Object Design

---

1. User Management

• User

o Attributes:

- **userId:** ObjectId (unique identifier).
- **role:** String (student, organizer, sponsor, admin).
- **name:** Full name.
- **email:** Unique; validated against college domain.
- **profilePic:** URL (stored in Cloudinary).
- **linkedin:** Optional professional link.
- **github:** Optional portfolio link.
- **address:** Residential address.
- **dob:** Date of birth.
- **areasOfInterest:** List of interests for recommendations.
- **roleTag:** Tag indicating team role/responsibility.
- **resumeUrl:** URL (resume stored in Cloudinary).
- **achievements:** List<Achievement> (with proof).
- **status:** active / suspended.
- **college:** Reference to College collection.

o Methods:

- **register():** Creates a new user with validation.

- **login():** Authenticates user via JWT.
- **editProfile():** Updates profile fields.
- **uploadResume():** Uploads resume to Cloudinary.
- **addAchievement():** Adds new achievement entry.
- **changeRole():** Updates user role (student ↔ organizer ↔ sponsor).

• AuthenticationService

o Methods:

- **validateEmail(email):** Ensures email belongs to a registered college domain.
- **encryptPassword(password):** Hashes password using bcrypt.
- **generateJWT(user):** Generates a secure JWT access token.
- **verifyJWT(token):** Verifies token and extracts user info + role.
- **sendVerificationEmail():** Sends verification email (optional).
- **handleForgotPassword():** Sends password reset link.

• AuthorizationService

o Methods:

- **authorizeRoles(roles[]):** Grants access only to specific roles.
- **checkSuspension(user):** Restricts actions by suspended users.
- **verifyEventAccess(user, event):** Ensures only authorized organizers can edit/publish events.
- **verifyTeamAccess(user, team):** Allows team management only for authorized organizers.

• College

o Attributes:

- **name:** College name.
- **code:** Unique short identifier (e.g., NIT-SRT).
- **logo:** Cloudinary URL.
- **poc:** Point of contact (name & contact).
- **address:** Full structured address.
- **status:** Pending / Approved / Rejected.
- **approvedBy:** Reference to admin user.

o Methods:

- **approveCollege(admin):** Approves a college application.
- **rejectCollege(admin):** Rejects a college application.
- **suspendEntities():** Suspends all users/events belonging to the college.

2. Event Management

• Event

o Attributes:

- **eventId:** Unique event identifier.
- **title:** Event name.
- **description:** Detailed description.
- **categoryTags:** List of category labels (technical, cultural, workshop, etc.).
- **college:** Reference to the hosting college.
- **ruleBook:** URL of rulebook stored in Cloudinary.

- **posterUrl:** Event poster URL.
- **venue:** Venue name.
- **location:** Location object (coordinates + annotations).
- **timeline:** List of timeline items (rounds/sessions).
- **subEvents:** List of linked sub-events.
- **gallery:** List of event image URLs.
- **config:** EventConfig object for registration settings.
- **registrations:** List of registration IDs.
- **sponsors:** List of sponsor references.
- **announcements:** List of announcements.
- **ratings:** List of ratings and reviews.
- **chatRoom:** List of chat messages.
- **winners:** List of winner entries.
- **scoreboard:** List of competition scores.
- **status:** draft / published / completed / suspended.
- **createdBy:** Organizer team reference.
- **createdAt:** Timestamp.
- **updatedAt:** Timestamp.

o Methods:

- **saveDraft():** Save minimal details as draft.
- **publish():** Make event public.
- **complete():** Mark event as completed.
- **addSubEvent(eventId):** Add sub-event reference.
- **addSponsor(userId):** Attach sponsor.
- **addAnnouncement():** Add event announcement.

- **addGalleryImage(url):** Append gallery image.
- **getRegistrations():** Fetch registered participants.
- **updateTimeline():** Modify timeline items.
- **addRatingReview():** Add rating and review.
- **recordCheckIn():** Mark attendance.
- **suspendEvent():** Admin suspension action.

• TimelineItem

o Attributes:

- **title:** Name of the stage/round.
- **description:** Details of the stage.
- **date:** Scheduled date.
- **duration.from:** Start time.
- **duration.to:** End time.
- **venue:** Venue for this stage.
- **checkInRequired:** Whether check-in is needed.

o Methods:

- **isOngoing():** Check if time is in progress.
- **requiresCheckIn():** Return check-in requirement.
- **validateDuration():** Ensure valid timing.

