Course Project Report

Subject: WDP301

– Quy Nhon, September 2025 –

Table of Contents

Record of Changes

Date	A* M, D	In charge	Change Description

MỤC LỤC

I. Overview.

I.1 Project Information.

- **Project name:** FreeDay
- **Group:** 6
- **Software type:** FullStack Web

I.2 Project Team.

Full Name	Role	Email	Mobile
VangH	Lecturer		
Hoàng Lê Quý An	Leader		
Nguyễn Đăng Nhân	Member		

Lương Gia Khánh	Member		
Lê Anh Khôi	Member		
Nguyễn Trần Quang Nhật	Member		

II. Requirement Specification

II.1 Problem description

Students and young adults struggle to find suitable weekend events because information is fragmented and outdated, with no simple way to register or revisit favorites. Many also want companions but lack a lightweight, trusted space to coordinate.

Organizers face parallel issues: reaching the right audience without ads, handling scattered inquiries, tracking registrations and attendance intent, and measuring real interest—resulting in spammy, low-quality experiences and little insight.

FREDAY addresses this with a single web platform: email/Google/Facebook sign-in, profile management, event browse/filter, registration, and favorites; a forum with posts and comments to find companions; an organizer dashboard showing confirmed registrations and favorites; and administrator moderation of forum posts, event listings, and users to ensure quality and trust.

II.2 Major Features

- **FE-01:** Register/Login (Email, Google, Facebook)
- **FE-02:** Profile Management
- **FE-03:** Event Discovery (browse/search/filter by time, location, price, tags)
- **FE-04:** Event Details (description, schedule, location, capacity, organizer)
- **FE-05:** Event Registration (join/cancel)

- **FE-06:** Favorites & My Events (saved and registered)
- **FE-07:** Forum – Companion Finder (posts & comments)
- **FE-08:** Organizer Event Management & Dashboard (create/edit/delete, stats)
- **FE-09:** Admin Moderation & User Management (approve/hide/delete posts/events)
- **FE-10:** Notifications (email/in-app for registrations, updates, comments)
- **FE-11: Payment** (online checkout for paid events; promo codes; receipts/refunds via Stripe/VNPay/Momo)
- **FE-12: Google Calendar** (one-click add/sync registered events; auto updates & reminders)
- **FE-13: Chat with Organizer** (1:1 inbox for Q&A; notifications; optional image/file attachments)
- **FE-14: Map API** (geocoding & distance filter; map/marker on event detail; directions)
- **FE-15: Chatbot AI** (natural-language search like "events today near me under $X"; personalized recommendations)

II.3 Context Diagram

The platform interacts with external services and internal components. Below is the context (with suitable technologies for this project).

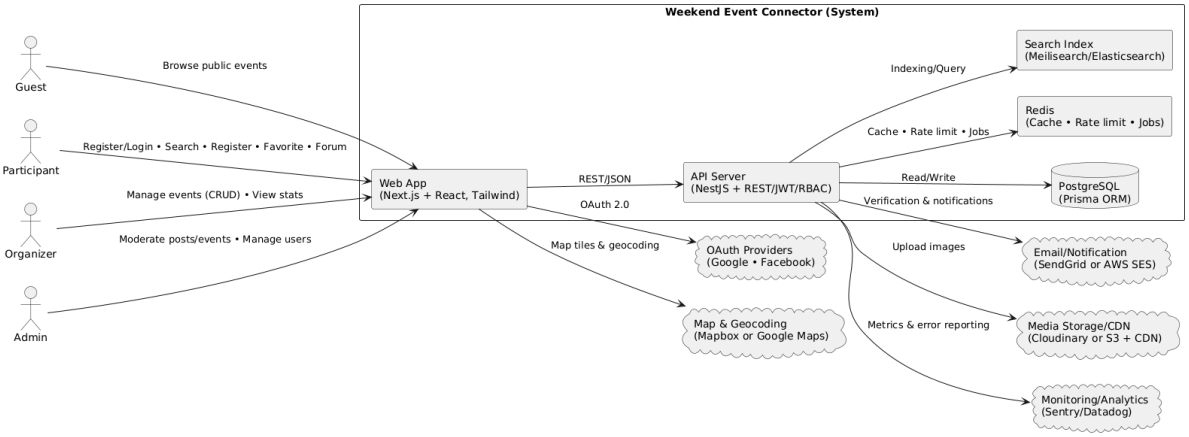
External entities:

- **Guest / Participant / Organizer / Admin:** Use the web app to sign up/sign in, browse events, register, favorite, post on the forum, and moderate (admin).
- **OAuth Providers (Google, Facebook):** Social login via OAuth 2.0.
- **Email/Notification Service (e.g., SendGrid/SES):** Account verification, registration updates, moderation notices.
- **Map & Geocoding (e.g., Mapbox/Google Maps):** Location display and search by city/venue.
- **Media Storage/CDN (e.g., Cloudinary or AWS S3 + CDN):** Event and post images.
- **(Optional) Analytics/Monitoring (e.g., Sentry/Datadog):** Error and performance tracking.

Internal components:

- **Web App (Next.js 14 + React, Tailwind CSS):** Responsive UI, SSR/SEO for event listings, calls backend APIs, handles OAuth flows.
- **API Server (Node.js + NestJS, REST, JWT + Passport OAuth2):** Business logic, validation, RBAC, moderation, rate limiting.

- **Database (PostgreSQL + Prisma ORM):** Users, profiles, events, registrations, favorites, posts, comments, reports, notifications, audit logs.
- **Cache/Queue (Redis):** Session/token blacklist, rate limits, background jobs (emails, counters).
- **Search Index (Meilisearch/Elasticsearch – optional):** Fast full-text search and filters on events and forum.



II.4 Nonfunction Requirements.

#	Feature	System Function	Description
1	**Performance**	System Performance	The system must handle up to 10,000 simultaneous users with a maximum response time of 3 seconds for any user interaction.
2	**Scalability**	Data Handling and System Load	The application must scale to accommodate growing numbers of users and data, ensuring consistent performance as the user base increases.
3	**Security**	User	Secure user registration, login, and password management using

		Authentication	Firebase Authentication with data encryption.
4	**Platform Support**	Multi-Platform Compatibility	The app must support both Android and iOS devices using React Native for cross-platform compatibility.
5	**Data Integrity**	Data Synchronization	The system must ensure that data between Firebase Firestore and the app is consistent and synchronized in real-time.
6	**Reliability**	System Availability	The system should have 99% uptime, with automatic failover capabilities in case of hardware or software failure.
7	**Maintainability**	Code Structure and Updates	The code must be modular, easy to maintain, and allow for regular updates without disrupting user experience.
8	**Accessibility**	Accessibility Features	The app must meet WCAG (Web Content Accessibility Guidelines) standards, ensuring it is usable for people with disabilities (e.g., voice commands).
9	**Localization**	Language Support	The app should be localized for at least two languages (English and Vietnamese) to cater to a wider user base.
10	**Backup and Recovery**	Data Backup and Recovery	Regular backups should be taken, and the system must support data recovery in case of loss or corruption.

II.5 Functional requirements.

II.5.1 Actor

Actor	Description
Guest	● Browse public event listings ● View limited event details ● Sign up / sign in
Participant (Event Seeker)	● Register/Login via Email, Google, or Facebook ● Manage profile ● Browse/search/filter events ● View event details ● Register/cancel attendance ● Save/unsave favorites ● View “My Events” (registered & favorites) ● Create forum posts to find companions ● Comment and report content ● Receive notifications ● Make online payments for paid events (Stripe/VNPay/Momo). ● Sync events to Google Calendar for reminders and automatic updates. ● View event map ● Chat with organizer ● Chat with AI ● Pay to become organizer
Organizer/Host	● Select Organizer role ● Create/Edit/Publish/Close/Cancel events ● Set capacity, deadlines, and policies ● View registrations list (with permitted contact info) ● Track favorites/views ● Manage own event content ● Chat with user ● Manage payment transactions and view revenue reports.
Admin	● Moderate forum posts/comments ● Approve/Hide/Delete events ● Manage users (ban/unban, assign/revoke Organizer role) ● Review violation reports and maintain audit logs ● Monitor system health and content quality
External Actors (System)	● **OAuth Providers (Google/Facebook):** authenticate users and return basic profile data. **Email/Notification Service:** send verification, registration updates, and comment/activity notifications. ● **Payment Gateway (Stripe/VNPay/Momo):** process online transactions and return payment status via webhook. ● **Google Calendar API:** sync user-registered events and send reminders. ● **AI Service (LLM/Chatbot):** handle natural-language queries and return personalized event recommendations.

	• **Map API (Google/Mapbox):** geocode event addresses, calculate distances, and render map markers.

II.5.2 Use Cases

Use Case ID	Use Case Name	Actors Involved	Description
UC-01	**Register User**	Participant, Organizer, Admin	Create a new account with email/password or social login (Google/Facebook); verify email and select role (Participant/Organizer).
UC-02	**Login User**	Participant, Organizer, Admin	Securely log in using credentials or social login to access personalized features.
UC-03	**Logout User**	Participant, Organizer, Admin	End the session and clear auth tokens/cookies.
UC-04	**Manage Profile**	Participant, Organizer	View and update profile (display name, avatar, city, interests, bio).
UC-05	**Browse Events**	Guest, Participant, Organizer	View public event listings with pagination and basic sorting.
UC-06	**Search/Filter Events**	Guest, Participant, Organizer	Search by keywords; filter by weekend/date, location, price, tags, capacity.

UC-07	**View Event Details**	Guest, Participant, Organizer	View full event info: description, schedule, location, organizer, capacity, favorites.
UC-08	**Register for Event**	Participant	Register to attend an event; receive confirmation; capacity/registered count updates.
UC-09	**Cancel Registration**	Participant	Cancel a previously registered event; notify organizer if applicable.
UC-10	**Favorite/Unfavorite Event**	Participant	Save or remove an event from favorites; update favorite count.
UC-11	**View "My Events"**	Participant	See events the user has registered for and saved as favorites.
UC-12	**Create Forum Post**	Participant, Organizer	Create a companion-finder post with title, content, tags, optional images.
UC-13	**Comment on Post**	Participant, Organizer	Add, edit, or delete comments on forum posts (permissions applied).
UC-14	**Report Content**	Participant, Organizer	Report posts, comments, or events for policy violations; create moderation tickets.
UC-15	**Create Event Listing**	Organizer	Create an event (Draft/Publish), set schedule, location, capacity, price, tags, policies.
UC-16	**Edit/Update Event**	Organizer	Update event details; changes notify registered users.
UC-17	**Close/Cancel**	Organizer	Close registrations or cancel an event; notify all registrants

	Event		automatically.
UC-18	**View Registrations List**	Organizer	View the list of registrants with permitted contact info and statuses.
UC-19	**View Event Dashboard/Stats**	Organizer	See counts of registrations, favorites, and views for each event.
UC-20	**Moderate Forum Content**	Admin	Approve/Hide/Delete posts and comments; act on violation reports; keep audit logs.
UC-21	**Moderate Event Listings**	Admin	Approve/Hide/Delete events that violate policies; process event-related reports.
UC-22	**Manage Users & Roles**	Admin	Ban/Unban users; assign/revoke Organizer role; review violation history.
UC-23	**System Notifications**		Send email/in-app notifications for registrations, updates, comments, and moderation actions.
UC-24	**Process Event Payment**	Participant, Organizer, Payment Gateway	Allows participants to make payments for paid events using Stripe/VNPay/Momo. System verifies the transaction, updates the registration status, and sends confirmation to user and organizer.
UC-25	**Sync Event to Google Calendar**	Participant, Google Calendar API	Enables users to add registered events to Google Calendar, automatically syncing updates when organizers change event time or location.
UC-26	**Chat with Organizer**	Participant, Organizer	Provides 1:1 chat between user and organizer for event-related inquiries, attachments, and updates. Notifications are sent when new messages are received.

UC-27	**View Event Map**	Participant, Map API	Displays the event location using Map API (Google/Mapbox), showing the exact venue, directions, and distance from user's position.
UC-28	**Ask AI Chatbot for Event Suggestions**	Participant, AI Chatbot Service	Allows users to ask natural-language questions (e.g., "What events are happening near me today?"). The AI interprets the query and returns relevant event recommendations based on time, location, and budget.

II.5.2.1 Use case descriptions

UC-1: Register User

UC ID and Name:	UC-1 - Register User		
Created By:		Date Created:	
Primary Actor:	Participant/Organizer	Secondary Actors:	OAuth Providers (Google/Facebook), Email Service
Trigger:	User selects "Sign up"		
Description:	Create account via email/password or social login; select role.		
Preconditions:	User not authenticated.		
Postconditions:	Account created; role stored; (if email) verification sent.		
Normal Flow:	1) Open Sign up → 2) Choose email or social → 3) Provide data/consent → 4) System creates user & role → 5) Send		

	verification/auto-login.
Alternative Flows:	• A1 Social login returns profile; account linked. • A2 Email sign-up but verify later.
Exceptions:	• E1 Duplicate email. • E2 OAuth failure. • E3 Weak password/invalid data.
Priority:	High
Frequency of Use:	High (Daily usage expected)
Business Rules:	Unique email; password policy; min age/legal consent; role required
Other Information:	The system should handle potential server errors gracefully and prompt the user to retry if necessary.
Assumptions:	OAuth providers available; email delivery working.

UC-2: Login User

UC ID and Name:	UC-02 - Login User		
Created By:		Date Created:	
Primary Actor:	Participant/Organizer/Admin	Secondary Actors:	OAuth Providers

Trigger:	User clicks **Log in**
Description:	Authenticate via credentials or social login.
Preconditions:	Account exists; (email verified if required).
Postconditions:	Session/JWT issued.
Normal Flow:	Submit creds/social → validate → issue tokens → redirect.
Alternative Flows:	Prompt to resend verification.
Exceptions:	Wrong creds; locked account; OAuth error
Priority:	High
Frequency of Use:	High
Business Rules:	Account lock after N failed attempts; session TTL; refresh token rotation.
Other Information:	Secure cookies/HTTPS; device logging.
Assumptions:	System clock in sync; network stable.

UC-3: Logout User

UC ID and Name:	UC-3: Logout User

Created By:		Date Created:	
Primary Actor:	Participant/Organizer/Admin	Secondary Actors:	N/A
Trigger:	Click **Log out**.		
Description:	End current session.		
Preconditions:	Authenticated.		
Postconditions:	Tokens revoked/cleared; back to home.		
Normal Flow:	Call logout endpoint → revoke → clear storage → redirect.		
Alternative Flows:	Network error (local clear, retry later).		
Exceptions:	Wrong creds; locked account; OAuth error		
Priority:	High		
Frequency of Use:	Medium		
Business Rules:	Revoke only current session by default.		
Other Information:	Token blacklist in Redis.		

Assumptions:	User intends to keep account active.

UC-4: Manage Profile

UC ID and Name:	UC-4: Manage Profile		
Created By:		Date Created:	
Primary Actor:	Participant/Organizer	Secondary Actors:	Media Storage
Trigger:	Open **Profile**.		
Description:	View/update display name, avatar, city, interests, bio.		
Preconditions:	Authenticated.		
Postconditions:	Profile saved.		
Normal Flow:	Edit fields → upload avatar → save → success toast.		
Alternative Flows:	Remove avatar.		
Exceptions:	Invalid fields; upload failure.		
Priority:	Medium		
Frequency of Use:	Medium		

Business Rules:	PII minified; profanity filter; avatar ≤ size/format limits.
Other Information:	Changes audited.
Assumptions:	User consents to public display of chosen fields.

UC-5: Browse Events

UC ID and Name:	UC-5: Browse Events		
Created By:		Date Created:	
Primary Actor:	Guest/Participant/Organizer	Secondary Actors:	N/A
Trigger:	Open **Events** list.		
Description:	View paginated public events.		
Preconditions:	Published events exist.		
Postconditions:	List rendered.		
Normal Flow:	Request page → render cards → next/prev.		
Alternative Flows:	Sort by time/popularity.		
Exceptions:	Empty state.		

Priority:	High
Frequency of Use:	High
Business Rules:	Show only Published & not Hidden; default sort nearest upcoming.
Other Information:	Server pagination & cache.
Assumptions:	Reasonable data volume per page.

UC-6: Search/Filter Events

UC ID and Name:	UC-6: Search/Filter Events		
Created By:		Date Created:	
Primary Actor:	Guest/Participant/Organizer	Secondary Actors:	Search Index (optional)
Trigger:	Enter keywords/filters.		
Description:	Search by keyword; filter by date/weekend, location, price, tags, capacity.		
Preconditions:	Events indexed/available.		
Postconditions:	Matching results returned.		

Normal Flow:	Input criteria → submit → fetch results → display.
Alternative Flows:	Clear filters → default list.
Exceptions:	Query timeout; invalid filter.
Priority:	High
Frequency of Use:	High
Business Rules:	Sanitize queries; AND filters by default, OR within same facet.
Other Information:	Rate limit aggressive queries.
Assumptions:	Search service responsive.

UC-7: View Event Details

UC ID and Name:	UC-7: View Event Details		
Created By:		Date Created:	
Primary Actor:	Guest/Participant/Organizer	Secondary Actors:	Maps, Media Storage
Trigger:	Click event card.		
Description:	Show full info, schedule, organizer, capacity, favorites.		

Preconditions:	Event Published/Visible.
Postconditions:	Details presented.
Normal Flow:	Load event by ID → render text/images/map.
Alternative Flows:	Event hidden → show unavailable.
Exceptions:	Not found/removed.
Priority:	High
Frequency of Use:	High
Business Rules:	Registrant list hidden to non-organizers.
Other Information:	Image CDN, map tiles cached.
Assumptions:	Valid event link.

UC-8: Register for Event

UC ID and Name:	UC-8: Register for Event		
Created By:		Date Created:	

Primary Actor:	Participant	Secondary Actors:	Email Service
Trigger:	Click **Register/Join**.		
Description:	Create a registration for the event.		
Preconditions:	Authenticated; event Published; capacity available; before deadline.		
Postconditions:	Registration saved; counters updated; confirmation sent.		
Normal Flow:	Validate rules → create record (txn) → update counts → send email.		
Alternative Flows:	Add to waitlist if enabled.		
Exceptions:	Full capacity; closed/cancelled event; duplicate registration.		
Priority:	High		
Frequency of Use:	High		
Business Rules:	One active registration per user/event; respect cancellation policy.		
Other Information:	ACID transaction to avoid overbooking.		
Assumptions:	Email deliverable.		

