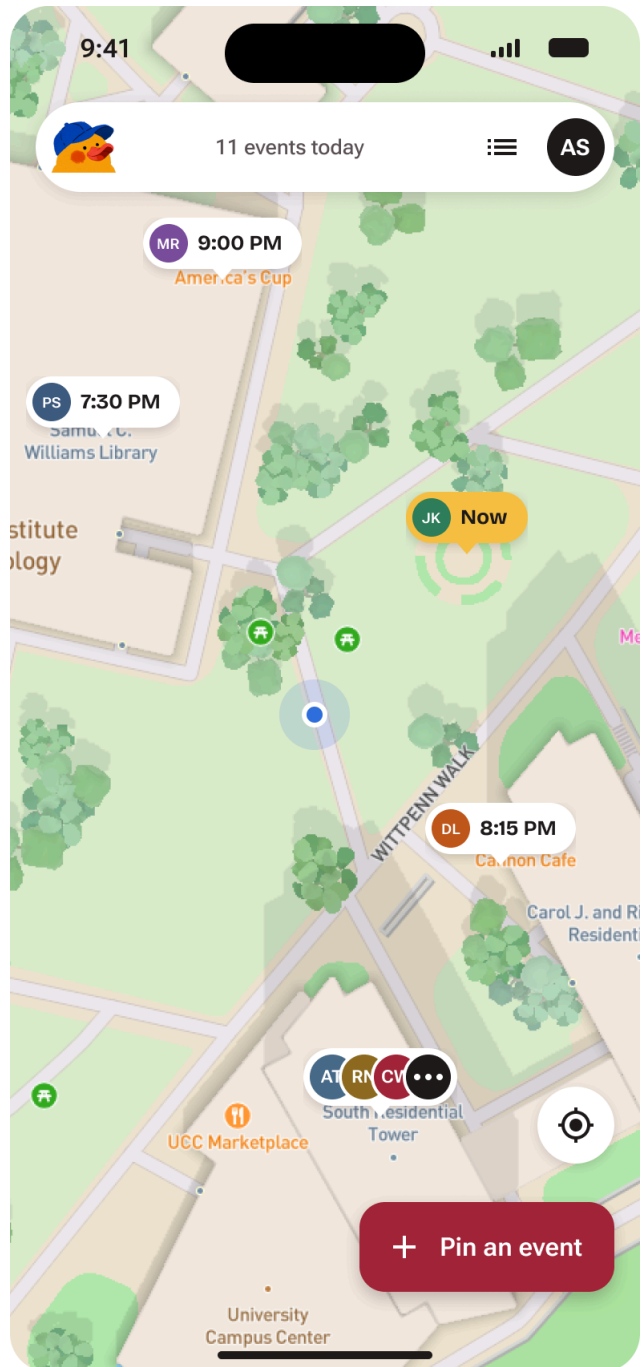
A view is logged on detail fetch, once per viewer per day, excluding the host, so refreshing cannot inflate F9.

## Concurrency

A unique constraint on (event\_id, user\_id) prevents double joins; the service treats the violation as success.

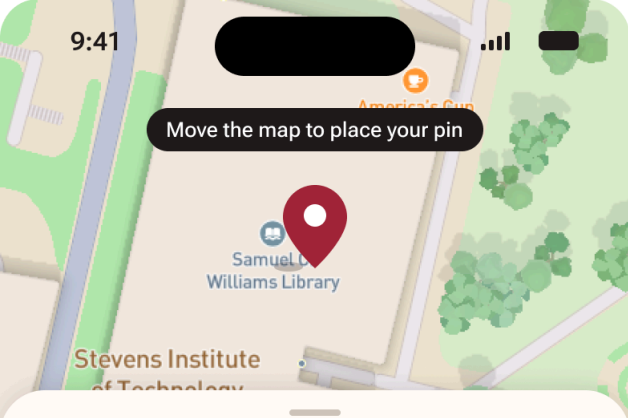
## Frontend Design

The map is the home screen and everything else is a sheet over it, which also avoids extra Mapbox map loads. View the [Figma Wireframes](#) for further UI layouts



9:41

Move the map to place your pin



Pin an event

What's happening?

Studying for CS102!

One line. Nearby students see it on the map. 19/140

Category

✓ Study

Food

Sports

Games

Movie

Music

Other

When

🕒 Today, 8:00 PM

Defaults to the next hour. Up to 48 hours ahead.

Where exactly?