• EventConfig

o Attributes:

- **fees:** Registration fee.
- **qrCodeUrl:** Payment QR code.
- **registrationType:** Individual / Team.
- **teamSizeRange:** Minimum and maximum team size.
- **registrationFields:** Dynamic registration form fields.

o Methods:

- **addField():** Add a custom field.
- **removeField():** Delete a field.
- **validateTeamSize():** Validate team size.
- **isPaidEvent():** Check if event has fees.

• Location

o Attributes:

- **address:** Venue address.
- **coordinates:** Latitude and longitude.
- **mapAnnotations:** List of annotations.

o Methods:

- **addAnnotation():** Add marker.
- **editAnnotation():** Edit marker.
- **removeAnnotation():** Remove marker.

• MapAnnotation

o Attributes:

- **label:** Marker label.
- **description:** Details.
- **coordinates:** Lat/long.
- **icon:** Marker icon.
- **color:** Marker color.

• Announcement

o Attributes:

- **announcementId:** Unique ID.
- **date:** Date of creation.
- **time:** Time of creation.
- **author:** Organizer reference.
- **message:** Announcement message.

• RatingReview

o Attributes:

- **by:** User reference.
- **rating:** Numeric rating (1–5).
- **review:** Text review.
- **createdAt:** Timestamp.

• ChatMessage

o Attributes:

- **sender:** User reference.
- **message:** Text message.
- **createdAt:** Timestamp.

• WinnerInfo

o Attributes:

- **name:** Winner name.
- **team:** Team members (if team event).
- **proof:** Proof or certificate URL.

• ScoreboardEntry

o Attributes:

- **participant:** Name/team.
- **score:** Numeric score.

• SubEvent

o Attributes:

- **subEventId:** Reference to event.
- **status:** Pending / Approved / Rejected.

• EventService

o Methods:

- **createEvent():** Create new event.
- **updateEvent():** Update details.
- **fetchEventById():** Get event by ID.
- **fetchAllEvents():** List all events.
- **fetchPublishedEvents():** Get published events.
- **fetchCompletedEvents():** Get completed events.
- **addRegistration():** Add participant/team.
- **addCheckIn():** Record attendance.
- **getAnalytics():** Fetch event analytics.
- **rebuildVectorIndex():** Sync event with recommendation DB.

3. ORGANIZER TEAM MANAGEMENT

• OrganizerTeam

o Attributes:

- **teamId:** Unique identifier.
- **teamName:** Unique team name.
- **description:** Summary or purpose.
- **leader:** Reference to User (team leader).
- **members:** List of OrganizerTeamMember entries.
- **createdAt:** Timestamp.
- **updatedAt:** Timestamp.

o Methods:

- **addMember(userId, role):** Add member with specific role.

- **removeMember(userId):** Remove a member.
- **updateRole(userId, role):** Change member role.
- **inviteMember(userId):** Send team invitation.
- **approveInvite(userId):** Approve join request.
- **rejectInvite(userId):** Reject invitation.
- **getTeamMembers():** Return member list.

• OrganizerTeamMember

o Attributes:

- **member:** User reference.
- **role:** Co-Organizer / Volunteer.
- **status:** Pending / Approved / Reject.

o Methods:

- **approve():** Mark as approved.
- **reject():** Mark as rejected.
- **updateRole(newRole):** Update member role.

• EventTeamManager

o Methods:

- **createTeam(teamData):** Create a new organizer team.
- **deleteTeam(teamId):** Remove organizer team.
- **getTeamDetails(teamId):** Get team info.
- **inviteMember(teamId, userId):** Create invitation.
- **respondToInvitation(inviteId, action):** Approve/reject.

- **validateTeamName(name):** Ensure unique team name.
- **getUserTeams(userId):** Teams a user belongs to.
- **authorizeEventAction(userId, eventId):** Check if user can edit/publish event.

• EventPublisher

o Methods:

- **publishEvent(eventId, user):** Publish after permission check.
- **saveDraft(eventData):** Save event as draft.
- **updateEvent(eventId, updates, user):** Apply updates.
- **completeEvent(eventId, user):** Mark as completed.
- **getEventStatus(eventId):** Return lifecycle stage.
- **hasPermission(userId, eventId):** Check user permissions.

• OrganizerDashboard

o Attributes:

- **organizerId:** User reference.
- **events:** Draft, published, completed events.
- **registrations:** Participant lists.
- **checkIns:** Attendance logs.
- **ratings:** Feedback entries.
- **teams:** Organizer's teams.

o Methods:

- **getDraftedEvents():** Fetch draft events.
- **getPublishedEvents():** Fetch live events.