UC-9: Cancel Registration

UC ID and Name:	UC-9: Cancel Registration		
Created By:		Date Created:	
Primary Actor:	Participant	Secondary Actors:	Email Service
Trigger:	Click **Cancel** in My Events.		
Description:	Cancel an existing registration.		
Preconditions:	Registered; within cancel window.		
Postconditions:	Status = Cancelled; counters adjusted; notifications sent.		
Normal Flow:	Confirm cancel → update record → update counts → notify.		
Alternative Flows:	Late cancel blocked; offer contact organizer.		
Exceptions:	Already cancelled; event started.		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	Open slot; optional waitlist auto-promote.		
Other	Store cancel reason (optional).		

Information:	
Assumptions:	Organizer allows cancellations until cutoff.

UC-10: Favorite/Unfavorite Event

UC ID and Name:	UC-10: Favorite/Unfavorite Event		
Created By:		Date Created:	
Primary Actor:	Participant	Secondary Actors:	N/A
Trigger:	Toggle **Favorite**.		
Description:	Save/remove event from favorites.		
Preconditions:	Authenticated; event visible.		
Postconditions:	Favorite state persisted; count updated.		
Normal Flow:	Toggle → update record → update UI count.		
Alternative Flows:	N/A		
Exceptions:	Event hidden/removed.		
Priority:	Medium		
Frequency of Use:	High		

Business Rules:	Unique favorite per user/event.
Other Information:	No email; realtime UI update.
Assumptions:	User uses it as a shortlist.

UC-11: View “My Events”

UC ID and Name:	UC-11: View “My Events”		
Created By:		Date Created:	
Primary Actor:	Participant	Secondary Actors:	N/A
Trigger:	Open **My Events**.		
Description:	View registered and favorited events.		
Preconditions:	Authenticated.		
Postconditions:	Lists shown.		
Normal Flow:	Fetch registrations & favorites → render tabs.		
Alternative Flows:	Filter/sort within list.		
Exceptions:	Empty lists.		

Priority:	Medium
Frequency of Use:	High
Business Rules:	Show future first; past registrations in history.
Other Information:	Server pagination.
Assumptions:	Data consistent.

UC-12: Create Forum Post

UC ID and Name:	UC-12: Create Forum Post		
Created By:		Date Created:	
Primary Actor:	Participant/Organizer	Secondary Actors:	Media Storage
Trigger:	Click **New Post**.		
Description:	Publish companion-finder post (title, content, tags, images).		
Preconditions:	Authenticated; not banned; rate limit OK.		
Postconditions:	Post created (Visible/Pending).		
Normal Flow:	Compose → validate → save → show in feed.		

Alternative Flows:	Auto-flag → pending moderation.
Exceptions:	Spam/rate-limit; invalid content
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	Content policy; tag required; max posts/day.
Other Information:	Images compressed; audit log.
Assumptions:	User aims to find companions.

UC-13: Comment on Post

UC ID and Name:	UC-13: Comment on Post		
Created By:		Date Created:	
Primary Actor:	Participant/Organizer	Secondary Actors:	N/A
Trigger:	Submit comment.		
Description:	Add/edit/delete comments (within permissions/time window).		
Preconditions:	Authenticated; post visible; not banned.		

Postconditions:	Comment stored/updated/removed.
Normal Flow:	Type → submit → store → display.
Alternative Flows:	Edit/delete own comment within X minutes.
Exceptions:	Post hidden/locked; rate-limit.
Priority:	Medium
Frequency of Use:	High
Business Rules:	Prohibit abusive content; link blocking.
Other Information:	Notify post author.
Assumptions:	Threading flat (no nested replies).

UC-14: Report Content

UC ID and Name:	UC-14: Report Content		
Created By:		Date Created:	
Primary Actor:	Participant/Organizer	Secondary Actors:	Admin
Trigger:	Click **Report**.		

Description:	Report post/comment/event violating policy.
Preconditions:	Authenticated; content exists.
Postconditions:	Report ticket created & queued.
Normal Flow:	Pick reason → submit → queue → moderator notified.
Alternative Flows:	Attach evidence (image/link).
Exceptions:	Duplicate recent report from same user.
Priority:	High
Frequency of Use:	Medium
Business Rules:	One active report per user/item; auto-hide after threshold.
Other Information:	SLA target for review (e.g., 24–48h).
Assumptions:	Admin coverage exists.

UC-15: Create Event Listing

UC ID and Name:	UC-15: Create Event Listing		
Created By:		Date Created:	

Primary Actor:	Organizer	Secondary Actors:	Media Storage, Maps
Trigger:	Click **Create Event**.		
Description:	Create Draft/Publish event with schedule, location, capacity, price, tags, policies.		
Preconditions:	Authenticated as Organizer.		
Postconditions:	Event saved.		
Normal Flow:	Fill form → validate → save Draft → Publish.		
Alternative Flows:	Keep as Draft.		
Exceptions:	Invalid dates/location/capacity.		
Priority:	High		
Frequency of Use:	Medium		
Business Rules:	Required fields; geocode location; capacity ≥ 0; content complies.		
Other Information:	Organizer email must be verified.		
Assumptions:	Venue info known.		

UC-16: Edit/Update Event

UC ID and Name:	UC-16: Edit/Update Event		
Created By:		Date Created:	
Primary Actor:	Organizer	Secondary Actors:	Email Service
Trigger:	Click **Edit**.		
Description:	Modify event; notify registrants on critical changes.		
Preconditions:	Organizer owns event; not Cancelled.		
Postconditions:	Event updated; notifications sent if needed.		
Normal Flow:	Edit fields → save → update → notify.		
Alternative Flows:	Require re-confirm attendance on major time change.		
Exceptions:	Edit after start; permission denied.		
Priority:	High		
Frequency of Use:	Medium		
Business Rules:	Organizer cannot change ownership; version history kept.		
Other	Diff stored in audit log.		

Information:	
Assumptions:	Email preferences respected.

UC-17: Close/Cancel Event

UC ID and Name:	UC-17: Close/Cancel Event		
Created By:		Date Created:	
Primary Actor:	Organizer	Secondary Actors:	Email Service
Trigger:	Choose **Close** or **Cancel**.		
Description:	Stop new registrations or cancel entirely.		
Preconditions:	Organizer owns event; Published.		
Postconditions:	Status updated; registrants notified.		
Normal Flow:	Confirm action → update status → send notifications.		
Alternative Flows:	Reopen after Close (not after Cancel).		
Exceptions:	Already Cancelled.		
Priority:	High		
Frequency of Use:	Low–Medium		

Business Rules:	Cancel reason required; refunds out of scope v1.
Other Information:	State machine enforces Draft→Published→(Closed|Cancelled).
Assumptions:	Organizer acts in good faith.

UC-18: View Registrations List

UC ID and Name:	UC-18: View Registrations List		
Created By:		Date Created:	
Primary Actor:	Organizer	Secondary Actors:	N/A
Trigger:	Open **Registrations**.		
Description:	View registrants with allowed contact info and status.		
Preconditions:	Organizer owns event.		
Postconditions:	List displayed/exported.		
Normal Flow:	Query → paginate → filter → (optional) export CSV.		
Alternative Flows:	Filter by status.		
Exceptions:	No registrants.		

Priority:	Medium
Frequency of Use:	Medium
Business Rules:	Show only consented contact fields; data privacy enforced.
Other Information:	Access logged.
Assumptions:	Reasonable list size.

UC-19: View Event Dashboard/Stats

UC ID and Name:	UC-19: View Event Dashboard/Stats		
Created By:		Date Created:	
Primary Actor:	Organizer	Secondary Actors:	N/A
Trigger:	Open **Dashboard**.		
Description:	View registrations, favorites, and views per event/time range.		
Preconditions:	Organizer has events.		
Postconditions:	Metrics presented.		
Normal Flow:	Aggregate metrics → render charts/cards.		

Alternative Flows:	Change date range; export.
Exceptions:	No data.
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	Only own events; near-real-time or daily refresh.
Other Information:	Cached aggregates.
Assumptions:	Tracking in place.

UC-20: Moderate Forum Content

UC ID and Name:	UC-20: Moderate Forum Content		
Created By:		Date Created:	
Primary Actor:	Admin	Secondary Actors:	Reporter (User)
Trigger:	Open moderation queue/case.		
Description:	Approve/Hide/Delete posts/comments; record decision.		
Preconditions:	Admin authenticated.		

Postconditions:	Content status updated; notifications/audit log created.
Normal Flow:	Review → choose action → apply → notify author/reporter.
Alternative Flows:	Escalate to user suspension.
Exceptions:	Item already actioned.
Priority:	High
Frequency of Use:	Medium
Business Rules:	Reason required; evidence kept; actions reversible when possible.
Other Information:	SLA; bulk actions.
Assumptions:	Clear content policy.

UC-21: Moderate Event Listings

UC ID and Name:	UC-21: Moderate Event Listings		
Created By:		Date Created:	
Primary Actor:	Admin	Secondary Actors:	Organizer

Trigger:	Review reported event.
Description:	Approve/Hide/Delete non-compliant events.
Preconditions:	Admin authenticated; event exists.
Postconditions:	Event status changed; stakeholders notified; audit logged.
Normal Flow:	Inspect → decide → apply → notify organizer/registrants if needed.
Alternative Flows:	Request edits from organizer.
Exceptions:	Event already removed.
Priority:	High
Frequency of Use:	Medium
Business Rules:	Preserve history; follow state transition rules.
Other Information:	Template messages for reasons.
Assumptions:	Organizer reachable.

UC-22: Manage Users & Roles

UC ID and Name:	UC-22: Manage Users & Roles

Created By:		Date Created:	
Primary Actor:	Admin	Secondary Actors:	Affected User
Trigger:	Open **Users** admin page.		
Description:	Ban/unban; assign/revoke Organizer role; view violations.		
Preconditions:	Admin authenticated.		
Postconditions:	Status/roles updated; audit entries created; user notified.		
Normal Flow:	Search → select user → choose action → confirm → persist & notify.		
Alternative Flows:	Temporary suspensions with expiry.		
Exceptions:	Cannot demote sole Admin.		
Priority:	High		
Frequency of Use:	Medium		
Business Rules:	Principle of least privilege; justification required.		
Other Information:	All changes audited.		
Assumptions:	Admin has proper authorization.		

UC-23: System Notifications

UC ID and Name:	UC-23: System Notifications		
Created By:		Date Created:	
Primary Actor:	System/Admin	Secondary Actors:	Email/Notification Service
Trigger:	Event-driven (registration, updates, comments, moderation).		
Description:	Send email/in-app notifications.		
Preconditions:	Trigger event occurred; user has not disabled notifications.		
Postconditions:	Notification queued/sent; status stored.		
Normal Flow:	Detect event → compose template → queue/send → record delivery.		
Alternative Flows:	Fallback to in-app if email fails.		
Exceptions:	Provider outage; exceeded rate limits.		
Priority:	High		
Frequency of Use:	High		
Business Rules:	Respect user preferences; throttle non-critical messages; critical updates override.		

Other Information:	Exponential retry/backoff; idempotent sends.
Assumptions:	Providers (SMTP/Push) are reachable.

UC-24: Process Event Payment

UC ID and Name:	UC-24: Process Event Payment		
Created By:		Date Created:	
Primary Actor:	Participant	Secondary Actors:	Payment Gateway (Stripe/VNPay/Momo), Organizer, Notification Service, Webhook Handler
Trigger:	User clicks **Pay Now/Checkout** on a paid event.		
Description:	Handles online event ticket payments; validates via webhook, updates registration status to *paid*, and sends receipts to both user and organizer.		
Preconditions:	User is logged in; event is *Published* and not full; valid payment method and order exist.		
Postconditions:	Payment is recorded; registration marked as *paid*; event capacity updated; confirmation sent via email and in-app notification.		
Normal Flow:	1. User selects ticket quantity 2. Applies promo code (if any) 3. System creates checkout session/order		

	Redirects to payment gateway 4. User completes payment successfully Webhook confirms transaction 5. System marks registration as *paid* and generates receipt/ticket 6. Notification sent to both sides.
Alternative Flows:	User cancels at gateway → registration marked *unpaid/canceled* Payment fails → retry or store as pending Refund or promo adjustment handled via Admin/Organizer.
Exceptions:	Invalid webhook signature, amount mismatch, timeout, or duplicate confirmation.
Priority:	High
Frequency of Use:	High
Business Rules:	Each transaction is *idempotent* by provider ID; receipts stored; refunds follow policy; applicable taxes configurable
Other Information:	No credit card data stored; HTTPS enforced; audit logs maintained.
Assumptions:	Payment provider is operational; webhooks arrive securely.

UC-25: Sync Event to Google Calendar

UC ID and Name:	UC-25: Sync Event to Google Calendar		
Created By:		Date Created:	

Primary Actor:	Participant	Secondary Actors:	Google Calendar API, Notification Service
Trigger:	User clicks **Add to Google Calendar** or enables auto-sync after event registration		
Description:	Creates and synchronizes registered events to Google Calendar; automatically updates when organizer changes time or venue.		
Preconditions:	User logged in and registered for the event; granted Calendar OAuth permission.		
Postconditions:	Calendar event created/updated; `calendar_event_id` stored in system; reminders activated.		
Normal Flow:	1. User selects *Add to Calendar* 2. OAuth consent if first-time use 3. API call creates event on Calendar 4. System stores `calendar_event_id` 5. Displays success message 6. Any future change in event updates Calendar automatically.		
Alternative Flows:	User revokes permission → prompt to reconnect User deletes Calendar entry → mark `sync_status = removed` Provide downloadable ICS file as fallback.		
Exceptions:	Token expired, API quota exceeded, or API errors (4xx/5xx).		
Priority:	Medium		
Frequency of Use:	Medium		

Business Rules:	Only registered events can sync; maintain timezone consistency; updates are *idempotent* by Calendar ID.
Other Information:	Must comply with Google API policies and store minimal user data.
Assumptions:	Google API available; user has valid Google account.

UC-26: Chat with Organizer

UC ID and Name:	UC-26: Chat with Organizer		
Created By:		Date Created:	
Primary Actor:	Participant, Organizer	Secondary Actors:	Notification Service, Media/CDN Service, Moderation Service
Trigger:	User clicks **Chat with Organizer** on event page or Organizer replies from dashboard.		
Description:	Provides 1:1 chat between participants and organizers for event-related inquiries, attachments, and logistics.		
Preconditions:	User logged in; event is *Published*; both users not blocked; conversation exists or will be created.		
Postconditions:	Message stored; read receipts updated; notifications sent to both sides.		
Normal Flow:	1. User opens conversation		

	2. Composes and sends message 3. Message (and attachment if any) saved in DB/CDN 4. Notification sent to receiver 5. Receiver replies 6. System updates read status.
Alternative Flows:	User reports or blocks sender → conversation closed; rate-limit for spam prevention; mute notifications.
Exceptions:	Attachment too large; file type not allowed; user banned.
Priority:	High
Frequency of Use:	High
Business Rules:	Limit message size/type; content moderation filter; retention period per policy; one conversation per event-user pair.
Other Information:	Real-time updates via WebSocket; fallback email digest if offline.
Assumptions:	Notification and CDN services are active; both users accept chat policy.

UC-27: View Event Map

UC ID and Name:	UC-27: View Event Map		
Created By:		Date Created:	
Primary Actor:	Participant	Secondary Actors:	Map API Provider (Google/Mapbox)

Trigger:	User opens event detail page or clicks **View Map/Directions**.
Description:	Displays map, venue marker, and distance or route from user's current location (if permission granted).
Preconditions:	Event has valid address or latitude/longitude; API key configured.
Postconditions:	Map rendered; distance and route displayed if available.
Normal Flow:	1. Page loads event details 2. System geocodes address if needed 3. Map and marker displayed 4. If user grants location permission, calculate distance 5. Display directions or open external map app.
Alternative Flows:	Location denied → show marker only; invalid address → show text fallback.
Exceptions:	API quota exceeded, key invalid, or connection failure.
Priority:	Medium
Frequency of Use:	High
Business Rules:	Cache geocode results; store lat/lng for efficiency; enforce API quota.
Other Information:	Encrypted API calls; optional dark mode map styling.
Assumptions:	Map API service available and accessible.

UC-28: Ask AI Chatbot for Event Suggestions

UC ID and Name:	UC-28: Ask AI Chatbot for Event Suggestions		
Created By:		Date Created:	
Primary Actor:	Participant	Secondary Actors:	AI Service (LLM), Search/Index Module, Map API
Trigger:	User opens Chatbot widget and sends natural-language query (e.g., “What events are happening near me under $10?”).		
Description:	AI Chatbot interprets the user query, extracts filters (time, location, budget, topic), queries the event database, and returns personalized recommendations with quick actions (Register, Save, Open Map).		
Preconditions:	AI integration enabled; LLM guardrails and logging active; search index updated.		
Postconditions:	Event recommendations displayed; query and extracted filters logged; user may continue chat for refinement.		
Normal Flow:	1. User sends question 2. AI extracts intent and parameters 3. System queries DB/Search index 4. AI formats and ranks results 5. Chatbot displays list with actions 6. Logs interaction and optional feedback.		
Alternative Flows:	No suitable events found → AI suggests relaxing filters; missing location → requests permission; fallback to regular filter search.		

Exceptions:	LLM timeout; provider rate-limit; invalid response → show safe fallback message.
Priority:	Medium–High
Frequency of Use:	High
Business Rules:	Prevent disclosure of personal data; monitor output safety; limit session token and API cost.
Other Information:	Prompts versioned and A/B tested; show disclaimer (“Recommendations may vary”).
Assumptions:	AI provider operational; user allows location sharing when requested.

