# Full-Stack Next.js Developer Technical Assessment

Project Overview Create a Pokemon team builder and Pokedex application using the PokeAPI. The application should allow users to explore Pokemon, build teams, and analyze Pokemon statistics.

## Technical Requirements

### Core Technologies

- Next.js 14+ with App Router
- TypeScript
- Tailwind CSS
- Zustand for state management
- TanStack Query (React Query) or SWR for API management
- NextAuth.js for authentication (optional for user-specific teams)(+ if implemented, optional)

### Features to Implement

1. **Pokemon Explorer (Pokedex)**

    - Display a grid of Pokemon with infinite scroll or pagination
    - Server-side rendering for initial data load
    - Client-side data fetching for subsequent requests
    - Advanced filtering system (by type, generation, stats)
    -

2. **Search Functionality**

    - Debounced search input
    - Dynamic search results with loading states
    - Filter options (by type, abilities, stats)
    - Implement proper error handling for failed API requests

3. **Detail View**

    - Dynamic routing for individual Pokemon pages
    - Display comprehensive information including:
        - Base stats with visual representations
        - Evolution chain
        - Moves and abilities
        - Type effectiveness

- Implement loading and error states

4. **Team Builder**

    - Allow users to create and save Pokemon teams
    - Team analysis (type coverage, weaknesses)
    - Team saving functionality (local storage or database)
    - Drag and drop interface for team management

5. **Compare Feature**

    - Side-by-side comparison of Pokemon
    - Stat comparison charts
    - Type effectiveness comparison
    - Move pool analysis

## Technical Focus Areas

1. **TypeScript Implementation**

    - Proper type definitions for API responses
    - Custom type utilities for Pokemon data
    - Strict type checking
    - Reusable interface definitions

2. **State Management**

    - Zustand store configuration
    - Team state management
    - Cache state handling
    - Type-safe store implementation

3. **API Integration**

    - Proper error boundary implementation
    - Loading state management
    - Data caching strategy
    - Parallel data fetching
    - Request optimization

4. **Code Organization**

    - Clear folder structure
    - Reusable components
    - Custom hooks for Pokemon data
    - API service layer
    - Type definitions
    - Constants and utilities

### API Information

Use the PokeAPI (https://pokeapi.co/)

- Completely free to use
- No authentication required
- Rich dataset with related data
- Multiple endpoints for different features
- Stable and well-documented

## Evaluation Criteria

1. **Code Quality**

    - TypeScript usage and type safety
    - Code organization and modularity
    - Naming conventions and readability
    - Error handling
    - Comments and documentation

2. **Architecture**

    - Component structure
    - State management implementation
    - API integration patterns
    - Performance considerations
    - Code reusability

3. **Functionality**

    - Feature completeness
    - UI/UX implementation
    - Error states handling
    - Loading states
    - Responsive design

4. **Extra Credit**

    - Performance optimizations
    - Accessibility considerations
    - Additional features

## Submission Guidelines

1. Provide a GitHub repository with:

    - Clear README with setup instructions
    - Development decisions documentation

    - Any known issues or future improvements

2. Deploy the application

3. Time limit: 1 week

## Notes for Candidates

- Focus on code quality over feature quantity
- Document any assumptions made
- Include a brief writeup of architectural decisions
- Mention any challenges faced and how they were solved
- think before you write, not everything needs to be laid out, something can be dynamic and created on-the-fly