- **getCompletedEvents():** Fetch finished events.
- **getRegistrationLogs(eventId):** List registrations.
- **getCheckIns(eventId):** Show attendance.
- **getEventRatings(eventId):** Ratings and reviews.
- **exportRegistrations(eventId):** Export CSV.
- **getOrganizerTeams(userId):** List teams of organizer.

• AnnouncementManager

o Methods:

- **addAnnouncement(eventId, data):** Create announcement.
- **editAnnouncement(eventId, announcementId, data):** Edit message.
- **deleteAnnouncement(eventId, announcementId):** Delete announcement.
- **broadcastToInbox(eventId):** Send to participant inbox.

• CheckInManager

o Methods:

- **generateCheckInCode(registrationId):** Create unique code.
- **validateCheckIn(eventId, studentId, code):** Validate authenticity.
- **addCheckIn(eventId, timelineId, studentId):** Record attendance.
- **getAttendees(eventId):** Return present students.
- **getAbsentees(eventId):** Return missing students.

• Organizer Subsystem — Interaction Summary

o Interactions:

- **Event Management:** Organizers create/edit/publish events.
- **Student Registration:** View participant lists.
- **Inbox System:** Invites + announcements sent to inbox.
- **Analytics Service:** Fetch event statistics.
- **AI Recommender:** Publishing event updates vector DB.
- **Admin System:** Handles suspensions and permissions.

4. Student Team & Student Dashboard Management

• StudentTeam

o Attributes:

- **teamId:** Unique team identifier.
- **teamName:** Globally unique name.
- **createdBy:** User who created the team.
- **leader:** Team leader (User reference).
- **members:** List of StudentTeamMember objects.
- **createdAt:** Timestamp.
- **updatedAt:** Timestamp.

o Methods:

- **createTeam():** Create student team.
- **addMember(userId):** Add member (via invitation).
- **removeMember(userId):** Remove team member.

- **inviteMember(userId):** Send invitation to inbox.
- **approveInvite(userId):** Accept invitation.
- **rejectInvite(userId):** Reject invitation.
- **updateTeamName(newName):** Change team name.
- **assignLeader(userId):** Change team leader.
- **getTeamMembers():** List team members.

• StudentTeamMember

o Attributes:

- **member:** User reference.
- **status:** Pending / Approved / Reject.

o Methods:

- **approve():** Mark invitation as approved.
- **reject():** Mark as rejected.

• StudentDashboard

o Attributes:

- **studentId:** Logged-in student ID.
- **registeredEvents:** Ongoing events registered by student.
- **completedEvents:** Finished events.
- **recommendedEvents:** ML-based suggestions.
- **timelineReminders:** Events within the next 7 days.
- **clashWarnings:** Schedule conflict alerts.
- **studentTeams:** Teams created/joined by student.

o Methods:

- **getRegisteredEvents():** Fetch list of registered events.
- **getCompletedEvents():** Fetch completed event history.
- **getRecommendedEvents():** Fetch ML suggestions.
- **computeTimelineReminders():** Find upcoming events (7-day window).
- **detectClashes():** Identify timeline conflicts.
- **getStudentTeams():** Get all student teams.
- **addToRegistered(eventId):** Called after registration.
- **updateDashboard():** Refresh all dashboard states.

• RecommendationHandler

o Methods:

- **sendEmbeddingRequest():** Prepare embeddings for student/event.
- **queryRecommendations(studentId):** Get recommended events from Python ML backend.
- **rebuildIndex():** Rebuild Qdrant vector index.
- **addEventToVectorDB(event):** Add event upon publishing.
- **deleteEventFromVectorDB(eventId):** Remove deleted event.
- **hybridRanking():** Combine collaborative + content + demographic filtering.

• TimelineReminderEngine

o Methods:

- **getUpcomingEvents(studentId):** Events in next 7 days.
- **sendReminders():** Send reminder alerts (future extension).

- **filterByDateRange(events):** Filter events by date.

• ClashDetectionEngine

o Methods:

- **getEventTimelines(events):** Retrieve timelines.
- **detectClashes(timelineList):** Find overlapping schedules.
- **generateWarning():** Provide user-friendly conflict messages.

• Student Subsystem — Interaction Summary

o Interacts With:

- **Event Management:** View/register for events.
- **Recommendation Engine:** ML-based suggestions.
- **Registration Management:** Participate in events.
- **Inbox System:** Invitations & notifications.
- **Team System:** Manage student teams.
- **Analytics:** View event feedback.
- **Organizer System:** Receive announcements.
- **Check-In System:** Mark attendance.