📍 Library, 3rd floor

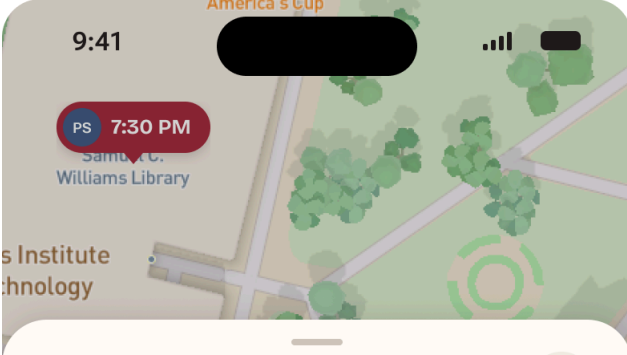
Type a room or landmark so people can find you.

Post event

9:41

PS 7:30 PM

Samuel C. Williams Library



Study

✓ You're going

×

Studying for CS102!

PS

Hosted by Priya Shah

🕒

Today, 7:30 PM

in 25 min

📍

Library, 3rd floor quiet room

Samuel C. Williams Library

👥 6 going

PS

Priya Shah

Host

AS

Andre S.

You

ML

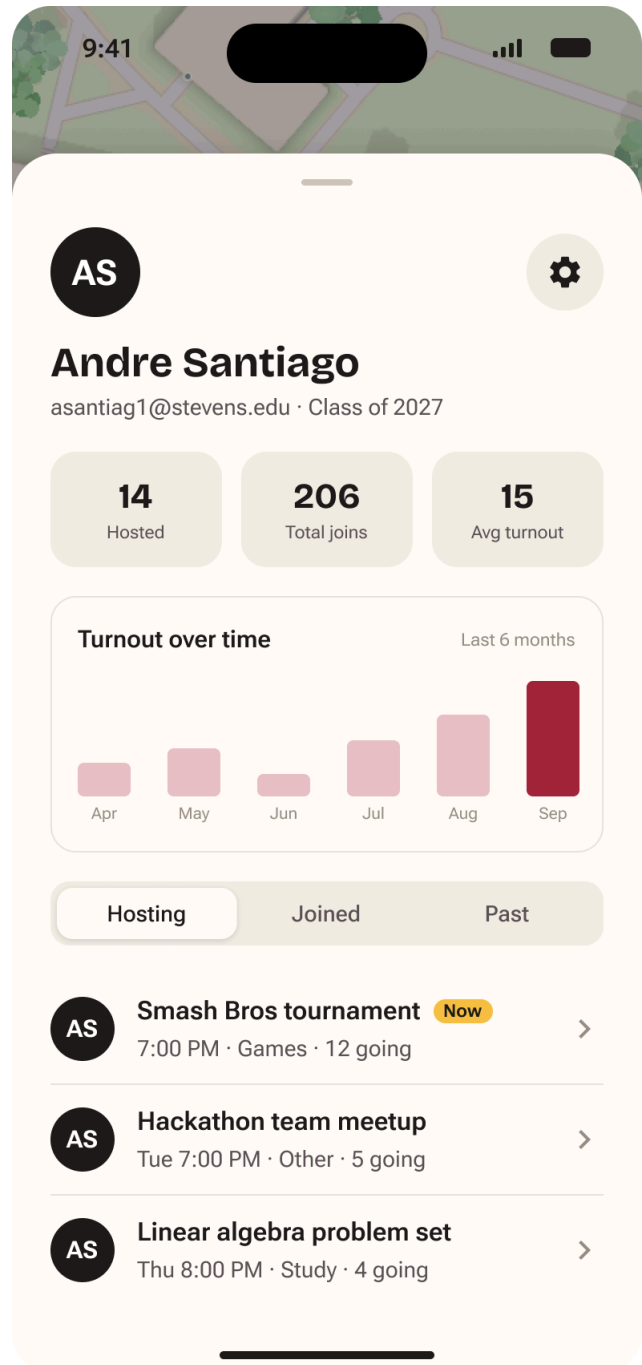
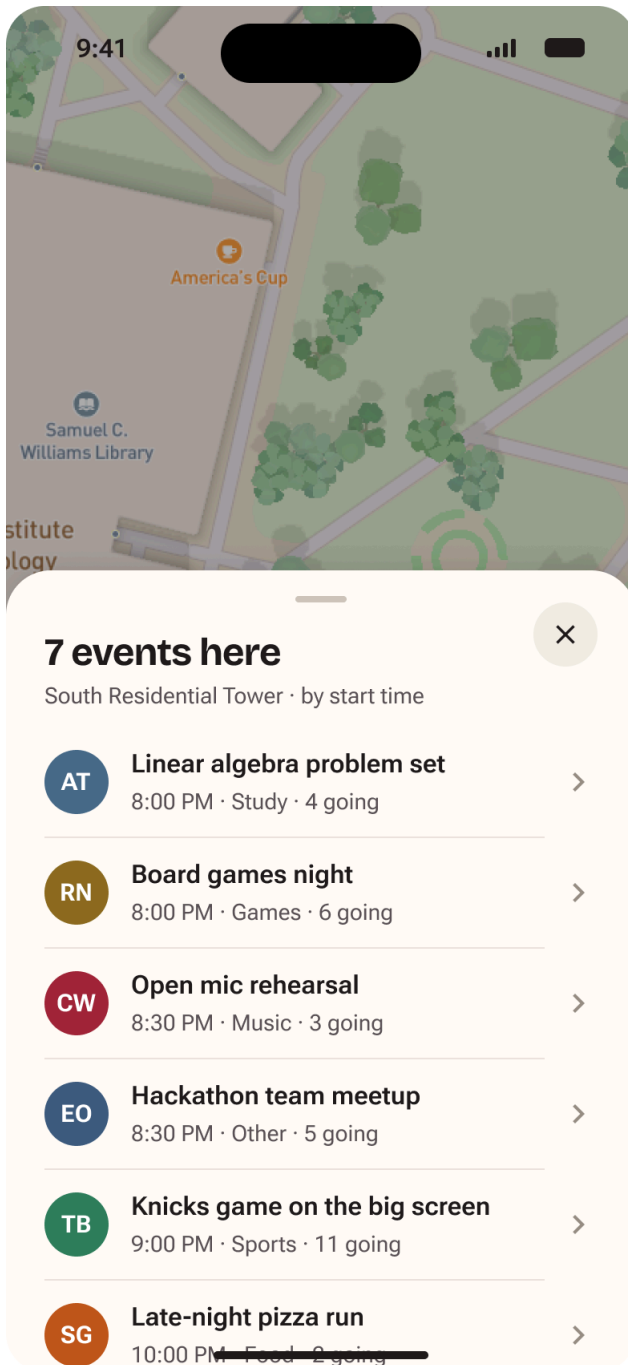
Maya Lin

