Sports Scheduling Constraint Parser - Developer Challenge

Overview

Build a semantic search interface that translates natural language scheduling objectives into structured constraint templates for sports league scheduling optimization.

Challenge Requirements

Core Functionality

Create a search interface where users can type business objectives in natural language (e.g., "Ensure all rivalry games on a weekend on ESPN") and receive:

1. The matching constraint template
2. Parsed parameters extracted from the query
3. Confidence score for the match
4. Alternative interpretations if ambiguous

Technical Stack Requirements

- **Framework:** Next.js 14+ (App Router)
- **Deployment:** Vercel or Netlify
- **Authentication:** Any provider (Supabase Auth recommended)
- **Code Repository:** Public GitHub repository
- **Search Implementation:** Your choice of:
    - OpenAI Embeddings + Vector Search
    - Algolia with AI Search
    - Pinecone
    - Supabase pgvector
    - Any other semantic search solution

Constraint Templates to Support

Template 1: Game Scheduling Constraints

```
None
Ensure that at least <min> and at most <max> games from <games or matchups or byes>
```

```
are scheduled across <rounds> and played in any venue from <venues>
and assigned to any of <networks>.
```

Example Inputs:

- "Ensure all rivalry games on a weekend on ESPN"
- "Don't schedule high profile games on a weekday"
- "At least 2 of UTN@VU, ALA@AU, MSU@UM should all be scheduled on the final 2 dates of the season and on either CBS or ESPN"
- "Make sure UTN, UK, USC, LSU do not have any weekday byes"

Template 2: Sequence Constraints

None

```
Ensure at least <min> and at most <max> cases where there is a sequence
<games or matchups or byes>, <games or matchups or byes>, ...
across rounds <round1>, <round2>.
```

Example Inputs:

- "Make sure Oregon, Washington, UCLA, USC do not play at home on either side of their bye week"
- "Make sure Penn State plays at UCLA and at USC in back-to-back weeks in the second half of the season"

Template 3: Team Schedule Pattern Constraints

None

```
Ensure that <each of/all> teams in <teams> have at least <min> and at most
<max> instances
where they play at least <k> and at most <m> <home/away/bye/active> games
across <rounds> where the game is assigned to any of <networks>
and played in any venue from <venues>.
```

Example Inputs:

- "No cases of 3 games in 3 nights for any NBA team"
- "No cases of 5 away games in 7 nights after the all star break"
- "At most 2 cases of 3 away games in 4 rounds for Western Conference teams"

Required Features

1. Search Interface

- Clean, intuitive search input
- Real-time search as you type (optional but preferred)
- Display search results with confidence scores
- Show matched constraint template
- Display extracted parameters in a readable format

2. Result Display

Each result should show:

- **Matched Template:** Which constraint template matched
- **Parsed Constraint:** The constraint in template sentence form
- **Extracted Parameters:** Structured breakdown (min, max, teams, rounds, networks, etc.) (BONUS)
- **Confidence Score:** How well the query matched
- **Alternative Interpretations:** If query is ambiguous

Example result format:

```JSON
{
  "template": "Template 1: Game Scheduling Constraints",
  "confidence": 0.92,
  "parsedConstraint": "Ensure that at least 1 and at most 999 games from
rivalry_games are scheduled across weekend_rounds and played in any venue from
all_venues and assigned to any of ESPN.",
  "parameters": {
    "min": 1,
    "max": 999,
    "games": ["rivalry_games"],
    "rounds": ["weekend_rounds"],
    "venues": ["all_venues"],
    "networks": ["ESPN"]
  },
  "alternatives": []
}
```

3. Authentication

- Implement user authentication (Supabase Auth recommended)
- Protect the search interface behind auth

- Simple login/signup flow

4. Sample Data

Include at least 15-20 pre-indexed constraint examples covering all three templates with various phrasings.

Evaluation Criteria

Technical Implementation (40%)

- Clean, well-organized Next.js code
- Proper use of App Router patterns
- Effective semantic search implementation
- Efficient API design
- Error handling

Search Quality (30%)

- Accurate matching of natural language to templates
- Handling of ambiguous queries
- Quality of parameter extraction
- Relevance ranking

User Experience (20%)

- Intuitive interface design
- Clear result presentation
- Responsive design
- Loading states and feedback

Code Quality (10%)

- Clean, readable code
- Proper TypeScript usage (if used)
- Documentation in README
- Git commit history

Deliverables

1. **Public GitHub Repository** containing:
    - Complete Next.js application code
    - README with:
        - Setup instructions
        - Architecture decisions
        - Search implementation explanation
        - Live demo URL

    - Environment variable template (.env.example)

2. **Live Deployment** on Vercel or Netlify with:

    - Working authentication
    - Functional search interface
    - Test credentials provided in README

3. **Brief Documentation** (in README or separate doc):

    - Your approach to semantic search
    - Why you chose your particular vendor/solution
    - Challenges faced and how you solved them
    - Trade-offs you considered

Submission

Please submit:

1. GitHub repository URL
2. Live demo URL
3. Test credentials for authentication
4. Any additional notes about your implementation

Questions?

Candidates should feel free to make reasonable assumptions and document them in their README. Creative solutions are encouraged!