Business Rules

Provide the business rules those are applied only to the use case

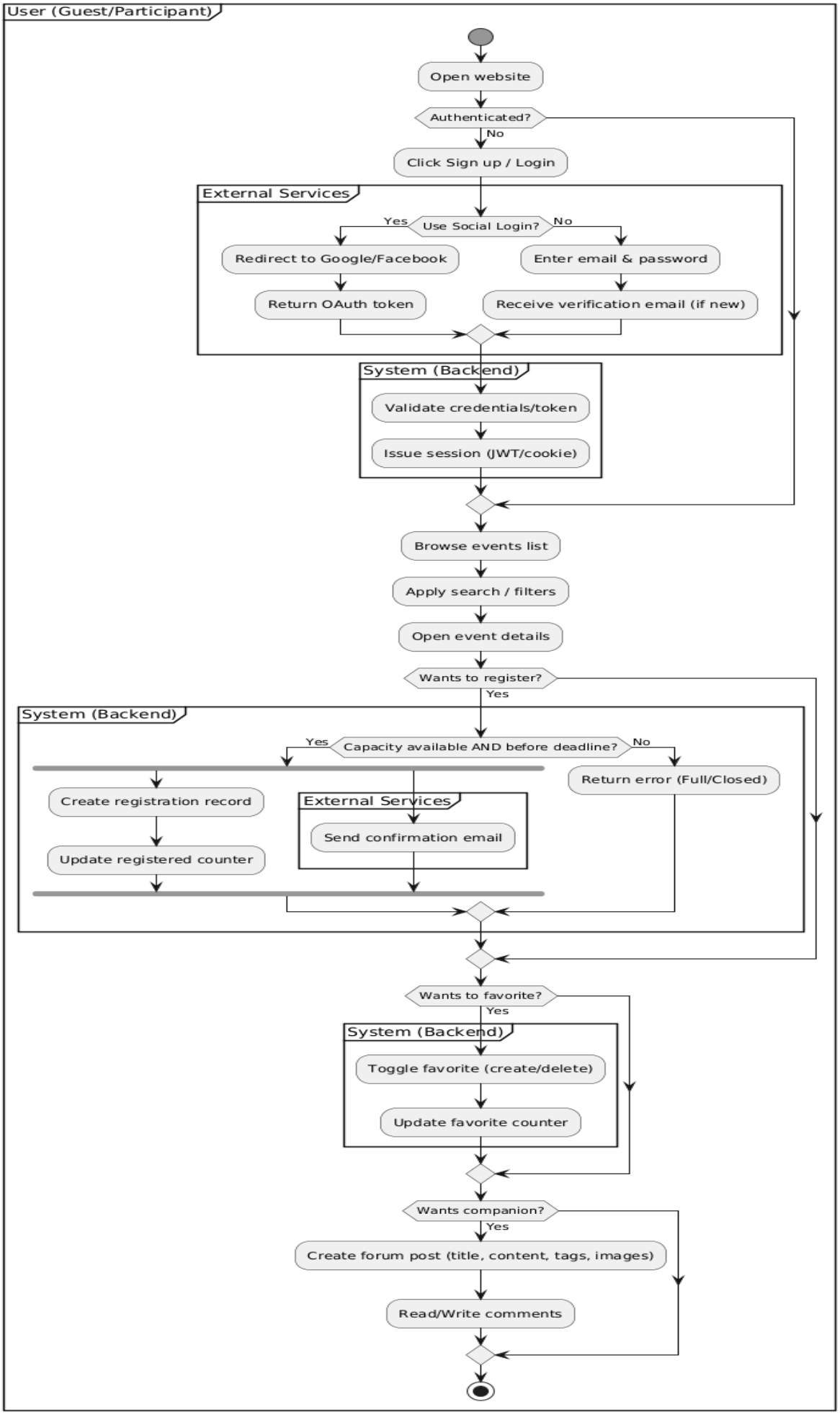
ID	Business Rule	Business Rule Description
FR1	**Password Hashing**	User passwords must be hashed with Argon2 or bcrypt with per-user salts before storage (MD5 is not allowed).
FR2	**Email Verification**	The system must send a verification email upon registration; accounts must be verified before creating events or posting in the forum.
FR3	**Role & RBAC**	Role is selected at sign-up (Participant/Organizer); access to organizer dashboard and event management is restricted to **Organizer**; **Admin** has moderation and user-management only.

FR4	**Event Publication Workflow**	Events have states Draft → Published → Closed
FR5	**Registration Capacity**	Registrations may not exceed event **capacity**; one active registration per user per event; optional waitlist may be used when full.
FR6	**Favorites Uniqueness**	Each user may favorite an event at most once; favorite counts reflect **unique** users and update on toggle.
FR7	**Content Moderation**	Forum posts, comments, and events must comply with community policy; items may be auto-flagged/hidden after a threshold of reports; all moderator actions are **audit-logged**.
FR8	**Data Privacy & Contact Sharing**	Participant contact info is hidden by default; organizers can view only fields the participant has **consented** to share; all PII must be handled per applicable privacy laws.

II.5.3 Activity diagram

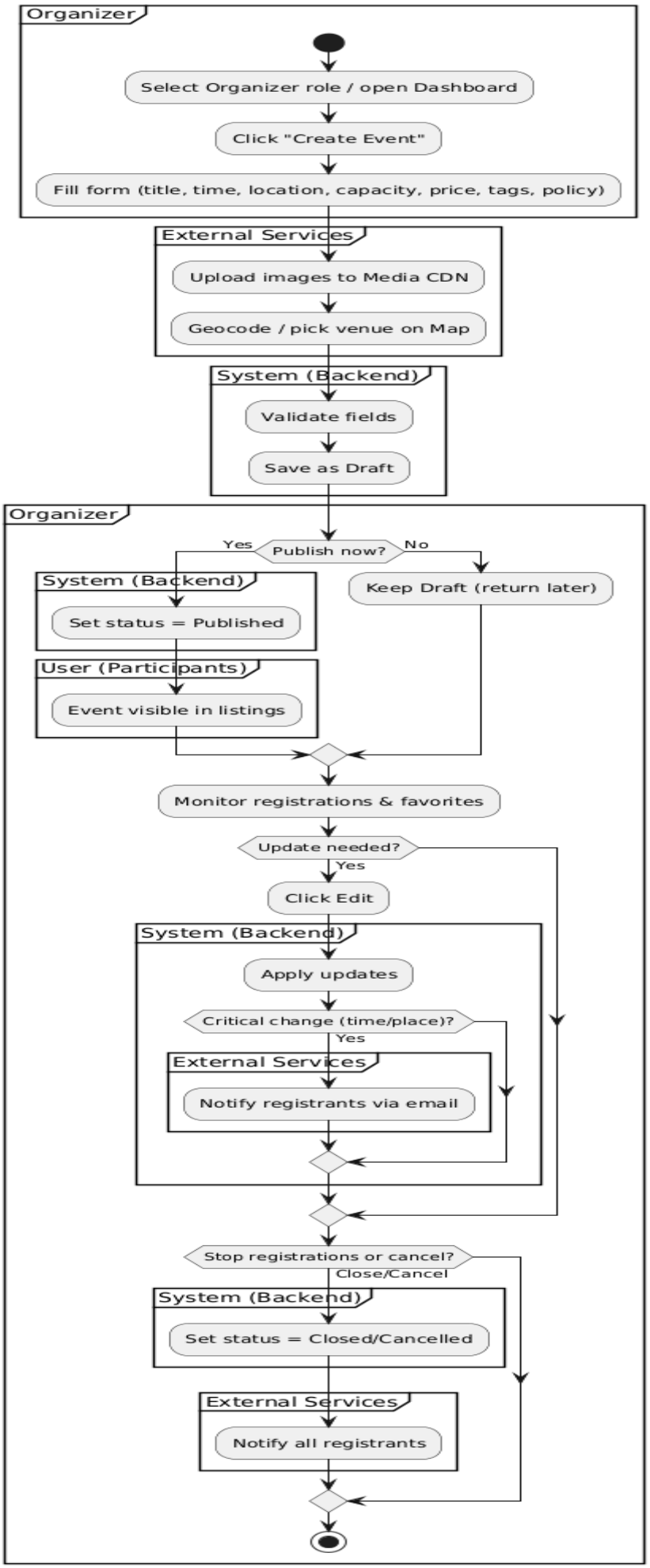
1) Participant(User):

Participant Journey: Discover, Register, Favorite, Forum



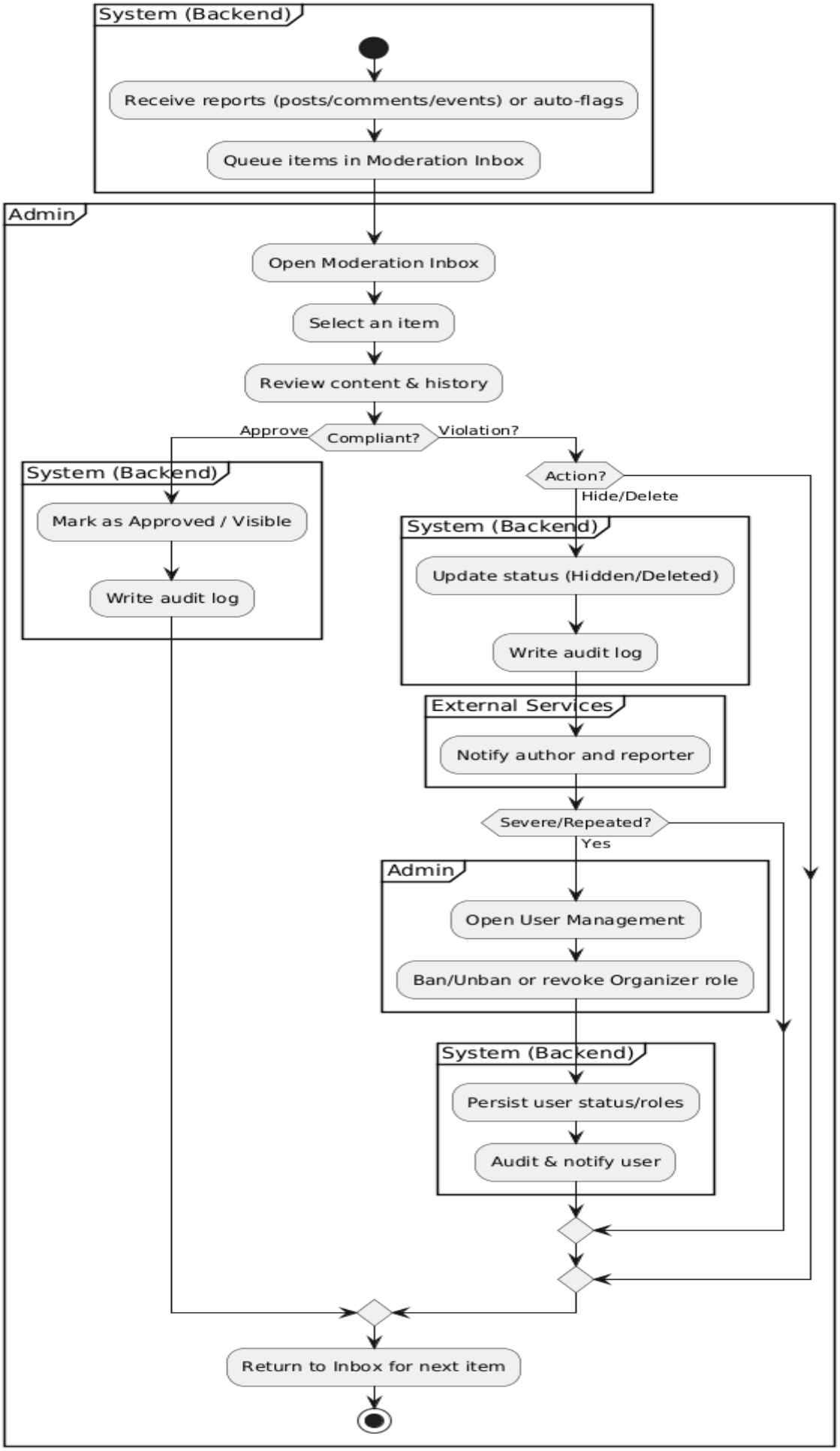
2) Organizer:

Organizer Flow: Create, Publish, Manage Event

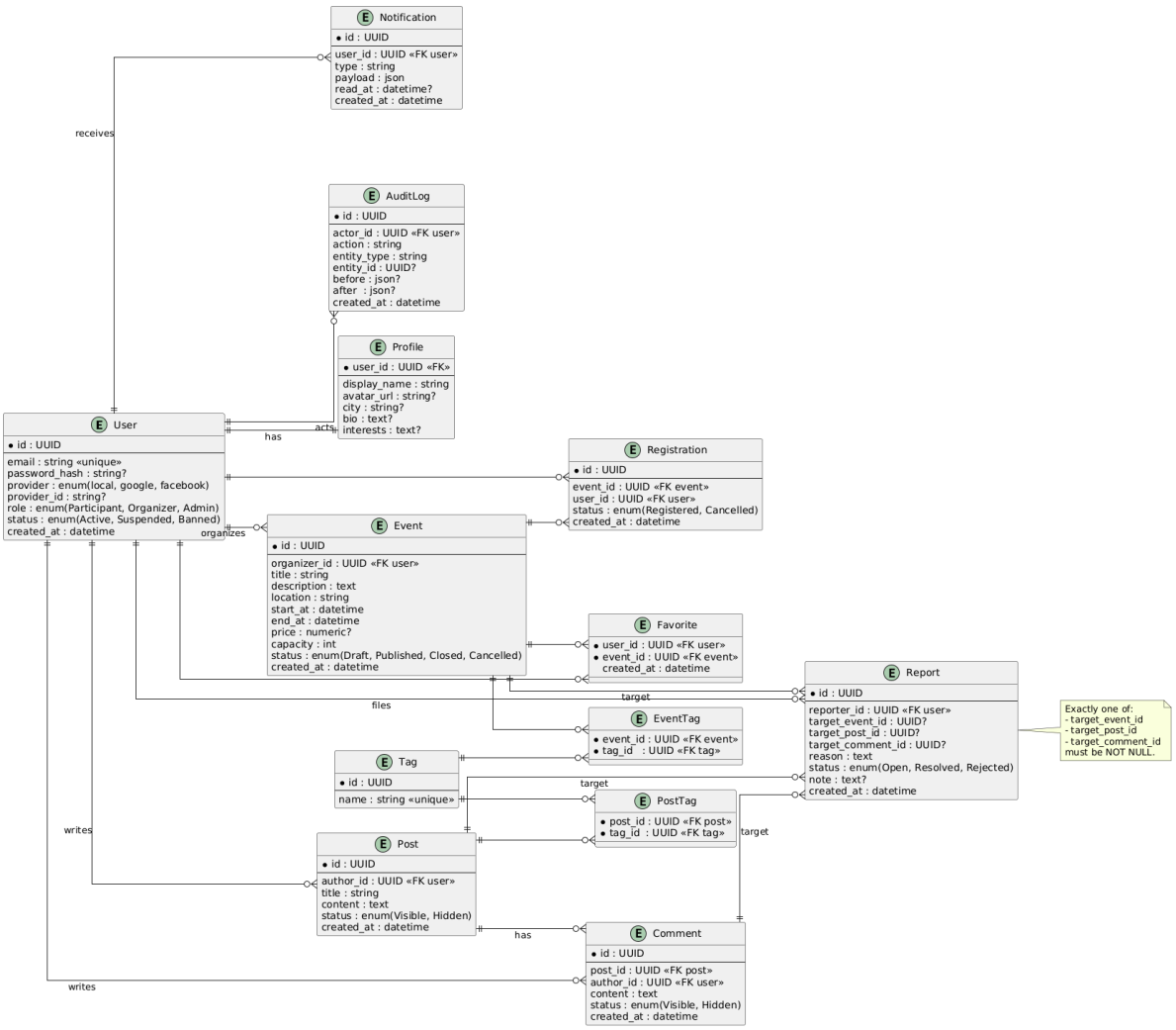


3) Admin

Admin Flow: Moderate Content and Manage Users



II.6 Entity Relationship Diagram



Entity Descriptions:

User

- **Description: Account entity controlling access and roles.**
- **Purpose: Authentication/authorization and ownership.**
- **Key fields: `id`, `email` (unique), `password_hash?`, `provider`, `provider_id?`, `role` (Participant/Organizer/Admin), `status`, `created_at`.**
- **Notes/Relations: 1–1 `Profile`; 1–N `Event` (if Organizer); N–N via `Registration`, `Favorite`; 1–N `Post`, `Comment`, `Notification`, `AuditLog`.**

Profile

- **Description: Public-facing user profile linked to a user.**
- **Purpose: Stores display information.**
- **Key fields: `user_id` (PK/FK), `display_name`, `avatar_url?`, `city?`, `bio?`,**

`interests?`.
- **Notes: Exactly one profile per user; privacy rules apply.**

Event

- **Description: An event listing created by an organizer.**
- **Purpose: Core entity for discovery and attendance.**
- **Key fields: `id`, `organizer_id`, `title`, `description`, `location`, `start_at`, `end_at`, `price?`, `capacity`, `status`, `created_at`.**
- **Notes: Tags via `EventTag`; attendees via `Registration`; likes via `Favorite`.**

Registration

- **Description: A user's attendance record for an event.**
- **Purpose: Track seats/attendance and status.**
- **Key fields: `id`, `event_id`, `user_id`, `status` (Registered/Cancelled), `created_at`.**
- **Notes: Unique (`event_id`,`user_id`); respects capacity and deadlines.**

Favorite

- **Description: A user's saved/liked event.**
- **Purpose: Quick access and popularity counting.**
- **Key fields: `user_id`, `event_id`, `created_at`.**
- **Notes: Composite PK (`user_id`,`event_id`) ensures uniqueness.**

Post

- **Description: Forum post to find companions or discuss events.**
- **Purpose: User-generated content for coordination.**
- **Key fields: `id`, `author_id`, `title`, `content`, `status` (Visible/Hidden), `created_at`.**
- **Notes: Tags via `PostTag`; comments via `Comment`.**

Comment

- **Description: User comment on a forum post.**
- **Purpose: Discussion thread unit.**
- **Key fields: `id`, `post_id`, `author_id`, `content`, `status`, `created_at`.**
- **Notes: Subject to moderation and rate limits.**

Report

- **Description: Violation report against content or events.**
- **Purpose: Feed for moderation workflow.**
- **Key fields: `id`, `reporter_id`, `target_event_id?`, `target_post_id?`, `target_comment_id?`, `reason`, `status`, `note?`, `created_at`.**
- **Notes: Exactly one target FK must be non-null.**

Notification

- **Description: Email/in-app message to a user.**
- **Purpose: Inform users about registrations, updates, moderation.**
- **Key fields: `id`, `user_id`, `type`, `payload` (JSON), `read_at?`, `created_at`.**
- **Notes: Generated by event-driven triggers.**

AuditLog

- **Description: Immutable record of sensitive actions and changes.**
- **Purpose: Compliance and traceability.**
- **Key fields: `id`, `actor_id`, `action`, `entity_type`, `entity_id?`, `before?` (JSON), `after?` (JSON), `created_at`.**
- **Notes: Covers admin and critical content updates.**

Tag

- **Description: Label used for classification and search.**
- **Purpose: Improve discovery and filtering.**
- **Key fields: `id`, `name` (unique).**
- **Notes: Linked to events/posts via junctions.**

EventTag

- **Description: Junction for Event–Tag many-to-many.**
- **Purpose: Associate tags with events.**
- **Key fields: `event_id`, `tag_id`.**
- **Notes: Composite PK (`event_id`,`tag_id`).**

