Virtual Event Platform - Design Document

1. Overview

The Virtual Event Platform is a web-based application that enables users to participate in live tech events, hackathons, and networking sessions.

It provides real-time chat, video conferencing, AI-powered matchmaking, and event tracking features.

2. Tech Stack

Backend:

- NestJS (with Express as default)
- GraphQL API (Efficient data fetching)
- WebSockets (Real-time chat & notifications)
- Threads & Microservices (Scalable architecture)
- PostgreSQL (Relational database for events & users)
- Redis (Session management & caching)
- Docker (Containerized deployments)
- CI/CD (Automated deployments via GitHub Actions)
- AWS Free Tier (ECS, Lambda, RDS, S3)

Frontend:

- Next.js (App Router, TypeScript)
- Tailwind CSS & ShadCN (Modern UI)
- WebRTC (Real-time video calls)
- GraphQL Client (Apollo) (Efficient data fetching)
- Socket.io (WebSockets) (Instant messaging)

3. System Architecture

Microservices Breakdown

- Auth Service: Handles authentication & authorization (OAuth, JWT)
- Event Service: Manages event creation, scheduling & tracking
- User Profile Service: Stores user data, interests & AI-based matchmaking
- Chat Service: Manages real-time chat using WebSockets
- Video Service: Manages WebRTC video conferencing
- Notification Service: Sends real-time notifications (email, in-app, push)

4. Database Schema (PostgreSQL)

Tables:

- Users (id, name, email, interests, role, created_at)
- Events (id, name, date, speakers, description, status)
- ChatMessages (id, sender_id, receiver_id, message, timestamp)
- Matchmaking (user_id, matched_user_id, status)

5. Key Features Implementation

1 Authentication (OAuth, JWT-based)

2 Event Management (GraphQL API, real-time updates)

3 AI Matchmaking (Interest-based networking suggestions)

4 Real-time Chat (WebSockets, chat history in PostgreSQL)

5 Video Conferencing (WebRTC, peer-to-peer calls)

6 Notifications (Redis pub/sub, email & push alerts)

6. Deployment Strategy

Backend Deployment (AWS ECS + RDS)

- Dockerized Microservices
- CI/CD Pipeline (GitHub Actions AWS)
- PostgreSQL on AWS RDS

Frontend Deployment (Vercel / AWS S3 + CloudFront)

- Next.js Server-Side Rendering (App Router)
- Static assets hosted on S3
- CDN for fast content delivery

7. CI/CD Setup

GitHub Actions for Backend

```yaml
name: Deploy Backend
on:
  push:
    branches:
      - main
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
```

```
      - uses: actions/checkout@v3
      - name: Install dependencies
        run: pnpm install
      - name: Run Tests
        run: pnpm test
      - name: Deploy to AWS
        run: |
          aws ecs update-service --cluster event-platform --service backend --force-new-deployment
```

GitHub Actions for Frontend
```yaml
name: Deploy Frontend
on:
  push:
    branches:
      - main
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Install dependencies
        run: pnpm install
      - name: Build
        run: pnpm build
      - name: Deploy to Vercel
        run: vercel --prod
```

8. Summary
This Virtual Event Platform leverages NestJS microservices, WebSockets, GraphQL, Next.js, and AWS to deliver a real-time, scalable, and engaging experience for tech events.