Sortonym Challenge – Frontend API Integration Guide

1. Overview

This document explains how the frontend (React application) integrates with backend REST APIs in the Sortonym Challenge project. It covers authentication, gameplay APIs, result handling, and leaderboard integration.

2. API Communication Strategy

The frontend communicates with the backend using REST APIs over HTTP. All requests are sent using JSON format and handled asynchronously using fetch or Axios.

3. Environment Configuration

The API base URL is configured using environment variables to support multiple environments.

Example:
REACT_APP_API_BASE_URL = http://localhost:8000/api

4. Authentication API Integration

Login / Register:
Frontend sends user credentials (email or phone and password) to the authentication API. On success, the backend returns a secure token which is stored in localStorage or memory.

Authorization:
The token is attached to all protected API requests using the Authorization header.

5. Game Start API Integration

Endpoint: POST /api/game/start/
Purpose: Initialize a new game round and fetch anchor word, options, and timer settings.

The frontend calls this API when the user clicks 'Start Game' or 'Start Daily Challenge' and stores the response in component state.

6. Game Submit API Integration

Endpoint: POST /api/game/submit/
Purpose: Submit user answers, calculate score, and save performance data.

After submission, the frontend navigates the user to the Results or Loading screen based on the challenge type.

7. Daily Challenge Handling

Before starting the Daily Challenge, the frontend checks eligibility using backend response flags. If already played within 24 hours, the game start is blocked and a message is shown to the user.

Results are requested only after the backend confirms that the 24-hour lock period has expired.

8. Leaderboard API Integration

Endpoint: GET /api/leaderboard/
Purpose: Fetch ranked leaderboard data (Global or Daily).

The frontend renders leaderboard tables using the API response and highlights the logged-in user position.

9. Error Handling & Loading States

The frontend handles API failures using try/catch blocks and displays user-friendly error messages. Loading indicators are shown while API requests are in progress to ensure a smooth user experience.

10. Security Best Practices

Sensitive data such as tokens are never logged. All protected routes verify authentication state before rendering. Admin override logic is handled through role-based checks returned by the backend.

11. Conclusion

This API integration approach ensures clean separation between frontend and backend, supports scalability, and provides a secure and responsive gameplay experience.