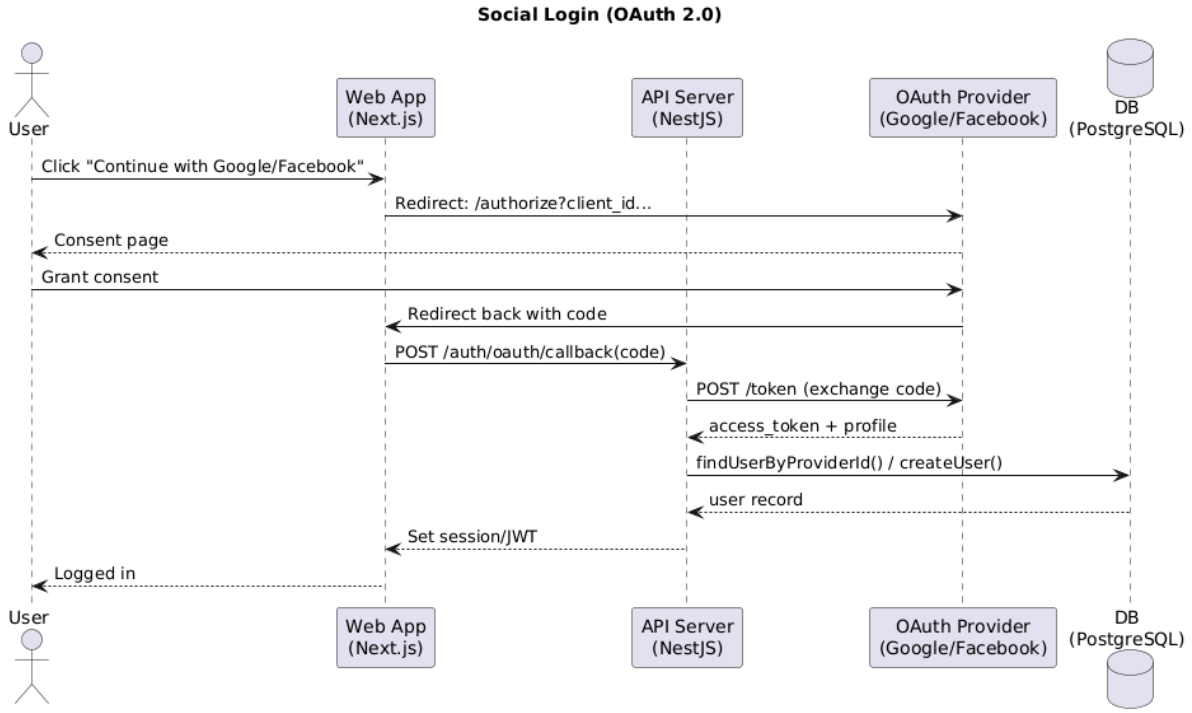
PostTag

- **Description: Junction for Post–Tag many-to-many.**
- **Purpose: Associate tags with posts.**
- **Key fields: `post_id`, `tag_id`.**
- **Notes: Composite PK (`post_id`,`tag_id`).**

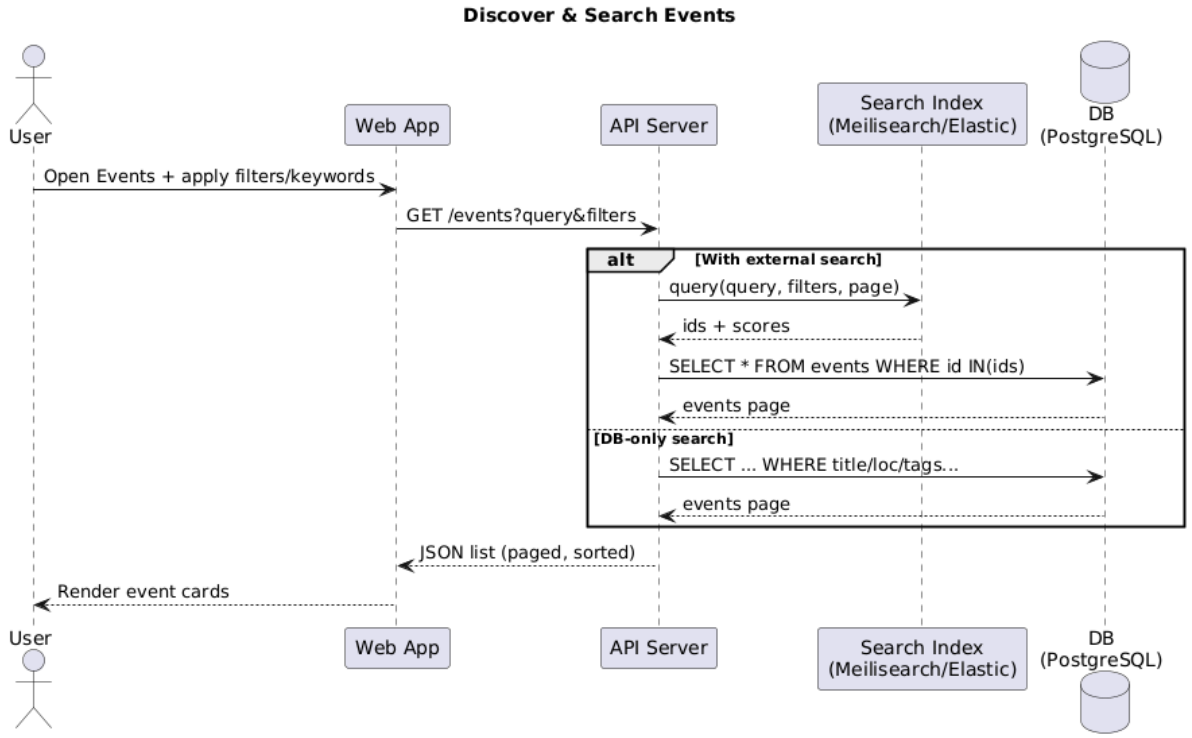
III. Analysis models.

III.1.1.Sequence Diagram

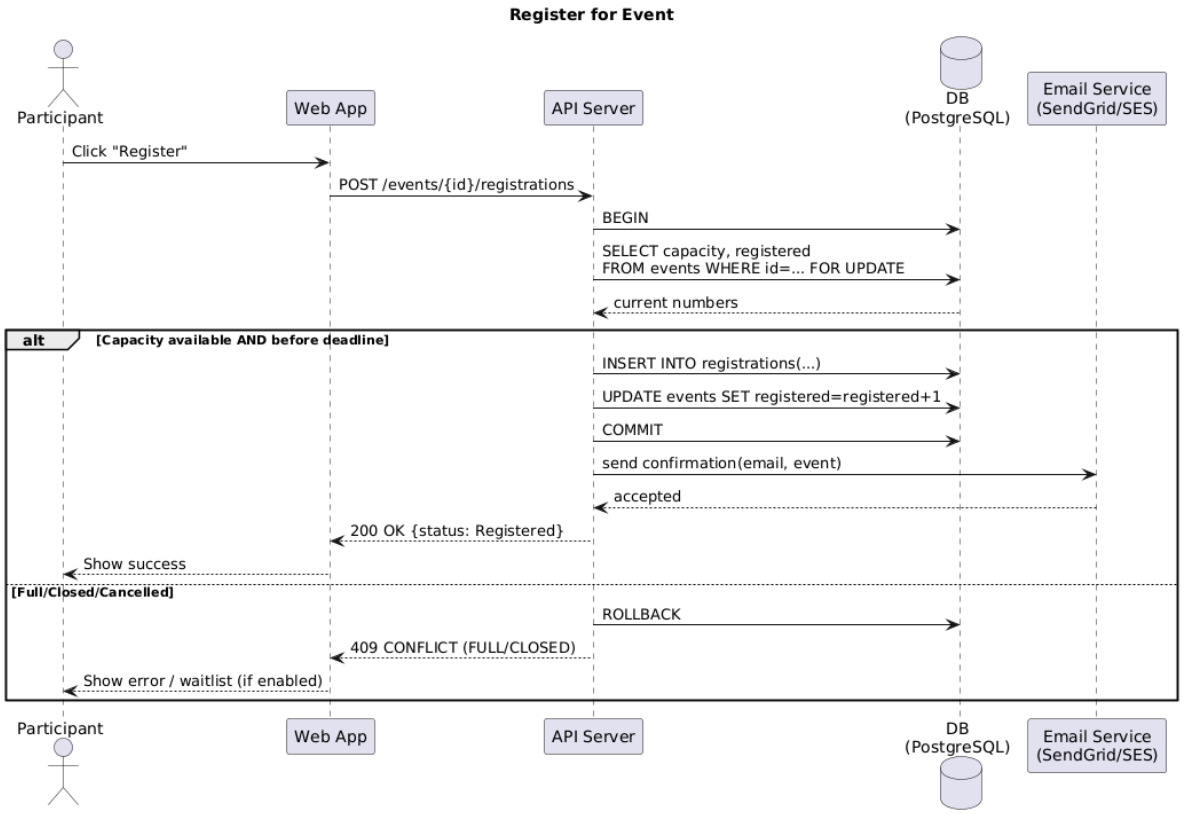
1) Social Login (Google/Facebook)



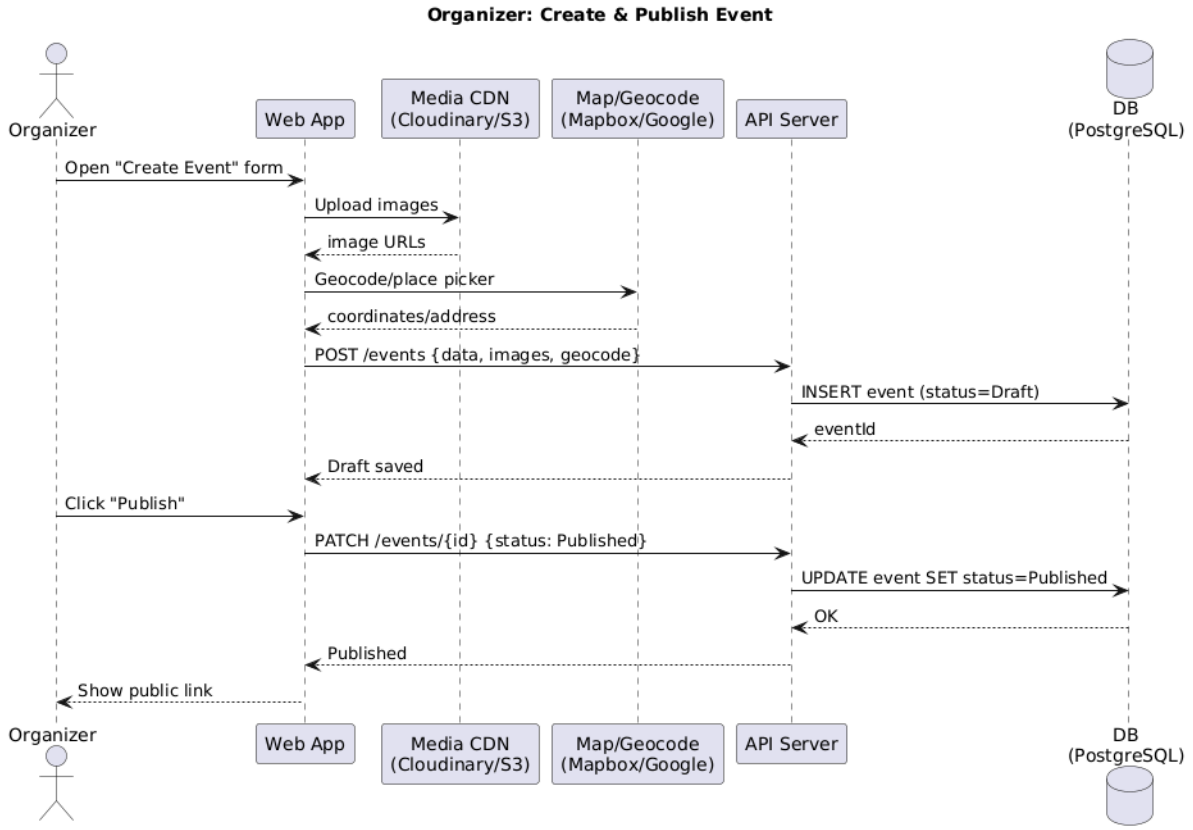
2) Discover & Search Events



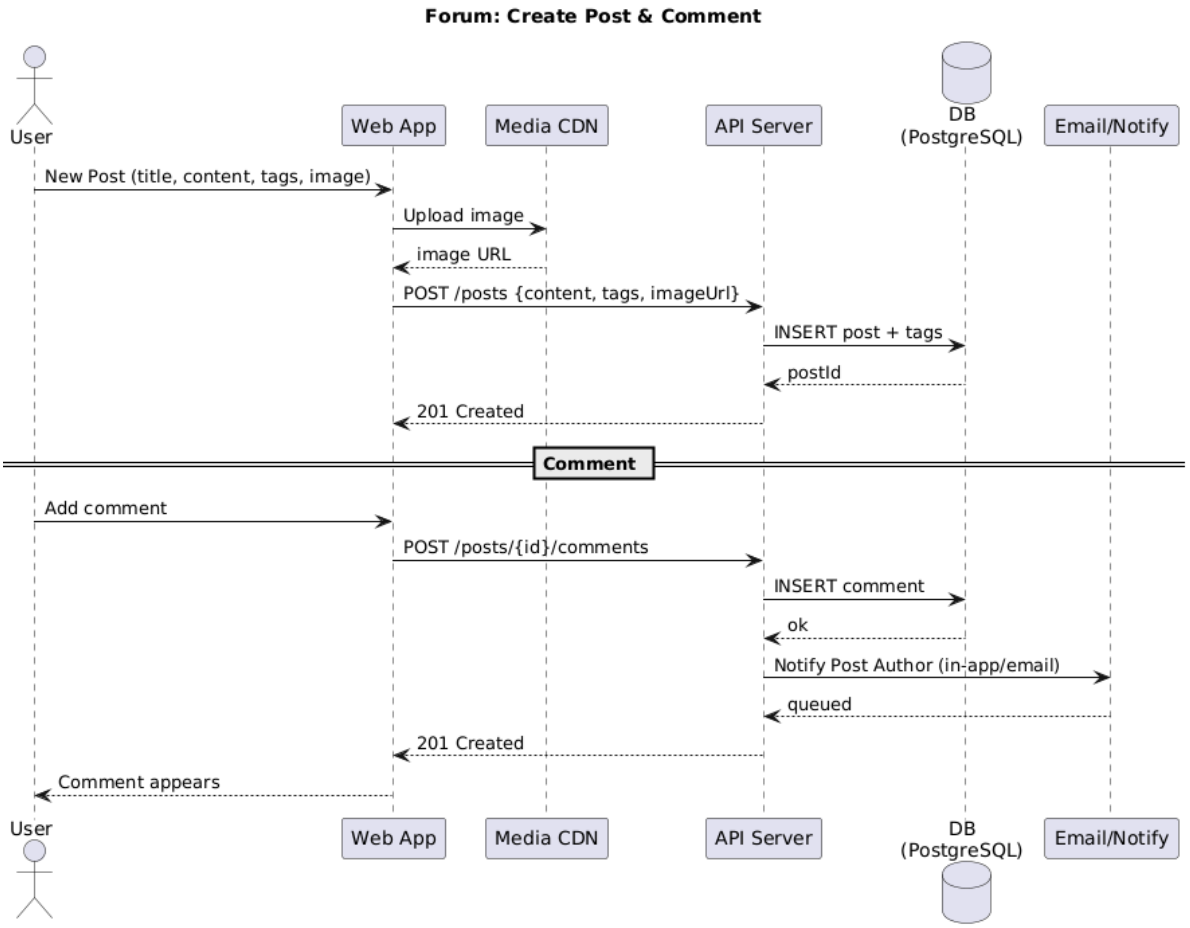
3) Register for Event



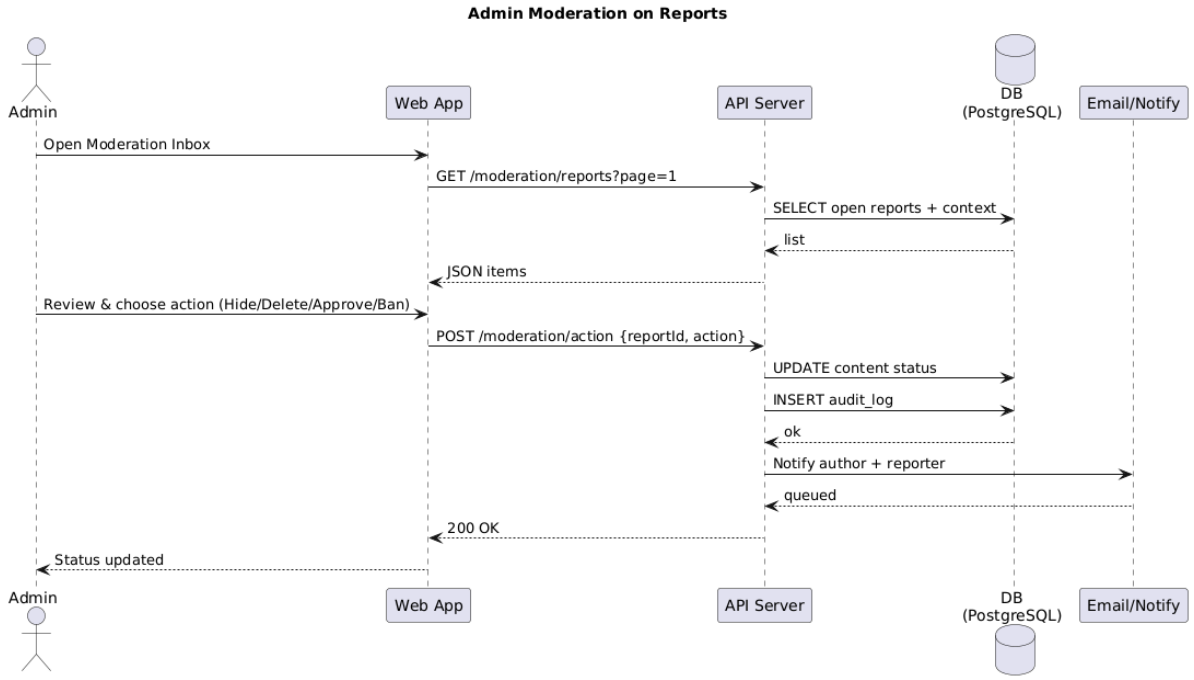
4) Organizer: Create & Publish Event



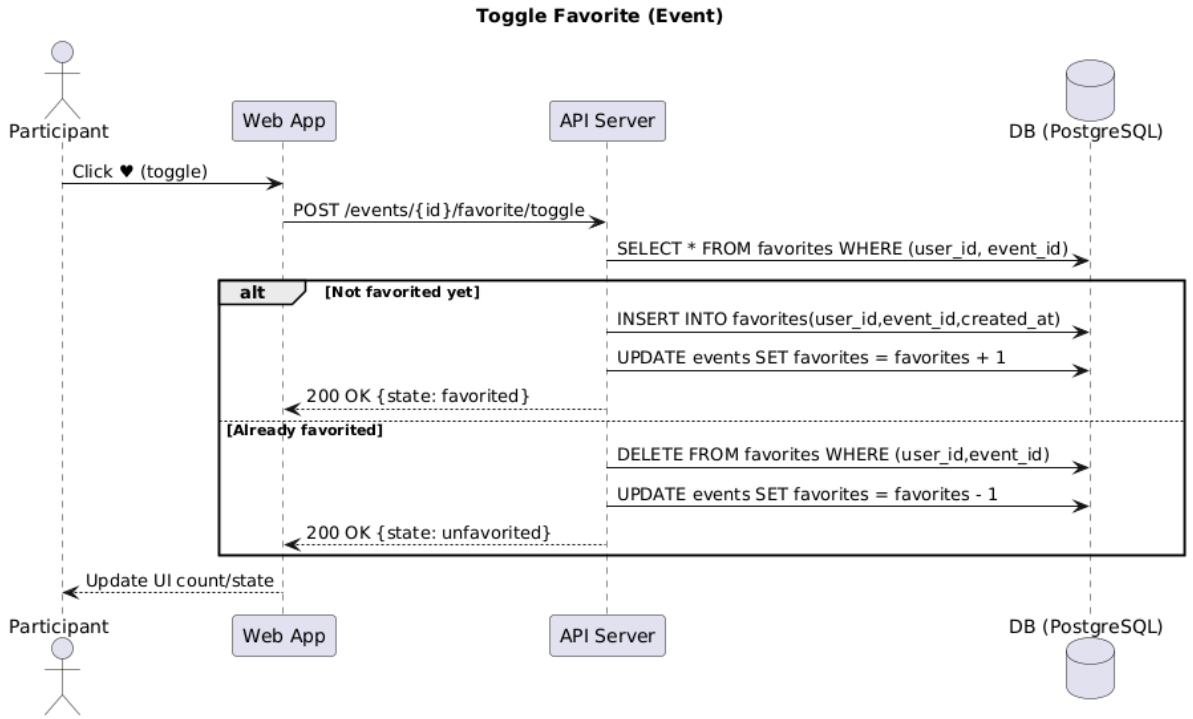
5) Forum: Create Post & Comment (+ notify author)