OK

Omar Khalil

See all 6 attendees

Leave



## Design principles

No feed or landing page before the map. Create is one screen: location from the current map position, time defaulting to the next hour, and one line of text. Joins update optimistically. A list view backs up the map for accessibility and tile failures.

## **Mapbox integration**

Mount the map once. Mapbox bills per map initialization, with panning and zooming free for 12 hours, so detail and create are overlays rather than routes.

The viewport bounding box drives every query. Requests are debounced while panning and hit our API, not Mapbox.

No geocoding: hosts type the location label. It is faster and avoids Mapbox's permanent geocoding pricing.

Mapbox has no spending cap, so use a scoped public token restricted at the provider.

Clustering is in scope, since it is how overlapping events are shown.

## **State management**

Server state is fetched per screen and refetched on navigation and viewport changes. Only the session is global.

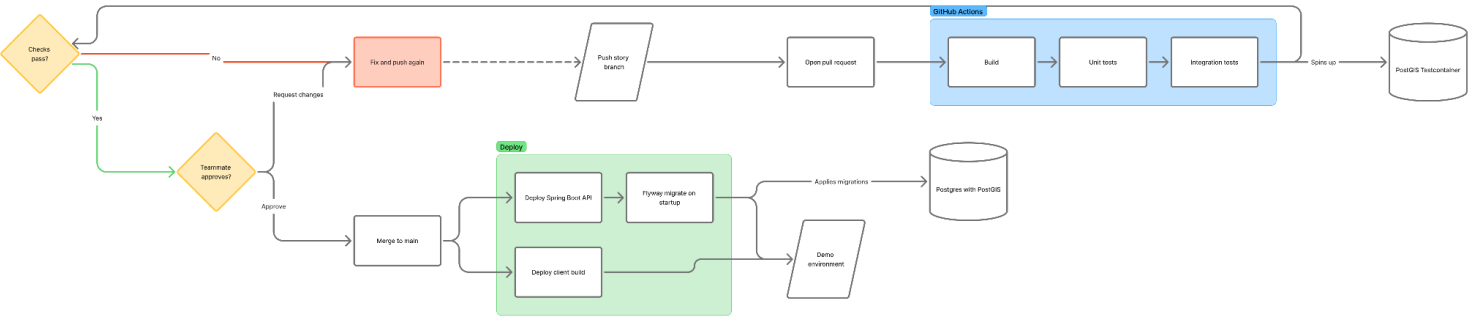
## **Deployment and Environments**

Two environments: local, and one deployed environment for staging and the demo.