5. Sponsor Advertisement & Sponsor Management

• SponsorAd

o Attributes:

- **adId:** Unique advertisement ID.
- **sponsorId:** Sponsor user reference.
- **title:** Advertisement title.
- **description:** Promotional content.
- **images:** List of Cloudinary image URLs.
- **videos:** Optional promotional videos.
- **address:** Sponsor's business address.
- **contact:** Sponsor contact details.
- **poster:** Featured poster URL.
- **status:** Drafted / Published.
- **views:** Impression count.
- **likes:** Like count.
- **createdAt:** Timestamp.
- **updatedAt:** Timestamp.

o Methods:

- **createAd():** Create advertisement draft.
- **updateAd():** Edit advertisement.
- **publishAd():** Make advertisement public.
- **deleteAd():** Delete advertisement.

• SponsorDashboard

o Attributes:

- **sponsorId:** Logged-in sponsor.
- **draftedAds:** List of draft ads.
- **publishedAds:** List of published ads.
- **viewsReport:** Total views across ads.
- **likesReport:** Total likes across ads.
- **pastSponsoredEvents:** Events previously sponsored.

o Methods:

- **getDraftedAds():** Fetch drafted ads.
- **getPublishedAds():** Fetch published ads.
- **getViewsReport():** Return view analytics.
- **getLikesReport():** Return engagement metrics.
- **getSponsoredEvents():** Show event partnerships.
- **exportAdAnalytics():** Future CSV export feature.

• SponsorProfile

o Attributes:

- **firmDescription:** Company/brand description.
- **firmLogo:** Logo URL.
- **links:** List of product or service links.

o Methods:

- **updateProfile():** Update sponsor info.

- **uploadLogo():** Upload logo to Cloudinary.
- **addProductLink():** Add product/service link.
- **removeProductLink():** Delete link.

• AdInteractionService

o Methods:

- **recordView(adId):** Increment ad view count.
- **getAdById(adId):** Return advertisement details.
- **getAdsBySponsor(sponsorId):** Fetch ads by sponsor.

• PublicSponsorListing

o Attributes:

- **sponsorCards:** Public list of sponsors.
- **adCards:** Public list of sponsor ads.

o Methods:

- **listAllSponsors():** Show all sponsors with meta details.
- **listAdsForSponsor(sponsorId):** List ads posted by a sponsor.
- **getAdPreview(adId):** Provide preview of ad.

• Sponsor Subsystem — Interaction Summary

o Interacts With:

- **Event Management:** Sponsors linked to events.
- **User Management:** Only sponsor-role users can create ads.

- **Cloudinary Service:** Upload sponsor logos, posters, images.
- **Student Dashboard:** Students view sponsor ads.
- **Analytics Service:** Track impressions and engagement.
- **Inbox System (future):** Sponsor–organizer/admin communications.

6. Inbox, Notification & Messaging Subsystem

• InboxEntity

o Attributes:

- **inboxId:** Unique identifier.
- **type:** announcement / team_invite / subevent_invite / sponsorship_req / mou_req / registration_approval / message / report.
- **title:** Short title for inbox item.
- **description:** Optional detailed description.
- **from:** Sender (User reference).
- **to:** List of recipient users.
- **status:** Draft / Sent / Approved / Rejected / Pending.
- **message:** Message body.
- **relatedEvent:** Optional event reference.
- **relatedTeam:** Optional team reference.
- **relatedTeamModel:** “OrganizerTeam” or “StudentTeam”.
- **createdAt:** Timestamp.
- **updatedAt:** Timestamp.
- **approvals:** List of ApprovalRecord entries.

- **attachments:** List of Cloudinary URLs.

o Methods:

- **saveDraft(userId):** Save inbox item as draft.
- **send():** Mark as Sent + trigger notifications.
- **addRecipient(userId):** Add receiver.
- **removeRecipient(userId):** Remove receiver.
- **changeStatus(newStatus):** Modify approval state.
- **addAttachment(url):** Attach file.
- **getDeliveryReceipts():** Track recipient actions.

• ApprovalRecord

o Attributes:

- **approver:** User reference.
- **action:** Approved / Rejected / Pending.
- **comments:** Optional remarks.
- **timestamp:** When action happened.

• InboxServiceo Methods:

- **createDraft(payload, user):** Save new draft.
- **editDraft(inboxId, updates, user):** Update draft.
- **deleteDraft(inboxId, user):** Remove draft.
- **sendMessage(inboxId, user):** Send message from draft.

- **sendDirectMessage(payload):** Create + send instantly.
- **getDrafts(userId):** Return user drafts.
- **getSent(userId):** Items user has sent.
- **getArrivals(userId):** Messages received by users.
- **changeApprovalStatus(inboxId, action, approver):** Approval or rejection flow.
- **notifyRecipients(inboxEntity):** Send in-app + email notifications.
- **archive(inboxId):** Archive inbox item.