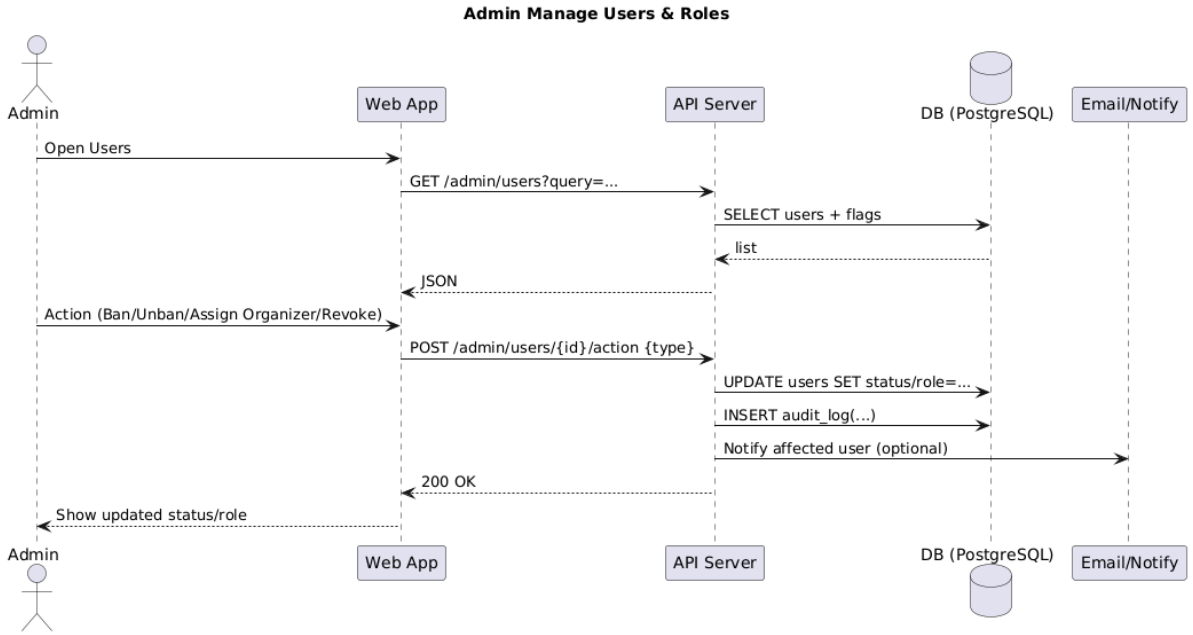
6) Admin Moderation: Act on Reported Content



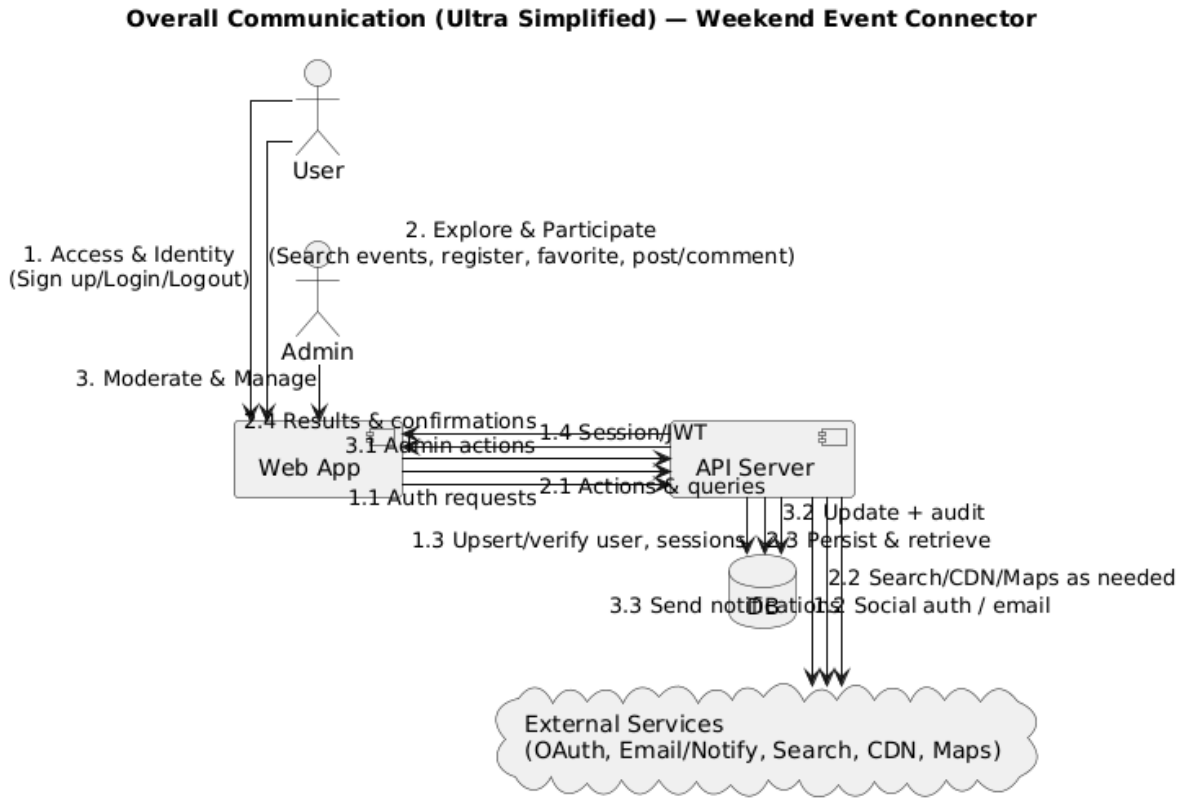
7) Favorite / Unfavorite Event (toggle)



8) Admin – Manage Users & Roles (ban / organizer role)



III.1.2. Communication Diagram



Explanation:

- Actors:
User (all regular users) and **Admin** (moderators/operators).

- Core components:
Web App (frontend UI) → **API Server** (single backend gateway) → **DB** (system source of truth).
External Services bundle OAuth, Email/Notifications, Search, CDN (images), and Maps.

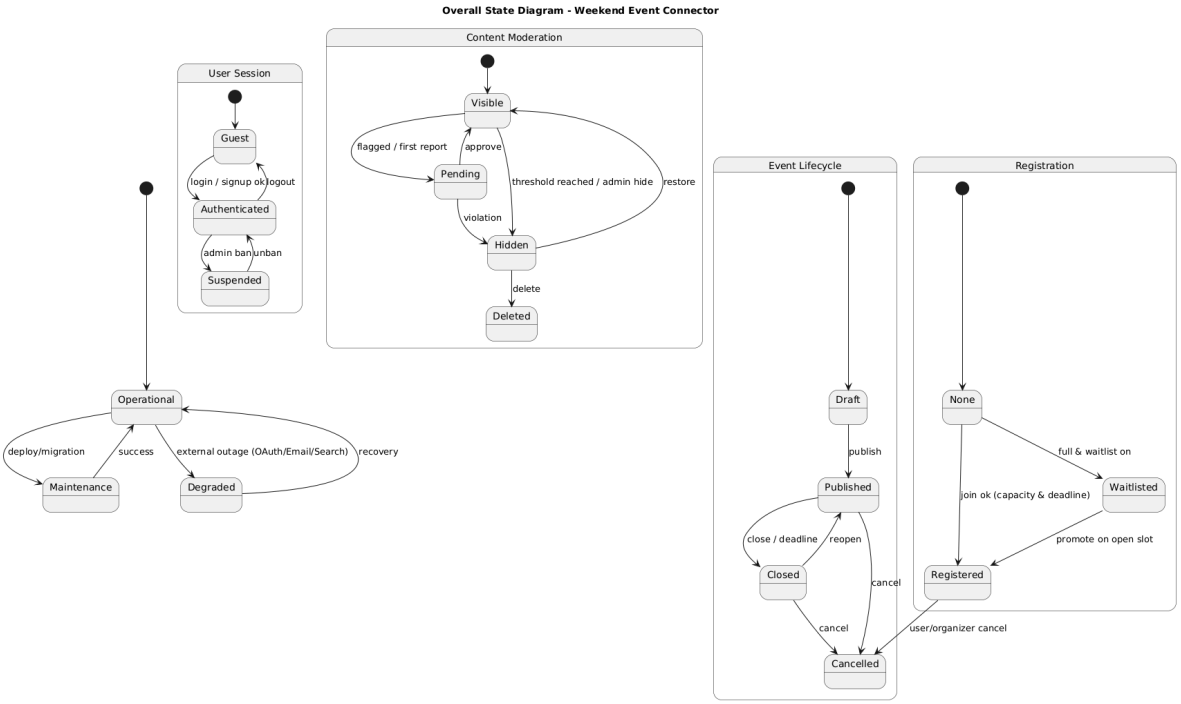
- Main flows:

1. **Access & Identity:** User signs up/logs in/logs out via the Web App. The API handles auth, talks to **External Services** for social login/email, and updates **DB** (users, sessions).
2. **Explore & Participate:** User browses/searches events, registers, favorites, and posts/comments. Web sends requests to API; API reads/writes **DB** and uses **External Services** as needed (search, images, maps); results/confirmations go back to the Web.

3. **Admin & Notifications:** Admin moderates content and manages users via the Web → API. API updates **DB** with audit logs and triggers notifications through **External Services**.

Key idea: The **Web App** is the single entry point; the **API Server** coordinates all logic and integrations; the **DB** keeps authoritative data; **External Services** are used on demand.

III.2 State Diagram



Explanation:

System (top level)

- **Operational** – platform runs normally.
- **Maintenance** – deploy/migrations; returns to Operational when done.
- **Degraded** – external outages (OAuth/Email/Search); recovers to Operational.

User Session *(runs concurrently under Operational)*

- **Guest** → **Authenticated** on successful sign-up/login.
- **Authenticated** → **Guest** on logout.
- **Authenticated** → **Suspended** when banned; **Suspended** → **Authenticated** when unbanned.

Event Lifecycle

- **Draft** → **Published** (publish).
- **Published** → **Closed** (registration closed/deadline) or → **Cancelled**.
- **Closed** ↔ **Published** (reopen), **Closed** → **Cancelled** (final).

Registration

- **None** → **Registered** if capacity ok and before deadline.
- **None** → **Waitlisted** when full (if waitlist enabled).
- **Registered** → **Cancelled** by user/organizer.
- **Waitlisted** → **Registered** when a slot opens.

Content Moderation

- **Visible** ↔ **Pending** (reported/auto-flagged → review; approved → visible).
- **Pending** → **Hidden** (violation).
- **Visible** → **Hidden** (threshold reached/admin hide).
- **Hidden** ↔ **Visible** (restore), **Hidden** → **Deleted** (remove).

IV. Design specification

IV.1 Integrated Communication Diagrams

1) Mục tiêu

Mô tả cách các tác nhân (User/Organizer/Admin) giao tiếp với các thành phần lõi (Web App, API Server, DB) và dịch vụ bên ngoài (OAuth, Email/Notify, Search, CDN, Maps) trong **một sơ đồ tích hợp**, giúp nhìn nhanh toàn bộ luồng dữ liệu – không tách rời theo chức năng.

2) Ranh giới & phạm vi

- **Trong phạm vi:** Giao tiếp đồng bộ kiểu request/response (HTTP/JSON), gọi dịch vụ ngoài, cập nhật DB, gửi thông báo.
- **Ngoài phạm vi:** Logic chi tiết từng API, retry/backoff cấp hạ tầng, batching nâng cao.

3) Thành phần & tác nhân

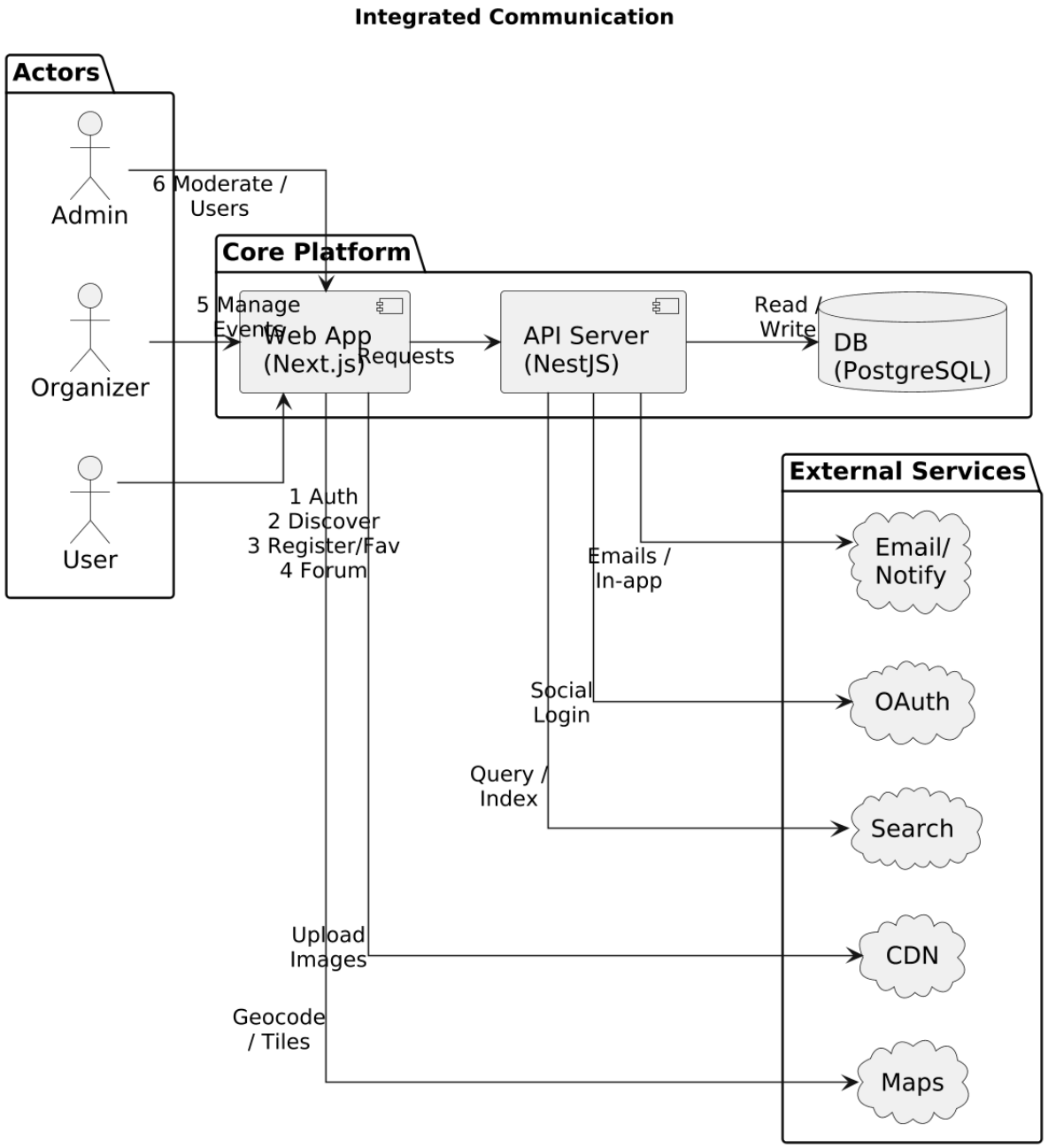
- **Tác nhân:** Guest/Participant (gọi chung **User**), **Organizer**, **Admin**.

- **Thành phần lõi: Web App (Next.js)**, **API Server (NestJS)**, **DB (PostgreSQL)**.
- **Dịch vụ ngoài: OAuth (Google/Facebook)**, **Email/Notify (SendGrid/SES + in-app)**, **Search Index (Meilisearch/Elastic)**, **Media CDN (Cloudinary/S3)**, **Maps/Geocoding (Mapbox/Google)**.

