NFR (Non-Functional Requirements) - Mavito Application

- **NFR1 Performance (QR)**
  - NFR1.1 Fast Indexing & Caching: The system shall employ efficient indexing and caching mechanisms to ensure fast search query responses.
  - NFR1.2 Lightweight & Efficient: The application shall be optimized for performance, especially in low-bandwidth and low-resource settings.
  - NFR1.3 Response Times: Define acceptable response times for key operations (e.g., search, page load).
- **NFR2 Scalability (QR)**
  - NFR2.1 Scalable Backend: The backend system shall be designed to handle a growing number of users, data, and requests. (Utilizing cloud-based systems like Firebase or Supabase is suggested).
  - NFR2.2 Modular Design: The system architecture shall be modular to allow for seamless integration of future datasets, features, and scaling of components.
- **NFR3 Usability (QR)**
  - NFR3.1 Responsive UI/UX: The UI shall be responsive and provide a consistent, user-friendly experience across common desktop, tablet, and mobile devices.
  - NFR3.2 Intuitive Navigation: The application interface shall be simple, clean, and easy to navigate.
  - NFR3.3 Offline-first Approach: The application design should prioritize functionality in low-bandwidth or offline environments.

    - NFR3.4 Accessibility (WCAG): (Implicitly important, can be made explicit) The application should strive to meet relevant web accessibility guidelines (e.g., WCAG 2.1 Level AA).
- **NFR4 Reliability (QR)**
    - NFR4.1 Data Synchronization Reliability: The data synchronization process between offline clients and the central repository must be reliable, preventing data loss or corruption.
    - NFR4.2 Availability: Define target uptime for the online services.
- **NFR5 Security (QR)**
    - NFR5.1 User Data Security: The system must ensure the security and privacy of user data (e.g., login credentials, contributions).
    - NFR5.2 Unauthorized Access Prevention: The system must restrict unauthorized edits or access to data and administrative functions.
    - NFR5.3 Secure Authentication: User authentication mechanisms must be secure.
- **NFR6 Maintainability & Extensibility (QR)**
    - NFR6.1 Well-documented Code: The codebase (both frontend and backend) shall be thoroughly documented to facilitate understanding and maintenance.
    - NFR6.2 Structured for Future Contributors: The project should follow consistent coding standards and architectural patterns to ease onboarding for new developers.
    - NFR6.3 Testability: The system should be designed in a way that facilitates unit, integration, and end-to-end testing.
- **NFR7 Deployment**
    - NFR7.1 Deployment Platform: The application should be easily deployable (e.g., via GitHub Pages/Vercel for frontend, cloud services for backend).
    - NFR7.2 (Optional) Dockerized Backend: The backend components may be containerized using Docker for easier deployment and environment consistency.

  - NFR7.3 CI/CD Pipeline: A Continuous Integration/Continuous Deployment pipeline should be implemented to automate testing and deployment.
- **NFR8 Data Management (System-level)**
  - NFR8.1 Version Control for Data: The system must ensure that updates to glossaries or data do not incorrectly override existing, validated information. Mechanisms for versioning or tracking changes to linguistic data should be considered.
  - NFR8.2 (Optional) Data Export/Import Feature: The system may allow researchers to download datasets in common formats (e.g., JSON, CSV).
- **NFR9 Legal and Ethical**
  - NFR9.1 Linguistic Copyrights and Licensing: The system and its data management must respect linguistic copyrights and licensing agreements of source materials.
- **NFR10 Technology Stack Specifics**
  - NFR10.1 Backend Technology: The backend will be developed using Python
  - NFR10.2 Frontend Technology: The frontend will be developed using React.js with TypeScript.
  - NFR10.3 (Optional) Open API Documentation: If APIs are provided (see FR6.2.1), they should be well-documented using standards like OpenAPI (Swagger).