• InboxController (REST interface)

o Endpoints:

- POST `/inbox/drafts` → createDraft
- PUT `/inbox/drafts/:id` → editDraft
- DELETE `/inbox/drafts/:id` → deleteDraft
- POST `/inbox/send` → sendDirectMessage / send draft
- GET `/inbox/drafts` → getDrafts
- GET `/inbox/sent` → getSent
- GET `/inbox/arrivals` → getArrivals
- PUT `/inbox/approve/:id` → changeApprovalStatus
- PUT `/inbox/reject/:id` → changeApprovalStatus

o Security: Requires authentication + role-based authorization.

• NotificationService

o Attributes:

- **queue:** Job queue (e.g., Bull/Redis).
- **emailClient:** SMTP/SendGrid provider.
- **pushClient:** Socket/WebSocket handler.

o Methods:

- **enqueueNotification(payload):** Push job to queue.
- **deliverInApp(userId, payload):** Store + push via socket.
- **sendEmail(email, subject, body):** Email delivery.
- **sendSMS(phone, body):** Optional SMS.
- **markAsRead(notificationId, userId):** Mark read.
- **batchDeliver(recipients, payload):** Optimized batch send.

• Notification (Model)

o Attributes:

- **notificationId:** Unique ID.
- **to:** Target user.
- **type:** announcement / invite / request / system.
- **payload:** Minimal JSON (title, body, link).
- **read:** Boolean.
- **createdAt:** Timestamp.

• MessageDraft

o Attributes:

- **draftId:** Unique draft ID.

- **owner:** User who owns draft.
- **content:** Message text.
- **recipients:** List of user IDs.
- **attachments:** List of URLs.
- **createdAt:** Timestamp.
- **updatedAt:** Timestamp.

o Methods:

- **save():** Save draft.
- **update():** Update draft.
- **publish():** Convert to InboxEntity via InboxService.

• InboxQueryService

o Methods:

- **listDrafts(userId, filters):** Paginated draft list.
- **listSent(userId, filters):** Sent items.
- **listArrivals(userId, filters):** Received items.
- **filterByType(type):** By inbox type.
- **getEntityDetails(inboxId):** Expand event/team info.

• ReportHandler

o Attributes:

- **reportId:** Unique ID.
- **reportedBy:** User who filed report.
- **reportedEntity:** { type, id }.

- **status:** Pending / Reviewed / ActionTaken.
- **reason:** Report reason.
- **createdAt:** Timestamp.

o Methods:

- **createReport(reporter, entity, reason):** Insert new report.
- **escalate(reportId):** Mark urgent.
- **resolve(reportId, action):** Update status + notify users.

• Inbox & Notification Interaction Summary

o Integrations:

- **Authentication:** Validate sender/approver.
- **User Management:** Fetch user profiles & emails.
- **Event Management:** Announcements and approval flows.
- **Team Management:** Invitation and team join flows.
- **Admin Panel:** Suspend/approve actions via inbox.
- **Notification Service:** Deliver in-app + email alerts.
- **Realtime System:** Use sockets for live notifications.
- **Cloudinary:** Store attachments.

8. Registration, Payment & Check-In Subsystem

• Registration

o Attributes:

- **registrationId:** Unique ID.
- **eventId:** Event reference.
- **studentId:** User reference.
- **studentTeamId:** Team reference (for team registrations).
- **registrationType:** Individual / Team.
- **paymentStatus:** Unpaid / Paid / Pending Approval.
- **paymentScreenshotUrl:** Proof of payment.
- **registrationData:** List of RegistrationFieldAnswer objects.
- **checkInCode:** Unique code for attendance tracking.
- **checkIns:** List of CheckInRecord entries.
- **status:** Pending / Approved / Rejected.
- **createdAt:** Timestamp.
- **updatedAt:** Timestamp.

o Methods:

- **validateBeforeSubmit():** Check deadline & rules.
- **submit():** Create registration entry.
- **markPaymentPending():** Await organizer approval.
- **approvePayment():** Verify payment proof.
- **rejectPayment():** Reject payment or proof.
- **addCheckIn(timelineId):** Add attendance record.
- **generateCheckInCode():** Create QR/check-in code.

• RegistrationFieldAnswer

o Attributes:

- **title:** Field name.

- **description:** Optional help text.
- **answer:** Student-provided value.