4) Quy ước đánh số thông điệp

- **1.x** Xác thực & phiên đăng nhập
- **2.x** Khám phá/Tìm kiếm sự kiện
- **3.x** Đăng ký & Yêu thích
- **4.x** Diễn đàn (bài viết/bình luận)
- **5.x** Quản lý sự kiện (Organizer)
- **6.x** Kiểm duyệt & quản trị (Admin)

5) Sơ đồ giao tiếp tích hợp



6) Ghi chú thiết kế

- **Bảo mật: Tất cả luồng qua HTTPS; JWT cho phiên; RBAC ở API; rate-limit các endpoint nhạy cảm (auth, post, comment, register).**
- **Hiệu năng: Pagination server-side; cache ngắn cho danh sách sự kiện; tải bản đồ/CDN phía client.**
- **Khả dụng: Nếu Search/Maps/CDN lỗi, hệ thống vẫn phục vụ danh sách cơ bản từ DB.**
- **Theo dõi: Ghi audit cho hành động admin/organizer và thay đổi quan trọng.**

IV.2 System High-Level Design

1) Tổng quan

FREEDAY là nền tảng web kết nối người tìm kiếm sự kiện cuối tuần với người đăng sự kiện, kèm diễn đàn tìm bạn đồng hành và khu vực quản trị. Kiến trúc gồm **Web App (Next.js)**, **API Server (NestJS)**, **PostgreSQL (Prisma ORM)**, bổ trợ **Redis** (cache, rate-limit, job), **Email/Notify**, **Media CDN**, **Search Index** và **Maps/Geocoding**. Xác thực hỗ trợ **OAuth 2.0 (Google/Facebook)** và JWT. Thiết kế ưu tiên bảo mật (RBAC, audit), khả năng mở rộng (scale ngang) và khả dụng cao.

2) Thành phần

Frontend (Web App)

- **Framework:** Next.js 14 + React, Tailwind CSS (SSR/SSG cho SEO danh sách sự kiện).
- **Tính năng UI chính:**
  - Khám phá/tìm kiếm/lọc sự kiện; xem chi tiết.
  - Đăng ký/đăng xuất/đăng nhập (Email, Google, Facebook); quản lý hồ sơ.
  - Đăng ký tham gia, đánh dấu yêu thích, “Sự kiện của tôi”.
  - Diễn đàn: đăng bài tìm bạn đồng hành, bình luận, báo cáo vi phạm.
  - Dashboard Organizer: tạo/sửa/đóng/hủy sự kiện, xem số liệu.
  - Admin: hàng đợi kiểm duyệt, quản lý người dùng và nội dung.
- **Điều hướng & trạng thái:** Next Router; Zustand/Context; SWR/React Query cho cache dữ liệu phía client.

Backend (API Server)

- **Công nghệ:** Node.js + NestJS (REST), Passport (JWT + OAuth2), class-validator, Helmet.
- **Trách nhiệm:**
  - Xác thực/OAuth, RBAC (Guest/Participant/Organizer/Admin), rate-limit.
  - Nghiệp vụ sự kiện (CRUD, trạng thái Draft/Published/Closed/Cancelled).
  - Đăng ký tham gia, yêu thích, diễn đàn (post/comment), báo cáo vi phạm.
  - Hàng đợi kiểm duyệt, audit log, thông báo email/in-app.
- **Thiết kế API (ví dụ):**
  - **Auth:** `POST /auth/signup`, `POST /auth/login`, `GET /auth/oauth/callback`
  - **User/Profile:** `GET/PUT /me`, `GET /me/events`
  - **Events:** `GET /events?query&filters`, `GET /events/:id`
  - **Organizer:** `POST /events`, `PATCH /events/:id`, `PATCH /events/:id/status`
  - **Registration:** `POST /events/:id/registrations`, `DELETE /events/:id/registrations/me`
  - **Favorite:** `POST /events/:id/favorite/toggle`
  - **Forum:** `POST /posts`, `GET /posts`, `POST /posts/:id/comments`

    - **Report:** `POST /reports`
    - **Admin:** `GET /moderation/reports`, `POST /moderation/action`, `POST /admin/users/:id/action`

CSDL (PostgreSQL qua Prisma)

- **Mô hình chính:** `User`, `Profile`, `Event`, `Registration`, `Favorite`, `Post`, `Comment`, `Report`, `Notification`, `AuditLog`, `Tag` (+ bảng nối).
- **Ràng buộc & toàn vẹn:**
    - Uniq: email, (user_id,event_id) cho Favorite/Registration.
    - Ràng buộc capacity, deadline, trạng thái sự kiện (FSM).
    - Ghi **audit log** cho hành động admin/organizer & thay đổi quan trọng.

Cache/Queue (Redis)

- **Dùng cho:** rate-limit, session blacklist/refresh token revoke, đếm lượt xem/yêu thích tức thời, job gửi email/thông báo, retry với backoff.

Search Index (tùy chọn: Meilisearch/Elasticsearch)

- **Mục đích:** tìm kiếm full-text, lọc theo tag/địa điểm/thời gian; đồng bộ chỉ mục khi sự kiện/bài viết thay đổi.

Media Storage & CDN

- **Giải pháp:** Cloudinary hoặc S3 + CDN.
- **Dùng cho:** ảnh bìa/album sự kiện, ảnh bài viết; nén/biến thể kích thước; liên kết lưu trong DB.

Maps & Geocoding

- **Giải pháp:** Mapbox/Google Maps.
- **Dùng cho:** chọn địa điểm, geocode, hiển thị bản đồ trong chi tiết sự kiện.

Thông báo (Email/In-App/Web Push)

- **Email:** SendGrid/AWS SES cho xác minh tài khoản, xác nhận đăng ký, thay đổi thời gian/địa điểm, thông báo kiểm duyệt.
- **In-App/Web Push:** thông báo hoạt động (bình luận mới, bài bị ẩn/duyệt, cập nhật sự kiện).
- **Trigger:** sự kiện hệ thống (đăng ký, hủy, cập nhật, báo cáo, quyết định kiểm duyệt).

Bảo mật & Tuân thủ

- **Chuẩn:** HTTPS, JWT (HttpOnly cookie hoặc bearer), CORS, Helmet, input validation.
- **RBAC:** kiểm soát theo vai trò; endpoint organizer/admin tách biệt.

- **Chống tấn công:** CSRF (nếu cookie), XSS, SQLi (Prisma), rate-limit, captcha cho hành vi nhạy cảm.
- **Riêng tư:** tối thiểu PII, opt-in chia sẻ liên hệ với organizer, quyền tải/xóa dữ liệu.

Quan sát & Vận hành

- **Logging & APM:** Sentry/Datadog/OpenTelemetry; log tập trung JSON.
- **Metrics:** error rate, latency, 5xx, tỉ lệ thành công OAuth, backlog job.
- **Sao lưu:** backup DB hằng ngày; RPO ≤ 24h, RTO ≤ 4h.
- **Triển khai:** CI/CD (build/test/lint/scan), nhiều môi trường Dev/Staging/Prod; scale ngang API & Search; CDN cho media.

---

3) Kiến Trúc Hệ Thống

3.1 Mục tiêu kiến trúc (Drivers)

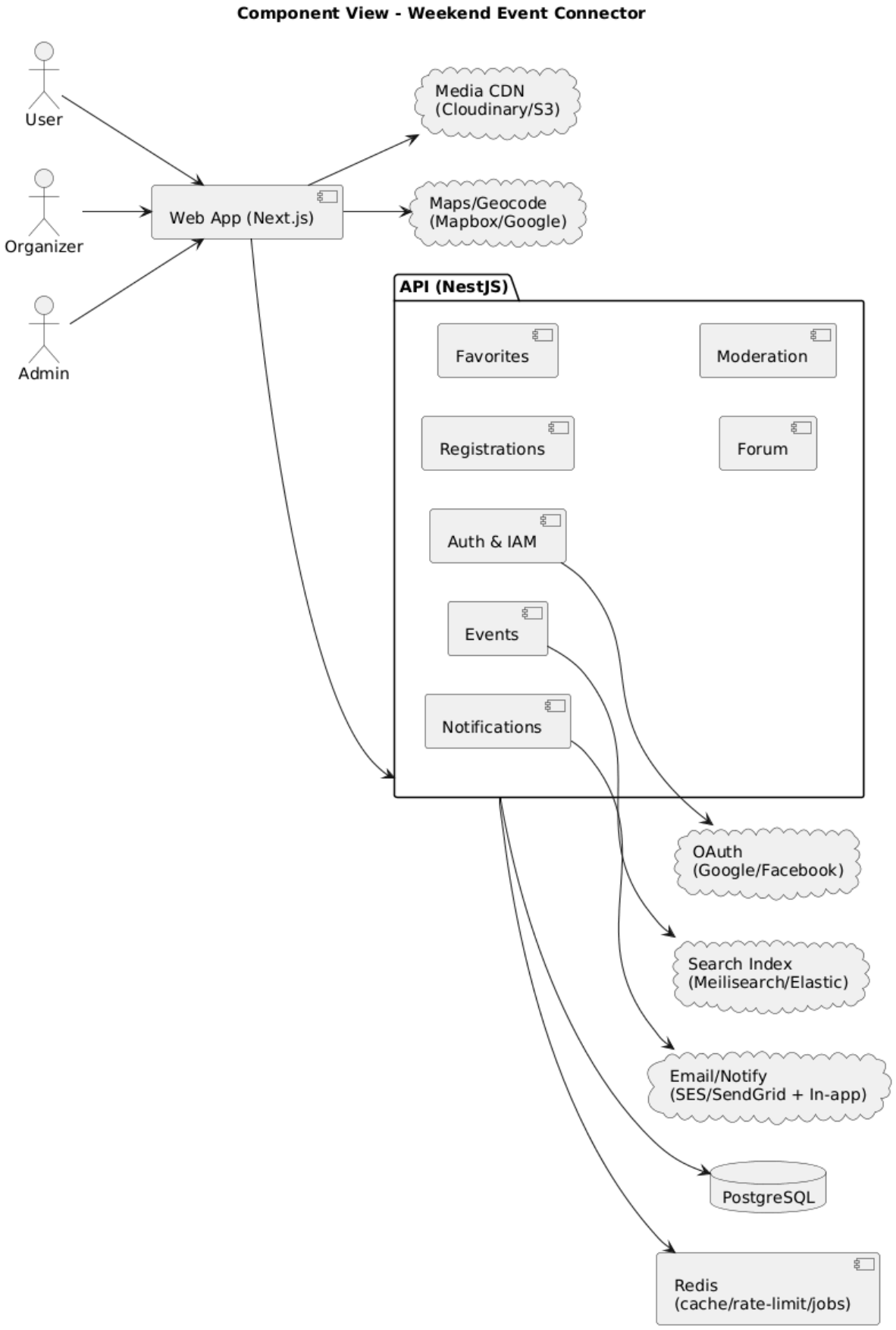
- **Chức năng cốt lõi: khám phá/tìm kiếm sự kiện, đăng ký, yêu thích, diễn đàn, quản trị kiểm duyệt.**
- **Phi chức năng: bảo mật (OAuth2 + JWT, RBAC, audit), hiệu năng (paging, cache), mở rộng ngang, sẵn sàng cao, dễ vận hành/quan sát.**
- **Ràng buộc: Web-first (SEO cho danh sách sự kiện), dùng dịch vụ ngoài OAuth/Email/CDN/Maps/Search.**

3.2 Phong cách & biên giới (Bounded Modules)

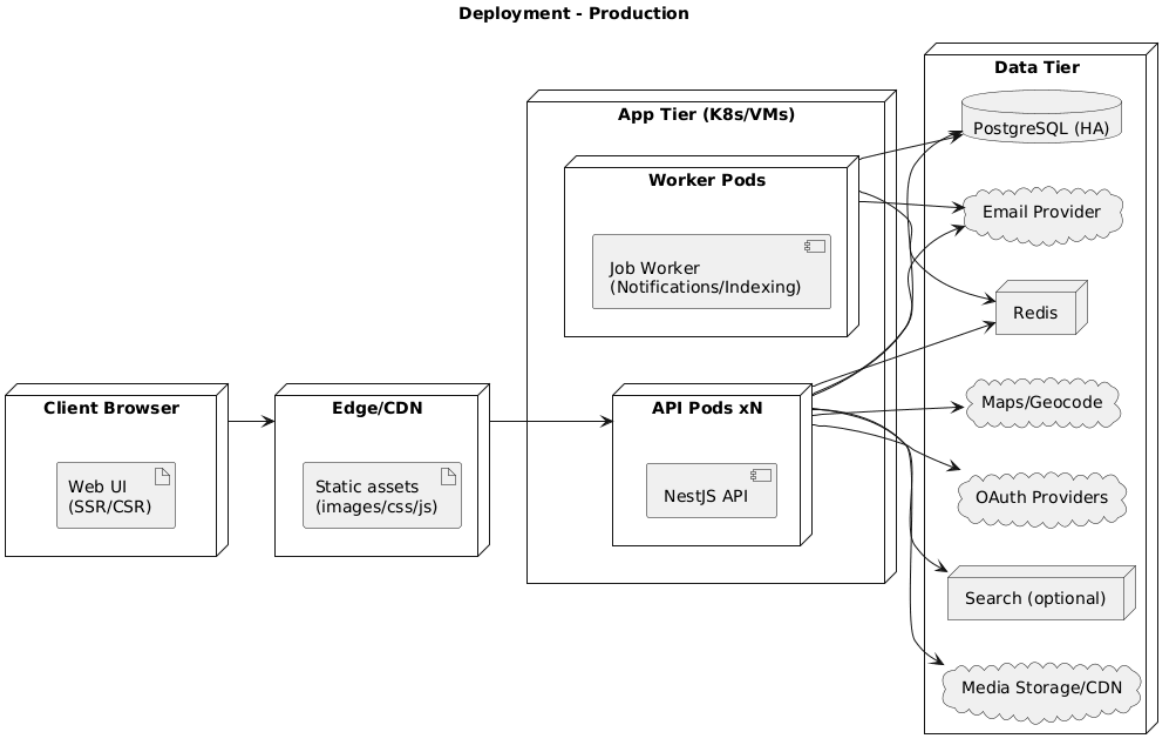
- **Kiểu: Modular-monolith (NestJS) + tác vụ nền (worker).**
- **Modules (chịu trách nhiệm & "sở hữu" bảng dữ liệu):**
    - **Auth & IAM: đăng ký/đăng nhập, OAuth2, JWT, khóa/mở khóa tài khoản. *(User, Profile)***
    - **Events: CRUD sự kiện, trạng thái Draft/Published/Closed/Cancelled. *(Event, Tag, EventTag)***
    - **Registrations: tham gia/hủy, chờ (waitlist, tùy chọn), đếm chỗ. *(Registration)***
    - **Favorites: đánh dấu/bỏ yêu thích, đếm lượt thích. *(Favorite)***
    - **Forum: bài viết/bình luận/tags. *(Post, Comment, PostTag)***

    - **Moderation: báo cáo, duyệt/ẩn/xóa, nhật ký. *(Report, AuditLog)***
    - **Notifications: email/in-app, hàng đợi gửi. *(Notification)***

3.3 Logical / Component View



3.4 Deployment View



3.5 Data Architecture

- **CSDL: PostgreSQL (Prisma ORM); khóa chính UUID; ràng buộc unique cho (user_id,event_id) ở Registration/Favorite.**
- **Chỉ mục gợi ý: `events(start_at)`, `events(status)`, `registrations(event_id,user_id)`, `favorites(user_id,event_id)`, `posts(created_at)`, `comments(post_id,created_at)`.**
- **Giao dịch quan trọng: đăng ký sự kiện dùng `SELECT ... FOR UPDATE` + transaction để tránh overbooking.**
- **Media: chỉ lưu URL vào DB; nội dung thật ở CDN.**
- **Lưu vết: AuditLog cho hành động admin/organizer và thay đổi trạng thái.**

3.6 Integration

- **OAuth2: Google/Facebook (sign-in via code exchange).**
- **Email/Notify: SES/SendGrid (SMTP/API) + in-app; worker xử lý retry/backoff.**
- **Search: Meilisearch/Elastic (tùy chọn) cho full-text + facet; đồng bộ chỉ mục qua job.**
- **Maps/Geocoding: Mapbox/Google để chọn/hiển thị địa điểm.**
- **Rate-limit/Jobs: Redis (bucket/token), hàng đợi gửi mail, reindex.**

3.7 Security Architecture

- **Auth: JWT (HttpOnly cookie hoặc Bearer), refresh token rotation, logout/blacklist.**

- **RBAC: Guest/Participant/Organizer/Admin; chặn theo route/guard.**
- **AppSec: HTTPS, CORS, Helmet, input validation, chống XSS/SQLi/CSRF, captcha cho hành vi nhạy cảm.**
- **Privacy: tối thiểu PII, opt-in chia sẻ liên hệ; quyền tải/xóa dữ liệu.**
- **Audit: ghi log mọi hành động admin/duyệt; bất biến (append-only).**

3.8 Observability & Ops

- **Logs/Tracing: JSON logs, OpenTelemetry, Sentry/Datadog.**
- **Metrics/Alerts: 5xx rate, latency p95, lỗi OAuth/Email, queue backlog.**
- **Backup/DR: snapshot DB hằng ngày; RPO ≤ 24h, RTO ≤ 4h.**
- **CI/CD: build → test → lint → scan → deploy (blue/green/rolling); config qua env/secrets vault.**

