**Business requirements**

We’re building a system for managing online events (e.g. conferences).

We already have a 3rd party video streaming service purchased, but the rest of the software and infrastructure should be built from scratch.

The video streaming service provides us with the following API

* create and dispose virtual rooms - the environment for holding the events
* issue time-limited tokens for accessing a virtual room (as a streamer or a spectator)
* creating a video stream by token - returns a stream id
* connecting to a stream as a spectator by token and stream id
* starting/stopping a video stream by token and stream id
* retrieving a video record by stream id

Thus, we have users who can

* register using their email
* enroll as a spectator for an event
* request an event as a speaker - this leads to allocating a virtual room for a certain time slot

An event request includes

* the speaker id
* expected duration
* content plan - free-form text
* topic tags

Event requests are handled by moderators

* they monitor pending event requests
* may ask to correct the content plan/topic tags
* may refuse event requests
* speakers should be notified via email about what’s going on with their event requests

Users may subscribe to a newsletter about upcoming events by specific tags.

Once an event is complete, its recording may be published on YouTube based upon the speakers’ decision.

We would like to build a vendor-agnostic solution in the cloud. We plan to launch in specific countries only. We plan to start with an audience of about 10000 watchers, 2000 speakers, and 500 events a week.

**Online Event Management System Documentation**

**Overview**

* The system is designed to manage online events, enabling users to register, enroll as spectators, and request to host events as speakers. The system integrates with a third-party video streaming service for managing virtual rooms and video streams.
* Initial launch in selected countries with around 10,000 spectators, 2,000 speakers, and 500 events per week.
* Vendor-agnostic, cloud-based infrastructure to support scaling

**Functional Requirements**

**User Registration**

* **Action:** Users register using their email addresses.
* **Outcome:** Registered users can either enroll as spectators or request events as speakers.

**Event Management**

* **Speakers:**  
  Speakers can request events by providing the following:
  + Speaker ID
  + Expected duration
  + Content plan (free-form text)
  + Topic tags
* **Moderators:**  
  Moderators are responsible for managing event requests:
  + Monitor pending requests.
  + Request corrections to content plans or tags.
  + Approve or deny event requests.
  + Send email notifications to speakers regarding the status of their requests.

**Video Streaming Service Integration**

* The third-party video service offers APIs for:
  + Creating and disposing of virtual rooms.
  + Issuing time-limited tokens for streamers and spectators.
  + Starting/stopping video streams by token and stream ID.
  + Retrieving video records after the stream.

**Newsletters & Subscriptions**

* **Action:** Users can subscribe to newsletters based on specific tags (e.g., topics of interest).
* **Outcome:** Users will be notified about upcoming events related to their subscribed tags.

**Non-Functional Requirements**

**Performance**

* Must support an audience of 10,000 watchers, 2,000 speakers, and 500 events per week.

**Scalability**

* The system should be scalable to accommodate growth and peak traffic.

**Reliability**

* Ensure high availability of event streams and related infrastructure.

**Security**

* Tokens for accessing streams should be time-limited and securely managed.
* Personal data (emails, subscriptions) should be encrypted and secured.

**Technical Architecture**

**Cloud Infrastructure**

* Vendor-agnostic cloud setup to ensure flexibility and cost-efficiency.

**Third-Party Video Streaming Service**

* **API Integration:**
  + Creating virtual rooms.
  + Managing stream tokens.
  + Retrieving video streams and records.

**Database**

* **Users:** Store user information, subscriptions, and event participation history.
* **Events:** Store event details

**Event Request Lifecycle**

* **Creation:** Speaker submits a request → Moderator reviews → Event approved or rejected.
* **Notifications:** Email notifications for status updates to speakers.

**User Roles**

**Spectators**

* Register via email.
* Enroll for events.

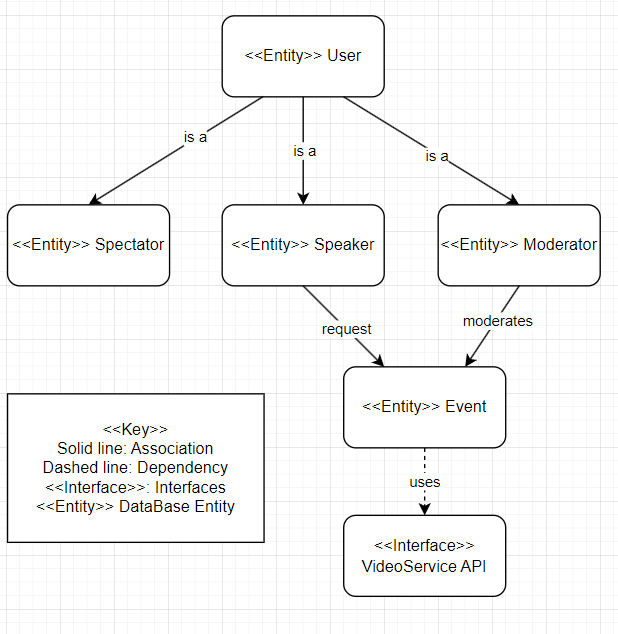
**Speakers**

* Request to host events.

**Moderators**

* Monitor, approve, or reject event requests.
* Request corrections to event details if needed.
* Notify speakers of the event status.

**UML Online Event Management System**



**Key**: Explaining the notation used (solid lines for associations, dashed lines for dependencies).

**Entities**: User, Spectator, Speaker, Moderator, Event, and VideoServiceAPI.

**Relationships**: Shows how users interact with events and the VideoServiceAPI

The purpose of the diagram is to visually represent the key entities and relationships in the online event management system, illustrating how users (spectators, speakers, moderators) interact with events and the video streaming service. It provides a simplified overview of the system's core components and their dependencies.