• EventRegistrationForm

o Attributes:

- **formId:** Unique ID.
- **fields:** List of RegistrationField objects.
- **teamSizeRange:** Min–max team size.
- **fees:** Registration fee.
- **qrCodeUrl:** Payment QR.
- **registrationType:** Individual / Team.

o Methods:

- **addField(field):** Add input field.
- **removeField(id):** Delete field.
- **updateField(id, updates):** Modify field.
- **validateFormInput(data):** Check required fields.
- **isPaidEvent():** Return fees > 0.

• RegistrationField

o Attributes:

- **fieldId:** Unique identifier.
- **title:** Field label.
- **description:** Additional help text.

- **inputType:** text / number / checkbox / options / date / time.
- **required:** Boolean.
- **options:** List of selectable values.

• RegistrationService

o Methods:

- **getRegistrationForm(eventId):** Return dynamic form.
- **submitRegistration(data):** Create registration.
- **validateTeamRegistration(teamId):** Check team eligibility.
- **processPaymentScreenshot():** Upload proof to Cloudinary.
- **getRegistrationStatus(eventId, userId):** Check status.
- **requestPaymentApproval(registrationId):** Set "Pending Approval".
- **approveRegistration(registrationId):** Organizer approval.
- **rejectRegistration(registrationId):** Organizer rejection.
- **checkDeadline(eventId):** Prevent late registrations.

• CheckInRecord

o Attributes:

- **timelineId:** Stage/session ID.
- **checkInTime:** Timestamp.
- **status:** present / absent.

• CheckInService

o Methods:

- **validateCheckIn(code, eventId, timelineId):** Verify registration.
- **addCheckIn(eventId, studentId, timelineId):** Mark attendance.
- **generateCodesForParticipants(eventId):** Create QR codes.
- **getAttendees(eventId):** List present participants.
- **getAbsentees(eventId):** List absentees.

• PaymentService

o Methods:

- **validatePaymentScreenshot(url):** Validate Cloudinary resource.
- **assignPaymentStatus(registrationId, status):** Set state.
- **verifyPayment(registrationId):** Manual approval.
- **rejectPayment(registrationId):** Reject payment.
- **sendPaymentNotification(userId):** Inbox + email alerts.

• RegistrationAnalytics

o Methods:

- **totalRegistrations(eventId):** Count of registrations.
- **paidVsUnpaid(eventId):** Payment insights.
- **attendanceRate(eventId):** Check-ins ÷ registrations.
- **peakRegistrationTime(eventId):** Identify peak timing.

• Registration Subsystem — Interaction Summary

o Interacts With:

- **Event Management:** Registration settings & restrictions.
- **User System:** Student information.
- **Student Dashboard:** Show registration status.
- **Organizer Dashboard:** Manage approvals & logs.
- **Check-In System:** Attendance marking.
- **Cloudinary:** Upload payment proofs.
- **Inbox System:** Payment approvals/notifications.
- **Recommendation Engine:** Use past registrations for ML.

9. Search, Filtering & Map Annotation Subsystem

• SearchService

o Attributes:

- **rankingModel:** RecommendationHandler (optional).
- **cache:** Redis client for caching queries.

o Methods:

- **indexEvent(event):** Index event fields.
- **updateIndex(eventId, data):** Update document.
- **deleteIndex(eventId):** Remove from index.
- **search(query, filters, sort, page):** Full-text search.
- **suggest(prefix):** Autocomplete hints.

- **getTrending(params):** Trending events by signals.
- **rebuildIndex():** Admin reindex.

o Notes:

- Use n-grams, synonyms, and tokenization.
- Store coordinates for geo-search.

• GeocodingService

o Attributes:

- **provider:** Nominatim / Mapbox / Google proxy.
- **cache:** Redis cache.

o Methods:

- **searchLocation(query):** Location suggestions.
- **reverseGeocode(lat, lng):** Address lookup.
- **normalizeAddress(address):** Clean formatted address.
- **proximitySearch(lat, lng, radiusKm):** Nearby events.

o Security:

- Use backend proxy.
- Rate-limit & cache.

• MapAnnotator (MapAnnotationService)

o Attributes:

- **tileProvider:** OSM base URL.
- **leafletAdapter:** Leaflet wrapper.

- **allowedIcons:** Icons registry.
- **storage:** Save annotations to event model.

o Methods:

- **createAnnotation(eventId, data):** Add marker.
- **editAnnotation(eventId, annotationId, updates):** Edit marker.
- **deleteAnnotation(eventId, annotationId):** Remove marker.
- **listAnnotations(eventId):** Return GeoJSON.
- **exportAnnotations(eventId, format):** Export GeoJSON/KML.
- **importAnnotations(eventId, geojson):** Bulk import.
- **validateAnnotationPayload(payload):** Validate coordinates.

o Data Model: GeoJSON-like structure.