Client and server deploy separately, so OAuth redirect URIs are registered for both.

Pull requests run build and tests; merging to main deploys. Flyway migrations run on startup, never by hand.

Seeded users are flagged `is_seed` and cannot sign in, so the demo needs two real accounts. Warm the API before presenting.



Test Plan

| Area | What is tested                                                                                                                                                                               | Level                        | Owner |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|-------|
| Auth | Allowed domain signs in, disallowed domain rejected with no user created, unverified email rejected, existing user matched by provider subject id, protected route rejects anonymous callers | Integration, provider mocked | TBD   |

| Area           | What is tested                                                                                                                                    | Level                | Owner |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-------|
| Event creation | Valid create, missing fields, missing or unknown category, out-of-bounds coordinates, start time beyond 48 hours, description over 140 characters | Unit and integration | TBD   |
| Map query      | Bounding box correctness including edge coordinates, date filtering, link-only exclusion, result limit                                            | Integration          | TBD   |
| Visibility     | Every cell of the authorization matrix                                                                                                            | Integration          | TBD   |
| Join and leave | Join, double join, leave, leave when not joined, count accuracy                                                                                   | Integration          | TBD   |
| Host controls  | Edit by host, edit by non-host rejected, cancel, edit after cancel rejected                                                                       | Integration          | TBD   |
| View counting  | Dedup per viewer per day, host views excluded                                                                                                     | Unit                 | TBD   |
| Core loop      | Sign in, post, discover, join, leave as one scripted path                                                                                         | End to end           | TBD   |

Integration tests use a real PostGIS database via Testcontainers, since in-memory databases mishandle spatial queries and constraints.

**Edge cases:** events on the bounding box edge, events starting now, simultaneous joins, cancelled events with attendees, empty viewports, seven events at one spot, tampered share tokens, and non-stevens.edu Google accounts.

**Demo check:** run the full core loop on a phone with seed data and two real stevens.edu accounts.

## Alternatives Considered

**Direct Stevens Okta integration.** Needs IT to register us on the institutional tenant, which we cannot schedule around a week 3 auth milestone. Federation already shows students the Okta page through Google.

**Email and password auth.** More work than OAuth, and proves nothing about student status.

**Go backend.** Lighter on memory, but unfamiliar to the team and without a built-in OIDC client.

**Node backend.** One language across the stack, but less familiar and weaker for integration testing.

**React Native client.** Better for use on the move, but @rnmapbox/maps needs native builds, iOS needs a Mac and a paid developer account, and the deliverable becomes an install rather than a link.

**Plain latitude and longitude columns.** Enough for bounding boxes, but adding PostGIS later would mean a migration and test rework during hardening.

**WebSocket live counts.** A new transport and failure mode for numbers that refreshing already keeps current.

## Open Questions

- Pin granularity: exact coordinates, or snapped to named campus locations? Snapping is safer and easier to label, but loses the specific bench case
- Can a host remove one of their past events from the day-back view, or does cancelling and letting it age out cover it?
- Final category list. Six plus other is the starting proposal, but the team should agree the set before the first migration

## Timeline and Milestones

| Weeks  | Milestone        | Deliverables                                                                                                                                           | Dependencies |
|--------|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| 1 to 2 | Design and setup | This document approved, data model final with PostGIS enabled, wireframes, repository, CI, deploy pipeline, Mapbox spike, OAuth provider confirmed and | None         |

| Weeks    | Milestone              | Deliverables                                                                                 | Dependencies            |
|----------|------------------------|----------------------------------------------------------------------------------------------|-------------------------|
|          |                        | client registered                                                                            |                         |
| 3 to 4   | Accounts and map       | F1. OAuth end to end with domain allowlist, map renders with hardcoded pins, schema migrated | OAuth client registered |
| 5 to 7   | Create and view events | F2, F3, F4. Events persist, viewport query works, detail sheet renders                       | Auth, schema            |
| 8 to 9   | Joining                | F5. Core loop complete and demoable end to end. <b>Checkpoint week</b>                       | Event CRUD              |
| 10 to 11 | Host controls          | F6, F7. Edit, cancel, visibility rules, full authorization matrix under test                 | Core loop               |
| 12       | History and stats      | F8, F9. Date filtering and host counts                                                       | Core loop               |
| 13       | Test and harden        | Edge cases, bug fixes, seed data, manual pass on deployed URL                                | All features            |
| 14       | Demo and writeup       | Final presentation, README, retrospective                                                    | Everything              |