We modeled a real-time notification system for a community events app. Our goals were to support simultaneous registrations, instant notifications, simple AI suggestions, and safe logging.

Design choices

We used Observer for real-time subscriptions, Factory to create notification objects (Push/Email/SMS), and Singleton for Event-Manager as the single source of truth. For concurrency we recommend optimistic locking with a version field and optional Redis counters for performance.

A user registers → Event-Manager checks capacity → reserves atomically → Notification-Factory creates a confirmation which is sent asynchronously → AI-Suggester suggests similar events → Admin-Logger records the action. If the event is full, users receive an 'event full' notification instantly.

Trade-offs & reflection

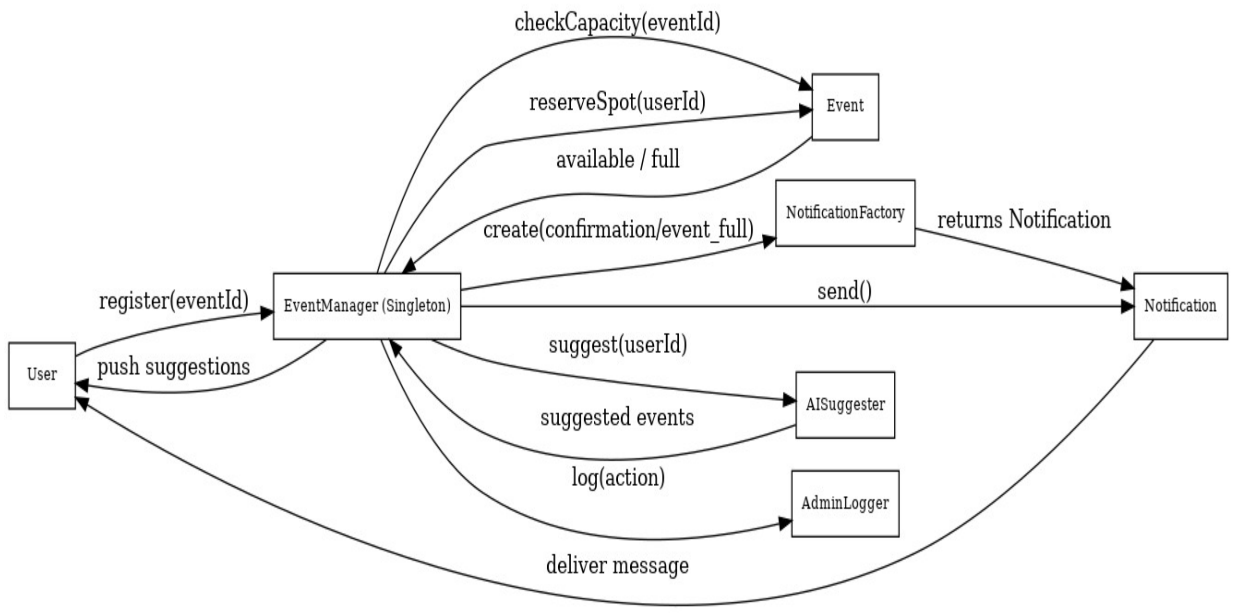
We chose **Observer** because notifying multiple users when an event changes is naturally a one-to-many problem; it fits cleanly with Web-Socket/SSE subscriptions. The **Factory** pattern decouples notification creation from sending logic so we can extend delivery channels without changing business logic. **Singleton** for `Event-Manager` simplifies the mental model there is one authoritative manager that coordinates checks and notifications.

Trade-offs: this design keeps the system simple to understand and extend, but places more responsibility on `Event-Manager` (risking a bottleneck). To avoid that in production we would split responsibilities across micro-services (e.g., Event Service, Notification Service) and use queues for durability and throughput. We also prioritized **optimistic locking** which favors throughput and simplicity, but requires retry logic for edge-case collisions.

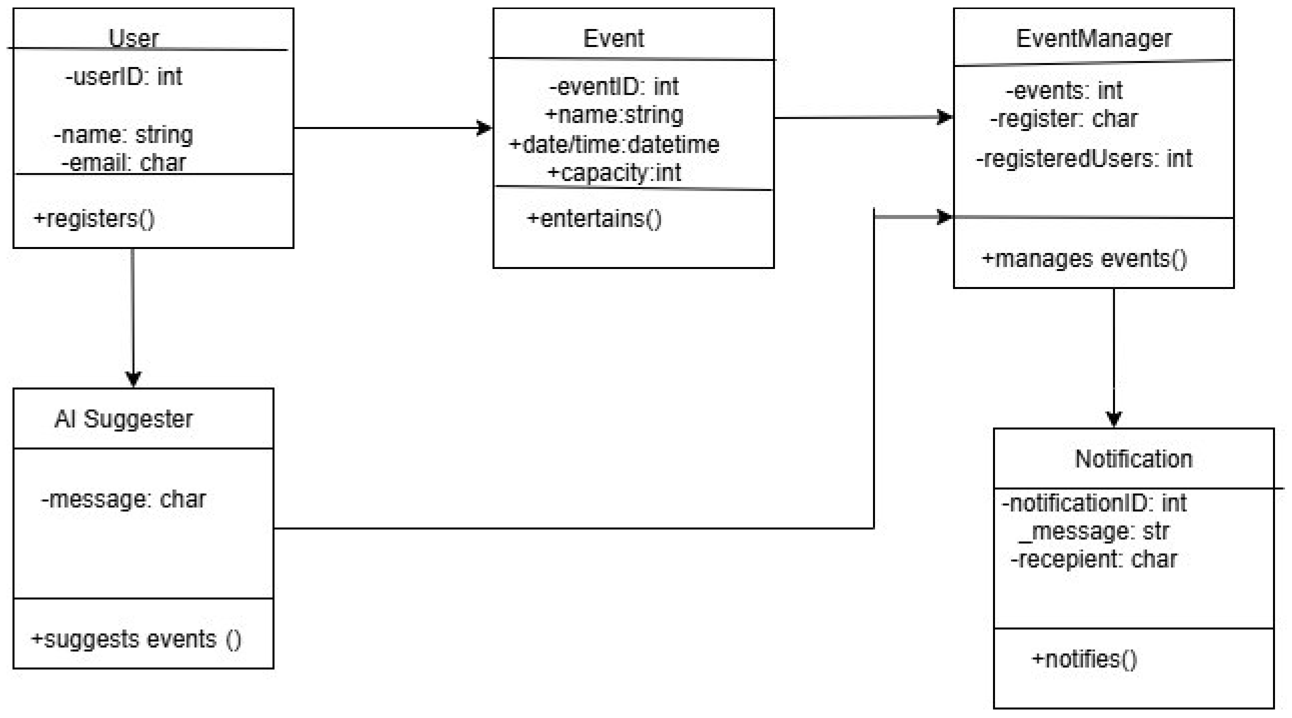
Ethical note: store minimal personal data, protect device tokens and emails, and provide users with options to opt out of notifications.

Trade-offs: simple and extensible but Event-Manager can become a bottleneck at scale; we suggested queuing and micro-services as scaling strategies. Any questions?

SEQUENCE DIAGRAM



CLASS DIAGRAM



User
-userID: int
-name: string
-email: char
+registers()
Event
-eventID: int
+name:string
+date/time:datetime
+capacity:int
+entertains()
EventManager
-events: int
-register: char
-registeredUsers: int
+manages events()
AI Suggester
-message: char
+suggests events ()
Notification
-notificationID: int
_message: str
-recepient: char
+notifies()