**📝 Project Description: Eventivity**

**Eventivity** is a **Progressive Web Application (PWA)** designed to support and enhance **real-world events** such as conferences, cultural festivals, or educational workshops. Its key concept is **adaptability** — while the core functionality remains consistent across events, the visual theme and branding dynamically adjust to suit the identity of each event.

The app aims to deliver a seamless, responsive experience through modern web technologies, providing attendees with:

* A **dynamic landing page** tailored to the event’s theme
* **Schedules and timetables** for workshops, talks, or key sessions
* **Google Maps integration** for navigating the local area
* A **forum/blog area** for attendee interaction and feedback
* **Custom QR codes** that link to **brief video overviews** or construction sequences related to workshops
* **User accounts with simple authentication**, enabling participation in forum discussions

Eventivity is built using a modern JavaScript stack including **ReactJS (or basic JS + templating)**, **Node.js** for backend logic, and **Tailwind CSS** or **Material UI** for design. The app is deployed using **AWS services**, specifically:

* **EC2** for backend hosting
* **S3** and **CloudFront** for serving the static frontend
* **DynamoDB** for fast, scalable NoSQL storage
* **Cognito** for managing user authentication securely

**✅ DynamoDB + S3 Combo (Scalable & Serverless)**

| **Purpose** | **AWS Service** |
| --- | --- |
| Blog/forum post data (text, user, timestamp, etc.) | 🗃️ **DynamoDB** (NoSQL) |
| Attached media (images, videos, PDFs) | 📦 **S3** (object storage) |

**🧱 DynamoDB Design: Forum Post Table**

Let’s sketch out what your Posts table might look like:

| **Attribute** | **Type** | **Notes** |
| --- | --- | --- |
| postId | String (PK) | Unique post ID (UUID) |
| eventId | String | If you support multiple events |
| authorId | String | From Cognito / JWT |
| authorEmail | String | Optional — for reference |
| title | String | Post title |
| content | String | Main text body |
| timestamp | Number | Unix timestamp |
| mediaUrl | String | Link to S3 asset (optional) |
| tags | List | Optional: workshop, Q&A, etc. |

You can query by eventId if you want to fetch all posts from one event.

**📦 S3 Bucket: Media Storage**

When a user attaches media:

* Upload it to S3 (via your Node backend or signed URL)
* Store the resulting URL in the mediaUrl field of the post

**🔐 Auth and Access Control**

Since you’re using Cognito:

* Use the Cognito sub or email to associate posts with users
* Restrict media uploads to signed-in users

**✅ Suggested Branch Name**

Let’s spin up a new feature branch:

bash

CopyEdit

git checkout -b feature/forum-dynamodb

Then I can help you:

1. Set up DynamoDB schema
2. Create basic post form (title, body, optional upload)
3. Integrate with your Node backend
4. Display posts in the /forum route

Ready to kick off the schema and backend work?

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Complexity isn't about building huge systems — it's about having meaningful, connected, valuable features.

A black screen with white text

AI-generated content may be incorrect.

Serverless architecture (AWS Lambda + API Gateway) was selected to prioritize rapid feature development, scalability, and reduced infrastructure maintenance. This allowed focusing project effort on delivering user-facing functionality rather than backend server management."

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Further Improvements: Dynamic Workshop Pages**

Currently, the workshop detail pages in Eventivity are statically created, meaning each page's content is hardcoded into the application. While this approach is sufficient for a limited number of workshops and ensures reliable behavior during the project's deadline, it does not scale well if the number of workshops increases.

In a future version of Eventivity, I would refactor the application to store workshop information in a database, such as AWS DynamoDB. Each workshop would include fields like title, description, image URL, topics covered, and event details. When a user navigates to a workshop URL (e.g., /workshops/react-basics), the application would dynamically fetch the relevant workshop data based on the slug parameter from the URL.

This approach would allow new workshops to be added, updated, or removed without needing to redeploy the frontend application. It would improve scalability, simplify management, and make the application more maintainable long-term.

"Currently, workshop pages are static, but in future versions, I would make them dynamic by fetching workshop details from a DynamoDB table based on the URL."