3.9 Scalability & Performance

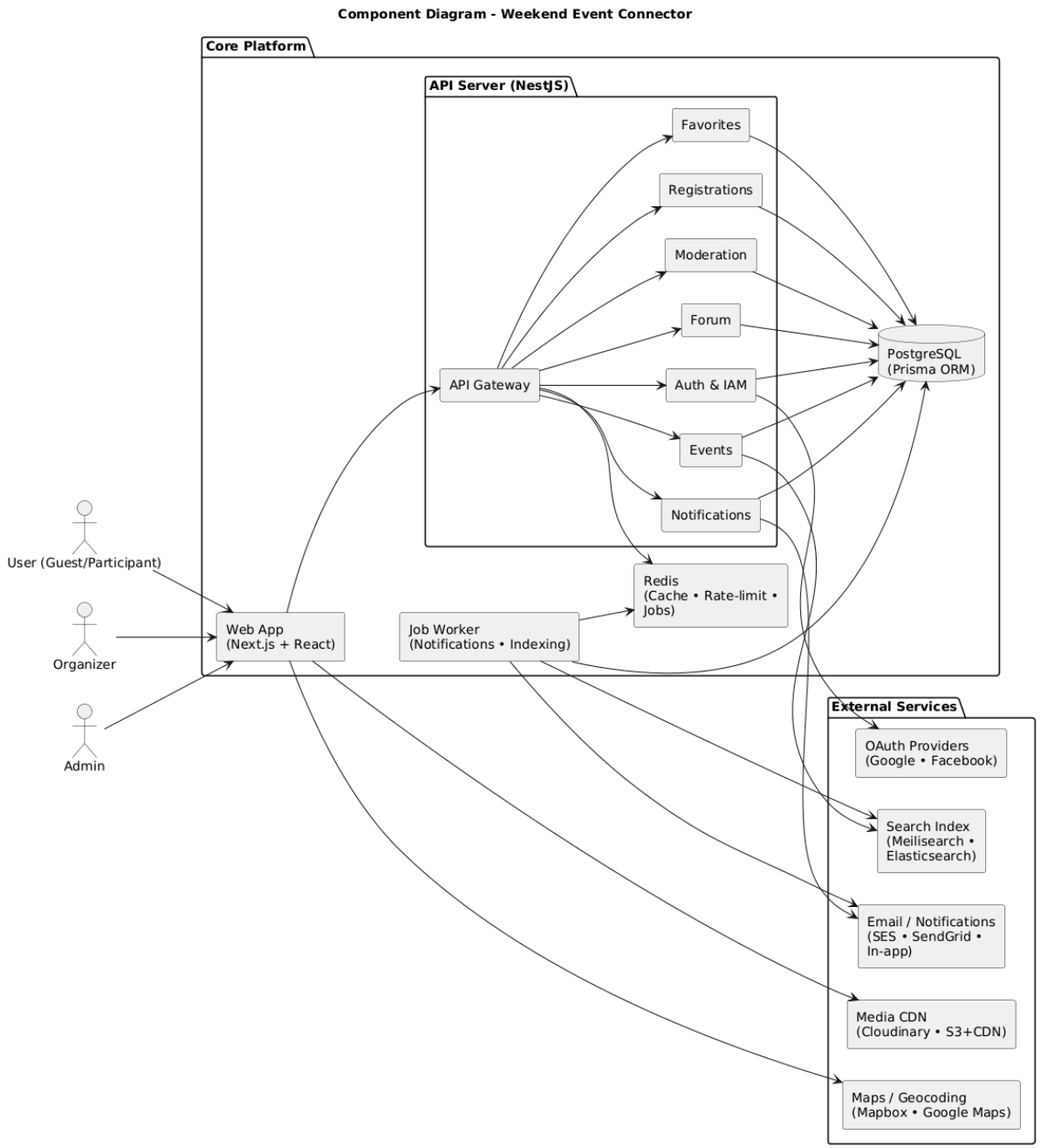
- **Scale ngang: API pods, worker pods; DB read-replica (nếu cần); Search cluster (tùy chọn).**
- **Hiệu năng: pagination, nén ảnh CDN, cache danh sách ngắn, tiền xử lý thống kê (dashboard).**
- **Graceful degradation: nếu Search/Maps/CDN lỗi → fallback DB/basic list; xếp hàng gửi mail.**

3.10 Rủi ro & Giảm thiểu

- **Overbooking: khoá hàng & transaction (đã nêu).**
- **Spam/abuse: rate-limit, captcha, moderation queue, auto-flag theo ngưỡng report.**
- **Vendor lock-in: trừu tượng adapter (Email/Search/Maps/CDN) để thay thế nhà cung cấp.**
- **Tải đột biến: autoscale pods, queue để "làm mượt" đỉnh tải, CDN cho media.**

IV.3 Component and Package Diagram

IV.3.1 Component Diagram



- Tác nhân:

User (Guest/Participant) duyệt/đăng ký/yêu thích/viết bài; **Organizer** tạo–quản lý sự kiện; **Admin** kiểm duyệt & quản lý người dùng.

- Lõi hệ thống (Core Platform):

- **Web App (Next.js + React):** giao diện người dùng, gọi API, tải ảnh lên CDN, hiển thị bản đồ.
- **API Server (NestJS)** gồm các **module**:
    - **API Gateway:** đầu mối nhận mọi request HTTP và phân tuyến vào module tương ứng.
    - **Auth & IAM:** đăng ký/đăng nhập (email + OAuth), JWT, RBAC, khóa/mở khóa tài khoản.

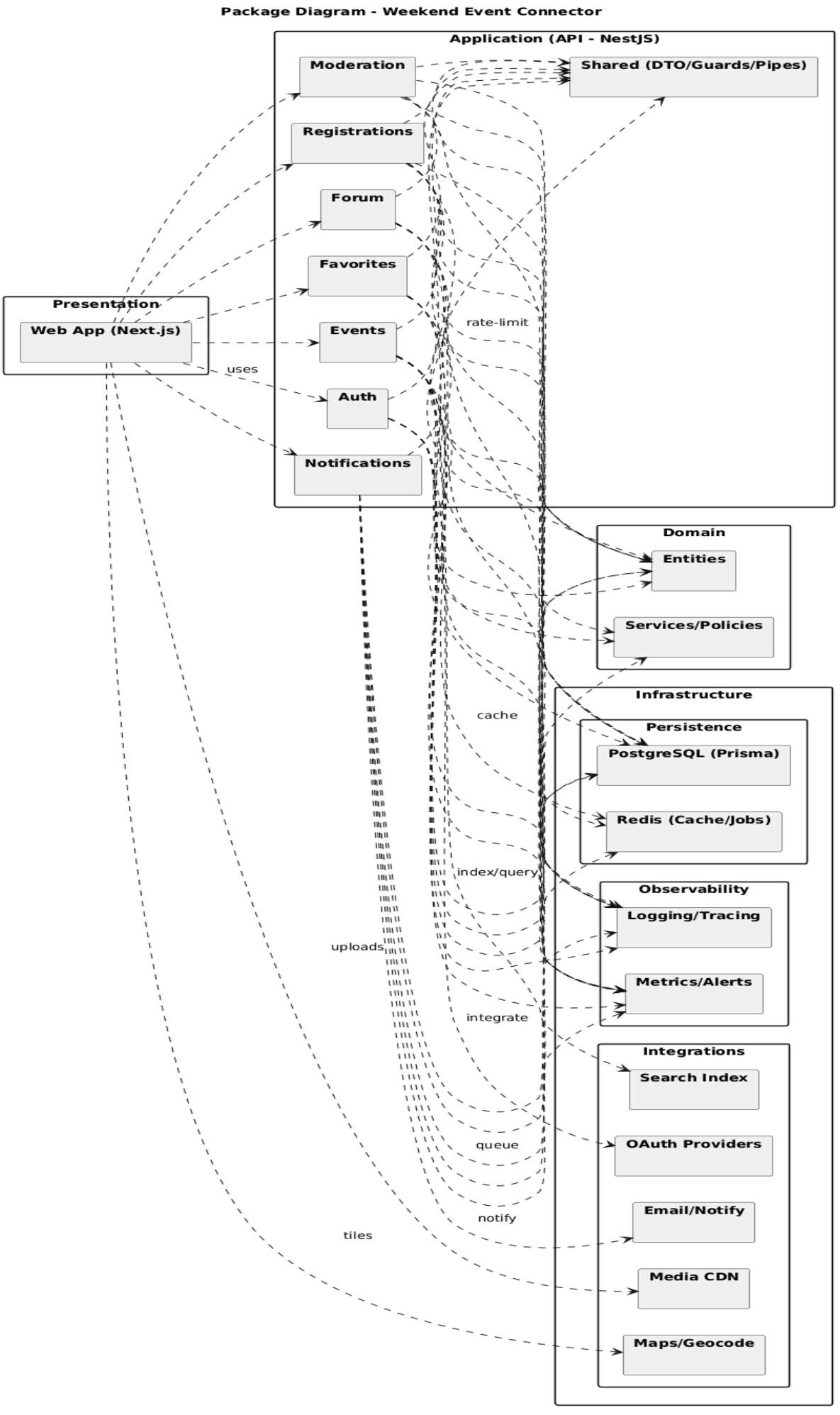
    - **Events:** CRUD sự kiện, trạng thái Draft/Published/Closed/Cancelled, gắn tag, đồng bộ search.
    - **Registrations:** tham gia/hủy, kiểm tra capacity/deadline, tránh overbooking (transaction).
    - **Favorites:** đánh dấu/bỏ yêu thích, đếm lượt yêu thích.
    - **Forum:** bài viết, bình luận, tag bài viết.
    - **Moderation:** báo cáo, duyệt/ẩn/xóa nội dung, ghi **audit log**.
    - **Notifications:** gửi email/in-app theo sự kiện hệ thống (đăng ký, đổi lịch, bình luận, kiểm duyệt).
- **PostgreSQL (Prisma):** nguồn dữ liệu chuẩn (users, events, registrations, favorites, posts, comments, reports, notifications, audit…).
- **Redis:** cache/rate-limit/queue jobs; lưu blacklist token, hàng đợi gửi email, đồng bộ search.
- **Job Worker:** xử lý nền (gửi thông báo, re-index search, tác vụ định kỳ).

- Dịch vụ ngoài (External Services):
OAuth (Google/Facebook) cho social login; **Email/Notify (SES/SendGrid + in-app)**; **Search Index (Meilisearch/Elastic)** cho tìm kiếm nhanh; **Media CDN (Cloudinary/S3)** lưu ảnh; **Maps/Geocoding (Mapbox/Google)** tra cứu/hiển thị địa điểm.

- Luồng chính:
User/Organizer/Admin → Web App → API Gateway → (module tương ứng) → DB/Redis → (khi cần) External Services.
Các tác vụ nặng/không đồng bộ (email, index, thông báo) được **đẩy sang Job Worker** qua Redis.

- Ý nghĩa kiến trúc:
Phân tách rõ **UI – API – Data/Infra**, module hóa nghiệp vụ, dễ mở rộng (scale ngang API/Worker), **an toàn** (OAuth/JWT/RBAC/audit), **nhanh** (cache, search, CDN) và **dễ vận hành** (tách tích hợp ngoài qua adapter).

IV.3.2 . Package Diagram

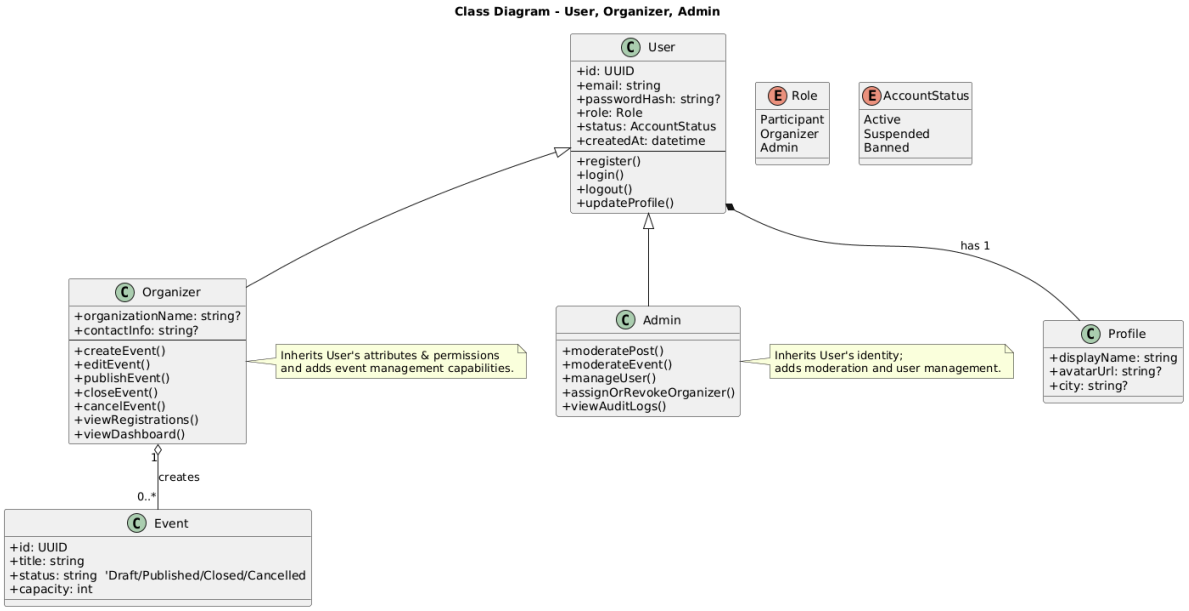

Package Diagram - Weekend Event Connector
Application (API - NestJS)
Moderation
Shared (DTO/Guards/Pipes)
Registrations
Forum
Favorites
Events
Auth
Notifications
Presentation
Web App (Next.js)
uses
rate-limit
Domain
Entities
Services/Policies
Infrastructure
cache
Persistence
PostgreSQL (Prisma)
Redis (Cache/Jobs)
index/query
Observability
Logging/Tracing
Metrics/Alerts
uploads
integrate
Integrations
Search Index
OAuth Providers
queue
Email/Notify
notify
tiles
Media CDN
Maps/Geocode

Package descriptions

No	Package	Description
01	**Tầng Presentation** – `Web App (Next.js)`	***Giao diện người dùng cho Guest/Participant/Organizer/Admin. Gọi REST tới các module API, upload ảnh trực tiếp lên Media CDN, lấy bản đồ/địa chỉ từ Maps.***
02	**Tầng Application (API – NestJS)**	Chứa các **module nghiệp vụ**: • `Auth`: đăng ký/đăng nhập (email + OAuth), JWT, RBAC. • `Events`: CRUD sự kiện, trạng thái (Draft/Published/Closed/Cancelled), đồng bộ Search. • `Registrations`: tham gia/hủy, kiểm tra capacity/deadline, chống overbooking. • `Favorites`: đánh dấu/bỏ yêu thích, đếm like. • `Forum`: bài viết & bình luận. • `Moderation`: báo cáo, duyệt/ẩn/xóa, audit log. • `Notifications`: email/in-app theo sự kiện. • `Shared`: DTO/guards/pipes dùng chung. ***Presentation chỉ "uses" các module này; không gọi trực tiếp tầng dưới.***
03	**Tầng Domain**	• `Entities`: mô hình dữ liệu/đối tượng miền (User, Event, Registration, …). • `Services/Policies`: quy tắc nghiệp vụ thuần (không phụ thuộc hạ tầng). ***Các module Application phụ thuộc Domain để thực thi logic.***
04	**Tầng Infrastructure**	• ***`Persistence`: PostgreSQL (Prisma) là nguồn dữ liệu chuẩn; Redis dùng cache, rate-limit, hàng đợi job.***

		• ***`Integrations`: OAuth (Google/Facebook), Email/Notify (SES/SendGrid), Search Index (Meilisearch/Elastic), Media CDN, Maps/Geocode.*** • ***`Observability`: Logging/Tracing và Metrics/Alerts cho giám sát. Module Application gọi tầng này để lưu dữ liệu, gửi email, tra cứu search, v.v.***

IV.4 Class diagram



Explantion:

Kế thừa & vai trò

- `User` là **lớp cơ sở** giữ danh tính và phiên: `id`, `email`, `passwordHash?`, `role`, `status`, `createdAt`.
- `Organizer` **kế thừa User** → có đầy đủ thuộc tính/ hành vi của User và **mở rộng** năng lực quản lý sự kiện: `createEvent()`, `editEvent()`, `publishEvent()`, `closeEvent()`, `cancelEvent()`, `viewRegistrations()`, `viewDashboard()`. Có thể có thêm `organizationName`, `contactInfo`.
- `Admin` **kế thừa User** → thêm hành vi **quản trị/kiểm duyệt**: `moderatePost()`, `moderateEvent()`, `manageUser()`, `assignOrRevokeOrganizer()`, `viewAuditLogs()`.

Liên kết

- `User` *— `Profile` (1–1, composition): mỗi tài khoản có đúng **một** hồ sơ hiển thị (`displayName`, `avatarUrl`, `city`...).

- `Organizer` o— `Event` (1–N, aggregation): một organizer **tạo nhiều** sự kiện; sự kiện tồn tại độc lập khi organizer còn tài khoản.

Enums & trạng thái

- `Role` (Participant, Organizer, Admin) giúp **RBAC** ở tầng API/UI.
- `AccountStatus` (Active, Suspended, Banned) điều khiển khả năng đăng nhập/ thao tác.

Hành vi cốt lõi ở User

- `register()`, `login()`, `logout()`, `updateProfile()`—mọi vai trò dùng chung; quyền nâng cao được phân biệt bằng `role`.

Lợi ích thiết kế

- Tái sử dụng logic xác thực/phiên ở `User`, chỉ **mở rộng** ở vai trò đặc thù (Organizer/Admin) → dễ bảo trì, phù hợp nguyên tắc OCP/LSP.
- Có thể ánh xạ sang CSDL theo **single-table + cột role** (đơn giản, hiệu quả) hoặc **table phụ** cho thuộc tính mở rộng của Organizer.

Ràng buộc nghiệp vụ chính

- Chỉ `Organizer` được thao tác CRUD sự kiện; chỉ `Admin` được kiểm duyệt & quản lý người dùng.
- `Suspended/Banned` chặn tất cả hành vi ngoài đăng xuất/kháng nghị.

IV.5 Database Design

1) Nguyên tắc & chuẩn đặt tên

- **CSDL:** PostgreSQL. Kiểu thời gian `timestamptz` (UTC). Khóa chính **UUID**.
- **Quy ước:** `snake_case` cho bảng/cột; timestamp: `created_at`, `updated_at`.
- **Bảo toàn dữ liệu:** FK dùng `ON DELETE CASCADE/RESTRICT` hợp lý. Chỉ mục trên mọi FK.
- **Bảo mật/riêng tư:** tối thiểu PII; không lưu mật khẩu thô (hash Argon2/bcrypt); ẩn liên hệ người tham gia trừ khi có consent.