• LeafletAdapter

o Attributes:

- **defaultLayers:** Base & overlays.
- **pluginList:** draw / markercluster / geojson.

o Methods:

- **initializeMap(containerId, options):** Setup map.
- **addLayer(config):** Add layer.
- **addMarker(annotation):** Render marker.
- **removeMarker(annotationId):** Remove marker.
- **onAnnotationEdit(callback):** Hook for edit events.
- **serializeLayerToGeoJSON(layer):** Export shapes.

• MapTileService

o Methods:

- **getTileURL(z, x, y):** Tile endpoint.
- **getAttribution():** Attribution text.
- **configureTileProvider(providerConfig):** Swap provider.

• SpatialIndex

o Attributes:

- **dbIndexType:** Mongo 2dsphere.
- **gridCaching:** Optional region caching.

o Methods:

- **findWithinRadius(lat, lng, radius, filters):** Radius search.
- **findWithinBounds(bounds, filters):** Bounding box search.
- **nearestNeighbors(point, k):** k-nearest events.

• SearchController

o Endpoints:

- GET `/search` → full search.
- GET `/search/suggest` → autocomplete.
- GET `/search/trending` → trending.
- GET `/events/nearby` → proximity search.

- GET `/events/:id/annotations` → list annotations.
- POST `/events/:id/annotations` → create annotation.
- PUT `/events/:id/annotations/:aid` → edit annotation.
- DELETE `/events/:id/annotations/:aid` → delete annotation.
- GET `/geocode/search` → geocoding proxy.

o Security:

- Annotation changes restricted to organizers.
- Search public but personalized ranking requires login.

• Search & Map Interaction Summary

o Integrates With:

- **Event Management:** Index events.
- **Recommendation Engine:** Re-rank results.
- **Student Dashboard:** Nearby events + map view.
- **Inbox/Notifications:** Venue-update alerts.
- **MapTileService:** Fetch map tiles.
- **AI/NLP:** Auto-tagging (optional).
- **Qdrant:** Semantic vector search.

10. Admin Panel & College Management Subsystem

• College

o Attributes:

- **collegeId:** Unique identifier.
- **name:** College name.
- **code:** Unique short code (e.g., IITB, NIT-XXX).
- **logo:** Logo URL (Cloudinary/CDN).
- **poc:** College point-of-contact info.
- **address:** City, state, country, pincode.
- **website:** College website URL.
- **description:** About the college.
- **tags:** Category/feature tags.
- **status:** Pending / Approved / Rejected.
- **approvedBy:** Admin user who approved.
- **createdAt / updatedAt:** Timestamps.

o Methods:

- **applyRegistration(details):** Submit registration request.
- **approve(adminId):** Mark as approved.
- **reject(adminId):** Mark as rejected.
- **suspend():** Suspend college + cascade suspension.
- **unsuspend():** Remove suspension cascade.

• AdminUser

o Attributes:

- **adminId:** Unique ID.

- **name:** Admin name.
- **email:** Admin login.
- **role:** Always “admin”.
- **permissions:** approveCollege, suspendUser, viewAllEntities, handleReports.

o Methods:

- **approveCollege(collegeId):** Approve college request.
- **handleReport(modelType, id):** Process report.
- **suspendEntity(modelType, id):** Suspend user/event/ad.
- **unsuspendEntity(modelType, id):** Reverse suspension.
- **viewStats():** Platform overview.

• SuspensionService

o Attributes:

- **suspendedModels:** User, Event, SponsorAd, College.

o Methods:

- **suspendCollege(collegeId):** Suspend all users/events in college.
- **unsuspendCollege(collegeId):** Restore entities.
- **suspendUser(userId):** Mark user suspended.
- **suspendEvent(eventId):** Mark event suspended.
- **suspendAd(adId):** Suspend advertisement.
- **recursiveSuspend():** Apply cascading suspension rules.

o Cascading Rule:

- Suspend a college → automatically suspend users, events, ads under it.

• Report

o Attributes:

- **reportId:** Unique ID.
- **modelType:** event / user / ad.
- **modelId:** Target being reported.
- **reason:** Reason tag.
- **description:** Full explanation.
- **reportedBy:** User who filed report.
- **status:** Pending / Reviewed / ActionTaken.
- **createdAt:** Timestamp.

o Methods:

- **create(modelType, modelId, reason, byUser):** Insert report.
- **resolve(adminId, action):** Mark resolved.
- **linkInbox():** Notify admin + team leads.

