Marito Project: Non-Functional Requirements

This document outlines the non-functional requirements (quality attributes) for the Mavito platform, reflecting the current system architecture.

NFR1 Performance

- **NFR1.1 Fast Query Responses:** The system utilizes indexed tables in PostgreSQL, queried by an asynchronous backend, to ensure fast and efficient search and data retrieval operations.
- **NFR1.2 Lightweight Frontend:** The frontend application is built with React and Vite, producing optimized static assets for fast initial page loads and a responsive user experience.
- **NFR1.3 Asynchronous API:** All backend I/O operations are non-blocking, using FastAPI's async capabilities to handle concurrent requests efficiently.

NFR2 Scalability

- **NFR2.1 Scalable Backend:** The backend is built on a microservices architecture. Each service is containerized with Docker and deployed to Google Cloud Run, which automatically scales based on incoming request traffic.
- **NFR2.2 Scalable Database:** The system uses Google Cloud SQL for PostgreSQL, a managed service that can be scaled vertically (by increasing machine size) and horizontally (with read replicas) as needed.

NFR3 Usability

- **NFR3.1 Responsive UI/UX:** The UI provides a consistent and user-friendly experience across common desktop and mobile devices.
- **NFR3.2 Intuitive Navigation:** The application interface is designed to be simple, clean, and easy to navigate.
- **NFR3.3 Offline Capability:** The frontend is a Progressive Web App (PWA) that supports offline functionality. A Service Worker with Workbox Background Sync queues write-operations (like votes) made while offline and automatically syncs them when the connection is restored.

NFR4 Reliability

- **NFR4.1 Data Integrity:** Using a centralized PostgreSQL database as the single source of truth ensures strong data integrity and consistency across all services.
- **NFR4.2 Availability:** The use of managed Google Cloud services (Cloud Run for services, Cloud SQL for the database) provides high availability and automatic recovery from failures.

NFR5 Security

- **NFR5.1 Secure Authentication:** User authentication is handled by a dedicated 'auth-service' using JWT (JSON Web Tokens). Passwords are not stored in plain text and are hashed using bcrypt.
- **NFR5.2 Secure API Access:** All backend services are fronted by a Google API Gateway, which enforces TLS/HTTPS. Protected endpoints within the services validate the JWT token to prevent unauthorized access.
- **NFR5.3 Secure File Uploads:** User-submitted documents are securely uploaded to a private Google Cloud Storage bucket via temporary, time-limited signed URLs, ensuring files are never exposed publicly.

NFR6 Maintainability & Extensibility

- **NFR6.1 Modular Design:** The microservices architecture allows for independent development, testing, and deployment of each component (e.g., ‘auth-service‘, ‘search-service‘, ‘vote-service‘).
- **NFR6.2 Shared Codebase:** A central ‘mavito-common-lib‘ contains shared database models, schemas, and configurations to ensure consistency and reduce code duplication.
- **NFR6.3 Database Versioning:** Database schema changes are managed through a dedicated ‘alembic-service‘, providing version control and a repeatable process for migrations.

NFR7 Deployment

- **NFR7.1 Containerization:** All backend services are containerized using Docker, providing a consistent runtime environment for both local development and production.
- **NFR7.2 CI/CD Pipeline:** A GitHub Actions workflow automates code quality checks (linting, type-checking), testing, Docker image builds, and deployments to Google Cloud Run.

NFR8 Data Management

- **NFR8.1 Data Ingestion:** An initial, one-time script populates the PostgreSQL database from a source JSON file, migrating the data to a persistent and structured format.
- **NFR8.2 Data Export:** The system provides an API endpoint to allow users to download terminology datasets in common formats like JSON and CSV.