Event Manager

# Schema

## Users { "\_id": ObjectId, "name": String, "friendlyName": String, "zipcode": String, "birthyear": Number, "sex": String, "radius": Number, "preferences": { /\* key-value pairs \*/ }, "updatedAt": ISODate }

## Events { "\_id": ObjectId, "title": String, "dateTime": ISODate, "location": String, "cost": Number, "numOfExpectedAttendees": Number, "recurrence": String, "tags": [String], "updatedAt": ISODate }

## userEvents { "\_id": ObjectId, "userId": ObjectId, // references users.\_id "eventId": ObjectId, // references events.\_id "attended": Boolean, "rating": Number, "note": String, "updatedAt": ISODate }

## userTagAffinities { "\_id": ObjectId, "userId": ObjectId, // references users.\_id "tag": String, "affinity": Number, // negative=dislike, positive=like "updatedAt": ISODate }

## urls { "\_id": ObjectId, "url": String, // The URL to crawl "params": { // JSON object of key-value pairs for query parameters "key": "value" }, "updatedAt": ISODate // The last time the URL was updated }

## crawls { "\_id": ObjectId, "url\_id": ObjectId, // references urls.\_id "last\_parsed\_date": ISODate, // The last time this URL was crawled "parse\_status": { // Status of the crawl "http\_code": Number, // HTTP status code from the crawl (e.g., 200, 404) "status": String // Enum: "success", "error", "pending", "in\_progress" }, "errors\_encountered": [ // Array of error messages from the crawl "String" ], "updatedAt": ISODate // The last time this record was updated }

# Rest Endpoints

## Users POST /users → Create a user GET /users/{userId} → Get user details PATCH /users/{userId} (or PUT) → Update user details (like name, preferences) DELETE /users/{userId} → Delete user GET /users → List all users

## Events POST /events → Create an event GET /events/{eventId} → Retrieve event details PATCH /events/{eventId} → Update event fields (title, dateTime, tags, etc.) DELETE /events/{eventId} → Remove event GET /events → List all events

## userEvents POST /userEvents → Create a link record (user + event) GET /userEvents/{id} → Retrieve one link’s details (attended, rating, note) PATCH /userEvents/{id} → Update rating, note, etc. DELETE /userEvents/{id} → Remove that link GET /users/{userId}/events → Get all userEvents for a user

## userTagAffinities POST /userTagAffinities → Create user’s affinity for a tag GET /userTagAffinities/{id} → Retrieve one record PATCH /userTagAffinities/{id} → Update affinity DELETE /userTagAffinities/{id} → Remove that affinity GET /users/{userId}/tagAffinities → List all affinities for a user

## urls POST /urls → Add a new URL to crawl GET /urls/{id} → Retrieve details for a specific URL PATCH /urls/{id} → Update an existing URL (e.g., parameters) DELETE /urls/{id} → Remove a URL from the system GET /urls → List all URLs

## crawls POST /crawls → Log a new crawl attempt for a URL GET /crawls/{id} → Retrieve crawl details (status, errors, etc.) PATCH /crawls/{id} → Update crawl status or errors DELETE /crawls/{id} → Remove a crawl log GET /crawls → List all crawl logs

# File Hierarchy A computer screen shot of a program Description automatically generated

# Enhancements

## Users

User profiles – a user may search based on a profile/preference set similar to a different user. Each profile will have it’s own set of tags and location/zip

## Urls

Each URL will have a set of tags so that searches may be triggered based on a user/user profile set of tags.