• AdminDashboard

o Responsibilities:

- Manage college approvals.
- View users by role/college.
- View events globally.
- View sponsor ads.
- Resolve reports.

- Toggle suspensions.

o Methods:

- **loadSummaryWidgets():** Stats cards.
- **loadPendingColleges():** Approval list.
- **loadReports():** Pending reports.
- **renderEntityTable():** Table for users/events/ads.

• AdminService

o Methods:

- **handleCollegeRegistration(collegeId):** Approve/reject college.
- **suspendCollege(collegeId):** Cascade suspension.
- **unsuspendCollege(collegeId):** Reverse cascade.
- **toggleSuspension(modelType, id):** Suspend/unsuspend entity.
- **createReport(modelType, id, details):** Create report entry.
- **getAllColleges():** Admin listing.
- **getAllEvents():** Global event view.
- **getAllUsers():** Global user view.
- **getAllAds():** Sponsor ads list.

• Admin Panel Workflows

A. College Registration:

1. User submits application.
2. Admin checks pending list.

3. Admin approves/rejects.
4. Status updated.
5. All users linked become valid/invalid.
6. Notification sent via Inbox.

B. Suspension:

1. Admin clicks “Suspend”.
2. Entity marked suspended.
3. If college suspended → suspend all related entities.
4. Suspended users blocked by middleware.

C. Report Handling:

1. User files report.
2. Backend creates Report.
3. Inbox alerts to admin/team.
4. Admin inspects.
5. Admin either:
    - Action: suspend entity
    - Reject: mark as “Reviewed”
6. Report finalized.

• Interaction Summary

o Interacts With:

- **User System:** Suspend/view users.
- **Event System:** Suspend events.

- **Sponsor Ads:** Suspend ads.
- **Inbox System:** Notifications for reports.
- **College System:** Approve/manages colleges.
- **Middleware:** Block suspended accounts.
- **Report System:** Resolve reports.

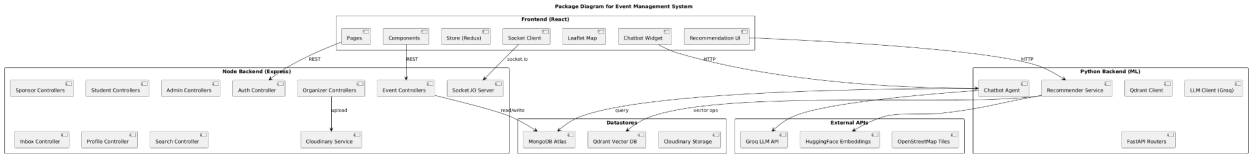
• Roles & Permissions

- **Admin:** Full platform control.
- **Organizer:** Manage only their events/college.
- **Student:** No admin permissions.
- **Sponsor:** No admin permissions.

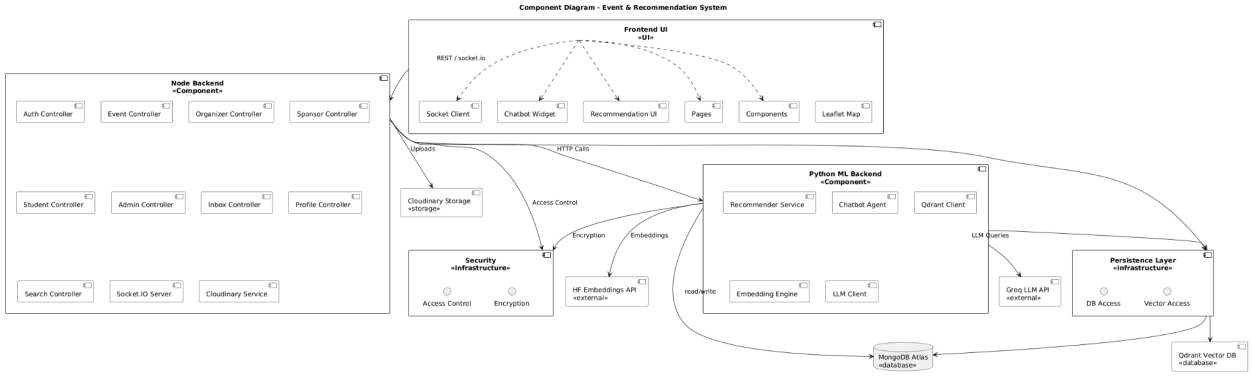
• Error Handling Rules

- Suspended organizer → all organizer routes blocked.
- Suspended event → registrations disabled.
- Suspended college → all its users/events locked.
- Invalid report → return HTTP 400.
- All admin APIs protected by **authorizeRoles("admin")**.

Package Diagram



Component Diagram



Deployment Diagram