2) Kiểu liệt kê (ENUM) & tiện ích

```sql
-- Tiện ích
CREATE EXTENSION IF NOT EXISTS pgcrypto;   -- gen_random_uuid()
CREATE EXTENSION IF NOT EXISTS citext;     -- case-insensitive email, tag

-- ENUMs
DO $$ BEGIN
  CREATE TYPE role_enum           AS ENUM ('participant','organizer','admin');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;

DO $$ BEGIN
  CREATE TYPE account_status_enum AS ENUM ('active','suspended','banned');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;

DO $$ BEGIN
  CREATE TYPE event_status_enum   AS ENUM ('draft','published','closed','cancelled');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;

DO $$ BEGIN
  CREATE TYPE reg_status_enum     AS ENUM ('registered','cancelled');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;

DO $$ BEGIN
  CREATE TYPE visibility_enum     AS ENUM ('visible','hidden');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;

DO $$ BEGIN
  CREATE TYPE report_status_enum  AS ENUM ('open','resolved','rejected');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;
```

3) Lược đồ bảng (DDL cốt lõi)

3.1 Người dùng & hồ sơ

```sql
CREATE TABLE users (
  id             UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  email          CITEXT NOT NULL UNIQUE,
  password_hash  TEXT,                                  -- NULL nếu login xã hội
  provider       TEXT NOT NULL DEFAULT 'local',         -- local/google/facebook
  provider_id    TEXT,                                  -- id từ OAuth
  role           role_enum NOT NULL DEFAULT 'participant',
  status         account_status_enum NOT NULL DEFAULT 'active',
  created_at     TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at     TIMESTAMPTZ NOT NULL DEFAULT now(),
  CONSTRAINT chk_local_pwd CHECK (
    (provider <> 'local') OR (password_hash IS NOT NULL)
  )
);

-- unique (provider, provider_id) cho tài khoản OAuth (không áp cho local)
CREATE UNIQUE INDEX ux_users_provider ON users(provider, provider_id)
WHERE provider <> 'local' AND provider_id IS NOT NULL;

CREATE TABLE profiles (
  user_id      UUID PRIMARY KEY REFERENCES users(id) ON DELETE CASCADE,
  display_name TEXT NOT NULL,
  avatar_url   TEXT,
  city         TEXT,
  bio          TEXT,
  interests    JSONB,
  updated_at   TIMESTAMPTZ NOT NULL DEFAULT now()
);
```

3.2 Sự kiện & gắn thẻ

```sql
CREATE TABLE events (
  id               UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  organizer_id     UUID NOT NULL REFERENCES users(id) ON DELETE RESTRICT,
  title            TEXT NOT NULL,
  slug             TEXT UNIQUE,                          -- tùy chọn SEO
  description      TEXT NOT NULL,
  location_text    TEXT NOT NULL,
  lat              NUMERIC(9,6),                         -- tùy chọn
  lng              NUMERIC(9,6),
  start_at         TIMESTAMPTZ NOT NULL,
  end_at           TIMESTAMPTZ NOT NULL,
  price            NUMERIC(12,2),
  capacity         INT NOT NULL DEFAULT 0 CHECK (capacity >= 0),
  status           event_status_enum NOT NULL DEFAULT 'draft',
  favorites_count  INT NOT NULL DEFAULT 0 CHECK (favorites_count >= 0),
  registered_count INT NOT NULL DEFAULT 0 CHECK (registered_count >= 0),
  created_at       TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at       TIMESTAMPTZ NOT NULL DEFAULT now(),
  CONSTRAINT chk_time_order CHECK (start_at < end_at)
);

CREATE INDEX ix_events_status_time ON events(status, start_at);
CREATE INDEX ix_events_geo ON events USING BTREE (lat, lng);

CREATE TABLE tags (
  id    UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name  CITEXT NOT NULL UNIQUE
);

CREATE TABLE event_tags (
  event_id UUID NOT NULL REFERENCES events(id) ON DELETE CASCADE,
  tag_id   UUID NOT NULL REFERENCES tags(id) ON DELETE CASCADE,
  PRIMARY KEY (event_id, tag_id)
);
```

3.3 Đăng ký & yêu thích

```sql
CREATE TABLE registrations (
  id          UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  event_id    UUID NOT NULL REFERENCES events(id) ON DELETE CASCADE,
  user_id     UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  status      reg_status_enum NOT NULL DEFAULT 'registered',
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  UNIQUE (event_id, user_id)
);

CREATE INDEX ix_registrations_user ON registrations(user_id, status);
CREATE INDEX ix_registrations_event ON registrations(event_id, status);

CREATE TABLE favorites (
  user_id     UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  event_id    UUID NOT NULL REFERENCES events(id) ON DELETE CASCADE,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  PRIMARY KEY (user_id, event_id)
);
```

3.4 Diễn đàn, bình luận, báo cáo

```sql
CREATE TABLE posts (
  id         UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  author_id  UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  title      TEXT NOT NULL,
  content    TEXT NOT NULL,
  status     visibility_enum NOT NULL DEFAULT 'visible',
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE INDEX ix_posts_status_created ON posts(status, created_at DESC);

CREATE TABLE post_tags (
  post_id UUID NOT NULL REFERENCES posts(id) ON DELETE CASCADE,
  tag_id  UUID NOT NULL REFERENCES tags(id) ON DELETE CASCADE,
  PRIMARY KEY (post_id, tag_id)
);

CREATE TABLE comments (
  id         UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  post_id    UUID NOT NULL REFERENCES posts(id) ON DELETE CASCADE,
  author_id  UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  content    TEXT NOT NULL,
  status     visibility_enum NOT NULL DEFAULT 'visible',
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX ix_comments_post ON comments(post_id, created_at);

CREATE TABLE reports (
  id                UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  reporter_id       UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  target_event_id   UUID REFERENCES events(id)   ON DELETE CASCADE,
  target_post_id    UUID REFERENCES posts(id)    ON DELETE CASCADE,
  target_comment_id UUID REFERENCES comments(id) ON DELETE CASCADE,
  reason            TEXT NOT NULL,
  status            report_status_enum NOT NULL DEFAULT 'open',
  note              TEXT,
  created_at        TIMESTAMPTZ NOT NULL DEFAULT now(),
  CONSTRAINT chk_one_target
    CHECK (
      (target_event_id IS NOT NULL)::int +
      (target_post_id IS NOT NULL)::int +
      (target_comment_id IS NOT NULL)::int = 1
    )
);

CREATE INDEX ix_reports_status ON reports(status, created_at DESC);
```

3.5 Thông báo & audit

```sql
CREATE TABLE notifications (
  id         UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id    UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  type       TEXT NOT NULL,
  payload    JSONB,
  read_at    TIMESTAMPTZ,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX ix_notifications_user ON notifications(user_id, read_at);

CREATE TABLE audit_logs (
  id          UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  actor_id    UUID REFERENCES users(id) ON DELETE SET NULL,
  action      TEXT NOT NULL,
  entity_type TEXT NOT NULL,
  entity_id   UUID,
  before      JSONB,
  after       JSONB,
  created_at  TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX ix_audit_entity ON audit_logs(entity_type, entity_id, created_at DESC);
```

Danh sách chờ (Waitlist)

```sql
CREATE TABLE event_waitlist (
  id         UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  event_id   UUID NOT NULL REFERENCES events(id) ON DELETE CASCADE,
  user_id    UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  UNIQUE (event_id, user_id)
);
CREATE INDEX ix_waitlist_event ON event_waitlist(event_id, created_at);
```

4) Chỉ mục & tối ưu hóa (gợi ý)

- **Tra cứu sự kiện: `ix_events_status_time (status, start_at)`; thêm `start_at DESC` cho feed sắp tới.**
- **FK: index trên mọi FK: `*_id` ở registrations, favorites, posts, comments, reports.**
- **Tìm kiếm fallback: có thể thêm GIN trigram (`pg_trgm`) trên `events.title`, `events.description` (nếu không dùng Search Index).**

- **Phân trang: mọi danh sách có `ORDER BY created_at DESC` + index phù hợp.**
- **Đếm nhanh: `registered_count`, `favorites_count` lưu trong bảng `events` (có thể cập nhật qua trigger hoặc worker) để tránh COUNT nặng.**

5) Ràng buộc nghiệp vụ chính (DB-level)

- **Đăng ký: `UNIQUE (event_id, user_id)`; kiểm tra capacity/deadline ở API + transaction `SELECT ... FOR UPDATE` trên `events`.**
- **Thời gian sự kiện: `start_at < end_at`.**
- **Báo cáo vi phạm: `chk_one_target` đảm bảo chỉ report một loại mục tiêu.**
- **Quyền: RBAC ở tầng API; DB không mã hóa quyền nhưng FK/DELETE bảo toàn toàn vẹn.**

6) Chính sách dữ liệu & vận hành

- **Lưu giữ: `audit_logs` & `reports` ≥ 12 tháng (theo policy); có job dọn dẹp cũ.**
- **Backup: snapshot hằng ngày, RPO ≤ 24h, RTO ≤ 4h.**
- **PII: cho phép người dùng tải/xóa dữ liệu (xóa cascade profile, ẩn/giấu nội dung đã duyệt theo policy).**

V. Implementation

V.1 Tổng quan kiến trúc đã chọn

FREEDAY áp dụng kiến trúc **Layered + Modular Monolith** (module hóa theo miền nghiệp vụ) với khả năng tách dần thành microservices khi tải tăng. Kiến trúc nhấn mạnh bảo mật (OAuth2/JWT, RBAC, audit), hiệu năng (SSR/SSG, cache, hàng đợi), và khả năng mở rộng theo chiều ngang.

Phong cách kiến trúc

- **Layered Architecture + Modular Monolith**: tách rõ **Presentation** → **Application** → **Domain** → **Infrastructure**, các chức năng được đóng gói thành module (Auth, Events, Registrations, Favorites, Forum, Moderation, Notifications).
- **Worker nền**: xử lý tác vụ không đồng bộ (gửi email/thông báo, re-index search, tính toán chỉ số).

Các lớp và thành phần chính

- **Presentation (Web App):**
  - **Next.js + React, Tailwind** (SSR/SSG để SEO tốt cho danh sách sự kiện).

    - Chức năng: duyệt/tìm kiếm/lọc sự kiện, đăng ký/đăng nhập (Email/Google/Facebook), quản lý hồ sơ, yêu thích, “Sự kiện của tôi”, diễn đàn, bảng điều khiển của Organizer, công cụ kiểm duyệt của Admin.
- **Application (API – NestJS, REST):**
    - Module **Auth & IAM** (OAuth2, JWT, RBAC), **Events** (CRUD + trạng thái Draft/Published/Closed/Cancelled), **Registrations** (tham gia/hủy, đảm bảo capacity & deadline), **Favorites**, **Forum** (post/comment), **Moderation** (báo cáo, duyệt/ẩn/xóa, audit), **Notifications** (email/in-app).
    - Bảo vệ endpoint bằng Guards/Policies, rate-limit, validation.
- **Domain (Entities/Services):**
    - Mô tả quy tắc nghiệp vụ thuần (không phụ thuộc hạ tầng): kiểm soát trạng thái sự kiện, logic chống overbooking, chính sách chỉnh sửa/cancel, ngưỡng auto-flag.
- **Infrastructure:**
    - **PostgreSQL + Prisma ORM** (nguồn dữ liệu chuẩn, ACID).
    - **Redis** (cache, rate-limit, hàng đợi job).
    - **Tích hợp ngoài: OAuth** (Google/Facebook), **Email/Notify** (SES/SendGrid + in-app), **Search Index** (Meilisearch/Elastic – tùy chọn), **Media CDN** (Cloudinary/S3), **Maps/Geocoding** (Mapbox/Google).
    - **Observability:** Logging/Tracing (Sentry/Datadog/OpenTelemetry), Metrics/Alerts.

Vì sao lựa chọn kiến trúc này

- **Bảo trì & phát triển nhanh:**
  Phân lớp rõ ràng + module hóa theo nghiệp vụ giúp đội ngũ phát triển thay đổi độc lập (ví dụ chỉnh UI không ảnh hưởng Domain/DB). Đường biên module rõ (Auth, Events, Forum…) rút ngắn vòng đời tính năng.
- **Phù hợp Web-first & SEO:**
  Next.js (SSR/SSG) tối ưu lập chỉ mục sự kiện, tải nhanh, thân thiện chia sẻ liên kết; thích hợp cho nền tảng khám phá sự kiện.
- **Khả năng mở rộng & chi phí hợp lý:**
  Modular Monolith đơn giản để vận hành ở giai đoạn đầu, nhưng vẫn dễ “bóc tách” thành service riêng (Notifications/Search) khi tải tăng. **Redis + worker** làm mượt tải đỉnh (email, index, thống kê).
- **An toàn & tuân thủ:**
  OAuth2/JWT, RBAC, audit log; dữ liệu cá nhân tối thiểu, chia sẻ liên hệ theo **opt-in**; ràng buộc cấp DB (FK/unique) + giao dịch đảm bảo **đăng ký không vượt capacity**.
- **Hiệu năng & độ tin cậy:**
  Cache/pagination, CDN ảnh, search index (tùy chọn) cho truy vấn nhanh; **fallback** về truy vấn DB khi Search/Maps/CDN gặp sự cố (degraded mode).
- **Dễ kiểm thử & CI/CD:**
  Phân tách UI–API–Domain–Infra giúp viết unit/integration test hiệu quả; pipeline CI/CD (build, test, lint, scan, deploy) áp dụng thuận lợi..

V.2. Mapping to Project Structure

Below is the **actual folder/package structure** for the **FREEDAY** project, mapped to the architecture:

```
/freeday
├─ README.md
├─ .env.example                    # Mẫu biến môi trường
├─ docker-compose.yml              # Postgres, Redis, Mailhog (dev)
├─ infra/                          # Hạ tầng & CI/CD
│  ├─ k8s/                         # Manifests triển khai (prod/staging)
│  └─ github-actions/              # Pipeline CI (build/test/lint)
│
├─ apps/
│  ├─ web/                         # Presentation Layer (Next.js)
│  │  ├─ next.config.js
│  │  ├─ public/                   # Ảnh tĩnh, icons
│  │  └─ src/
│  │     ├─ app/                   # App Router (SSR/SSG)
│  │     │  ├─ (public)/events/    # Danh sách / chi tiết sự kiện
│  │     │  ├─ (auth)/login/ signup/ # Đăng nhập/đăng ký (email + OAuth)
│  │     │  ├─ (me)/profile/       # Hồ sơ, Sự kiện của tôi
│  │     │  ├─ organizer/          # Dashboard quản lý sự kiện
│  │     │  └─ admin/              # Moderation & user management
│  │     ├─ components/            # UI tái sử dụng
│  │     ├─ features/              # Theo tính năng (events, forum, ...)
│  │     ├─ hooks/                 # Hooks dùng chung
│  │     ├─ lib/                   # Fetcher, auth client, utils
│  │     ├─ services/              # Gọi API (REST) phía client
│  │     ├─ store/                 # Zustand/Context
│  │     ├─ styles/                # Tailwind/SCSS
│  │     └─ middleware.ts          # Bảo vệ route (auth)
│  │
│  ├─ api/                         # Application + Domain + Infrastructure (NestJS)
│  │  ├─ nest-cli.json  tsconfig.json
│  │  ├─ src/
│  │  │  ├─ main.ts                # Bootstrap Nest app
│  │  │  ├─ app.module.ts
│  │  │  ├─ config/                # env, cors, rate-limit, helmet
│  │  │  ├─ common/                # guards, interceptors, pipes, dto-base
│  │  │  ├─ database/              # PrismaService, seed
│  │  │  ├─ integrations/          # email, oauth, search, storage, maps
│  │  │  ├─ jobs/                  # Worker in-process (BullMQ/Queues)
│  │  │  ├─ modules/               # Modules nghiệp vụ
│  │  │  │  ├─ auth/               # OAuth2, JWT, RBAC (User, Profile)
│  │  │  │  ├─ users/
│  │  │  │  ├─ events/             # CRUD + FSM Draft/Published/Closed/Cancelled
│  │  │  │  ├─ registrations/      # Tham gia/hủy, chống overbooking (txn)
│  │  │  │  ├─ favorites/          # Toggle yêu thích
│  │  │  │  ├─ forum/              # posts, comments, tags
│  │  │  │  ├─ moderation/         # reports, actions, audit-log
│  │  │  │  └─ notifications/      # gửi email/in-app, template
│  │  │  └─ domain/                # Entities/Policies/Services thuần miền
│  │  └─ prisma/
│  │     ├─ schema.prisma          # Mô hình dữ liệu (PostgreSQL)
│  │     └─ migrations/            # Migration do Prisma sinh
│  │
│  └─ worker/                      # Background Worker (tách process, tùy chọn)
│     ├─ src/
│     │  ├─ main.ts                # Consumer queue (email, re-index, thống kê)
│     │  └─ queues/                # Định nghĩa job
│     └─ tsconfig.json
│
├─ packages/
│  └─ shared/                      # Types/DTO chung, schema validation, constants
│     ├─ dto/                      # Hợp đồng REST dùng chung client/server
│     ├─ types/
│     └─ utils/
│
└─ tests/
   ├─ api/                         # Integration tests (NestJS e2e)
   └─ web/                         # Playwright/Cypress e2e cho Web
```

Mapping to Architecture:

1. Presentation Layer (Web App)

Path: `/freeday/apps/web/src`

Bên trong: `app/` (events, me, organizer, admin), `components/`, `features/`, `services/`, `store/`, `lib/`.

Nhiệm vụ: hiển thị danh sách/chi tiết sự kiện, đăng ký–yêu thích, diễn đàn; dashboard Organizer; công cụ Admin. Gọi API backend (REST) cho đăng nhập (Email/Google/Facebook), CRUD sự kiện, đăng bài/bình luận…

Ánh xạ kiến trúc: ***Presentation Layer***.

2. Application Layer (Backend API)

Path: `/freeday/apps/api/src/modules/`

Bên trong: `auth/`, `events/`, `registrations/`, `favorites/`, `forum/`, `moderation/`, `notifications/`; dùng chung tại `common/` (guards/DTO/pipes) và `config/`.

Nhiệm vụ: xử lý nghiệp vụ cốt lõi, RBAC, kiểm tra capacity/deadline, điều phối Domain & Infrastructure, trả response.

Ánh xạ kiến trúc: ***Application/Business Logic Layer***.

3. Domain Layer

Path: `/freeday/apps/api/src/domain/`

Nhiệm vụ: Entities/Policies/Services thuần miền (FSM sự kiện, chính sách hủy/sửa, auto-flag).

Ánh xạ kiến trúc: ***Domain Layer***.

4. Data Access & Infrastructure Layer

Paths:

- **Database/ORM:** `/freeday/apps/api/prisma/` (schema.prisma, migrations), `/freeday/apps/api/src/database/`
- **Cache/Queue:** `/freeday/apps/api/src/jobs/` (hoặc tách process ở `apps/worker/`)

Nhiệm vụ: PostgreSQL + Prisma (Users, Events, Registrations, Favorites,

Posts, Comments, Reports, Notifications, AuditLogs, Tags); Redis cho cache/rate-limit/queue.

Ánh xạ kiến trúc: *Data Access & Infrastructure*.

5. External Services Layer

Path: `/freeday/apps/api/src/integrations/` (`oauth/`, `email/`, `search/`, `storage/`, `maps/`).

Nhiệm vụ: tích hợp Google/Facebook OAuth, Email/Notify (SES/SendGrid), Search (Meilisearch/Elastic – tùy chọn), Media CDN (Cloudinary/S3), Maps/Geocoding (Mapbox/Google).

Ánh xạ kiến trúc: *External Services Layer*.

6. Background Worker (Asynchronous)

Path: `/freeday/apps/worker/`

Nhiệm vụ: gửi email/thông báo, re-index search, job định kỳ; giúp không chặn request đồng bộ.
